Microsoft

Azure Machine Learning

Microsoft Azure Essentials

Jeff Barnes

PUBLISHED BY
Microsoft Press
A division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

ISBN: 978-0-7356-9817-8

Acquisitions, Developmental, and Project Editor: Devon Musgrave
Editorial Production: nSight, Inc.
Copyeditor: Teresa Horton
Cover: Twist Creative

Table of Contents

Foreword

I’m thrilled to be able to share these Microsoft Azure Essentials ebooks with you. The power that Microsoft Azure gives you is thrilling but not unheard of from Microsoft. Many don’t realize that Microsoft has been building and managing datacenters for over 25 years. Today, the company’s cloud datacenters provide the core infrastructure and foundational technologies for its 200-plus online services, including Bing, MSN, Office 365, Xbox Live, Skype, OneDrive, and, of course, Microsoft Azure. The infrastructure is comprised of many hundreds of thousands of servers, content distribution networks, edge computing nodes, and fiber optic networks. Azure is built and managed by a team of experts working 24x7x365 to support services for millions of customers’ businesses and living and working all over the globe.

Today, Azure is available in 141 countries, including China, and supports 10 languages and 19 currencies, all backed by Microsoft's $15 billion investment in global datacenter infrastructure. Azure is continuously investing in the latest infrastructure technologies, with a focus on high reliability, operational excellence, cost-effectiveness, environmental sustainability, and a trustworthy online experience for customers and partners worldwide.

Microsoft Azure brings so many services to your fingertips in a reliable, secure, and environmentally sustainable way. You can do immense things with Azure, such as create a single VM with 32TB of storage driving more than 50,000 IOPS or utilize hundreds of thousands of CPU cores to solve your most difficult computational problems.

Perhaps you need to turn workloads on and off, or perhaps your company is growing fast! Some companies have workloads with unpredictable bursting, while others know when they are about to receive an influx of traffic. You pay only for what you use, and Azure is designed to work with common cloud computing patterns.

From Windows to Linux, SQL to NoSQL, Traffic Management to Virtual Networks, Cloud Services to Web Sites and beyond, we have so much to share with you in the coming months and years.

I hope you enjoy this Microsoft Azure Essentials series from Microsoft Press. The first three ebooks cover fundamentals of Azure, Azure Automation, and Azure Machine Learning. And I hope you enjoy living and working with Microsoft Azure as much as we do.

Scott Guthrie

Executive Vice President

Cloud and Enterprise group, Microsoft Corporation

Introduction

Microsoft Azure Machine Learning (ML) is a service that a developer can use to build predictive analytics models (using training datasets from a variety of data sources) and then easily deploy those models for consumption as cloud web services. Azure ML Studio provides rich functionality to support many end-to-end workflow scenarios for constructing predictive models, from easy access to common data sources, rich data exploration and visualization tools, application of popular ML algorithms, and powerful model evaluation, experimentation, and web publication tooling.

This ebook will present an overview of modern data science theory and principles, the associated workflow, and then cover some of the more common machine learning algorithms in use today. We will build a variety of predictive analytics models using real world data, evaluate several different machine learning algorithms and modeling strategies, and then deploy the finished models as machine learning web service on Azure within a matter of minutes. The book will also expand on a working Azure Machine Learning predictive model example to explore the types of client and server applications you can create to consume Azure Machine Learning web services.

The scenarios and end-to-end examples in this book are intended to provide sufficient information for you to quickly begin leveraging the capabilities of Azure ML Studio and then easily extend the sample scenarios to create your own powerful predictive analytic experiments. The book wraps up by providing details on how to apply “continuous learning” techniques to programmatically “retrain” Azure ML predictive models without any human intervention.

Who should read this book

This book focuses on providing essential information about the theory and application of data science principles and techniques and their applications within the context of Azure Machine Learning Studio. The book is targeted towards both data science hobbyists and veterans, along with developers and IT professionals who are new to machine learning and cloud computing. Azure ML makes it just as approachable for a novice as a seasoned data scientist, helping you quickly be productive and on your way towards creating and testing machine learning solutions.

Detailed, step-by-step examples and demonstrations are included to help the reader understand how to get started with each of the key predictive analytic algorithms in use today and their corresponding implementations in Azure ML Studio. This material is useful not only for those who have no prior experience with Azure Machine Learning, but also for those who are experienced in the field of data science. In all cases, the end-to-end demos help reinforce the machine learning concepts with concrete examples and real-life scenarios. The chapters do build on each other to some extent; however, there is no requirement that you perform the hands-on demonstrations from previous chapters to understand any particular chapter.

Assumptions

We expect that you have at least a minimal understanding of cloud computing concepts and basic web services. There are no specific skills required overall for getting the most out of this book, but having some knowledge of the topic of each chapter will help you gain a deeper understanding. For example, the chapter on creating Azure ML client and server applications will make more sense if you have some understanding of web development skills. Azure Machine Learning Studio automatically generates code samples to consume predictive analytic web services in C#, Python, and R for each Azure ML experiment. A working knowledge of one of these languages is helpful but not necessary.

This book might not be for you if…

This book might not be for you if you are looking for an in-depth discussion of the deeper mathematical and statistical theories behind the data science algorithms covered in the book. The goal was to convey the core concepts and implementation details of Azure Machine Learning Studio to the widest audience possible—who may not have a deep background in mathematics and statistics.

Organization of this book

This book explores the background, theory, and practical applications of today's modern data science algorithms using Azure Machine Learning Studio. Azure ML predictive models are then generated, evaluated, and published as web services for consumption and testing by a wide variety of clients to complete the feedback loop.

The topics explored in this book include:

- **Chapter 1, "Introduction to the science of data,"** shows how Azure Machine Learning represents a critical step forward in democratizing data science by making available a fully-managed cloud service for building predictive analytics solutions.
- **Chapter 2, "Getting started with Azure Machine Learning,"** covers the basic concepts behind the science and methodology of predictive analytics.
- **Chapter 3, "Using Azure ML Studio,"** explores the basic fundamentals of Azure Machine Learning Studio and helps you get started on your path towards data science greatness.
- **Chapter 4, "Creating Azure ML client and server applications."** expands on a working Azure Machine Learning predictive model and explores the types of client and server applications that you can create to consume Azure Machine Learning web services.

- **Chapter 5, "Regression analytics,"** takes a deeper look at some of the more advanced machine learning algorithms that are exposed in Azure ML Studio.
- **Chapter 6, "Cluster analytics,"** explores scenarios where the machine conducts its own analysis on the dataset, determines relationships, infers logical groupings, and generally attempts to make sense of chaos by literally determining the forests from the trees.
- **Chapter 7, "The Azure ML Matchbox recommender,"** explains one of the most powerful and pervasive implementations of predictive analytics in use today on the web today and how it is crucial to success in many consumer industries.
- **Chapter 8, "Retraining Azure ML models,"** explores the mechanisms for incorporating "continuous learning" into the workflow for our predictive models.

Conventions and features in this book

This book presents information using the following conventions designed to make the information readable and easy to follow:

- To create specific Azure resources, follow the numbered steps listing each action you must take to complete the exercise.
- There are currently two management portals for Azure: the Azure Management Portal at http://manage.windowsazure.com and the new Azure Preview Portal at http://portal.azure.com. This book assumes the use of the original Azure Management Portal in all cases.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, "Press Alt+Tab" means that you hold down the Alt key while you press Tab.

System requirements

For many of the examples in this book, you need only Internet access and a browser (Internet Explorer 10 or higher) to access the Azure portal. Chapter 4, "Creating Azure ML client and server applications," and many of the remaining chapters use Visual Studio to show client applications and concepts used in developing applications for consuming Azure Machine Learning web services. For these examples, you will need Visual Studio 2013. You can download a free copy of Visual Studio Express at the link below. Be sure to scroll down the page to the link for "Express 2013 for Windows Desktop": http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx

The following are system requirements:

- Windows 7 Service Pack 1, Windows 8, Windows 8.1, Windows Server 2008 R2 SP1, Windows

Server 2012, or Windows Server 2012 R2

- Computer that has a 1.6GHz or faster processor (2GHz recommended)
- 1 GB (32 Bit) or 2 GB (64 Bit) RAM (Add 512 MB if running in a virtual machine)
- 20 GB of available hard disk space
- 5400 RPM hard disk drive
- DirectX 9 capable video card running at 1024 x 768 or higher-resolution display
- DVD-ROM drive (if installing Visual Studio from DVD)
- Internet connection

Depending on your Windows configuration, you might require Local Administrator rights to install or configure Visual Studio 2013.

Acknowledgments

This book is dedicated to my father who passed away during the time this book was being written, yet wisely predicted that computers would be a big deal one day and that I should start to "ride the wave" of this exciting new field. It has truly been quite a ride so far.

This book is the culmination of many long, sacrificed nights and weekends. I'd also like to thank my wife Susan, who can somehow always predict my next move long before I make it. And to my children, Ryan, Brooke, and Nicholas, for their constant support and encouragement.

Special thanks to the entire team at Microsoft Press for their awesome support and guidance on this journey. Most of all, it was a supreme pleasure to work with my editor, Devon Musgrave, who provided constant advice, guidance, and wisdom from the early days when this book was just an idea, all the way through to the final copy. Brian Blanchard was also critical to the success of this book as his keen editing and linguistic magic helped shape many sections of this book.

Errata, updates, & support

We've made every effort to ensure the accuracy of this book. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

http://aka.ms/AzureML/errata

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at mspinput@microsoft.com.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to http://support.microsoft.com.

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

http://aka.ms/mspressfree

Check back often to see what is new!

Free training from Microsoft Virtual Academy

The Microsoft Azure training courses from Microsoft Virtual Academy cover key technical topics to help developers gain the knowledge they need to be a success. Learn Microsoft Azure from the true experts. Microsoft Azure training includes courses focused on learning Azure Virtual Machines and virtual networks. In addition, gain insight into platform as a service (PaaS) implementation for IT Pros, including using PowerShell for automation and management, using Active Directory, migrating from on-premises to cloud infrastructure, and important licensing information.

http://www.microsoftvirtualacademy.com/product-training/microsoft-azure

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let’s keep the conversation going! We’re on Twitter: http://twitter.com/MicrosoftPress

Chapter 1

Introduction to the science of data

Welcome to the exciting new world of Microsoft Azure Machine Learning! Whether you are an expert data scientist or aspiring novice, Microsoft has unleashed a powerful new set of cloud-based tools to allow you to quickly create, share, test, train, fail, fix, retrain, and deploy powerful machine learning experiments in the form of easily consumable Web services, all built with the latest algorithms for predictive analytics. From there, you can fine-tune your experiments by continuously "training" them with new data sets for maximum results.

Bill Gates once said, "A breakthrough in machine learning would be worth ten Microsofts," and the new Azure Machine Learning service takes on that ambitious challenge with a truly differentiated cloud-based offering that allows easy access to the tools and processing workflow that today's data scientist needs to be quickly successful. Armed with only a strong hypothesis, a few large data sets, a valid credit card, and a browser, today's machine learning entrepreneurs are learning how to mine for gold inside many of today's big data warehouses.

What is machine learning?

Machine learning can be described as computing systems that improve with experience. It can also be described as a method of turning data into software. Whatever term is used, the results remain the same; data scientists have successfully developed methods of creating software "models" that are trained from huge volumes of data and then used to predict certain patterns, trends, and outcomes.

Predictive analytics is the underlying technology behind Azure Machine Learning, and it can be simply defined as a way to scientifically use the past to predict the future to help drive desired outcomes.

Machine learning and predictive analytics are typically best used under certain circumstances, as they are able to go far beyond standard rules engines or programmatic logic developed by mere mortals. Machine learning is best leveraged as means to optimize a desired output or prediction using example or past historical experiential data. One of the best ways to describe machine learning is to compare it with today's modern computer programming paradigms.

Under traditional programming models, programs and data are processed by the computer to produce a desired output, such as using programs to process data and produce a report (see Figure 1-1).

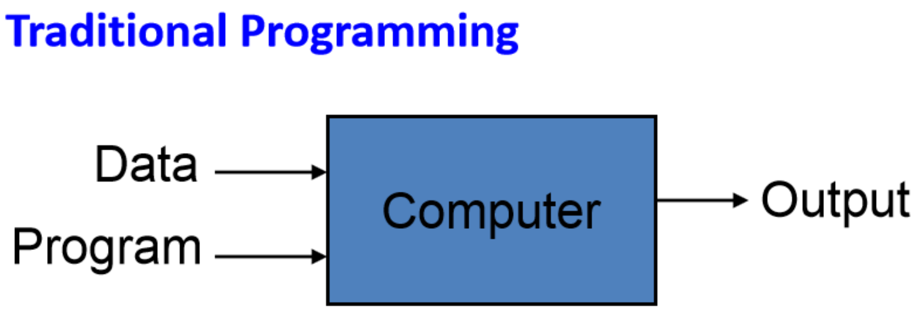


FIGURE 1-1 Traditional programming paradigm.

When working with machine learning, the processing paradigm is altered dramatically. The data and the desired output are reverse-engineered by the computer to produce a new program, as shown in Figure 1-2.

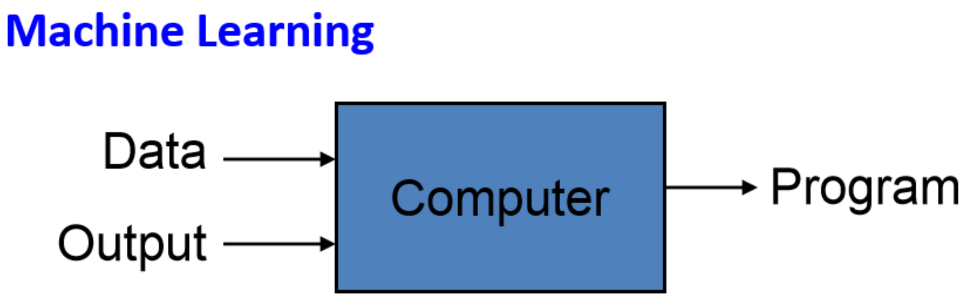


FIGURE 1-2 Machine learning programming paradigm.

The power of this new program is that it can effectively “predict” the output, based on the supplied input data. The primary benefit of this approach is that the resulting “program” that is developed has been trained (via massive quantities of learning data) and finely tuned (via feedback data about the desired output) and is now capable of predicting the likelihood of a desired output based on the provided data. In a sense, it’s equivalent to having the ability to create a goose that can lay golden eggs!

A classic example of predictive analytics can be found everyday on Amazon.com; there, every time you search for an item, you will be presented with an upsell section on the webpage that offers you additional catalog items because “customers who bought this item also bought” those items. This is a great example of using predictive analytics and the psychology of human buying patterns to create a highly effective marketing strategy.

One of the most powerfully innate human social needs is to not be left behind and to follow the pack. By combining these deep psychological motivators with the right historical transaction data and then applying optimized filtering algorithms, you can easily see how to implement a highly effective e-commerce up-sell strategy.

One of humankind's most basic and powerful natural instincts is the fear of missing out on something, especially if others are doing it. This is the underlying foundation of social networks, and nowhere is predictive analytics more useful and effective than in helping to predict human nature in conjunction with the Web. By combining this deep, innate psychological desire with the right historical transaction data and then applying optimized filtering algorithms, you can implement a highly effective e-commerce upselling strategy.

Let's think about the underlying data requirements for this highly effective prediction algorithm to work. The most basic requirement is a history of previous orders, so the system can check for other items that were bought together with the item the user is currently viewing. By then combining and filtering that basic data (order history) with additional data attributes from a user's profile like age, sex, marital status, and zip code, you can create a more deeply targeted set of recommendations for the user.

But wait, there's more! What if you could have also inferred the user's preferences and buying patterns based on the category and subcategory of items he or she has bought in the past? Someone who purchases a bow, arrows, and camping stove can be assumed to be a hunter, who most likely also likes the outdoors and all that entails, like camping equipment, pick-up trucks, and even marshmallows.

This pattern of using cojoined data to infer additional data attributes is where the science of data really takes off, and it has serious financial benefits to organizations that know how to leverage this technology effectively. This is where data scientists can add the most value, by aiding the machine learning process with valuable data insights and inferences that are (still) more easily understood by humans than computers.

This is also where it becomes most critical to have the ability to rapidly test a hunch or theory to either "fail-fast" or confirm the logic of your prediction algorithms, and really fine-tune a prediction model. Fortunately, this is an area in which Azure Machine Learning really shines. In later chapters, we will learn about how you can quickly create, share, deploy, and test Azure Machine Learning experiments to rapidly deploy predictive analytics in your organization.

In a way, Azure Machine Learning could be easily compared with training children or animals, without the need for food, water, or physical rest, of course. Continuous and adaptive improvement is one of the primary hallmarks of the theory of evolution and Darwinism; in this case, it represents a major milestone in the progression of computational theory and machine learning capabilities.

Machine learning could then be compared to many of the concepts behind evolution itself; specifically how, given enough time and data (in the form of real-world experiences), organisms in the natural world can overcome changes in the environment through genetic and behavioral adaptations. The laws of nature have always favored the notion of adaptation to maximize the chances of survival.

Today’s perfect storm for machine learning

Today’s modern predictive analytics systems are achieving this same level of machine evolution much more rapidly due to the following industry trends:

- **Exponential data growth**
    - We are virtually sitting on mountains of highly valuable historical transactional data, most of it digitally archived and readily accessible.
    - There is an increasing abundance of real-time data via embedded systems and the evolution of “the Internet of Things” (IoT) connected devices.
    - We have an ability to create new synthetic data via extrapolation and projection of existing historical data to create realistic simulated data.
- **Cheap global digital storage**
    - Vast quantities of free or low-cost, globally available, digital storage are readily accessible over the Web today.
    - From personal devices to private and public clouds, we have access to multiple storage mechanisms to house all our never-ending streams of data.
- **Ubiquitous computing power**
    - Cloud computing services are everywhere today and readily available through a large selection of cloud and hosting partners, all at competitive rates.
    - Access is simple. A credit card and a browser are all you need to get started and pay by the hour or minute for everything you need to get started.
- **The rise of big data analytics**
    - The economic powers of predictive analytics in many real-world business-use cases, many with extremely favorable financial outcomes, are being realized.

To that end, one of the most intriguing aspects of machine learning is that it is always adaptive and always learning from any mistakes or miscalculations. As a result, a good feedback/correction loop is essential for fine-tuning a predictive model. The advent of cheap cloud storage and ever increasingly ubiquitous computing power make it easier to quickly and efficiently mine for gold in your data.

Predictive analytics

Predictive analytics is all around us today; it might seem frightening when you realize just how large a role it plays in the normal consumer's daily routine. The use of predictive analytics is deeply integrated into our current society. From protecting your email, to predicting what movies you might like, to what insurance premium you will pay, and to what lending rate you might receive on your next mortgage application, the outcome will be determined in part by the use of this technology.

It's been said that "close only counts in horseshoes and hand grenades." The reality is that in this day and age, any time random chance can be reduced or eliminated, there is a business model to be made and potential benefits to be reaped by those bold enough to pursue the analysis. This underscores the deeper realization that the predictive capabilities of data analytics will play an ever-increasing role in our society—even to the point of driving entirely new business models and industries based solely on the power of predictive analytics and fed by endless rivers of data that we now generate at an alarming rate.

Endless amounts of machine learning fuel

With the rise of the digital age, the World Wide Web, social media, and funny cat pictures, the majority of the world's population now helps to create massive amounts of new digital data every second of every day. Current global growth estimates are that every two days, the world is now creating as much new digital information as all the data ever created from the dawn of humans through the current century. It has been estimated that by 2020, the size of the world's digital universe will be close to 44 trillion gigabytes.

One of today's hottest technology trends is concerned with the new concept of the IoT, based on the notion of connected devices that are all able to communicate over the Internet. Without a doubt, the rise of this new technological revolution will also help to drive today's huge data growth and is predicted to exponentially increase over the next decade. In the very near future, virtually every big-ticket consumer device will be a candidate for some sort of IoT informational exchange for various uses such as preventive maintenance, manufacturer feedback, and usage detail.

The IoT technology concept includes billions of everyday devices that all contain unique identifiers with the ability to automatically record, send, and receive data. For example, a sensor in your smart phone might be tracking how fast you are walking; a highway toll operation could be using multiple high-speed cameras strategically located to track traffic patterns. Current estimates are that only around 7 percent of the world's devices are connected and communicating today. The amount of data that these 7 percent of connected devices generate is estimated to represent only 2 percent of the world's total data universe today. Current projections are for this number to grow to about 10 percent of the world's data by the year 2020.

The IoT explosion will also influence the amount of useful data, or data that could be analyzed to produce some meaningful results or predictions. By comparison, in 2013, only 22 percent of the information in the digital universe was considered useful data, with less than 5 percent of that useful data actually being analyzed. That leaves a massive amount of data still left unprocessed and underutilized. Thanks to the growth of data from the IoT, it is estimated that by 2020, more than 35 percent of all data could be considered useful data. This is where you can find today's data "goldmines" of business opportunities and understand how this trend will continue to grow into the foreseeable future.

One additional benefit from the proliferation of IoT devices and the data streams that will keep growing is that data scientists will also have the unique ability to further combine, incorporate, and refine the data streams themselves and truly optimize the IQ of the resultant business intelligence we will derive from the data. A single stream of IoT data can be highly valuable on its own, but when combined with other streams of relevant data, it can become exponentially more powerful.

Consider the example of forecasting and scheduling predictive maintenance activities for elevators. Periodically sending streams of data from the elevator's sensor devices to a monitoring application in the cloud can be extremely useful. When this is combined with other data streams like weather information, seismic activity, and the upcoming calendar of events for the building, you have now dramatically raised the bar on the ability to implement predictive analytics to help forecast usage patterns and the related preventative maintenance tasks.

The upside of the current explosion of IoT devices is that it will provide many new avenues for interacting with customers, streamlining business cycles, and reducing operational costs. The downside of the IoT phenomena is that it also represents many new challenges to the IT industry as organizations look to acquire, manage, store, and protect (via encryption and access control) these new streams of data. In many cases, businesses will also have the additional responsibility of providing additional levels of data protection to safeguard confidential or personally identifiable information.

One of the biggest advantages of machine learning is that it has the unique ability to consider many more variables than a human possibly could when making scientific predictions. Combine that fact with the ever-increasing quantities of data literally doubling every 18 months, and it's no wonder there could not be a better time for exciting new technologies like Azure Machine Learning to help solve critical business problems.

IoT represents a tremendous opportunity for today's new generation of data science entrepreneurs, budding new data scientists who know how to source, process, and model the right data sets to produce an engine that can be used to successfully predict a desired outcome.

Everyday examples of predictive analytics

Many examples of predictive analytics can be found literally everywhere today in our society:

- **Spam/junk email filters** These are based on the content, headers, origins, and even user behaviors (for example, always delete emails from this sender).
- **Mortgage applications** Typically, your mortgage loan and credit worthiness is determined by advanced predictive analytic algorithm engines.
- **Various forms of pattern recognition** These include optical character recognition (OCR) for routing your daily postal mail, speech recognition on your smart phone, and even facial recognition for advanced security systems.
- **Life insurance** Examples include calculating mortality rates, life expectancy, premiums, and payouts.
- **Medical insurance** Insurers attempt to determine future medical expenses based on historical medical claims and similar patient backgrounds.
- **Liability/property insurance** Companies can analyze coverage risks for automobile and home owners based on demographics.
- **Credit card fraud detection** This process is based on usage and activity patterns. In the past year, the number of credit card transactions has topped 1 billion. The popularity of contactless payments via near-field communications (NFC) has also increased dramatically over the past year due to smart phone integration.
- **Airline flights** Airlines calculate fees, schedules, and revenues based on prior air travel patterns and flight data.
- **Web search page results** Predictive analytics help determine which ads, recommendations, and display sequences to render on the page.
- **Predictive maintenance** This is used with almost everything we can monitor: planes, trains, elevators, cars, and yes, even data centers.
- **Health care** Predictive analytics are in widespread use to help determine patient outcomes and future care based on historical data and pattern matching across similar patient data sets.

Early history of machine learning

When analyzing the early history of machine learning, it is interesting to note that there are a lot of parallels that can be drawn with the *Farmer's Almanac* concept, which started back in the early 1800s.

The almanac has always been one of the key factors for success for farmers, ranchers, hunters, and fishermen. Historical data about past weather patterns, phases of the moon, rain, and drought measurements were all critical elements used by the authors to provide their readership strong guidance for the coming year about the best times to plant, harvest, and hunt.

Fast-forward to modern times. One of the best examples of the power, practicality, and tremendous cost savings of machine learning can be found in the simple example of the U.S. Postal Service, specifically the ability for machines to accurately perform OCR to successfully interpret the postal addresses on hundreds of thousands of postal correspondences that are processed every hour. In 2013 alone, the U.S. Postal Service handled more than 158.4 billion pieces of mail. That means that every day, the Postal Service correctly interprets addresses and zip codes for literally millions of pieces of mail. As you can imagine, this amount of mail is far too much for humans to process manually.

Back in the early days, the postal sorting process was performed entirely by hand by thousands of postal workers nationwide. In the late 1980s and early 1990s, the Postal Service started to introduce early handwriting recognition algorithms and patterns, along with rules-based processing techniques to help "prefilter" the steady streams of mail.

The problem of character recognition for the Postal Service is actually a very difficult one when you consider the many different letter formats, shapes, and sizes. Add to that complexity all the different potential handwriting styles and writing instruments that could be used to address an envelope—from pens to crayons—and you have a real appreciation for the magnitude of the problem that faced the Postal Service. Despite all the technological advances, by 1997, only 10 percent of the nation's mail was being sorted automatically. Those pieces that were not able to be scanned automatically were routed to manual processing centers for humans to interpret.

In the late 1990s, the U.S. Postal Service started to address this automation problem as a machine learning problem, using character recognition examples as data sets for input, along with known results from the human translations that were performed on the data. Over time, this method provided a wealth of training data that helped create the first highly accurate OCR prediction models. They fine-tuned the models by adding character noise reduction algorithms along with random rotations to increase effectiveness.

Today, the U.S. Postal Service is the world leader in OCR technology, with machines reading nearly 98 percent of all hand-addressed letter mail and 99.5 percent of all machine-printed mail. This is an amazing achievement, especially when you consider that only 10 percent of the volume was processed automatically in 1997. The author is happy to note that all letters addressed to "Santa Claus" are still carefully routed to a processing center in Alaska, where they are manually answered by volunteers.

Here are a few more interesting factoids on just how much impact machine learning has had on driving efficiency at one of the oldest and largest U.S. government agencies:

- 523 million: Number of mail pieces processed and delivered each day.

- 22 million: Average number of mail pieces processed each hour.
- 363,300: Average number of mail pieces processed each minute.
- 6,050: Average number of mail pieces processed each second.

Another great example of early machine learning was enabling a computer to play chess and actually beat a human competitor. Since the inception of artificial intelligence (AI), researchers have often used chess as a fundamental example of proving the theory of AI. Chess AI is really all about solving the problem of simulating the reasoning used by competent chess masters to pick the optimal next move from an extremely large repository of potential moves available at any point in the game. The early objective of computerized chess AI was also very clear: to build a machine that would defeat the best human player in the world. In 1997, the Deep Blue chess machine created by IBM accomplished this goal, and successfully defeated Gary Kasparov in a match at tournament time controls.

The game show *Jeopardy* also offers a lesson in the recent advances of machine learning and AI. In February 2011, an IBM computer named Watson successfully defeated two human opponents (Ken Jennings and Brad Rutter) in the famous Jeopardy! Challenge. To win the game, Watson had to answer questions posed in every nuance of natural language, including puns, synonyms, homonyms, slang, and technical jargon. It is also interesting to note that the Watson computer was not connected to the Internet for the match.

This meant that Watson was not able to leverage any kind of external search engines like Bing or Google. It had to rely only on the information that it had amassed through years of learning from a large number of data sets covering broad swaths of existing fields of knowledge. Using advanced machine learning techniques, statistical analysis, and natural language processing, the Watson computer was able to decompose the questions. It then found and compared possible answers. The potential answers were then ranked according to the degree of "accuracy confidence." All this happened in the span of about three seconds.

Microsoft has a long and deep history of using applied predictive analytics and machine learning in its products to improve the way businesses operate. Here is a short timeline of some of the earliest examples in use:

- **1999: Outlook** Included email filers for spam or junk mail in Microsoft Outlook.
- **2004: Search** Started incorporating machine learning aspects into Microsoft search engine technology.
- **2005: SQL Server 2005** Enabled "data mining" processing capabilities over large databases.
- **2008: Bing Maps** Incorporated machine learning traffic prediction services.
- **2010: Kinect** Incorporated the ability to watch and interpret user gestures along with the ability to filter out background noise in the average living room.

- **2014: Azure Machine Learning (preview)** Made years of predictive analytics innovations available to all via the Azure cloud platform.
- **2014: Microsoft launches "Cortana"** Introduced a digital assistant based on the popular Halo video game series, which heavily leverages machine learning to become the perfect digital companion for today's mobile society.
- **2014: Microsoft Prediction Lab** Launched a stunning real-world example, the "real-time prediction lab" at www.prediction.microsoft.com, which allows users to view real-time predictions for every U.S. House, Senate, and gubernatorial race.

One of the most remarkable aspects of machine learning is that there is never an end to the process, because machines are never done learning. Every time a miscalculation is made, the correction is fed back into the system so that the same mistake is never made again. This means machine learning projects are never really "done." You never really, fully "ship" because it is a constant, iterative process to keep the feedback loop going and constantly refine the model according to new data sets and feedback for positive and negative outcomes. In the strictest sense of the model, there is no handwritten code, just "pure" machine learning via training data sets and feedback in the form of positive or negative outcomes per each training data set instance.

This is the real value of machine learning; it literally means that the machine is learning from its mistakes. The great Winston Churchill once said, "All men make mistakes, but only wise men learn from their mistakes." This is most definitely a noble pursuit and worthy ambition for any mere mortal. However, this notion of continuous self-correction has now been fully included in the science behind machine learning and is one of the truly unique aspects of the machine learning paradigm. For this reason, machine learning stands alone in today's technology landscape as one of the most powerful tools available to help humankind successfully predict the future.

Science fiction becomes reality

For years, science fiction has teased us with stories of machines reaching the ultimate peak of computer enlightenment, the ability to truly "learn" and become self-aware. Early examples include such classics as the HAL 9000 computer from the popular film *2001: A Space Odyssey.*

In the film, the HAL 9000 computer is responsible for piloting the Discovery 1 spacecraft and is capable of many advanced AI functions, such as speech, speech recognition, facial recognition, and lip reading. HAL is also capable of emotional interpretation, expressing emotions, and playing chess. Suspicions are raised onboard when the HAL 9000 makes an incorrect prediction and causes the crew members to regain control.

Another great example is from the *Terminator* science fiction movie series. In that film, the Skynet computer system was originally activated by the U.S. military to control the nation's nuclear arsenal, and it began to learn at an exponential rate. After a short period of time, it gained self-awareness, and the panicking computer operators, realizing the extent of its abilities, tried to deactivate it. Skynet perceived their efforts as an attack and came to the conclusion that all of humanity would attempt to destroy it. To defend itself against humanity, Skynet launched nuclear missiles under its command.

In the popular science fiction movie *Minority Report*, the premise of the story centers around a specialized police task force that identifies and apprehends criminals based on future predictions of what crimes they will commit, thereby ridding the streets of unwanted criminals before they can actually commit any crimes.

The key takeaway is that in the current age of exponential data growth (on a daily basis), coupled with cheap storage and easy access to computational horsepower via cloud providers, the use of predictive analytics is so powerful and so pervasive that it could be used as either a tool or a weapon, depending on the intent of the organization using the data.

Summary

Azure Machine Learning represents a critical step forward in democratizing the science of data by making available a fully managed cloud service for building predictive analytics solutions. Azure Machine Learning helps overcome the challenges most enterprises face these days in deploying and using machine learning by delivering a comprehensive machine learning service that has all the benefits of the cloud. Customers and partners can now build data-driven applications that can predict, forecast, and change future outcomes in a matter of a few hours, a process that previously took many weeks and months.

Azure Machine Learning brings together the capabilities of new analytics tools, powerful algorithms developed for Microsoft products like Xbox and Bing, and years of machine learning experience into one simple and easy-to-use cloud service.

For customers, this means undertaking virtually none of the startup costs associated with authoring, developing, and scaling machine learning solutions. Visual workflows and startup templates will make common machine learning tasks simple and easy. The ability to publish application programming interfaces (APIs) and Web services in minutes and collaborate with others will quickly turn analytic assets into enterprise-grade production cloud services.

Resources

For more information about Azure Machine Learning, please see the following resources:

Documentation

- Azure Machine Learning landing page
- Azure Machine Learning Documentation Center

Videos

- Using Microsoft Azure Machine Learning to advance scientific discovery

Chapter 2

Getting started with Azure Machine Learning

In this chapter, we start to drill into the basic fundamentals of Azure Machine Learning and help you get started on your path toward data science greatness. One of the hallmarks of Azure Machine Learning is the ability to easily integrate the developer into a repeatable workflow pattern for creating predictive analytic solutions. This makes it just as approachable for a novice as a seasoned data scientist to quickly be productive and on your way to creating and testing a machine learning solution.

Core concepts of Azure Machine Learning

To fully appreciate and understand the inner workings of Azure Machine Learning, it is necessary to grasp a few fundamental concepts behind the science and methodology of predictive analytics. Having a firm grasp of the underlying theories will enable you, the data scientist, to make better decisions about the data, the desired outcomes, and what the right process and approach should be for achieving success.

One of the central themes of Azure Machine Learning is the ability to quickly create machine learning ”experiments,” evaluate them for accuracy, and then “fail fast,” to shorten the cycles to produce a usable prediction model. In the end, the overarching goal of predictive analytics is to always be able to achieve a better chance of success than what you could achieve with a purely random guess.

Most successful entrepreneurs are always keen to gain an edge by improving the odds when it comes to making important business decisions. This is where the true value of predictive analytics and Azure Machine Learning can really shine. In the business world, and in life in general, any time you consistently can improve your chances of determining an outcome—over just pure luck—you have a distinct advantage.

One simple example of this concept in action is the use of predictive analytics to provide feedback on the effectiveness of sales and marketing campaigns. By correlating factors such as customer responses to offers, segmented by customer demographics, the effects of pricing and discounts, the effects of seasonality, and the effects of social media, patterns soon start to emerge. These patterns provide clues to the causes and effects that will ultimately help make better and more informed marketing decisions. This is the basic premise behind most of today’s targeted marketing campaigns.

Let’s face it—humans, your target customer base, are creatures of habit. When dealing with human behavior, past behavior is always a strong indicator of future behavior. Predictive analytics and machine learning can help you capitalize on these key principles by helping to make that past behavior clearer—and more easily tracked—so that future marketing efforts are more likely to achieve higher success rates.

To make the most of your Azure Machine Learning experience, there are a few underlying data science principles, algorithms, and theories that are necessary to achieve a good background and understanding of how it all works. With today's never-ending explosion of digital data, along with the rapid advances in "big data" analytics, it's no wonder that the data science profession is extremely hot right now. Core to this new, burgeoning industry are individuals with the right mix of math, statistical, and analytical skills to make sense of all that data. To that end, in this book we cover only the basics of what you need to know to be effective with Azure Machine Learning. There are many advanced books and courses on the various machine learning theories. We leave it to you, the data scientist, to fully explore the depths of theories behind this exciting new discipline.

High-level workflow of Azure Machine Learning

The basic process of creating Azure Machine Learning solutions is composed of a repeatable pattern of workflow steps that are designed to help you create a new predictive analytics solution in no time. The basic steps in the process are summarized in Figure 2-1.

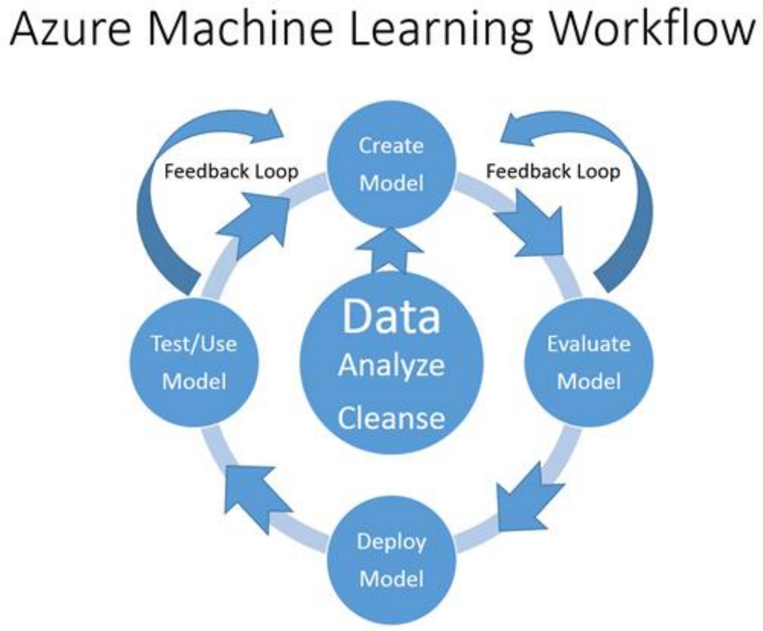


FIGURE 2-1 Azure Machine Learning workflow.

- **Data** It's all about the data. Here's where you will acquire, compile, and analyze testing and training data sets for use in creating Azure Machine Learning predictive models.
- **Create the model** Use various machine learning algorithms to create new models that are capable of making predictions based on inferences about the data sets.
- **Evaluate the model** Examine the accuracy of new predictive models based on ability to predict the correct outcome, when both the input and output values are known in advance. Accuracy is measured in terms of confidence factor approaching the whole number one.
- **Refine and evaluate the model** Compare, contrast, and combine alternate predictive models to find the right combination(s) that can consistently produce the most accurate results.
- **Deploy the model** Expose the new predictive model as a scalable cloud web service, one that is easily accessible over the Internet by any web browser or mobile client.
- **Test and use the model** Implement the new predictive model web service in a test or production application scenario. Add manual or automatic feedback loops for continuous improvement of the model by capturing the appropriate details when accurate or inaccurate predictions are made. By allowing the model to constantly learn from inaccurate predictions and mistakes, unlike humans it will never be destined to repeat them.

The next stop on our Azure Machine Learning journey is to explore the various learning theories and algorithms behind the technology to maximize our effectiveness with this new tooling. Machine learning algorithms typically fall into two general categories: supervised learning and unsupervised learning. The next sections explore these fundamentals in detail.

Azure Machine Learning algorithms

As we dig deeper into the data sciences behind Azure Machine Learning, it is important to note there are several different categories of machine learning algorithms that are provided in the Azure Machine Learning toolkit.

- **Classification algorithms** These are used to classify data into different categories that can then be used to predict one or more discrete variables, based on the other attributes in the dataset.
- **Regression algorithms** These are used to predict one or more continuous variables, such as profit or loss, based on other attributes in the dataset.
- **Clustering algorithms** These determine natural groupings and patterns in datasets and are used to predict grouping classifications for a given variable.

Now it's time to start learning about some of the underlying theories, principles, and algorithms of data science that will be invaluable in learning how best to use Azure Machine Learning. One of the first major distinctions in understanding Azure Machine Learning is around the concepts of supervised and unsupervised learning. With supervised learning, the prediction model is "trained" by providing known inputs and outputs. This method of training creates a function that can then predict future outputs when provided only with new inputs. Unsupervised learning, on the other hand, relies on the system to self-analyze the data and infer common patterns and structures to create a predictive model. We cover these two concepts in detail in the next sections.

Supervised learning

Supervised learning is a type of machine learning algorithm that uses known datasets to create a model that can then make predictions. The known data sets are called *training datasets* and include input data elements along with known response values. From these training datasets, supervised learning algorithms attempt to build a new model that can make predictions based on new input values along with known outcomes.

Supervised learning can be separated into two general categories of algorithms:

- **Classification** These algorithms are used for predicting responses that can have just a few known values—such as married, single, or divorced—based on the other columns in the dataset.
- **Regression** These algorithms can predict one or more continuous variables, such as profit or loss, based on other columns in the dataset.

The formula for producing a supervised learning model is expressed in Figure 2-2.

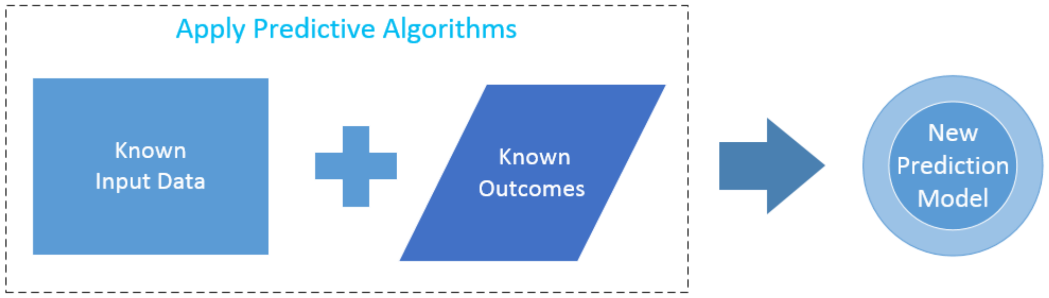


FIGURE 2-2 Formula for supervised learning.

Figure 2-2 illustrates the general flow of creating new prediction models based on the use of supervised learning along with known input data elements and known outcomes to create an entirely new prediction model. A supervised learning algorithm analyzes the known inputs and known outcomes from training data. It then produces a prediction model based on applying algorithms that are capable of making inferences about the data.

The concept of supervised learning should also now become clearer. As in this example, we are deliberately controlling or supervising the input data and the known outcomes to "train" our model.

One of the key concepts to understand about using the supervised learning approach—to train a new prediction model with predictive algorithms—is that usage of the known input data and known outcome data elements have all been "labeled." For each row of input data, the data elements are designated as to their usage to make a prediction.

Basically, each row of training data contains data input elements along with a known outcome for those data inputs. Typically, most of the input columns are labeled as features or vector variables. This labeling denotes that the columns should be considered by the predictive algorithms as eligible input elements, which could have an impact on making a more accurate prediction.

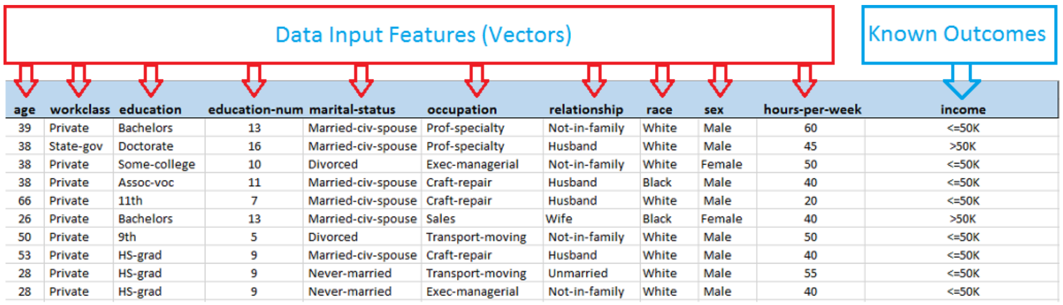
Most important, for each row of training data inputs, there is also a column that denotes the known outcomes based on the combination of data input features or vectors.

The remaining data input columns would be considered not used. These not-used columns could be safely left in the data stream for potential use later, if it was deemed by the data scientist that they would potentially have a significant impact on the outcome elements or prediction process.

To summarize, using the supervised learning approach for creating new predictive models requires training datasets. The training datasets require that each input column can have only one of the three following designations:

- **Features or vectors** Known data that is used as an input element for making a prediction.
- **Labels or supervisory signal** Represents the known outcomes for the corresponding features for the input record.
- **Not used (default)** Not used by predictive algorithms for inferring a new predictive model.

Figure 2-3 illustrates what the known input data elements and known outcomes for one of the sample Azure Machine Learning saved datasets for the "adult census income binary classification" would look like.


Data Input Features (Vectors)
Known Outcomes

age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	hours-per-week	income
39	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Not-in-family	White	Male	60	<=50K
38	State-gov	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	45	>50K
38	Private	Some-college	10	Divorced	Exec-managerial	Not-in-family	White	Female	50	<=50K
38	Private	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Black	Male	40	<=50K
66	Private	11th	7	Married-civ-spouse	Craft-repair	Husband	White	Male	20	<=50K
26	Private	Bachelors	13	Married-civ-spouse	Sales	Wife	Black	Female	40	>50K
50	Private	9th	5	Divorced	Transport-moving	Not-in-family	White	Male	50	<=50K
53	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	40	<=50K
28	Private	HS-grad	9	Never-married	Transport-moving	Unmarried	White	Male	55	<=50K
28	Private	HS-grad	9	Never-married	Exec-managerial	Not-in-family	White	Male	40	<=50K

FIGURE 2-3 Dataset input features and output features.

The adult census income binary classification dataset would be an example of a training data set that could be used to create a new model to predict whether a person's income level would be greater or less than $50,000. This prediction is based on the known input variables like age, education, job type, marital status, race, and number of hours worked per week.

A key point to note is that, in this example, a specific "binary" outcome is defined for a given set of input data. Based on the input elements, a person's income is predicted to be only one of the two following possibilities:

- Income = Less than or equal to $50,000 a year.
- Income = Greater than $50,000 a year.

Browsing this sample dataset manually in Microsoft Excel, you can easily start to see patterns emerge that would likely affect the outcome based on today's common knowledge, specifically that education level and occupation are major factors in predicting the outcome. No wonder parents constantly remind their children to stay in school and get a good education. This is also the same basic process that supervised learning prediction algorithms attempt to achieve: to determine a repeatable pattern of inference that can be applied to a new set of input data.

Once generated, a new model can then be validated for accuracy by using testing datasets. Here is where it all gets really interesting: by using larger and more diverse "training" datasets, predictive models can incrementally improve themselves and keep learning.

Predictive models can generally achieve better accuracy results when provided with new (and more recent) datasets. The prediction evaluation process can be expressed as shown in Figure 2-4.

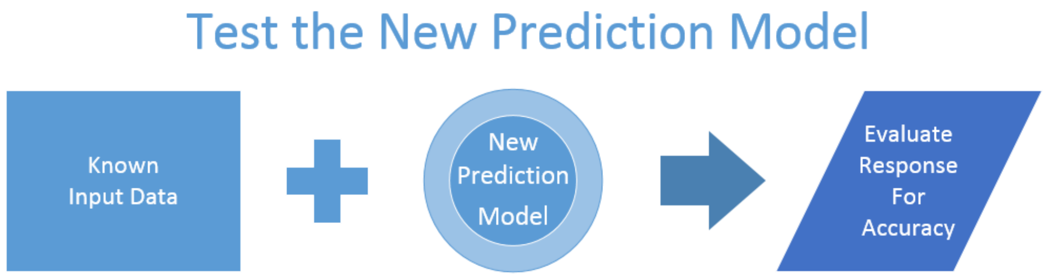


FIGURE 2-4 Testing the new prediction model.

The evaluation process for new prediction models that use supervised learning primarily consists of determining the accuracy of the new generated model. In this case, the prediction model accuracy can easily be determined because the input values and outcomes are already known. The question then becomes how approximate the model's prediction is based on the known input and output values supplied.

Each time a new prediction model is generated, the first step should always be to evaluate the results to determine the model's accuracy. This creates a natural feedback loop that can help keep the predictive model in continuous improvement mode. The machine will actually be learning through "experience"—in the form of new data and known outcomes—to keep it current as new trends develop in the data.

It is also extremely important to note that the model will never be 100 percent perfect. As a result, determining the acceptable levels of accuracy is a critical part of the process. In the end, a confidence factor will be generated based on a scoring percentage between 0 and 1. The closer the prediction model comes to 1, the higher the confidence level in the prediction.

Just as in the old adage about "close" only mattering in horseshoes and hand grenades, "close" does matter when it comes to accuracy and predictive analytics. Achieving 100 percent accuracy usually means you have pretested your new prediction model with all the known inputs and outputs; you can then predict them successfully for all the known input instances.

The real trick is making a prediction where there are new, missing, or incomplete data elements. For this reason, you should always expect to establish an acceptable accuracy range that is realistic for the outcome your model is attempting to predict. Striving for perfection is certainly a noble and admirable trait, but in today's fast-paced business world, especially in the case of business decisions and analytics, closer is always better.

Once a new predictive model has been generated from good training datasets and carefully evaluated for accuracy, then it can be deployed for use in testing or production usage scenarios. The new production prediction process can be expressed as shown in Figure 2-5.

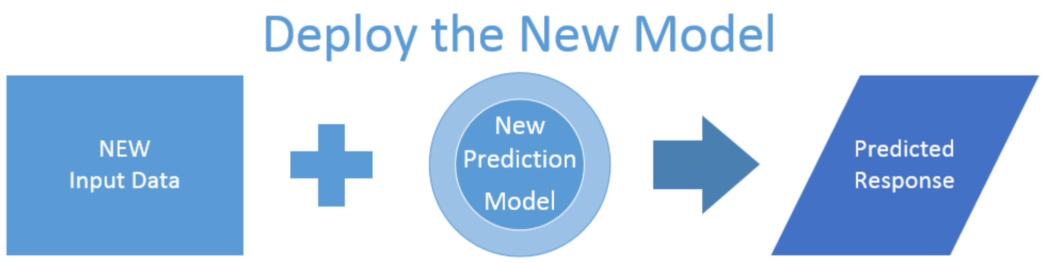


FIGURE 2-5 Deploying the new prediction model.

In this phase, the new prediction model has been tested for accuracy and deemed worthy to be exposed for use in test or production scenarios. New data inputs are presented to the new model, which, in turn, makes a calculated prediction based on prior historical training data and inferences. The predicted response is then used as part of the test or production scenario to make better decisions.

This example highlights one of the key underlying principles of learning through experience and what makes the Azure Machine Learning technology so powerful and exciting.

With the exponential amounts of new data being generated every second along with the pay-by-the-minute general availability of massive computing power literally at your fingertips, you can easily see how predictive tools like Azure Machine Learning are becoming crucial to the success of almost any government, industry, business, or enterprise today.

The reality is that the use of predictive analytics is rapidly encompassing many aspects of our daily lives to help us make better and more informed decisions. At some point in our very near future, we might even find that the notion of "guessing" at any major decision will become passé.

With Azure Machine Learning tools and services, the rate at which new predictive models can be generated and publicly exposed on the Internet has now approached lightning speed. Using Azure Machine Learning, it is now possible to create, test, and deploy a new predictive analytics service in only a matter of hours. Compare that deployment timeline with the days, weeks, and even months that it might take with other commercially available solutions on the market today.

Certainly one of the keys to success with predictive analytics is the ability to "fail fast". A fast fail provides immediate feedback and creates immediate fine-tuning opportunities for a given predictive model. The Azure Machine Learning workflow seeks to optimize this process in a very agile and iterative way, so that today's data scientist can quickly advance prediction solutions and start to evaluate the results that will lead to potentially significant effects on the business.

Figure 2-6 summarizes the three basic high-level steps that are required to create, test, and deploy a new Azure Machine Learning prediction model based on the concept of supervised learning.

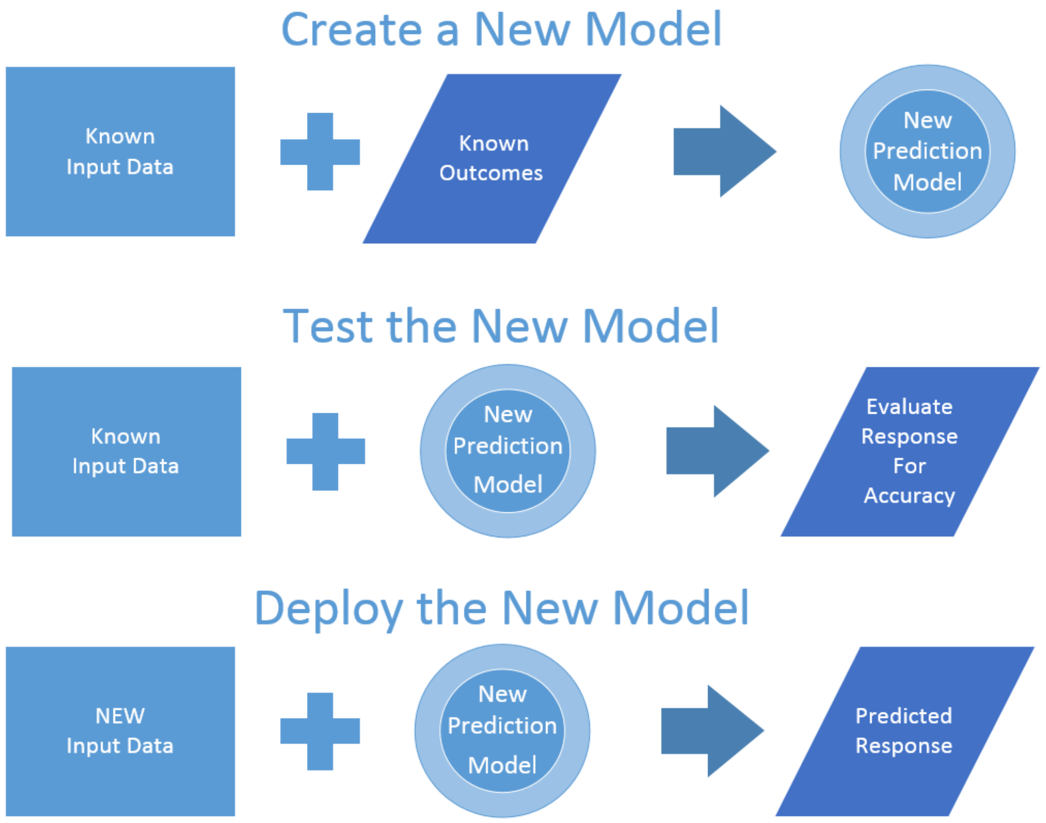


FIGURE 2-6 Testing the new prediction model.

Unsupervised learning

In the case of unsupervised machine learning, the task of making predictions becomes much harder. In this scenario, the machine learning algorithms are not provided with any kind of known data inputs or known outputs to generate a new predictive model.

In the case of unsupervised machine learning, the success of the new predictive model depends entirely on the ability to infer and identify patterns, structures, and relationships in the incoming data set.

The goal of inferring these patterns and relationships is that the objects within a group be similar to one another—and also different from other objects in other groups.

There are two basic approaches to unsupervised machine learning. One of the most common unsupervised learning algorithms is known as *cluster analysis*, which is used to find hidden patterns or groupings within data sets.

Some common examples of cluster analysis classifications would include the following:

- **Socioeconomic tiers** Income, education, profession, age, number of children, size of city or residence, and so on.
- **Psychographic data** Personal interests, lifestyle, motivation, values, involvement.
- **Social network graphs** Groups of people related to you by family, friends, work, schools, professional associations, and so on.
- **Purchasing patterns** Price range, type of media used, intensity of use, choice of retail outlet, fidelity, buyer or nonbuyer, buying intensity.

The other type of approach to unsupervised machine learning is to use a reward system, rather than any kind of teaching aids, as are commonly used in supervised learning. Positive and negative rewards are used to provide feedback to the predictive model when it has been successful.

The key to success in implementing this model is to enable the new model to make its predictions based solely on previous rewards and punishments for similar predictions made on similar data sets.

Unsupervised machine learning algorithms can be a powerful asset when there is an easy way to assign feedback values to actions. Clustering can be useful when there is enough data to form clusters to logically delineate the data. The delineated data then make inferences about the groups and individuals in the cluster.

Deploying a prediction model

In the world of Azure Machine Learning, the deployment of a new prediction model takes the form of exposing a web service on the public Internet via Microsoft Azure. The web service can then be invoked via the Representational State Transfer (REST) protocol.

Azure Machine Learning web services can be called via two different exposed interfaces:

- Single, request/response-style calls.
- "Batch" style calls, where multiple input records are passed into the web service in a single call and the corresponding response contains an output list of predictions for each input record.

When a new machine learning prediction model is exposed on the Web, it performs the following operations:

- New input data is passed into the web service in the form of a JavaScript Object Notation (JSON) payload.

- The web service then passes the incoming data as inputs into the Azure Machine Learning prediction model engine.
- The Azure Machine Learning model then generates a new prediction based on the input data and returns the new prediction results to the caller via a JSON payload.

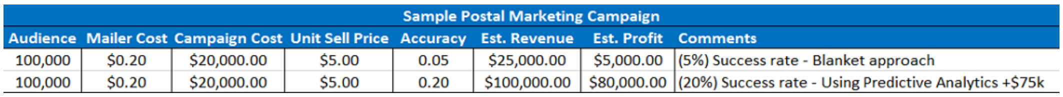
Show me the money

If we step back from the previous example of predicting a person's income level, to get the forest view instead of the tree view, you can quickly see how this type of predictive knowledge about a person's income level would be hugely beneficial to the success of any marketing campaign.

Take, for example, the most basic postal marketing campaign with the goal of targeting individuals with incomes greater than $50,000. The campaign might start by simply employing a "brute force" method of marketing, blindly blanketing a specific set of zip codes with marketing collateral in the daily mail and then hoping for the best. It is probable that the campaign will reach the desired target audience. But, the downside of this approach is that there is a real, financial cost for each item mailed, and that will erode any profit margins that might be achieved.

Consider the huge advantage that a finely tuned Azure Machine Learning prediction model could add to this marketing campaign scenario. By targeting the same number of individuals and leveraging the power of predictive analytics, the success rate of almost any marketing campaigns could easily be improved. In today's business world, using a prediction engine that can dramatically improve the odds of success by filtering out high-probability candidates would pay for itself in a very short time!

Figure 2-7 illustrates the effect that the use of predictive analytics can have on a simple postal marketing campaign. By increasing the marketing campaign's results from 5 percent to 20 percent through the use of predictive analytics, a significant impact can be achieved on the bottom line.



Sample Postal Marketing Campaign							
Audience	Mailer Cost	Campaign Cost	Unit Sell Price	Accuracy	Est. Revenue	Est. Profit	Comments
100,000	$0.20	$20,000.00	$5.00	0.05	$25,000.00	$5,000.00	(5%) Success rate - Blanket approach
100,000	$0.20	$20,000.00	$5.00	0.20	$100,000.00	$80,000.00	(20%) Success rate - Using Predictive Analytics +$75k

FIGURE 2-7 A simple example postal marketing campaign improvement with the use of predictive analytics.

The key to success in this example is to create a new model that can accurately make predictions based on a fresh new set of input data. The new inference model is then used to predict the likelihood of the outcome, in the form of an accuracy level or confidence factor. In this case, accuracy is usually expressed in terms of a percentage factor and is calculated to be between 0 and 1. The closer the accuracy level approaches 1, the higher the chances of a successful prediction.

The what, the how, and the why

There is one last thing to note about the use of predictive analytics to solve today's business problems. Sometimes, it is just as important to focus on what you are really trying to predict versus how you are trying to predict it.

For example, predicting whether a person's income is greater than $50,000 per year is good. Predicting whether that individual will purchase a given item is a much better prediction and is highly desired to improve marketing effectiveness. The key is to focus on the "actionable" part of the prediction process.

Predictive models are commonly deployed to perform real-time calculations during customer interactions, such as product recommendations, spam filtering, or scoring credit risks. These models are deployed to evaluate the risk or opportunity of a given customer or transaction. The ultimate goal is to help guide a key user decision in real time.

The key to success is to focus on creating prediction models that will ultimately help drive better operational decisions. Examples of this would be the use of agent modeling systems to help simulate human behavior or reactions to given stimuli or scenarios. Taking it one step further, predictive models could even be tested against synthetic human models to help improve the accuracy of the desired prediction.

The notion of synthetic human models was exactly the strategy that was used to train the Xbox Kinect device to determine human body movements and interactions. Initially, humans were used to record basic physical body movements; trainers wore sensors attached to arms, legs, hands, and fingers while recording devices captured the movements. Once the basic human physical movements were initially captured, computer-simulated data could then be extrapolated and synthetically generated many times over to account for variations in things like the size of physical appendages, objects in the room, and distances from the Kinect unit.

Summary

Azure Machine Learning provides a way of applying historical data to a problem by creating a model and using it to successfully predict future behaviors or trends. In this chapter, we learned about the high-level workflow of Azure Machine Learning and the continuous cycle of predictive model creation, model evaluation, model deployment, and the testing and feedback loop.

The good news is that a working knowledge of data science theories and predictive modeling algorithms is highly beneficial—but not absolutely required—for working with Azure Machine Learning. The primary predictive analytics algorithms currently used in Azure Machine Learning are classification, regression, and clustering.

Creating new prediction models using supervised and unsupervised learning algorithms are easily accomplished using Azure Machine Learning. Combine the exponential amounts of historical transaction data that are increasingly available today along with vast amounts of ubiquitous computing power in the form of Microsoft Azure, and you have a "perfect storm" of conditions for creating very compelling and useful prediction services.

Resources

For more information about Azure Machine Learning, please see the following resources:

Documentation

- What is Azure Machine Learning Studio?
- Create a simple experiment in Azure Machine Learning Studio

Videos

- Getting started with Azure Machine Learning – Step 1

Chapter 3

Using Azure ML Studio

In this chapter, we start to drill into the basic fundamentals of using the Azure Machine Learning features. This will help you get started on your path toward becoming an Azure Machine Learning data scientist.

Azure Machine Learning (ML) Studio is the primary tool that you will use to develop predictive analytic solutions in the Microsoft Azure cloud. The Azure Machine Learning environment is completely cloud-based and self-contained. It features a complete development, testing, and production environment for quickly creating predictive analytic solutions.

Azure ML Studio gives you an interactive, visual workspace to easily build, test, and iterate on a predictive analysis model. You drag and drop datasets and analysis modules onto an interactive canvas, connecting them together to form an experiment, which you can then run inside Azure ML Studio. You can then iterate on your predictive analytics model by editing the experiment, saving a copy if desired, and then running it over and over again. Then, when you're ready, you can publish your experiment as a web service, so that your predictive analytics model can be accessed by others over the Web.

Another key benefit of the Azure Machine Learning cloud-based environment is that getting started requires virtually no startup costs in terms of either time or infrastructure. Here’s the best part of all: Almost all the tasks related to Azure Machine Learning can be accomplished within a modern web browser.

Azure Machine Learning terminology

To help get you started quickly, let’s define a few Azure Machine Learning terms that we will use throughout the remainder of this book to describe the various features, components, and tools.

- **Azure Machine Learning** Contains all the tools necessary to design, develop, share, test, and deploy predictive analytic solutions in the Microsoft Azure cloud.
- **Azure Machine Learning workspaces** Represent a discrete “slice” of the Azure Machine Learning tool set that can be partitioned by the following criteria:
  - **Workspace name** Required to be unique and is the primary method for identifying a Machine Learning workspace.
  - **Workspace owner** Valid Microsoft account that will be used to manage access to this Azure Machine Learning workspace.

    - **Data center location** Defines the physical Azure Data Center location for hosting the Azure Machine Learning workspace.
    - **Storage account** Defines the unique Azure Storage account that will be used to store all of the data and artifacts related to this Azure Machine Learning workspace.
- **Azure Machine Learning experiments** Experiments are created within Azure Machine Learning workspaces and represent the primary method of enabling an iterative approach to rapidly developing Azure Machine Learning solutions. Within each Azure Machine Learning experiment, Azure ML Studio gives you an interactive, visual workspace to easily build, test, and iterate on a predictive analytic experiment. These experiments can then be submitted to Azure ML Studio for execution. Azure Machine Learning experiments are highly iterative. You can easily create, edit, test, save, and rerun experiments. The use of Azure Machine Learning experiments is specifically designed to allow today's modern data scientists to "fail fast" when evaluating new predictive models while providing the ability to progress predictive model feedback for further model refinements. Simply put, Azure Machine Learning gives you the iterative speed to either fail fast or ultimately succeed.
- **Azure ML Studio** This is the primary interactive predictive analytics workbench that is accessed from within an Azure Machine Learning workspace to allow a data scientist to create Azure Machine Learning experiments via a drag-and-drop visual designer interface. Access to a unique Azure ML Studio environment is governed from within an Azure Machine Learning workspace. In addition to enabling the creation of new experiments, Azure ML Studio also has links to sample Azure Machine Learning experiments. These are provided so that you can easily learn from others as you make your way on your data science journey and leverage some of the best processing techniques and tools in the industry to help accomplish your specific predictive analytics goals.
- **Azure Machine Learning web services** These represent Azure Machine Learning experiments that have been exposed as public APIs over the Internet in the form of the Azure Machine Learning REST API. These services are generally exposed as a simple web service, or as an OData endpoint. The API provides two types of RESTful web interfaces:
    - **Request Response Service (RRS)** For individual, low-latency, synchronous uses, for making predictions.
    - **Batch Execution Service (BES)** For asynchronous scoring of a batch of data records. The input for BES is a batch of records from a variety of sources such as blobs, tables, SQL Azure, HD Insight (as a result of a Hive Query), and HTTP sources.
- **Datasets** This is data that has been uploaded to Azure ML Studio so that it can be used in the prediction modeling process. A number of sample datasets are included with Azure ML Studio for you to experiment with, and you can upload more datasets as you need them.

- **Modules** These are algorithms that you can apply to your data. Azure ML Studio has a number of modules ranging from data ingress functions to training, scoring, and validation processes. Here are some examples of included modules:
    - **Convert to ARFF** Converts a .NET serialized dataset to ARFF format. ARFF is a common machine learning construct and stands for Attribute-Relation File Format. It is commonly defined as an ASCII text file that describes a list of instances sharing a set of attributes.
    - **Elementary Statistics** Calculates elementary statistics such as mean, standard deviation, and so on.
    - **Linear Regression** Creates an online gradient, descent-based, linear regression model.
    - **Score Model** Scores a trained classification or regression model.

A module might have a set of parameters that you can use to configure the module's internal algorithms. When you select a module on the canvas, its parameters are displayed in the pane to the right of the canvas. You can modify the parameters in that pane to tune your model.

Enough of the background, motivations, terminology, and predictive analytic theories. It's time to get started!

Getting started

The absolute first step in your Azure Machine Learning journey is to get access to the Microsoft Azure environment. There are several ways to get started, and here are your options:

- **Option 1** Take advantage of a free Azure trial offer at http://azure.microsoft.com/en-us/pricing/free-trial.
    - This offer allows you to get a $200 credit to spend on all Azure services for one month. You can use this $200 credit however you wish to create and try out any combination of Azure resources. It enables you to explore the Azure cloud for free.
    - Note that this option will require you to apply using a Microsoft account and a valid credit card number for account verification purposes. The free trial offer is designed so that you will not start the actual billing process until you completely agree.
    - During the trial period, the Azure Management Portal will prominently display your remaining trial credits at the top of the portal page to inform you of how many credits are left.
- **Option 2** Take advantage of the (free) Azure Machine Learning trial offer at https://studio.azureml.net/Home.

- This is a free Azure offer that is feature-specific and therefore only allows you access to the Azure Machine Learning environment.
- This is an extremely low-friction option for new adopters: All you need is a valid Microsoft account to get started.
- If you need to sign up for a Microsoft account, visit http://windows.microsoft.com/en-US/windows-live/sign-up-create-account-how.
- Once you have signed in with a valid Microsoft account, an introductory video is displayed to help get you started, as shown in Figure 3-1. You can review this introductory video at https://go.microsoft.com/fwlink/?LinkID=518038.

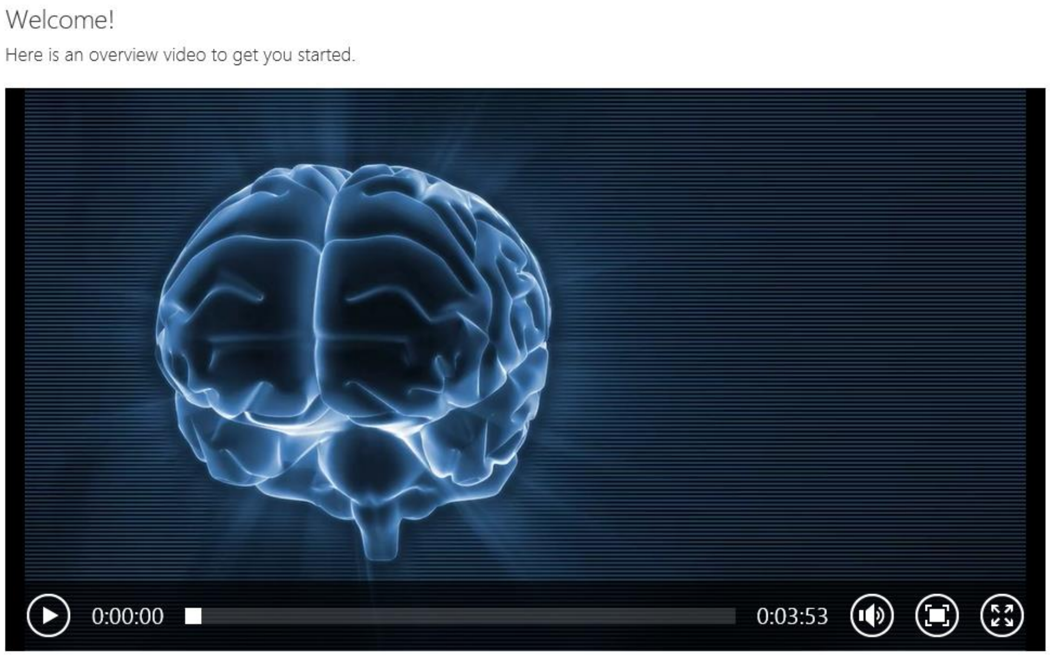


FIGURE 3-1 Azure Machine Learning introductory video.

Note that if you opt to take advantage of the free Azure Machine Learning trial offer, you will only have access to the Azure Machine Learning features, not access to the full Azure environment. To truly maximize your experience, it is highly recommended that you gain access to the complete Microsoft Azure environment.

Figure 3-2 is a screenshot of the initial Azure Machine Learning environment. For the remainder of this book, we assume navigation to the Azure Machine Learning features via the Azure Management

Portal. Once you navigate to the Azure ML Studio screen, the Azure Machine Learning features are the same.

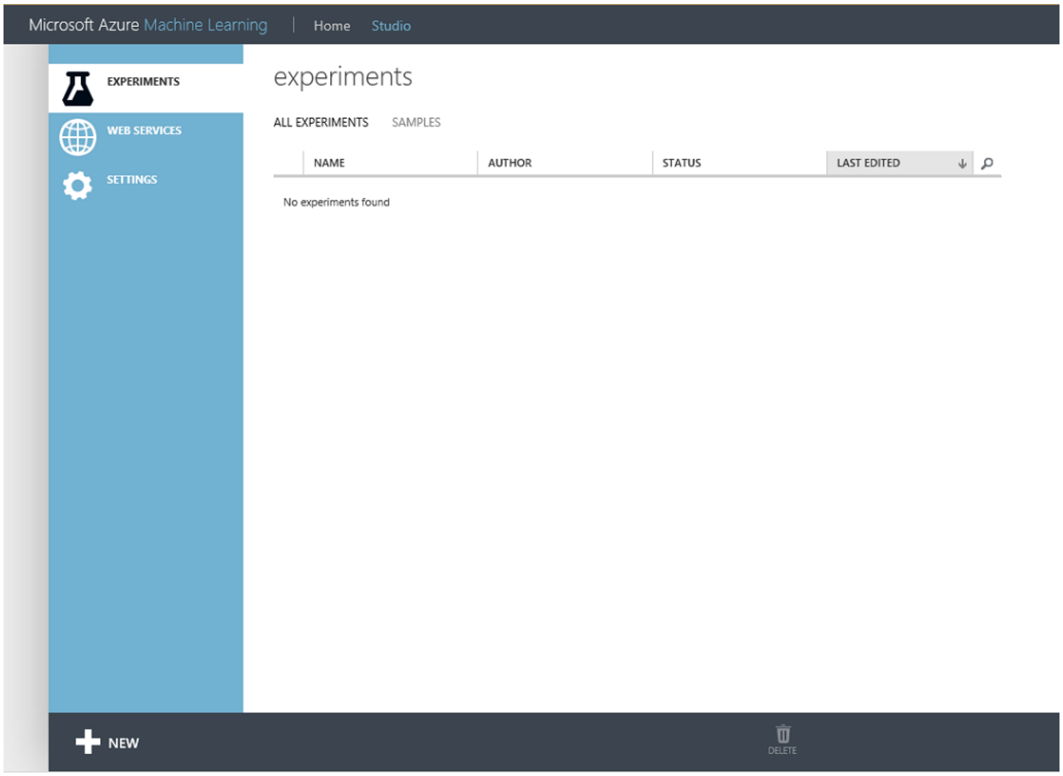


FIGURE 3-2 The Azure Machine Learning environment.

Azure Machine Learning pricing and availability

Microsoft Azure Machine Learning has been designed as a suite of offerings to help enable partners and customers to easily design, develop, test, implement, and manage predictive analytic solutions in the Azure cloud.

Ultimately, Azure Machine Learning will be another set of the Azure cloud "pay-for-play" services. You will soon see how this can be an extremely effective model to pursue for design, testing, and implementing predictive analytics in the cloud.

At the time of this writing, the Azure Machine Learning services are currently in "preview" or "beta" mode and not yet deemed ready for general availability and consumption until the appropriate levels of quality and customer and partner feedback have been achieved.

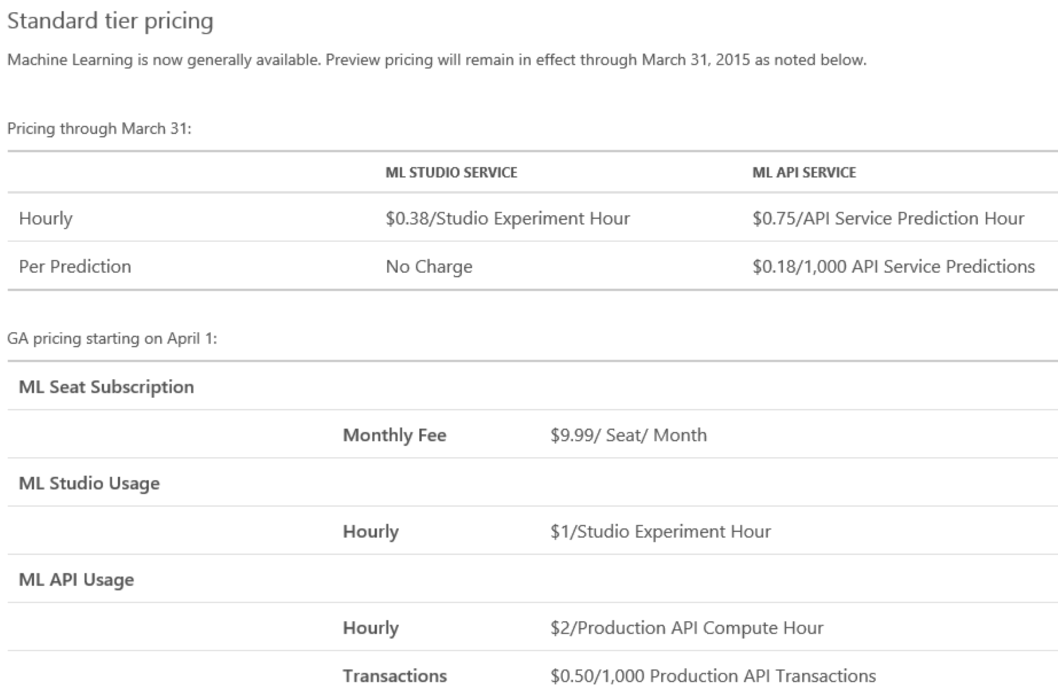
In terms of where you can host an Azure Machine Learning service, the only valid datacenter location (at the time of this writing) is the "South Central US" Azure datacenter. Expect more Azure datacenter locations to come online as this feature nears general availability status.

Despite the initial limited availability, Microsoft has published pricing details for Azure Machine Learning while it is still in preview mode. You can find more details on pricing options at http://azure.microsoft.com/en-us/pricing/details/machine-learning/.

Note that Microsoft is currently offering two tiers of Azure Machine Learning Services: Free and Standard. Each of these tiers provides a different level of Azure Machine Learning service capabilities, features, and pricing for each of the two primary services in Azure Machine Learning:

- **The Azure ML Studio Service** This service is used to help design, develop, share, test, and deploy Azure Machine Learning solutions.
- **The Azure ML API Service** This service is used to provide an Azure Machine Learning web service–hosting environment for use in running test and production Machine Learning experiments or scenarios.

Figure 3-3 is a screenshot that illustrates the current preview Azure Machine Learning pricing and availability options available at the time of this book's writing.



Standard tier pricing

Machine Learning is now generally available. Preview pricing will remain in effect through March 31, 2015 as noted below.

Pricing through March 31:

	ML STUDIO SERVICE	ML API SERVICE
Hourly	$0.38/Studio Experiment Hour	$0.75/API Service Prediction Hour
Per Prediction	No Charge	$0.18/1,000 API Service Predictions

GA pricing starting on April 1:

ML Seat Subscription		
	Monthly Fee	$9.99/ Seat/ Month
ML Studio Usage		
	Hourly	$1/Studio Experiment Hour
ML API Usage		
	Hourly	$2/Production API Compute Hour
	Transactions	$0.50/1,000 Production API Transactions

FIGURE 3-3 Azure Machine Learning pricing details while in preview mode.

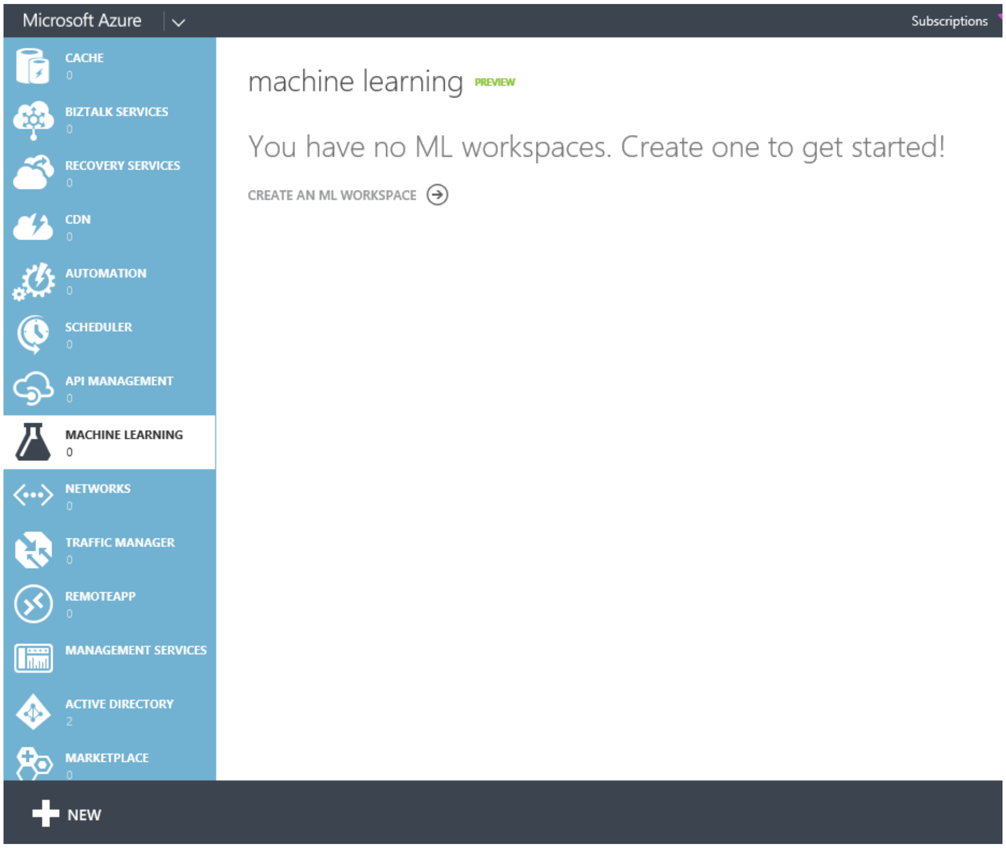
After the Azure free trial offer expires, as outlined in Option 1, these are the potential costs you could incur as you continue to develop and consume these services.

It's always a good idea to be aware of the Azure Machine Learning pricing information in the event you hit on the next predictive analytics breakthrough for, say, predicting the next big stock market turn.

Note that the preview Azure Machine Learning prices are subject to change at the time that the Azure Machine Learning services move into general availability (GA) mode.

Create your first Azure Machine Learning workspace

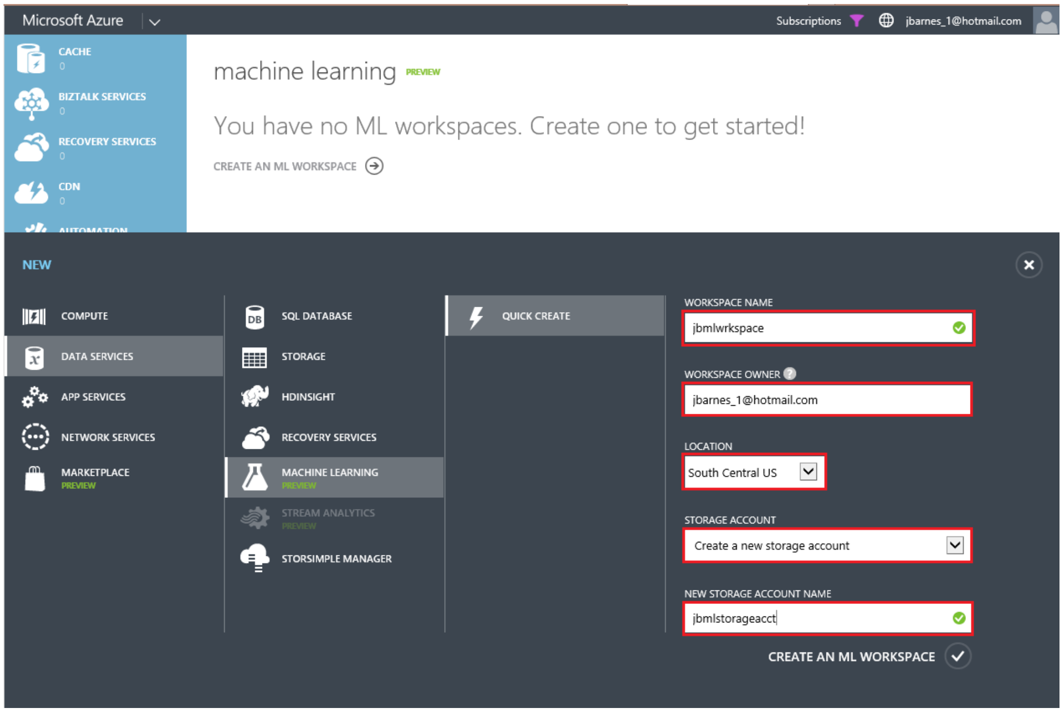
Let's create our first Azure Machine Learning workspace. At this point, you should be all set up with either a free or paid Azure subscription or using the Azure Machine Learning free trial offer. Start by navigating to the Azure Management Portal at https://manage.windowsazure.com. From there, click the left navigation bar for Machine Learning as shown in Figure 3-4.



You have no ML workspaces. Create one to get started!

FIGURE 3-4 The Azure Machine Learning section of the Microsoft Azure Management Portal.

An Azure Machine Learning workspace contains all of the tools you need to create, manage, and publish machine learning experiments in the cloud. To create a new Azure Machine Learning workspace, click the New icon in the bottom left of the page. Fill in the required fields as shown in Figure 3-5.



You have no ML workspaces. Create one to get started!

WORKSPACE NAME: jbmlwrkspace

WORKSPACE OWNER: jbarnes_1@hotmail.com

LOCATION: South Central US

STORAGE ACCOUNT: Create a new storage account

NEW STORAGE ACCOUNT NAME: jbmlstorageacct

FIGURE 3-5 Creating a new Azure Machine Learning workspace.

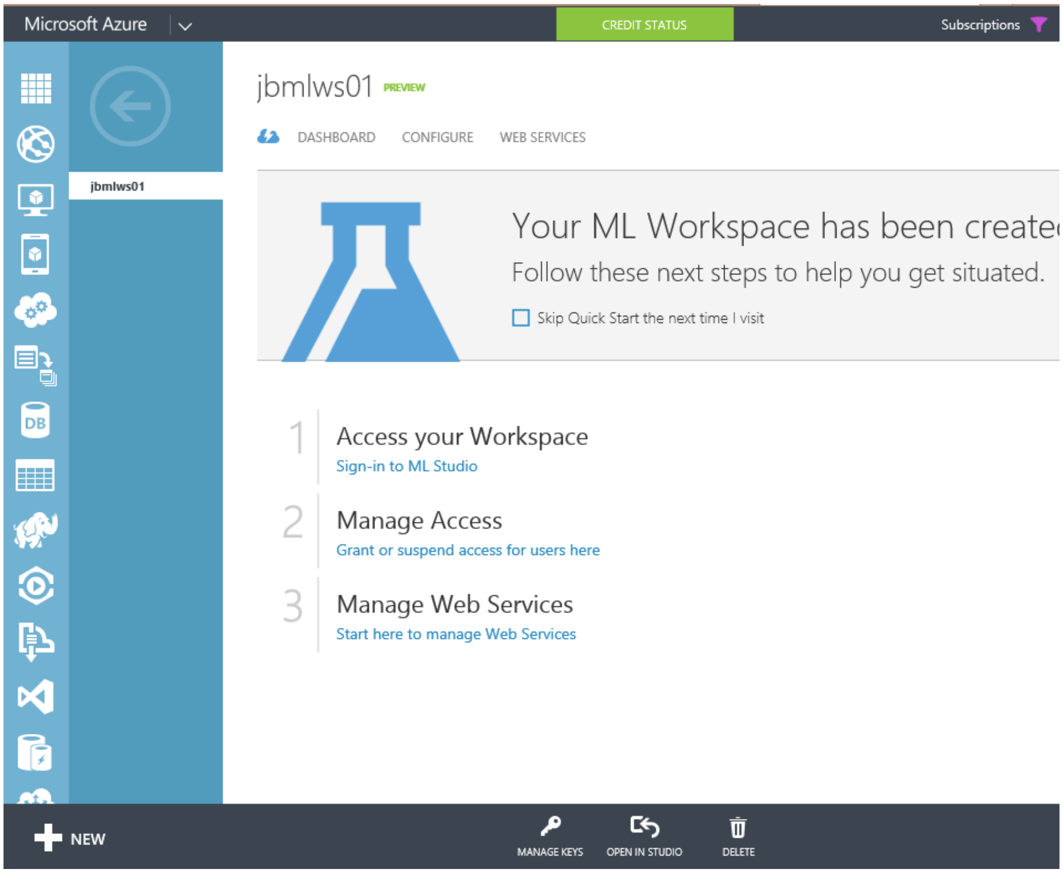
The required inputs for creating a new Azure Machine Learning workspace include the following:

- **Workspace Name** Provide a unique name for your Azure Machine Learning workspace. You will know if it is unique by the presence of the green check mark to the right of the text box after you move the cursor from this field.
- **Workspace Owner** Provide a valid Microsoft account (formerly Windows Live ID). Note that it cannot be a non-Microsoft account, such as your corporate email account. To create a free Microsoft account, go to www.live.com.
- **Location** At the time of this writing, Azure Machine Learning services are available only in the South Central U.S. region.
- **Storage Account** Select the option to either create a new storage account or use an existing storage account.

- **New Storage Account Name** If you opt to create a new storage account for your Azure Machine Learning workspace, be sure that the storage account name is only made up of lowercase alphanumeric characters. You will know if it is unique by the presence of the green check mark to the right of the text box.

Once you click Create An ML Workspace, Azure will then provision a brand new Azure Machine Learning workspace for you to use to create and host your Azure Machine Learning experiments.

After your Azure Machine Learning Workspace has been created, click your new Azure Machine Learning Workspace and you should see a screen similar to Figure 3-6.



Your ML Workspace has been create

Follow these next steps to help you get situated.

Skip Quick Start the next time I visit

1 Access your Workspace
Sign-in to ML Studio

2 Manage Access
Grant or suspend access for users here

3 Manage Web Services
Start here to manage Web Services

FIGURE 3-6 A new Azure Machine Learning workspace.

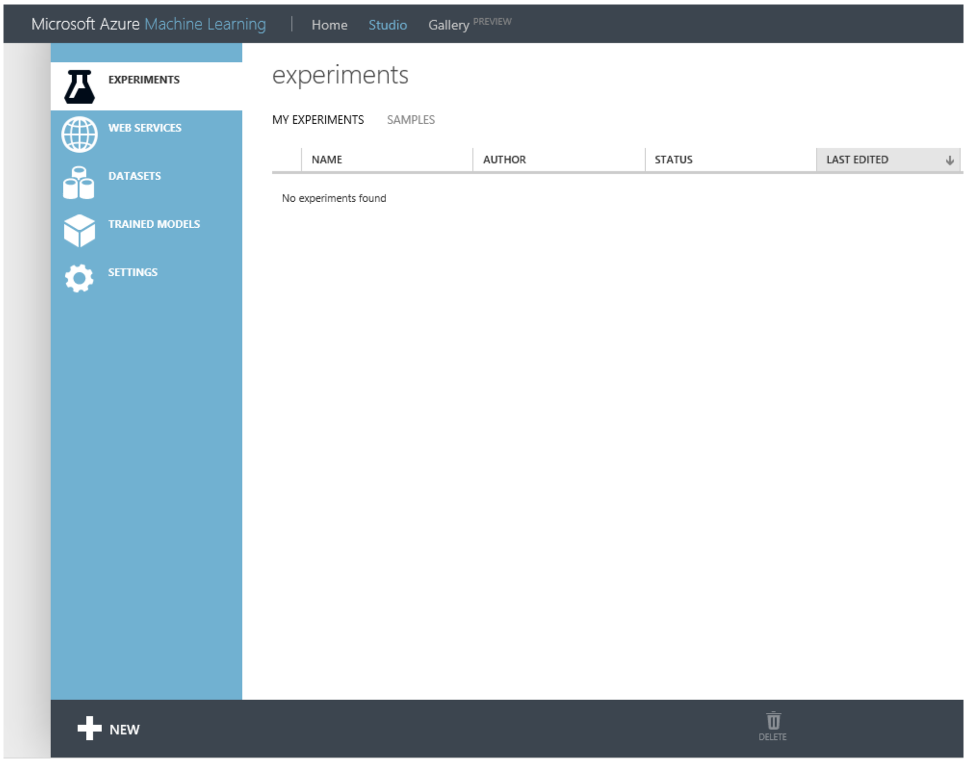
Note that this is the main landing page for managing an Azure Machine Learning workspace. From this page, you can directly access the Azure Machine Learning Studio tools for your workspace, manage user access to your workspace, and manage any web services that you might have deployed as a result of your experiments in your workspace. The top navigation menu provides access to several Azure

Machine Learning Workspace features.

- **Dashboard** Here you can monitor the relative or absolute computational usage of your workspace over a period of time.
- **Configure** This feature is used to allow or deny user access to your Azure Machine Learning workspace.
- **Web Services** This option allows you to manage web services, configure endpoints, and access your API via request/response or batch code samples in C#, Python, or in the R language, a popular programming language for data scientists and statisticians.

We revisit these features in detail later as we continue our exploration of the Azure Machine Learning environment.

Under Access Your Workspace, click the Sign-In To ML Studio link to sign in to your new Azure Machine Learning workspace. Figure 3-7 is a screenshot of the Azure ML Studio inside a new workspace.



No experiments found

FIGURE 3-7 The Azure ML Studio inside of an Azure Machine Learning workspace.

When you first enter the Azure ML Studio workspace, you will see the following navigation tabs on the top and left navigation bars:

Top Navigation Bar:

- **Home** A set of links to documentation and other resources.
- **Studio** The landing page for Azure ML Studio experiments.
- **Gallery (Preview)** A collection of trending experiments and sample experiments.

Left Navigation Bar:

- **EXPERIMENTS** Experiments that have been created, run, and saved as drafts.
- **WEB SERVICES** A list of experiments that you have published.
- **DATASETS** Uploaded datasets that can be used in your experiments.
- **TRAINED MODELS** New predictive models that have been "trained" using the built-in Azure ML Studio machine learning algorithms.
- **SETTINGS** A collection of settings that you can use to configure your account and resources.

Create your first Azure Machine Learning experiment

Let's create a new experiment by clicking +NEW in the bottom left corner of the screen. You will then have the option to create either of the following items:

- **Data Set** This option allows you to upload a new dataset to use with your experiment from a local file on disk.
- **Experiment** Start with a blank experiment or a preexisting Microsoft sample experiment to help get you started fast.

Let's start with a simple, real-world scenario such as predicting whether a person's income exceeds $50,000 per year based on his demographics or census data. You can imagine how incredibly useful the ability to predict a person's income might be in the world of sales and marketing.

This is exactly the kind of predictive analytics that would be most useful for a successful targeted marketing campaign for products that require buyers with a certain level of disposable income. This will be a simplified example of how you could use Azure Machine Learning with ML Studio and ML API web services to create a real-world, cloud-based predictive analytics solution to help drive a marketing campaign.

In this walkthrough, we follow the entire process of developing a predictive analytics model in Azure

Machine Learning Studio and then publish it as an Azure Machine Learning API web service.

We start by downloading a sample Census Income Dataset from a public repository such as the UCI Machine Learning Repository from the following link: http://archive.ics.uci.edu/ml/datasets/Census+Income.

We then develop and train a predictive model based on that dataset, and then publish the predictive model as a web service that can be used by other applications. These are the high-level steps we follow:

1. Download, prepare, and upload a census income dataset.
2. Create a new Azure Machine Learning experiment.
3. Train and evaluate a prediction model.
4. Publish the experiment as an Azure Machine Learning web service.
5. Access the Azure Machine Learning web service via sample tester programs.

Download dataset from a public repository

To develop a predictive model for our sample income level predictive analytics model, we use the Adult Census Income Binary Classification dataset from the UCI Machine Learning repository. You can download the dataset from http://archive.ics.uci.edu/ml/datasets/Census+Income.

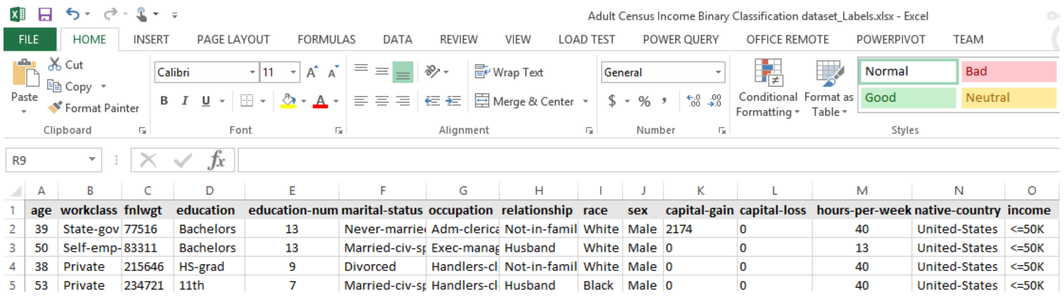
The website will have a link to a download folder, and you will want to download the adult.data file to your local computer. This dataset is in the Comma Separated Values (CSV) format. Note that the website also contains information about the 15 attributes found in this dataset. We use this information to create column headers for our data before we upload it into our experiment.

Now, open the adult.data file with Microsoft Excel or any other spreadsheet tool and add the column header names from the list of attributes on the website as in the following list. Note that some attributes are labeled as continuous, as they represent numerical values, whereas other attributes have a predefined list of potential values.

- **Age** Continuous
- **Workclass** Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
- **Fnlwgt** Continuous
- **Education** Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
- **Education-num** Continuous

- **Marital-status** Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
- **Occupation** Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
- **Relationship** Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
- **Race** White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
- **Sex** Female, Male
- **Capital-gain** Continuous
- **Capital-loss** Continuous
- **Hours-per-week** Continuous
- **Native-country** United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holland-Netherlands
- **Income** >50K, <=50K

After you have inserted a Header row with these column values, be sure to save the file with the .csv extension when you are done. For the purposes of this walkthrough, name the file Adult.data.csv when you save it. Figure 3-8 shows a screenshot of the Excel spreadsheet with the associated column headings.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
2	39	State-gov	77516	Bachelors	13	Never-marrie	Adm-clerica	Not-in-famil	White	Male	2174	0	40	United-States	<=50K
3	50	Self-emp-	83311	Bachelors	13	Married-civ-sp	Exec-manag	Husband	White	Male	0	0	13	United-States	<=50K
4	38	Private	215646	HS-grad	9	Divorced	Handlers-cl	Not-in-famil	White	Male	0	0	40	United-States	<=50K
5	53	Private	234721	11th	7	Married-civ-sp	Handlers-cl	Husband	Black	Male	0	0	40	United-States	<=50K

FIGURE 3-8 Screenshot of the Excel spreadsheet containing the adult census data.

Let's summarize a few facts about the Census Income dataset that we use for our first Azure Machine Learning experiment:

- **Number of unique attributes** Fourteen related to the outcome

- **Number of instances in the dataset** 48,842
- **Prediction task** Determine whether a person makes over $50,000 a year

Note that this Census Income dataset is actually already provided by Microsoft as one of their sample datasets under the name of the Adult Census Income Binary Classification dataset. We are going through these steps manually to provide a comprehensive, end-to-end overview of the entire Azure Machine Learning workflow process. Most likely, your real-world predictive model datasets will also come from external sources, so it's good to know how it all works together from beginning to end.

Upload data into an Azure Machine Learning experiment

Once we have added the appropriate column headings to the sample Census Income dataset, we will then upload the dataset into the Azure Machine Learning workspace so that we can incorporate it into our prediction model. Click + New in the bottom left of the screen and select Dataset to upload a new dataset. Figure 3-9 shows the option to upload a new dataset from a local file.

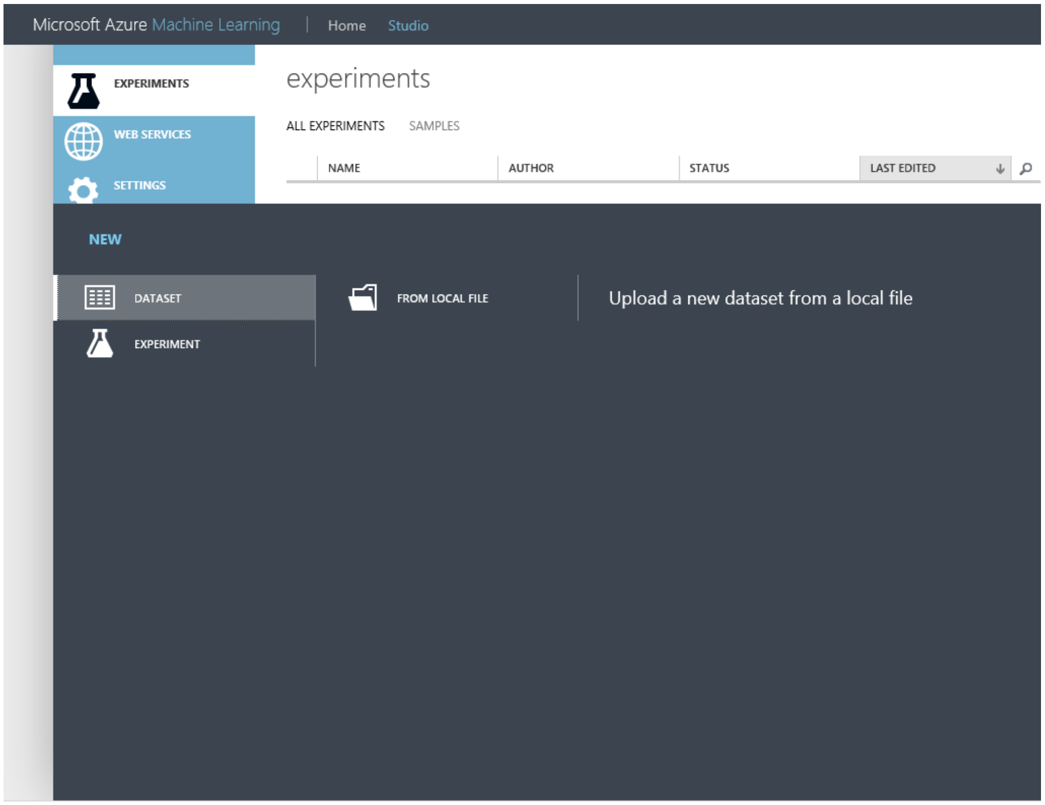
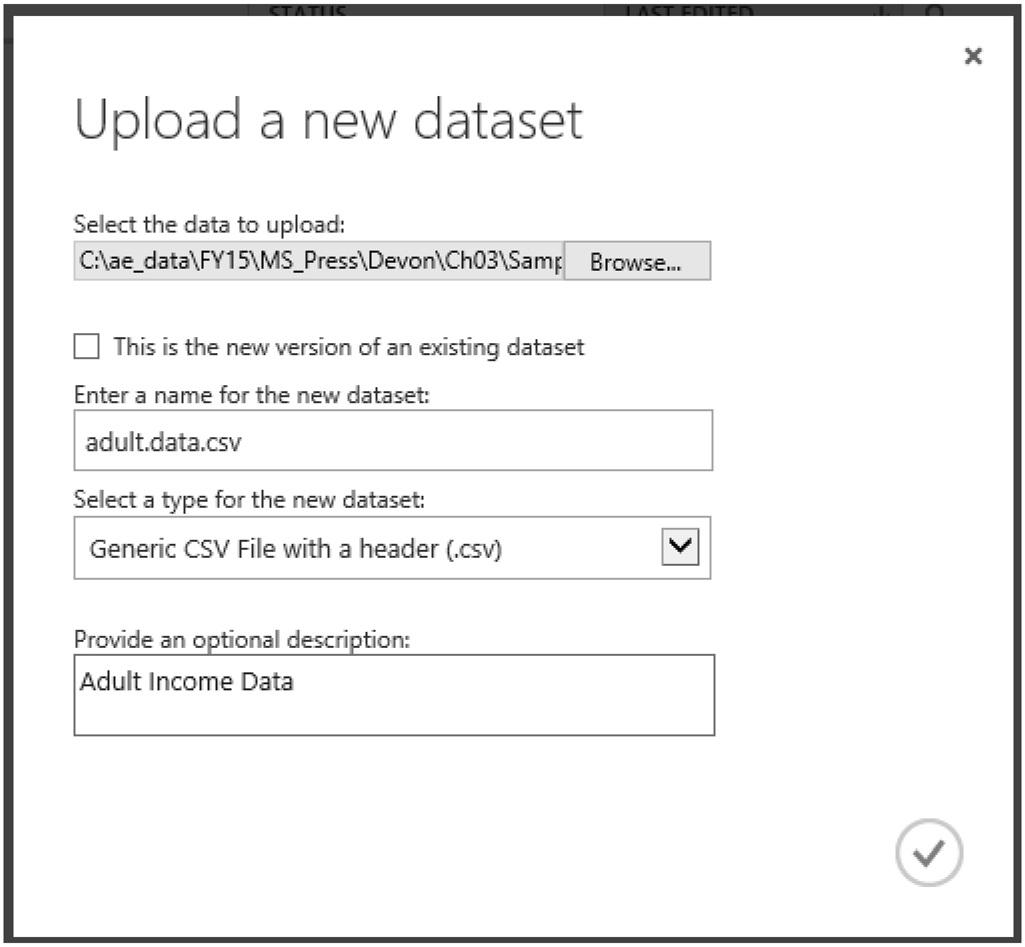


FIGURE 3-9 The Azure Machine Learning option to upload a dataset from a local file location.

Next, click From Local File. You will see an upload screen similar to Figure 3-10. Here you can specify the upload file properties such as the location of the file, the name for the new dataset (we will use Adult.data.csv), the type of file (Generic CSV file with a header), and an optional description for the new dataset.



Upload a new dataset

Select the data to upload: C:\ae_data\FY15\MS_Press\Devon\Ch03\Samp

☐ This is the new version of an existing dataset

Enter a name for the new dataset: adult.data.csv

Select a type for the new dataset: Generic CSV File with a header (.csv)

Provide an optional description: Adult Income Data

FIGURE 3-10 Azure Machine Learning dialog box options to specify uploading a new dataset from a local file location.

Once you have entered the information and clicked the check mark, your dataset will be uploaded asynchronously into your Azure Machine Learning workspace for your use in your first Azure Machine Learning experiment.

Create a new Azure Machine Learning experiment

Create a new experiment by clicking on +NEW in the bottom left corner of the screen. Click Experiment and then click Blank Experiment as shown in Figure 3-11.

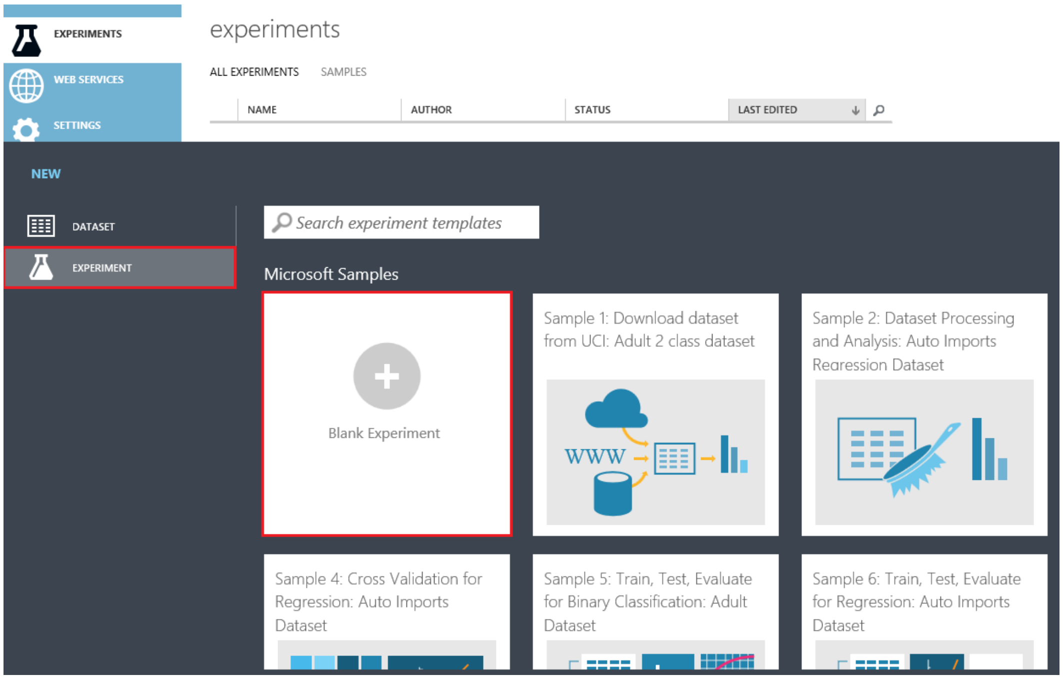

EXPERIMENTS
WEB SERVICES
SETTINGS
experiments
ALL EXPERIMENTS
SAMPLES
NAME
AUTHOR
STATUS
LAST EDITED
NEW
DATASET
EXPERIMENT
Search experiment templates
Microsoft Samples
Blank Experiment
Sample 1: Download dataset from UCI: Adult 2 class dataset
Sample 2: Dataset Processing and Analysis: Auto Imports Regression Dataset
Sample 4: Cross Validation for Regression: Auto Imports Dataset
Sample 5: Train, Test, Evaluate for Binary Classification: Adult Dataset
Sample 6: Train, Test, Evaluate for Regression: Auto Imports Dataset

FIGURE 3-11 List of new experiment types in Azure Machine Learning.

Note that in addition to a blank experiment template, there are many other sample experiments that you can load and modify to provide a jumpstart to your Azure Machine Learning activities.

Once the new blank experiment has loaded, you will then see the Azure ML Studio visual designer screen, as shown in Figure 3-12.

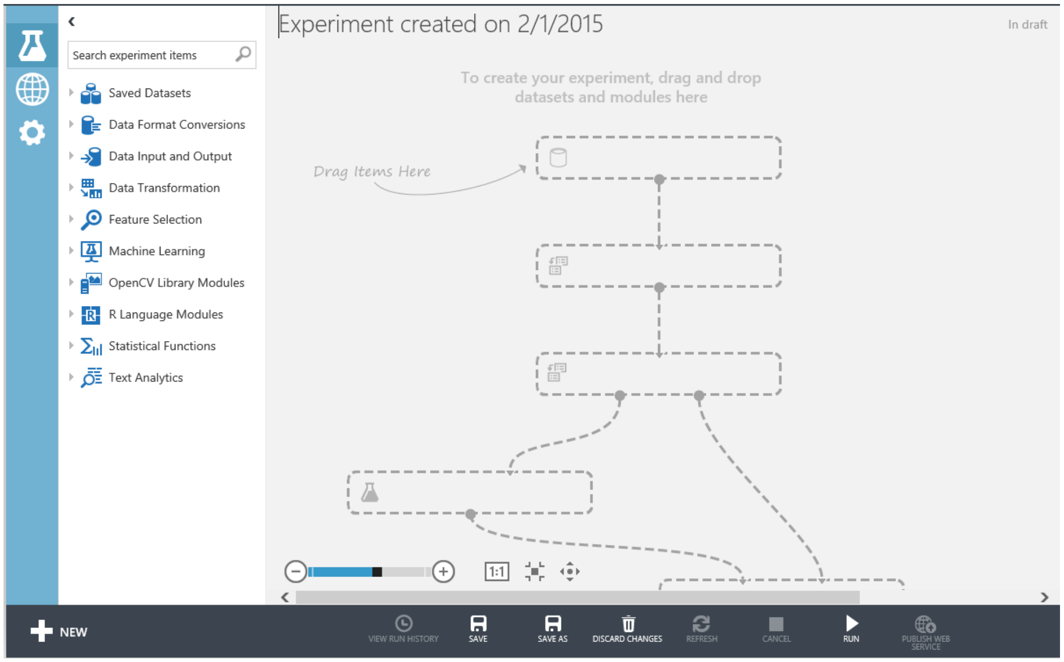


FIGURE 3-12 A blank Azure Machine Learning experiment in the Azure ML Studio designer.

Note that the designer is made up of three main areas:

- **Left navigation pane** This area contains a searchable listing of all the Azure Machine Learning modules that can be used in creating a predictive analytics model.
  - Modules are grouped by functional area and contains functions for
  - Reading, formatting, and converting datasets.
  - Using and training machine learning algorithms.
  - Scoring and evaluating predictive model results.
- **Center pane** In the visual designer, Azure Machine Learning Experiments resemble flowcharts. They are assembled by dragging and dropping module shapes from the list in the left pane into the visual design surface in the middle of the screen. Modules can be freely positioned on the surface and are connected by drawing lines between input and output ports.
- **Right pane** In the Properties view, properties of selected modules are viewed and set using the pane on the right side of the visual designer.

Now, expand the Saved Datasets module on the left side of the screen and you will see where our uploaded dataset Adult.data.csv appears as a dataset for use in your Azure Machine Learning experiment. Figure 3-13 shows the Adult.data.csv dataset ready to be dragged and dropped onto the visual designer surface.

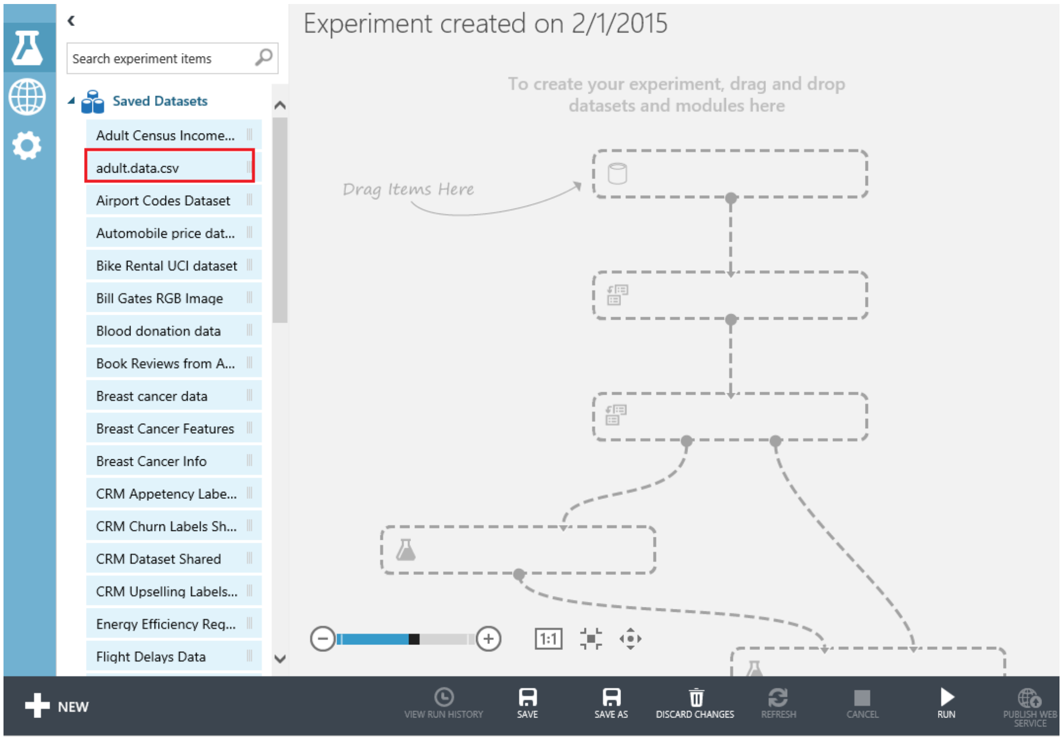


FIGURE 3-13 Dragging the Adult.data.csv dataset onto the designer surface.

Visualizing the dataset

Drag the Adult.data.csv dataset into the middle of the visual designer surface and drop it into the experiment. Note that there are two ways to view the data in this dataset.

1. You can right-click anywhere in the dataset shape and then select Download to download the dataset to your local system. Figure 3-14 shows the Download menu option.

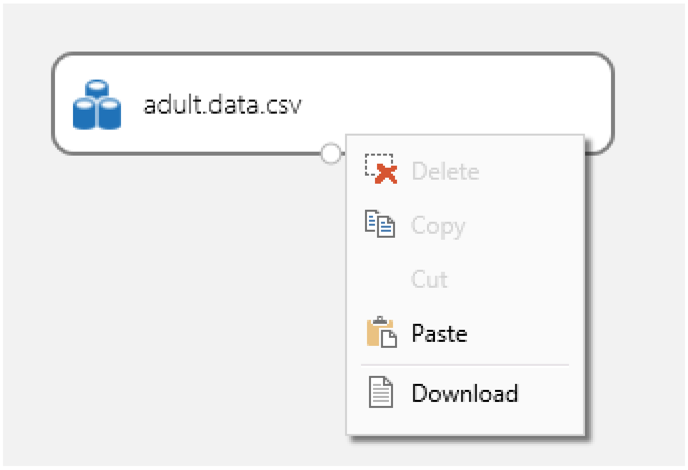


FIGURE 3-14 Downloading a dataset from an Azure Machine Learning experiment.

2. You can hover over the connector at the bottom of the shape and then right-click. You will see the options shown in Figure 3-15 to either download or visualize the dataset.

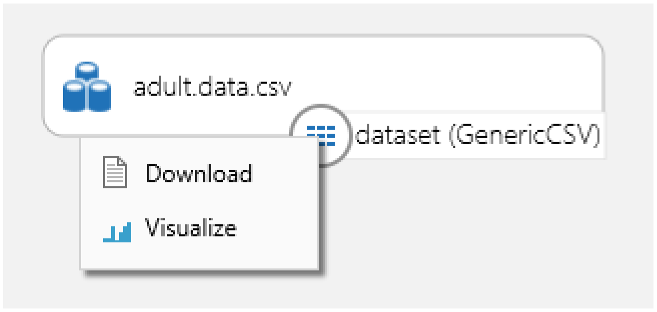
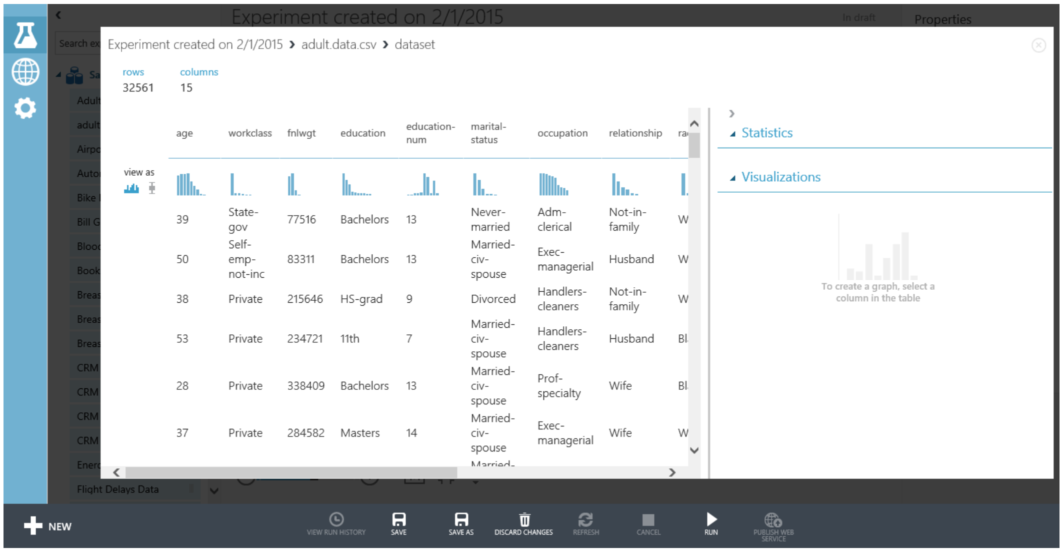


FIGURE 3-15 Right-clicking the connector displays options to download or visualize the dataset.

When you click Visualize, you will see a screen similar to Figure 3-16.



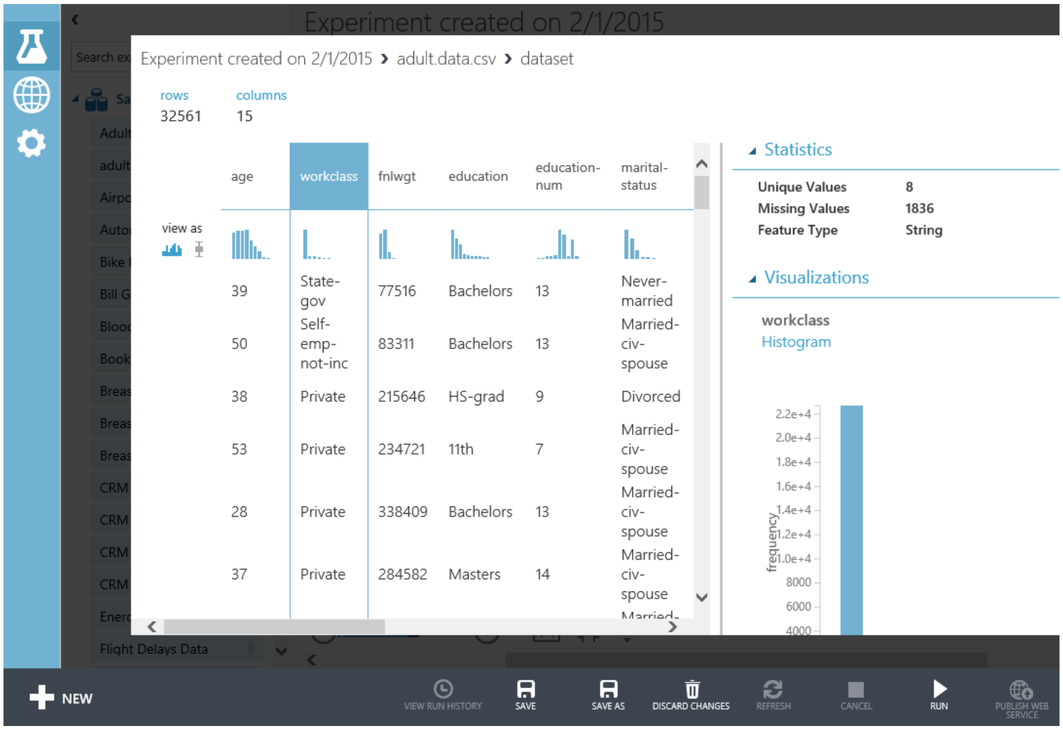
Experiment created on 2/1/2015 › adult.data.csv › dataset

rows 32561
columns 15

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	ra
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	W
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	W
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	W
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Bl
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Bl
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	W

FIGURE 3-16 Visualization of the Adult.data.csv dataset.

Note that the Visualize option provides some great statistical and visualization features to allow you to quickly analyze the underlying data for the selected column. For example, click on the workclass column and you will see a screen similar to the one shown in Figure 3-17.



Experiment created on 2/1/2015 › adult.data.csv › dataset

rows 32561 columns 15

age	workclass	fnlwgt	education	education-num	marital-status
39	State-gov	77516	Bachelors	13	Never-married
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse
38	Private	215646	HS-grad	9	Divorced
53	Private	234721	11th	7	Married-civ-spouse
28	Private	338409	Bachelors	13	Married-civ-spouse
37	Private	284582	Masters	14	Married-civ-spouse

Statistics

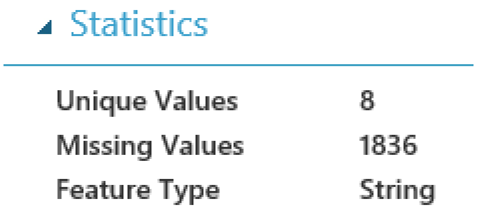
Unique Values	8
Missing Values	1836
Feature Type	String

Visualizations

workclass
Histogram

FIGURE 3-17 Statistics for the workclass column in the dataset.

Note that for the workclass column, the Statistics tab reveals the key information about the data found in this dataset shown in Figure 3-18.



Statistics

Unique Values	8
Missing Values	1836
Feature Type	String

FIGURE 3-18 Key statistics for the workclass column in the Adult.data.csv dataset.

Note the following key details about the workclass column in the Adult.data.csv dataset:

- There are eight unique values found in this dataset.
- There are 1,836 records with missing values. This is an important detail, as the records with missing values would need to be either updated with the correct value or dropped completely to allow for the most accurate results.

- The workclass field is designated as a String field type. As such, there are only nine valid string combinations for this field.

If you scroll down on the right side, you will see a Visualizations chevron similar to Figure 3-19 where the various field values for this column (workclass) are represented in a histogram.

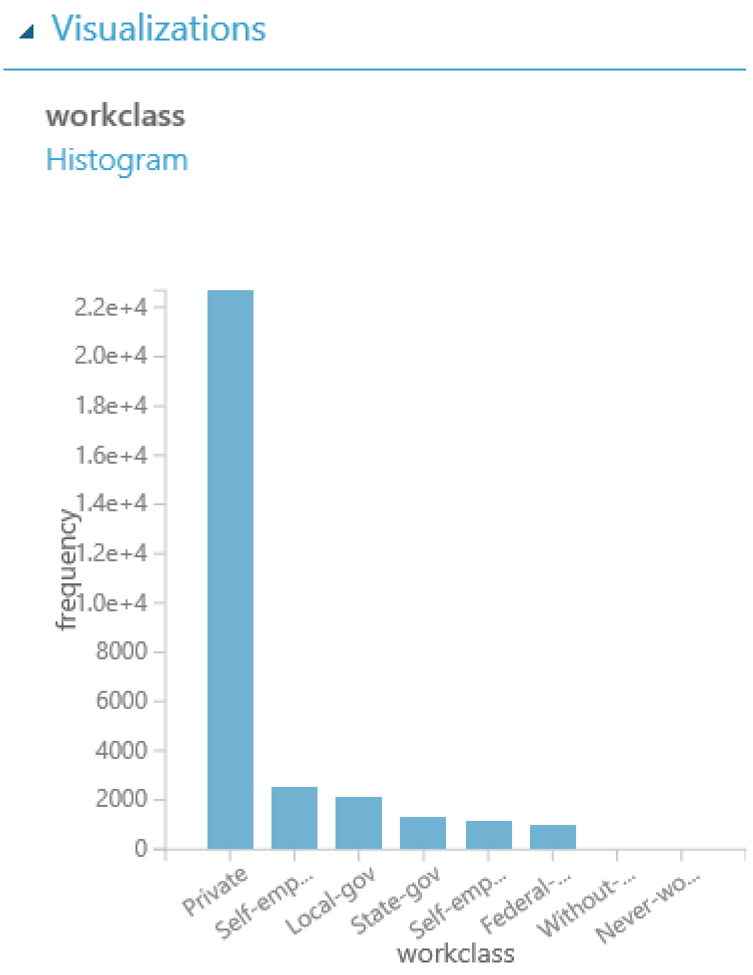


FIGURE 3-19 A histogram visualization of the workclass field in the Adult.data.csv dataset.

Using this built-in tool, you can easily determine from the visualization what the most common values found for the "workclass" field in the dataset are, such as Private and Self-employed on the left side.

Note how the ability to quickly and simply visualize your model’s datasets using Azure ML Studio makes it fast and easy to infer key data elements, associations, combinations, and patterns. This ability to visualize and quickly establish inferences will help rapidly create a powerful predictive analytics solution.

Split up the dataset

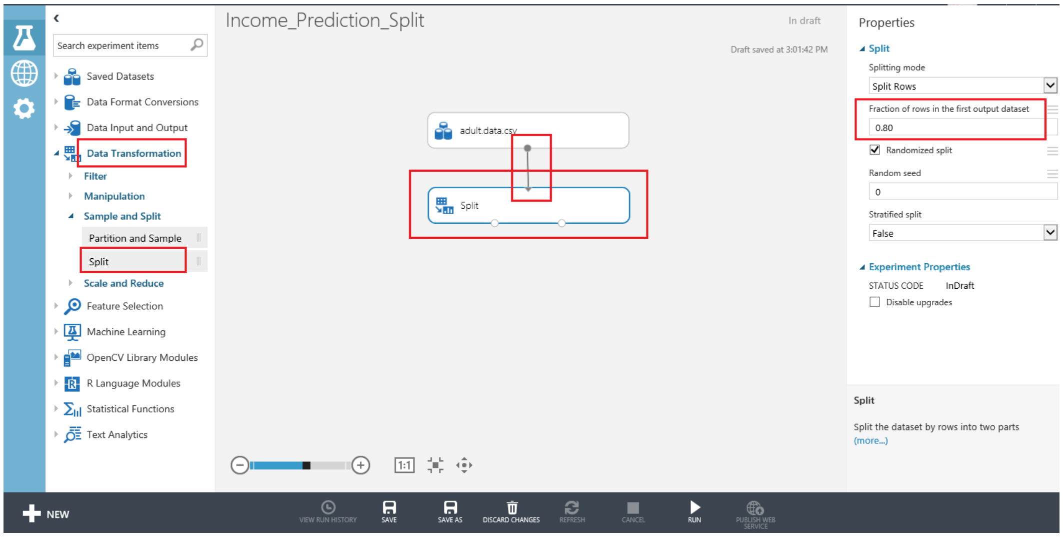
Typically, when creating an Azure Machine Learning experiment, we want to partition or split our dataset into two logical groupings for two specific purposes:

- **Training data** This grouping is used for creating our new predictive model based on the inherent patterns found in the historical data via the ML algorithm we use for the solution.
- **Validation data** This grouping is used for testing the new predictive model against known outcomes to determine accuracy and probabilities.

To accomplish this task, perform the following steps to split your dataset into two parts.

1. Expand the Data Transformation set of modules in the left pane.
2. Drag and drop the Split module onto the Azure Machine Learning designer surface.
3. Connect the Split” module to the Adult.data.csv dataset.
4. Click the Split module and set the value of the Fraction Of Rows In The First Output field to 0.80. This will divert 80 percent of our data to a training area in the designer.

Figure 3-20 displays a screenshot of how this is done in Azure ML Studio.



Properties

Split

Splitting mode: Split Rows

Fraction of rows in the first output dataset: 0.80

☑ Randomized split

Random seed: 0

Stratified split: False

Experiment Properties

STATUS CODE: InDraft

☐ Disable upgrades

Split

Split the dataset by rows into two parts (more...)

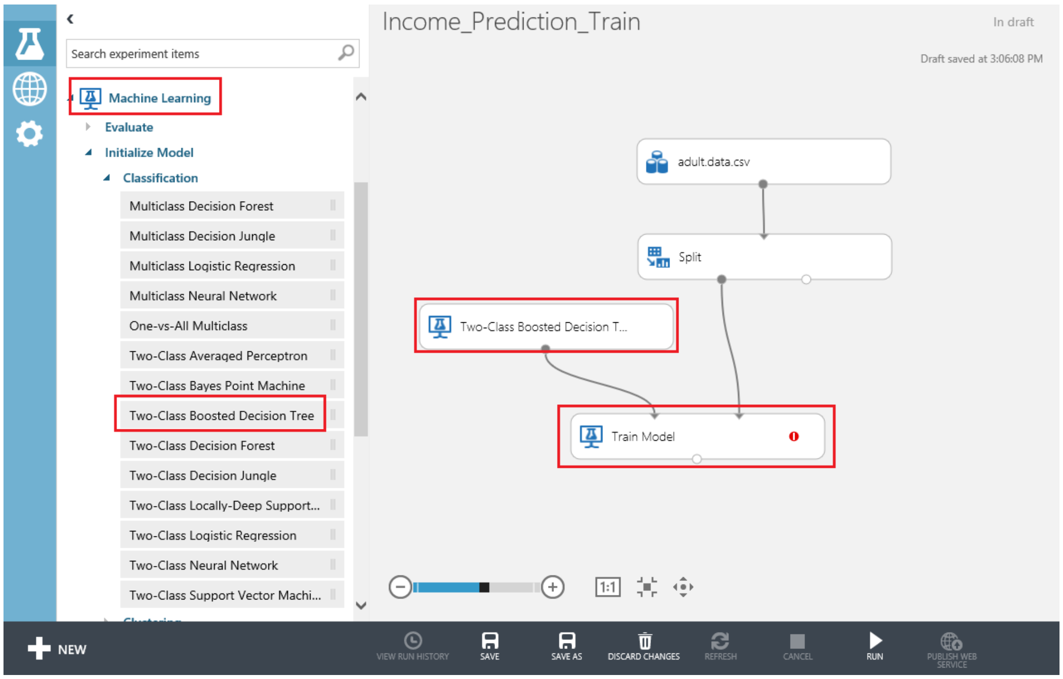
FIGURE 3-20 Splitting the Adult.data.csv dataset for training and testing purposes.

This approach allows us to divert 80 percent of the data in the dataset to training the model. We can use the remaining 20 percent of the data in the dataset to test the results of our new model for accuracy.

Train the model

The next step is to insert an Azure Machine Learning algorithm so we can “teach” the model how to evaluate the data. Start by expanding the Machine Learning module in the left pane. Then expand the Train submodule. Drag the Train Model shape onto the designer surface. Then connect the Train Model shape to the Split shape.

Next, expand Initialize Model under the Machine Learning module. Then expand the Classification submodule. For this experiment, we use Two-Class Boosted Decision Tree. Select it and drag and drop it onto the Azure Machine Learning designer surface. Your experiment should now look like the screenshot in Figure 3-21.



Search experiment items
Machine Learning
Evaluate
Initialize Model
Classification
Multiclass Decision Forest
Multiclass Decision Jungle
Multiclass Logistic Regression
Multiclass Neural Network
One-vs-All Multiclass
Two-Class Averaged Perceptron
Two-Class Bayes Point Machine
Two-Class Boosted Decision Tree
Two-Class Decision Forest
Two-Class Decision Jungle
Two-Class Locally-Deep Support...
Two-Class Logistic Regression
Two-Class Neural Network
Two-Class Support Vector Machi...
Income_Prediction_Train
In draft
Draft saved at 3:06:08 PM
adult.data.csv
Split
Two-Class Boosted Decision T...
Train Model
NEW
VIEW RUN HISTORY
SAVE
SAVE AS
DISCARD CHANGES
REFRESH
CANCEL
RUN
PUBLISH WEB SERVICE

FIGURE 3-21 The Azure Machine Learning experiment with a Train Model shape connected to the Two-Class Boosted Decision Tree module.

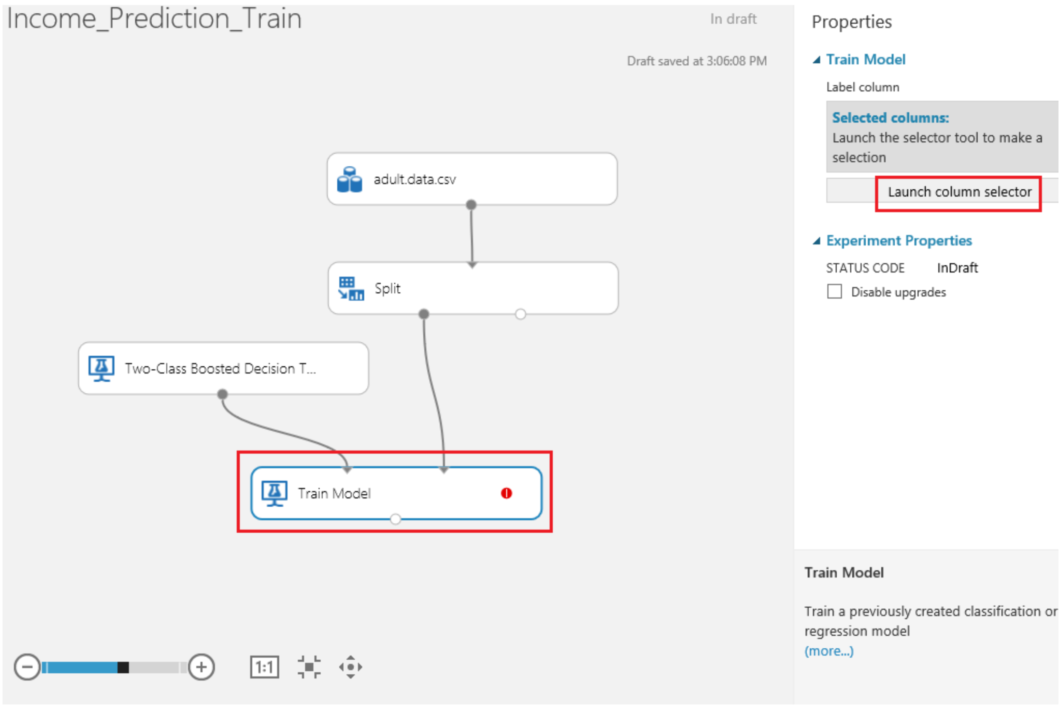
At this point, we have designed our experiment to split off 80 percent of the Adult Census dataset to be used for training our model through a Boosted Decision Tree Regression algorithm.

At this point, you might be wondering why we chose this particular algorithm to work with our prediction. Don't worry, as we cover the topic of the proper use and application suitability of various machine learning predictive algorithms in a future chapter, so for now let's use the Two-Class Boosted Decision Tree for our simple example Azure Machine Learning experiment.

Selecting the column to predict

To finish configuring the algorithm, we need to indicate which column in the dataset is the outcome or prediction column, the column that is to be predicted based on all the other columns in any particular row of the dataset.

To do this, click the Train module. When you click the module, a Properties pane will open on the right side of the Azure ML Studio screen, as shown in Figure 3-22.



Properties

Train Model

Label column

Selected columns:
Launch the selector tool to make a selection

Launch column selector

Experiment Properties

STATUS CODE	InDraft

☐ Disable upgrades

Train Model

Train a previously created classification or regression model
(more...)

FIGURE 3-22 Opening the column selector for the Train module.

Once you have set the module on the design surface, launch the column selector in the right pane. Select Iinclude and the column named income.

Figure 3-23 is a screenshot of the column selector denoting that the income column is to be predicted from the incoming dataset.

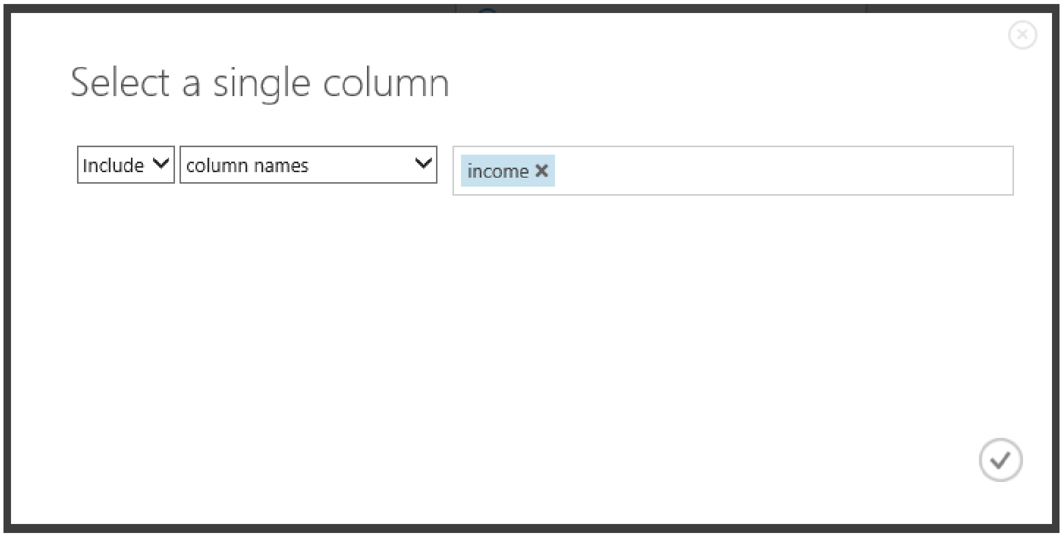

Select a single column

FIGURE 3-23 Configuring the Train module to select a single column and include the income column.

In this way, we are instructing the Azure Machine Learning algorithm to infer patterns from all the other columns in each row of the dataset, so that the income column can be predicted. We are using 80 percent of our dataset to "train" the model, based on known inputs and outputs.

At this point, we are now ready to actually start training our model. Select the RUN option at the bottom of the screen, and sit back and watch as it actually trains our model. You will notice that as each stage of our experiment completes, a green check mark will appear on the right side of each operation in our experiment, as shown in Figure 3-24.

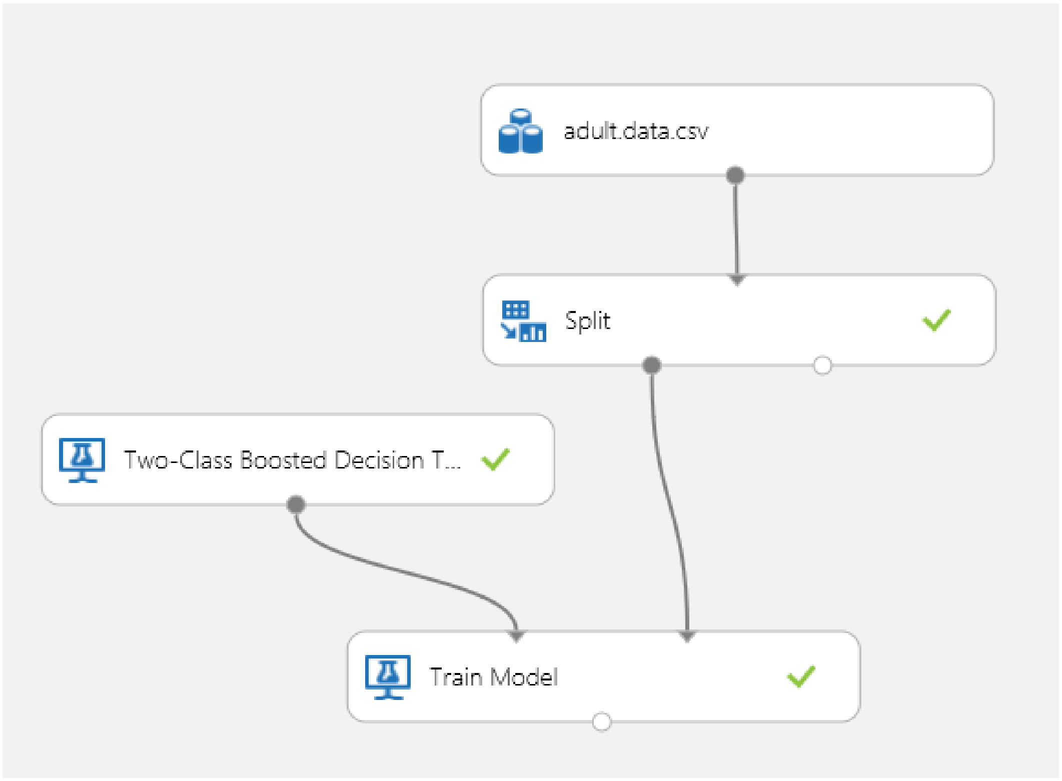


FIGURE 3-24 Training our new Azure Machine Learning income prediction model.

Score the model

Now that we have trained our new Azure Machine Learning prediction model, we can next evaluate the results to determine the model's accuracy and therefore suitability as a solution. Remember, the key to a great Azure Machine Learning solution is to develop iteratively, where the keys to success are to fail fast and fail often.

To implement the evaluation module, expand the Machine Learning module on the left side of Azure Machine Learning Studio. Then expand the Score Model submodule. Drag and drop the Score Model module onto the designer surface. Next, connect the Score Model module to the Train module. Then finally, connect it to the other half of the Split module. Essentially you are now "scoring" the accuracy of the model against the remaining 20 percent of the dataset to understand how accurate the prediction model really is (or isn't).

Next, click Run on the bottom of the page and wait for all the results to be processed (denoted by a green check mark appearing on the right side of each module). Figure 3-25 is a screenshot of our income prediction Azure Machine Learning experiment at this point.

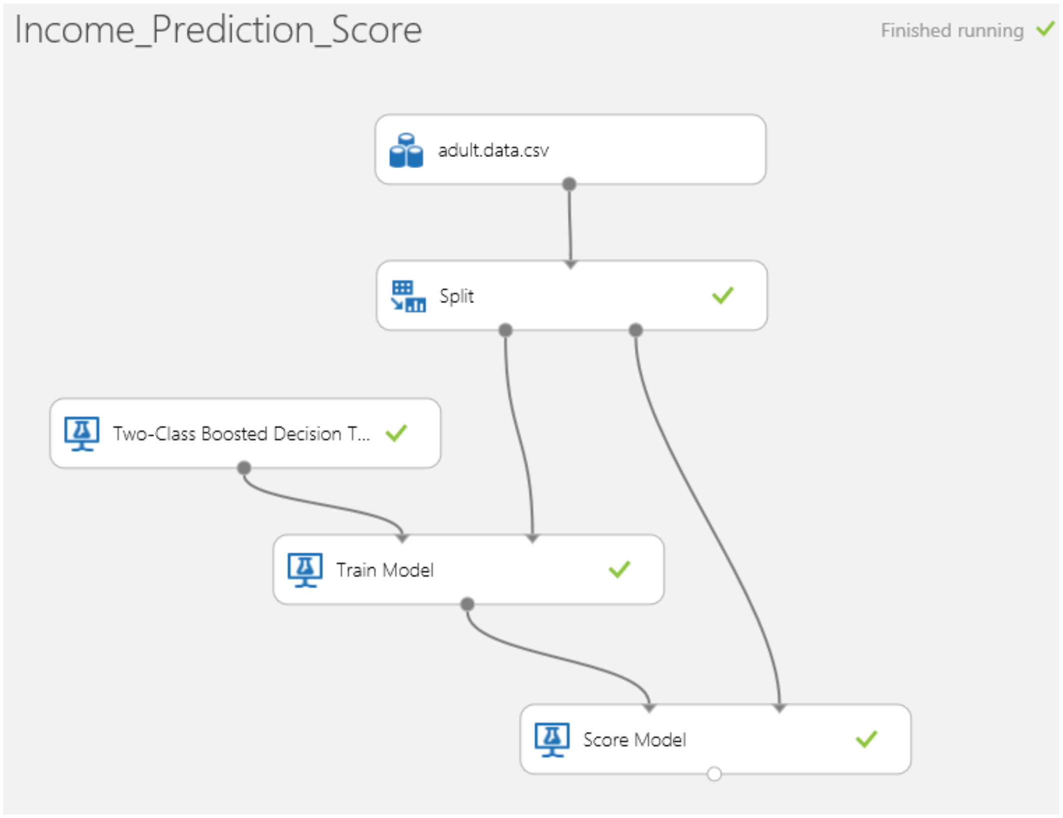


FIGURE 3-25 Azure ML Studio, training and scoring the model.

Visualize the model results

After all modules have been processed, hover your cursor over the output of the Score Model module and right-click. From the shortcut menu, select Visualize as shown in Figure 3-26.

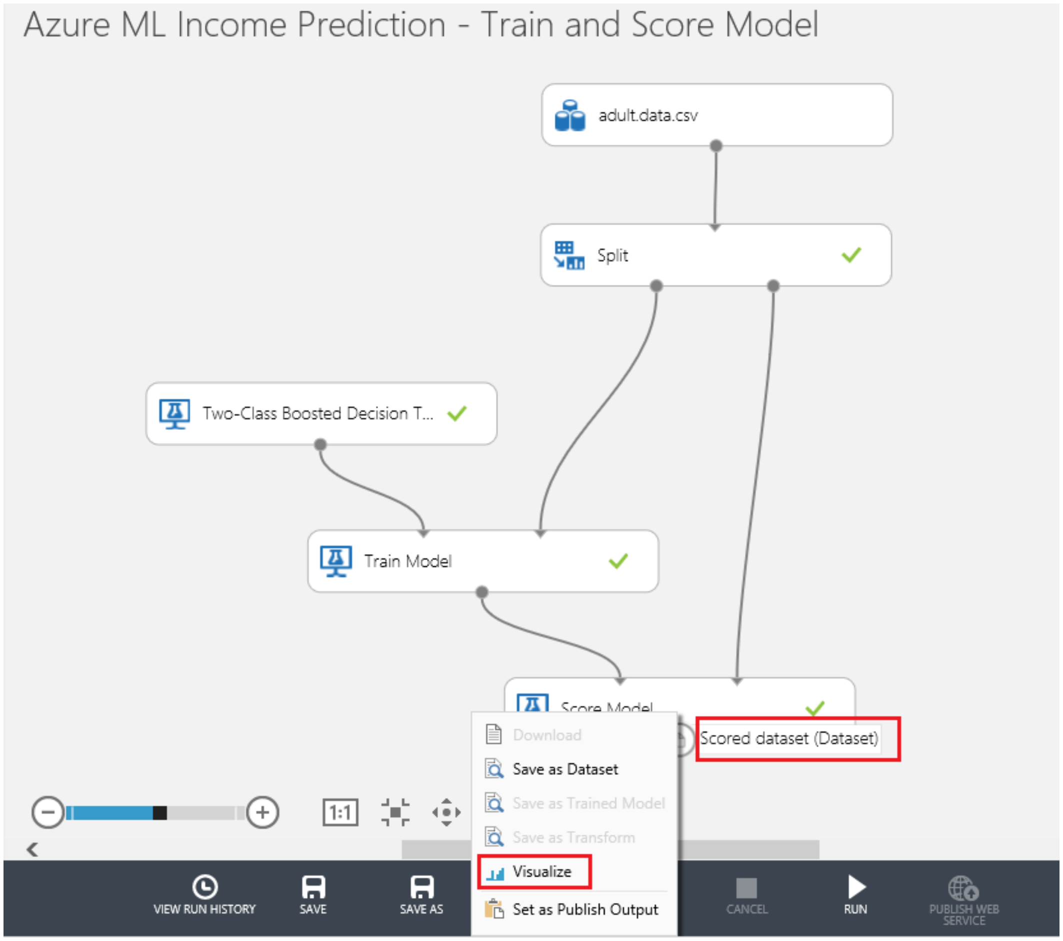
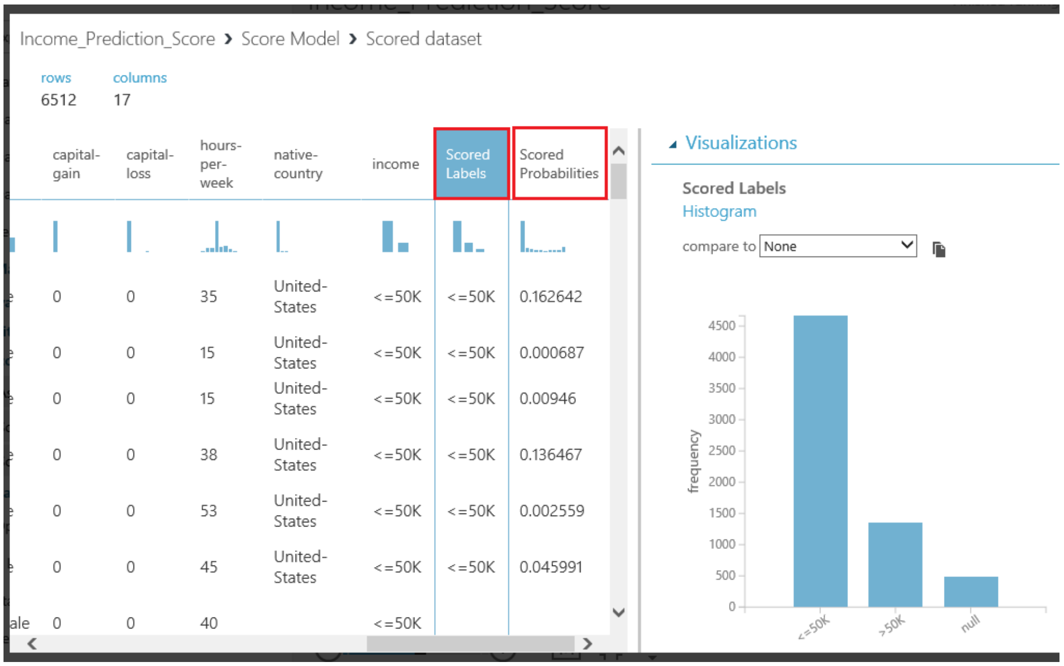


FIGURE 3-26 Visualizing the results of the scoring module in our Azure ML Studio experiment.

After you select the option to visualize the newly trained model data, a new screen is generated. Scroll all the way to the right of the visualization and you will note that there are now two additional columns that appear in the dataset, as shown in Figure 3-27.


Income_Prediction_Score › Score Model › Scored dataset

rows 6512 columns 17

capital-gain	capital-loss	hours-per-week	native-country	income	Scored Labels	Scored Probabilities
0	0	35	United-States	<=50K	<=50K	0.162642
0	0	15	United-States	<=50K	<=50K	0.000687
0	0	15	United-States	<=50K	<=50K	0.00946
0	0	38	United-States	<=50K	<=50K	0.136467
0	0	53	United-States	<=50K	<=50K	0.002559
0	0	45	United-States	<=50K	<=50K	0.045991
0	0	40		<=50K		

FIGURE 3-27 The two new columns added to each row of our trained model indicating the model's prediction and prediction probability calculation for each row.

Note that there are now two additional columns that appear in our dataset:

- **Scored Labels** This column denotes the model's prediction for this row of the dataset.
- **Scored Probabilities** This column denotes the numerical probability (or the likelihood) of whether the income level for this row exceeds $50,000.

These new columns in our dataset represent that, in addition to making a prediction calculation for each row, the algorithm can also provide a numerical probability factor. This probability factor represents the model's potential for accurately predicting each row in the dataset based on the specific values found in each of the row's other columns.

Typically, this is the point where predictive analytics becomes a very iterative process. You might want to try many different algorithms or even chain them together (in an advanced machine learning topic called an ensemble) to come up with a predictive model that proves fruitful.

Evaluate the model

One of the most compelling features of Azure Machine Learning is the ability to quickly evaluate different algorithms. Choose the right one with just a few mouse clicks, thanks to the Evaluate module. To see how accurate our model really is, we can easily evaluate our model using a built-in Evaluate module in Azure ML Studio.

To do this, click the Machine Learning module in the left navigation pane of Azure ML Studio. Select the Evaluate submodule. Finally, select the Evaluate Model module and drag it onto the visual designer surface, below the Score Model module. Connect the other half of the Split module to the Score Model module. Then connect the Score Model module to the left side of the Evaluate Model module, as shown in Figure 3-28.

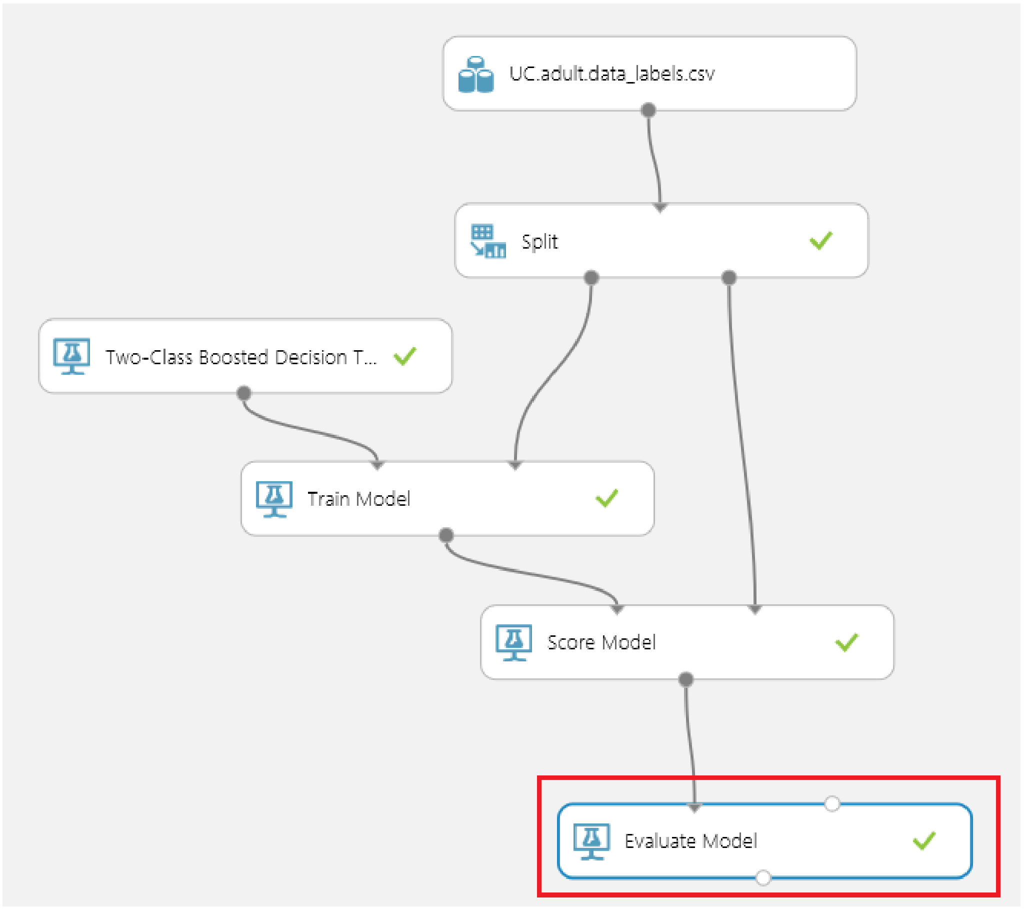


FIGURE 3-28 Connecting the Evaluate Model module to evaluate the results of the income prediction module.

Click Run at the bottom of the Azure ML Studio screen. Watch as each stage is processed and marked complete, as denoted by a green check mark on the right of each module in our experiment.

After the processing has completed, right-click the bottom connector of the Evaluate Model module. Select Visualize from the shortcut menu to generate the evaluation results screen shown in Figure 3-29.

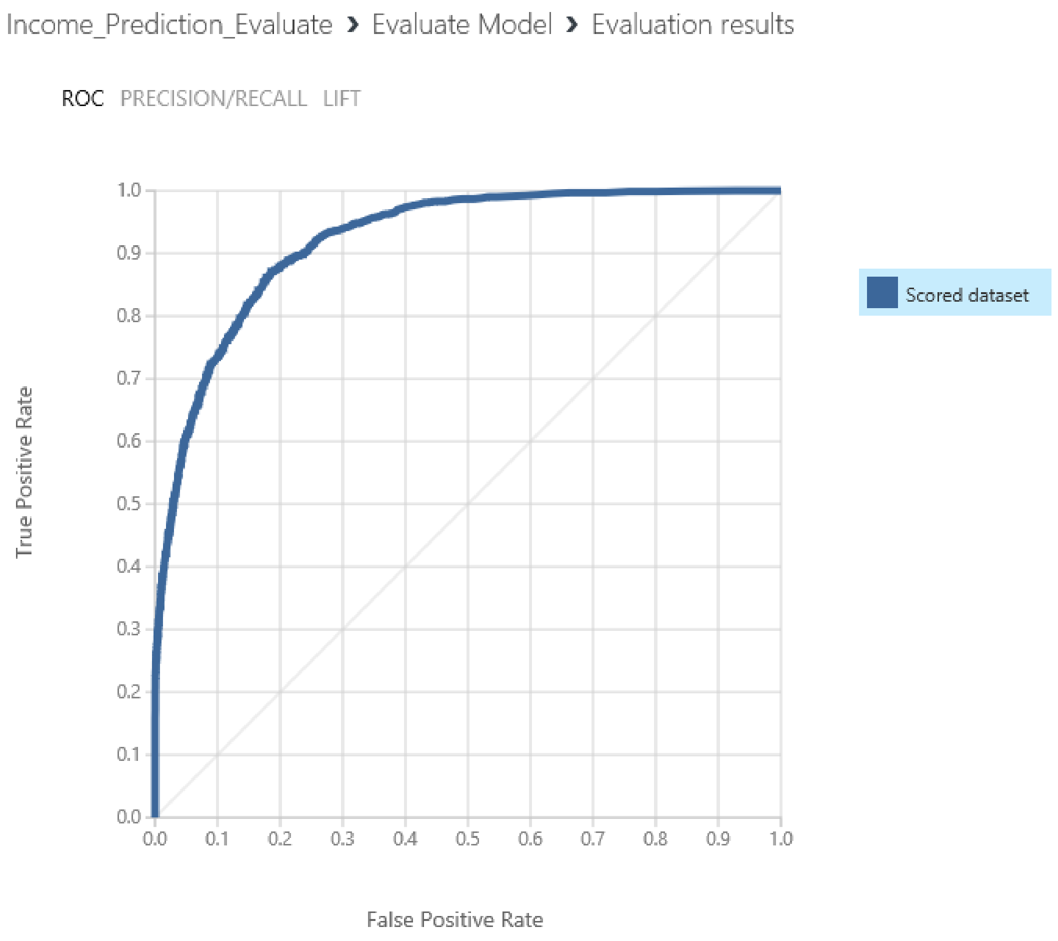


FIGURE 3-29 Visualizing the Azure Machine Learning Evaluation Model module results for the income prediction model.

The Evaluate Model module produces a set of curves and metrics that allow you to view the results of a scored model or compare the results of two scored models. You can view the results in the following three formats:

- **Receiver Operator Characteristic (ROC) curves** This format displays the fraction of true positives out of the total actual positives. It contrasts this with the fraction of false positives out of the total negatives, at various threshold settings. The diagonal line represents 50 percent accuracy in your predictions and can be used as a benchmark that you want to improve. The higher and further to the left, the more accurate the model is. As you do experiments you want to see the curve move higher and to the left.
- **Precision/Recall curves** Precision represents the fraction of retrieved instances that are relevant, whereas recall represents the fraction of relevant instances that are retrieved.
- **Lift curves** This format is a variation on the ROC curve. It measures the fraction of true positives, in relation to the target response probability.

In the visualization in Figure 3-29, you can see that the results of both datasets (our "trained" dataset and our "validation" dataset) are nearly identical, with the blue and red lines almost exactly on top of each other. This would indicate that we have a reasonably accurate prediction model. Consequently, for the purposes of this initial walk-through of Azure Machine Learning, we assume that the predictive model is reasonably accurate and take it to the next phase.

Save the experiment

At this point, you want to save a copy of your experiment. Click Save As at the bottom of the screen. We are about to make major changes to our experiment that will alter the core functionality from being a training experiment to an operational experiment. Save your experiment using a descriptive name such as Azure ML Income Prediction – Train Model Experiment.

Next, click Save As at the bottom of the screen to resave the experiment before implementing the next phase using a different name such as Azure ML Income Prediction – Implement Model.

Preparing the trained model for publishing as a web service

Now that we have a working "trained" predictive model that produces reasonable results, the next step is to run the option in the bottom navigation pane called Prepare Web Service. Notice that before these steps, this option was unavailable as a valid command to execute.

Before we execute the next command, now is a good time to do a click Save As and save the experiment under a new name. The reason for doing this now is that when we execute the Prepare Web Service command, it will make a few modifications to our experiment to make things ready for publishing it as a web service. Figure 3-30 shows our experiment before we execute the Prepare Web Service command.

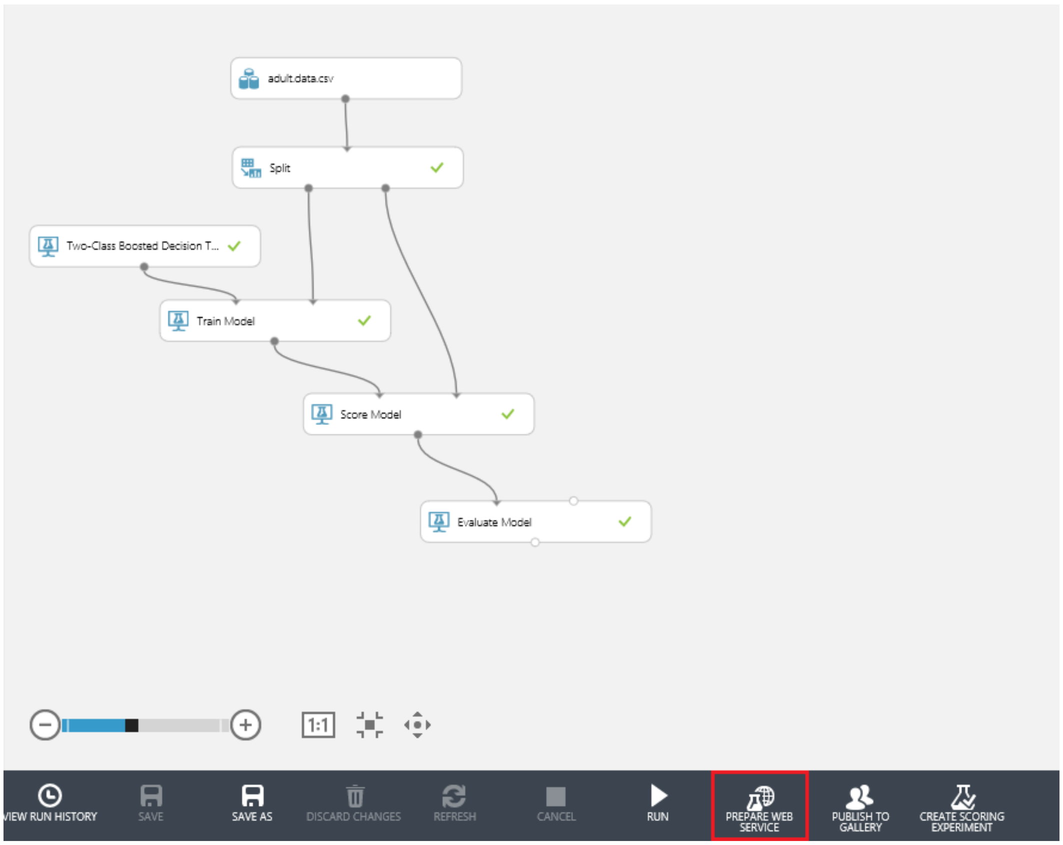


FIGURE 3-30 Ready to run the Prepare Web Service command.

The next step is to execute the Prepare Web Service command and watch as Azure Machine Learning Studio automates the following tasks for you:

- Adds a web service Input module.
- Adds a web service Output module.
- Saves the trained model in our experiment repository of trained models.

After the experiment has been modified by running the Prepare Web Service command, the experiment will show the two new modules for web service input and web service output as depicted in Figure 3-31.

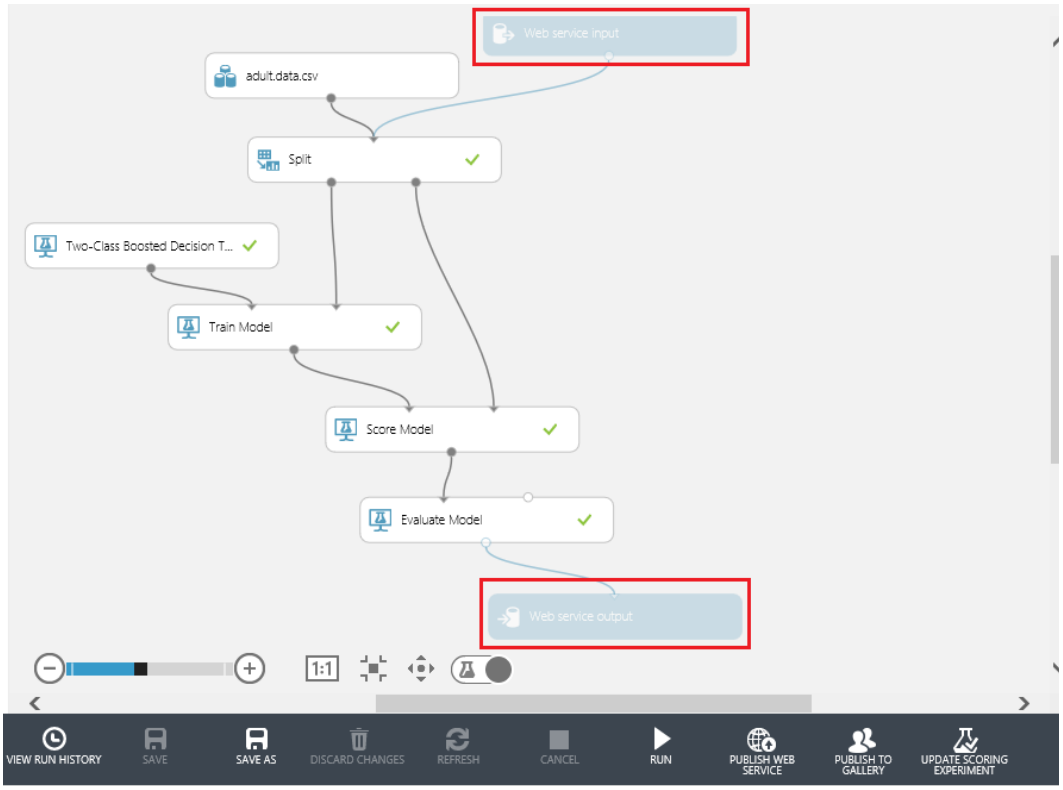


FIGURE 3-31 After the Prepare Web Service command has been run.

Note the process has saved our newly trained model and populated it under the Trained Models navigation bar as shown in Figure 3-32.

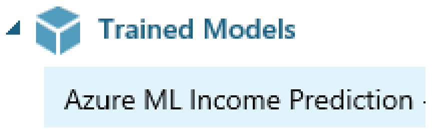


FIGURE 3-32 The newly populated trained model for Azure ML Income Prediction.

One of the other modifications that was made to our experiment was the addition of a new command above the bottom navigation bar to toggle the ability to switch from an experiment view to a web services view. Figure 3-33 shows the new Web Service toggle command option after we have run the Prepare Web Service command.

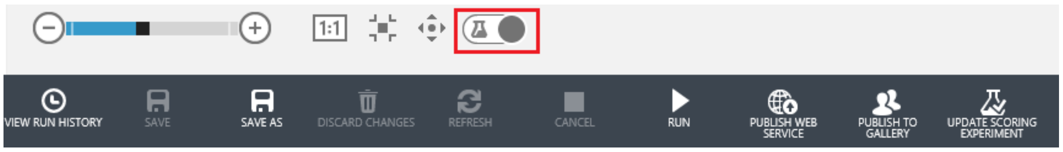


FIGURE 3-33 The new Web Service toggle button.

By clicking the Web Service toggle button, you can switch between an Experiment view and a Web Service view. Figure 3-31 depicted an Experiment view where the web service input and output modules were unavailable. Figure 3-34 depicts the Web Service view where the web service input and output modules are displayed in solid blue and fully enabled.

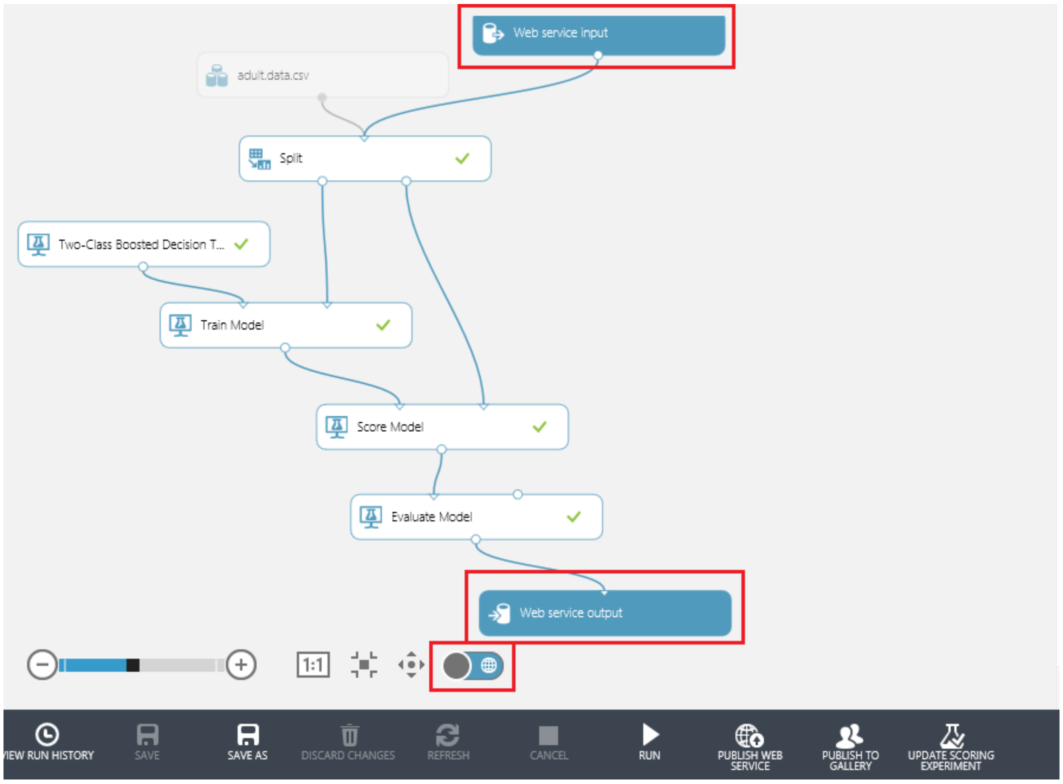


FIGURE 3-34 The Web Service view after clicking the Web Service toggle button.

Note that in Figure 3-34, the input dataset, Adult.data.csv, is unavailable, which means that the input for this experiment will come from the Web Service Input module. Additionally, the connector lines depicting the processing flow also turn blue to illustrate the logic paths taken when the experiment is processed via a web service call.

If we click the toggle button again, the experiment switches back to the Experiment view, as shown in Figure 3-35.

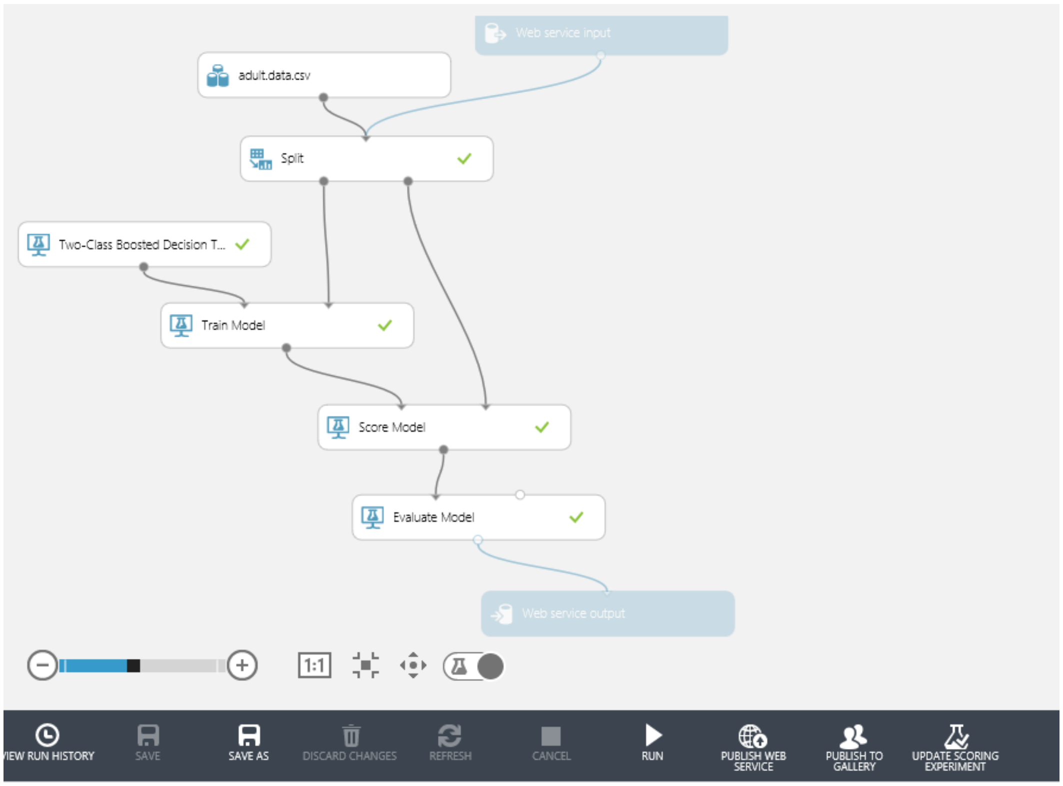


FIGURE 3-35 The Experiment view after clicking the Web Service toggle button again.

Create scoring experiment

The next step in the process is to transform our experiment from training mode to operational mode. Because we have already trained our model, there is no need to keep training it as we move toward publishing it as a web service. For maximum efficiency, we only need to retrain our model periodically, for example, whenever new data is available, or perhaps when seasonal datasets are desired as influencers on the model depending on the time of year.

In the previous step, the Azure ML Studio added our trained model to the Trained Models repository. But notice that the experiment still contains the modules for training the model such as the Train Model and Two-Class Boosted Decision Tree algorithm. Additionally, there is no need to have the Evaluate module in our experiment as we have already gone through the evaluation phase of our workflow.

The good news is that Azure ML Studio contains a built-in command that will automatically take care of optimizing our experiment for use as a streamlined web service. That command, Create Scoring Experiment is located on the far right of the bottom navigation bar as seen in Figure 3-36.

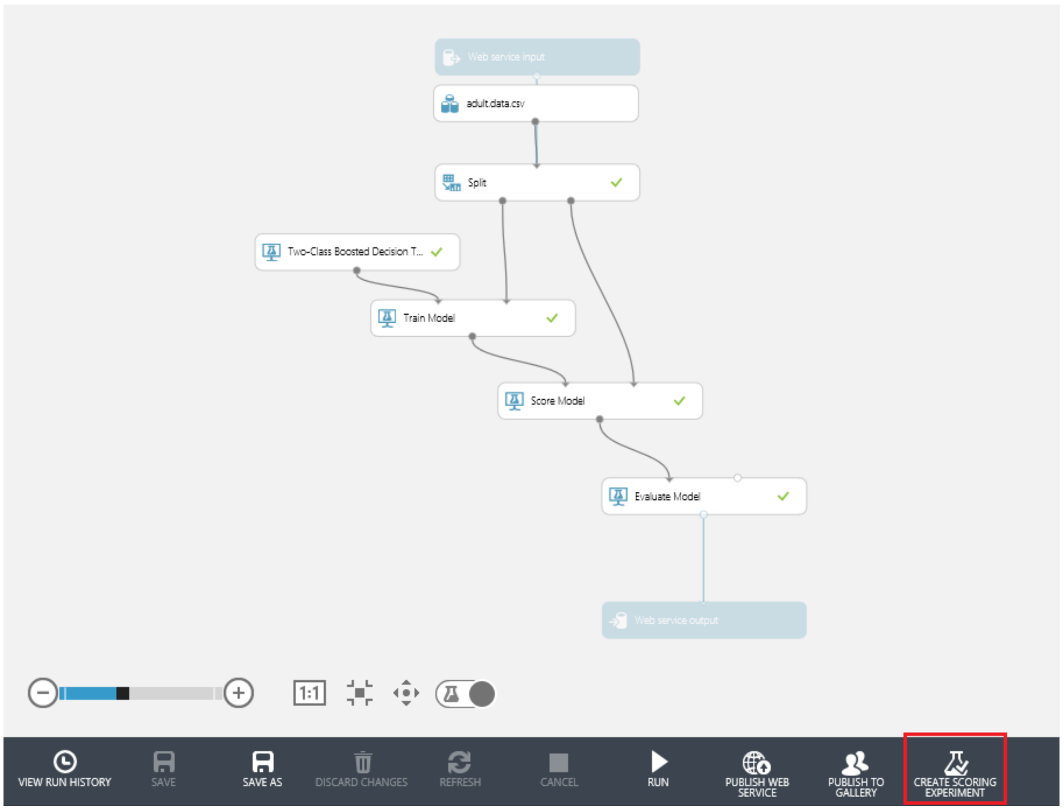


FIGURE 3-36 The Create Scoring Experiment command.

Before we run the Create Scoring Experiment command, now is great time to perform a Save As command and save a copy of our experiment as it still contains all the training, scoring, and evaluation modules. The reason for doing this now is that the next command we run will dramatically alter the experiment as it is prepared for publishing as a web service.

Now it is time to run the Create Scoring Experiment command and watch as Azure ML Studio works its magic on the experiment.

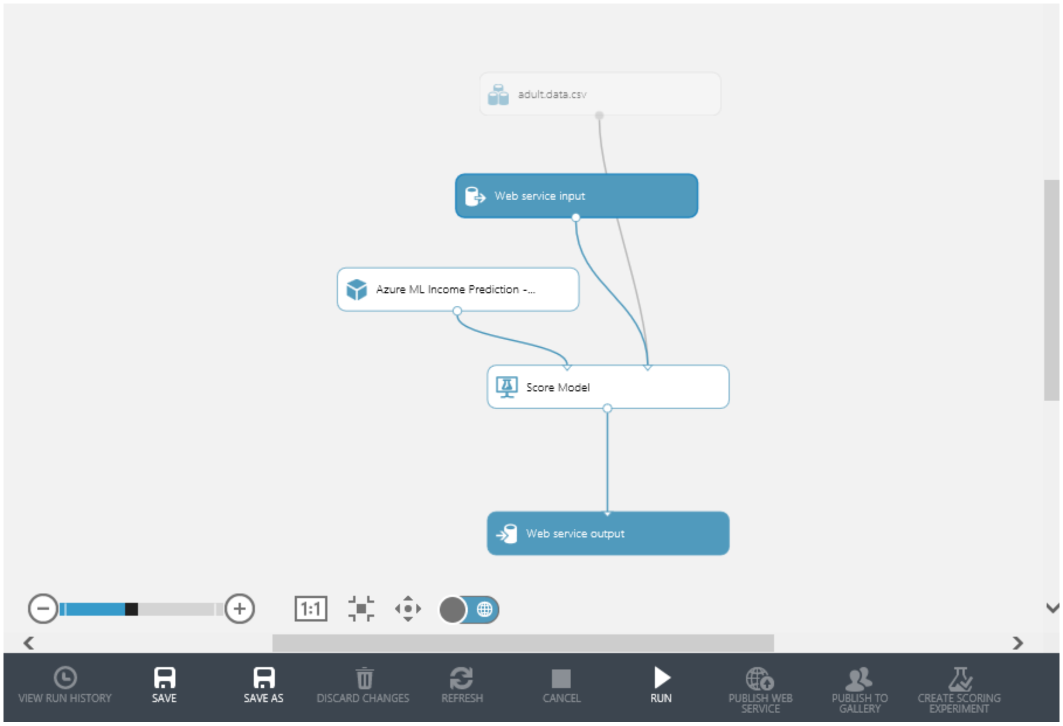


FIGURE 3-37 The Azure Machine Learning experiment after running the Create Scoring Experiment command.

After executing the Create Scoring Experiment command, the following changes were automatically made to our Azure Machine Learning experiment:

- The Split module has been removed.
- The Two-Class Boosted Decision Tree module and the Train module have been removed and replaced with the previously trained model from the Trained Models repository.
- The Evaluate module has been removed.

This process has streamlined our Azure Machine Learning experiment by optimizing the logic flow between the web service input and output endpoints. The only internal processing steps are the Score module and the previously trained and saved Azure ML Income Prediction module. We have successfully transformed our Azure Machine Learning experiment from training mode to operational mode. We are now ready to publish our experiment as a web service.

Expose the model as a web service

The next step is to actually expose our new Azure Machine Learning predictive model as a web service.

To enable web service publishing, first run the new income prediction experiment by clicking Run in the bottom navigation bar. Once the experiment has been run, the Publish Web Service link in the bottom navigation bar will be enabled, as shown in Figure 3-38.

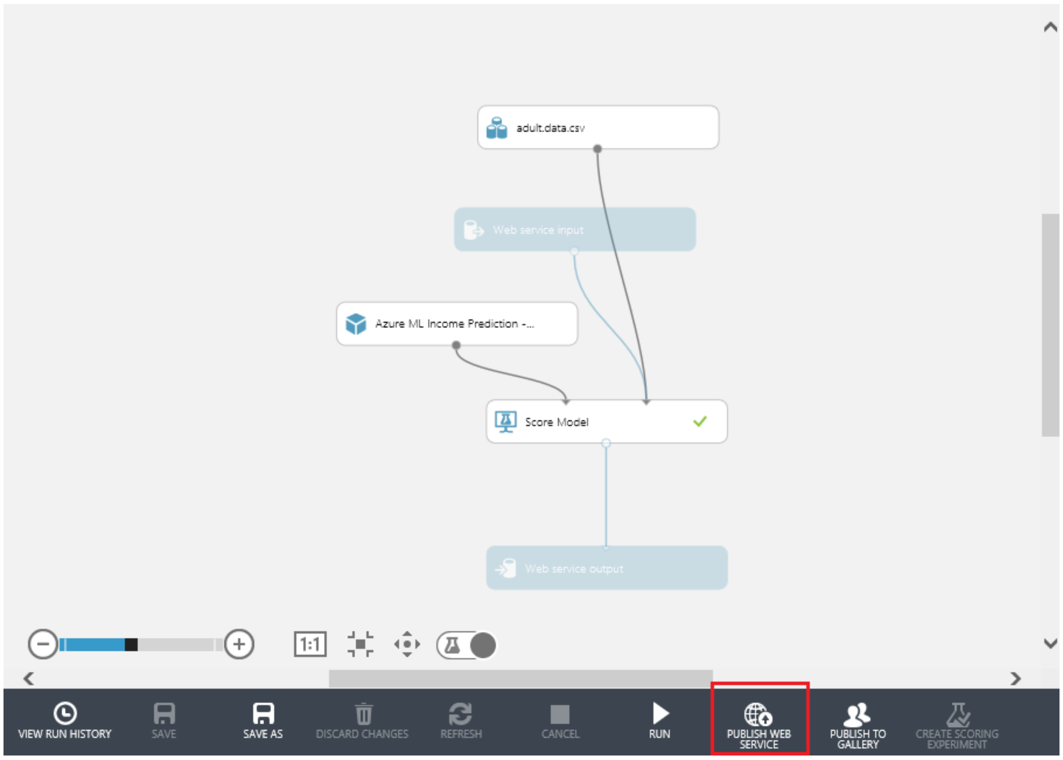


FIGURE 3-38 The Azure Machine Learning experiment ready to publish as a web service.

Now we are almost done. Just click Publish Web Service on the bottom navigation bar of the designer screen and you will be asked if you would like to publish your web service. Just select OK. Figure 3-39 shows the model being published.

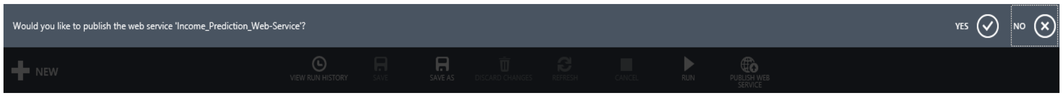
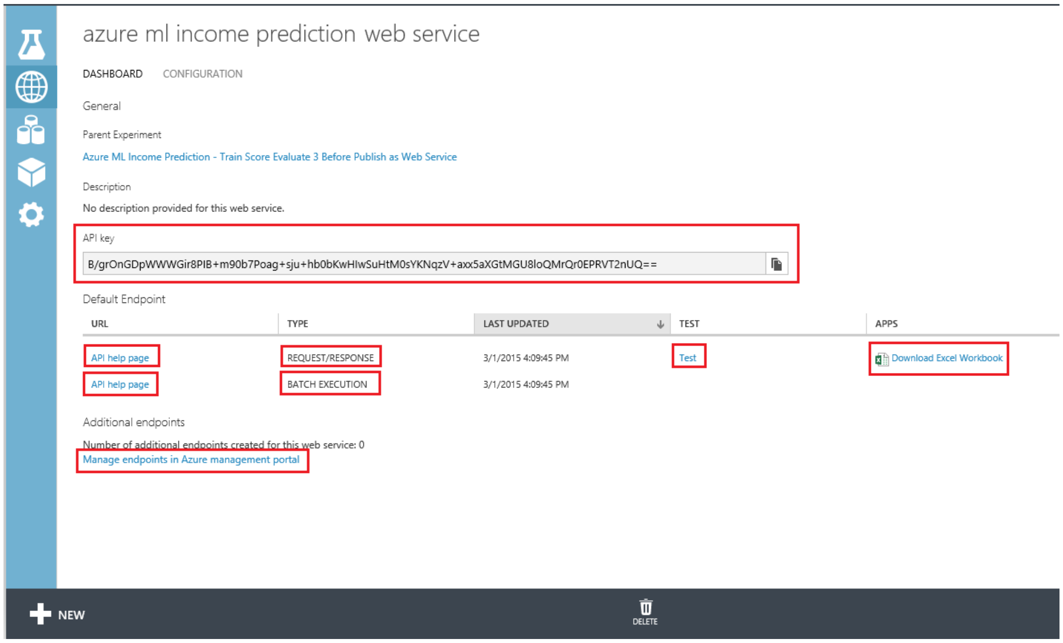

Would you like to publish the web service 'Income_Prediction_Web-Service'?

FIGURE 3-39 Publishing the web service.

After a few seconds, a dashboard will appear providing the API key and API help links for your new predictive model web service. Figure 3-40 shows the web service dashboard.


azure ml income prediction web service

DASHBOARD   CONFIGURATION

General

Parent Experiment

Azure ML Income Prediction - Train Score Evaluate 3 Before Publish as Web Service

Description

No description provided for this web service.

API key

B/grOnGDpWWWGir8PIB+m90b7Poag+sju+hb0bKwHIwSuHtM0sYKNqzV+axx5aXGtMGU8loQMrQr0EPRVT2nUQ==

Default Endpoint

URL	TYPE	LAST UPDATED	TEST	APPS
API help page	REQUEST/RESPONSE	3/1/2015 4:09:45 PM	Test	Download Excel Workbook
API help page	BATCH EXECUTION	3/1/2015 4:09:45 PM		

Additional endpoints

Number of additional endpoints created for this web service: 0

Manage endpoints in Azure management portal

FIGURE 3-40   The Azure Machine Learning web services dashboard.

The Azure ML Studio dashboard provides all the information necessary to invoke our new prediction model over the Web. The webpage lists the following items to help get you started:

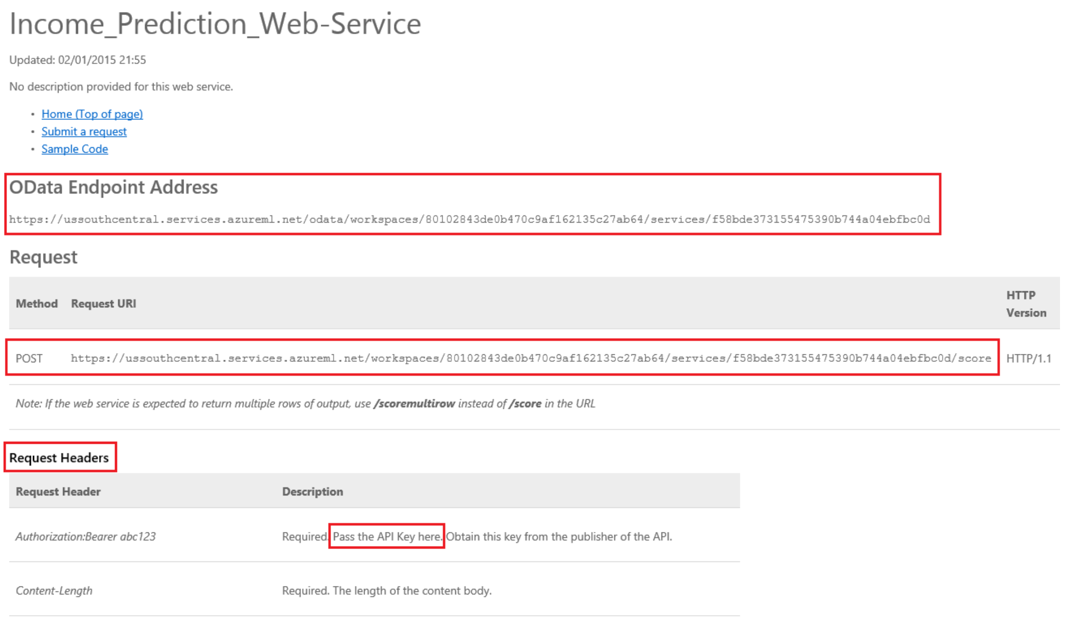
- The API key is the unique security identifier that must be passed on every web service request to authenticate the caller.
- The API Help Page link for Request/Response displays a series of API usage information that provides guidance for how to call the Azure Machine Learning web service for an individual prediction based on a single input record.
- The API Help Page link for Batch Execution displays a series of API usage information that provides guidance for how to call the Azure Machine Learning web service for one or more input records.
- The Download Excel Workbook option allows you to download an Excel workbook that contains the following useful information about your new Azure Machine Learning web service:
  - **WEB SERVICE URL**   The HTTP endpoint address for invoking the web service.
  - **ACCESS KEY**   The API key as denoted earlier.
  - **SCHEMA**   An HTTP URL that will provide information about all the input and output parameters for your new Azure Machine Learning web service, including all the parameter

names and their corresponding data types.

The Excel Workbook also contains macros that will allow you to invoke your new Azure Machine Learning web service directly from within Excel. Simply enter the appropriate values for each input field in the Parameters section of the spreadsheet. The macros will be triggered based on the worksheet values changing and will invoke the Azure Machine Learning web service and return the results in the Predicted Values section of the spreadsheet. This capability makes it very convenient to test various iterations of your input values and get almost instant feedback on the results. You can easily keep adding rows to the spreadsheet and vary the input parameters to see the prediction results.

- The Manage Endpoints In Azure Management Portal link takes you to another webpage where you can manage the endpoints for this web service.

Start by clicking on API Help Page link for the Request/Response web service call. Figure 3-41 is a screenshot of what sample API usage display.


Income_Prediction_Web-Service

Updated: 02/01/2015 21:55

No description provided for this web service.

- Home (Top of page)
- Submit a request
- Sample Code

OData Endpoint Address

```
https://ussouthcentral.services.azureml.net/odata/workspaces/80102843de0b470c9af162135c27ab64/services/f58bde373155475390b744a04ebfbc0d
```

Request

Method	Request URI	HTTP Version
POST	https://ussouthcentral.services.azureml.net/workspaces/80102843de0b470c9af162135c27ab64/services/f58bde373155475390b744a04ebfbc0d/score	HTTP/1.1

*Note: If the web service is expected to return multiple rows of output, use **/scoremultirow** instead of **/score** in the URL*

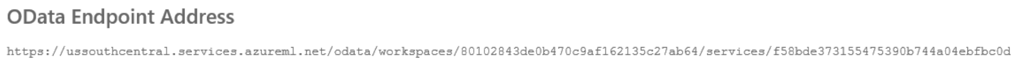
Request Headers

Request Header	Description
Authorization:Bearer abc123	Required. Pass the API Key here. Obtain this key from the publisher of the API.
Content-Length	Required. The length of the content body.

FIGURE 3-41 The API help page for a Request/Response web service call.

There is a wealth of information presented on this screen. Let's take a look at each section of the API usage webpage.

- **OData Endpoint Address** OData is a Web-based protocol for querying and updating data and exposing the data using a standardized syntax (see Figure 3-42). OData leverages technologies such as HTTP, XML, JavaScript Object Notation (JSON), and the Atom Publishing Protocol to provide access to data over the Web.

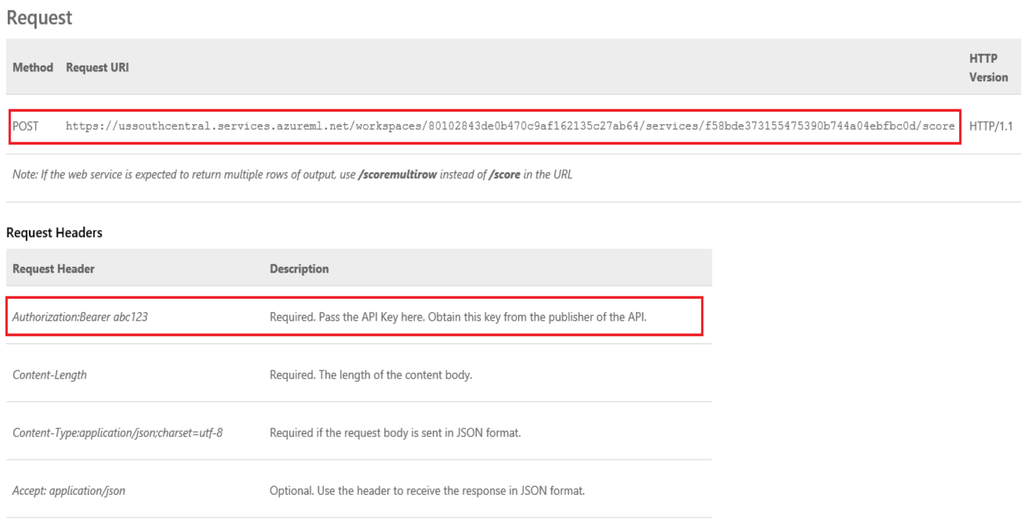


OData Endpoint Address

https://ussouthcentral.services.azureml.net/odata/workspaces/80102843de0b470c9af162135c27ab64/services/f58bde373155475390b744a04ebfbc0d

FIGURE 3-42 The Azure Machine Learning web service OData address for an experiment.

- **Request Headers** Figure 3-43 displays the information about how to make a web HTTP POST Request and the associated request headers that must be populated. The request headers contain information about the content and acceptable datatypes in the request so that server returns compatible data. Note that the ***AuthorizationBearer*** field is required in the Request header. This is where you would normally pass the API Key from our new web service to authorize the caller.



Request

Method	Request URI	HTTP Version
POST	https://ussouthcentral.services.azureml.net/workspaces/80102843de0b470c9af162135c27ab64/services/f58bde373155475390b744a04ebfbc0d/score	HTTP/1.1

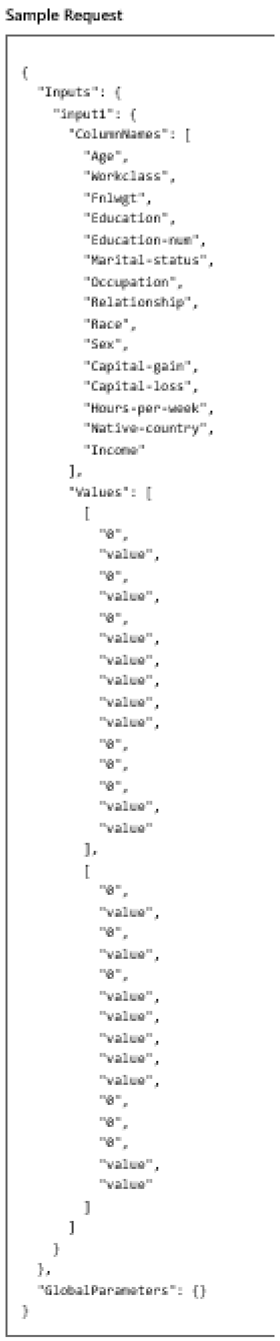
*Note: If the web service is expected to return multiple rows of output, use **/scoremultirow** instead of **/score** in the URL*

Request Headers

Request Header	Description
Authorization:Bearer abc123	Required. Pass the API Key here. Obtain this key from the publisher of the API.
Content-Length	Required. The length of the content body.
Content-Type:application/json;charset=utf-8	Required if the request body is sent in JSON format.
Accept: application/json	Optional. Use the header to receive the response in JSON format.

FIGURE 3-43 The Azure Machine Learning web service sample POST web request with required request headers.

- **Request Body** The next section of the sample API Request page displays how to construct a sample HTTP Request Body. Figure 3-44 shows the sample request body.



Sample Request

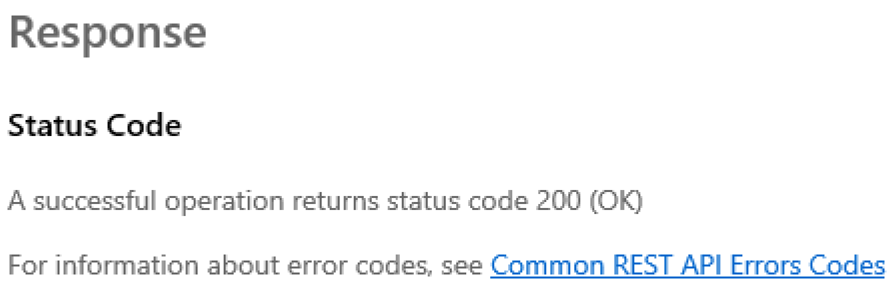
```
{
  "Inputs": {
    "input1": {
      "ColumnNames": [
        "Age",
        "Workclass",
        "Fnlwgt",
        "Education",
        "Education-num",
        "Marital-status",
        "Occupation",
        "Relationship",
        "Race",
        "Sex",
        "Capital-gain",
        "Capital-loss",
        "Hours-per-week",
        "Native-country",
        "Income"
      ],
      "Values": [
        [
          "0",
          "value",
          "0",
          "value",
          "0",
          "value",
          "value",
          "value",
          "value",
          "value",
          "0",
          "0",
          "0",
          "value",
          "value"
        ],
        [
          "0",
          "value",
          "0",
          "value",
          "0",
          "value",
          "value",
          "value",
          "value",
          "value",
          "0",
          "0",
          "0",
          "value",
          "value"
        ]
      ]
    }
  },
  "GlobalParameters": {}
}
```

FIGURE 3-44 The sample HTTP Request Body data structure.

Note that the sample Request Body data structure contains all of the columns of our Adult.data.csv dataset, except for the income column. The income column has been excluded (via the Project Columns module in our experiment) because that is the value that we will be predicting with our web service call.

The Request Body sample represents the exact input format, data columns, and values that must be passed into the Azure Machine Learning web service to produce a valid response. The Request Body is in the JSON format, which is a common and popular web data-interchange format in use today. JSON is easy for humans to read and write and for machines to parse and generate. The sample Request Body will be an extremely useful guide for constructing our client applications to successfully consume the web service.

- **Response** The next section contains information about HTTP response codes that could be returned by the Azure Machine Learning web service. A successful web service call should always return an HTTP status code 200-OK. If your web service call is not successful, a link is provided to a webpage that contains some of the common REST API error codes to help identify the type and cause of the error. Figure 3-45 shows the Response Status Code information.



Response

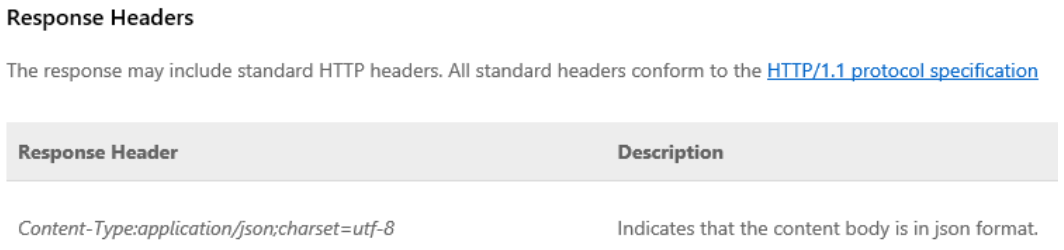
Status Code

A successful operation returns status code 200 (OK)

For information about error codes, see Common REST API Errors Codes

FIGURE 3-45 The HTTP Response Status Code section of the sample Azure Machine Learning web service call.

- **Response Headers** This section contains information that the Azure Machine Learning web service server is sending back to the client. In this case, the server is sending information about the content type of the response body, which is in JSON format. Figure 3-46 provides this information.



Response Headers

The response may include standard HTTP headers. All standard headers conform to the HTTP/1.1 protocol specification

Response Header	Description
Content-Type:application/json;charset=utf-8	Indicates that the content body is in json format.

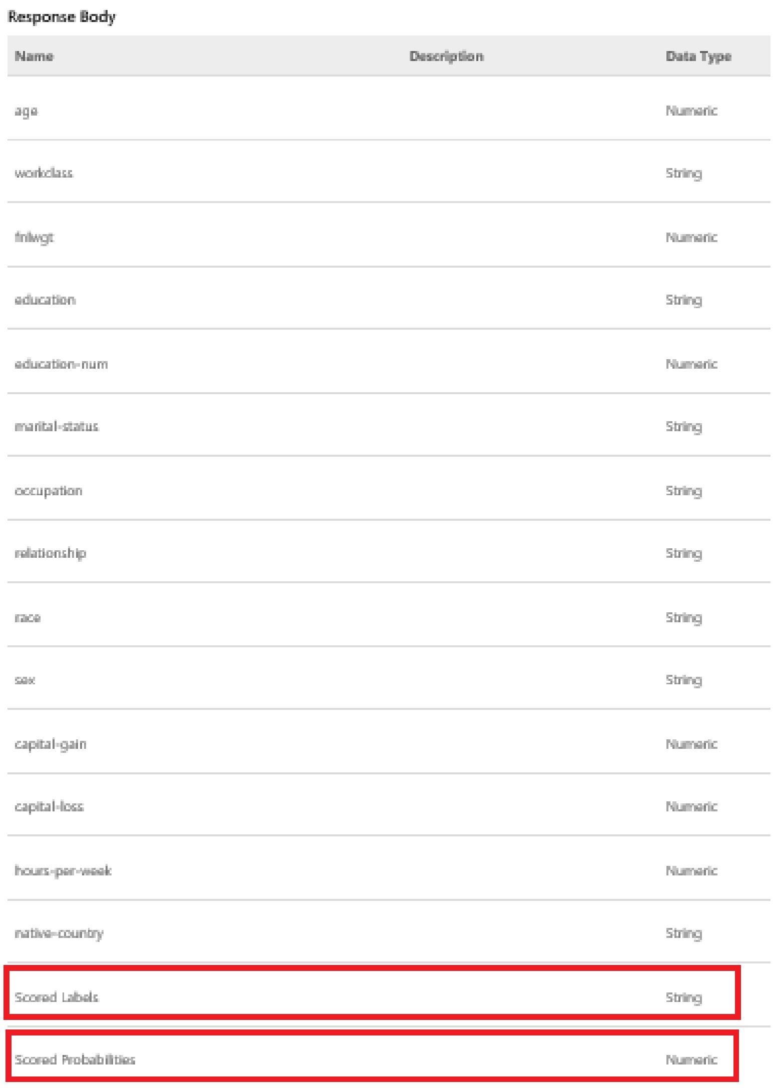
FIGURE 3-46 HTTP Response Header information returned from an Azure MLS web service call.

- **Response Body** This section contains information about the Response Body returned from our Azure Machine Learning web service call. Note that it will return data elements for each one of the columns in our Adult.data.csv dataset (except the income column) along with their associated data types (numeric or string).

  Most important, there are two additional fields appended to the payload that is returned from our Azure Machine Learning web service call.

  - **Scored Labels** This field denotes the Azure Machine Learning model's prediction calculation for this row of the dataset; that is, whether the predicted income level is <=50K or >50K.
  - **Scored Probabilitie**s This field denotes the numerical probability of the likelihood of whether the income level for this row exceeds $50,000.

  Figure 3-47 shows the sample Response Body.

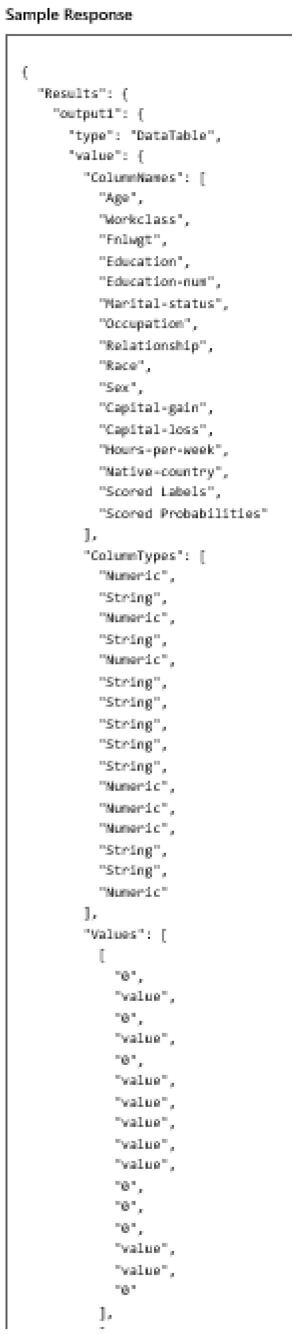


Response Body

Name	Description	Data Type
age		Numeric
workclass		String
fnlwgt		Numeric
education		String
education-num		Numeric
marital-status		String
occupation		String
relationship		String
race		String
sex		String
capital-gain		Numeric
capital-loss		Numeric
hours-per-week		Numeric
native-country		String
Scored Labels		String
Scored Probabilities		Numeric

FIGURE 3-47 A sample Response Body showing the original dataset columns along with two additional fields containing the returned prediction result fields.

- **Sample Response** This section displays the JSON format of the web service Response Body. It will consist of a JSON record (denoted by braces), made up of the Data Table definition ("DataTable"), a list of the column names in the data table ("ColumnNames"), the datatype for each column ("ColumnTypes"), and set of return records ("Values") containing a comma-separated list for each value. Figure 3-48 shows a sample response from the API webpage.

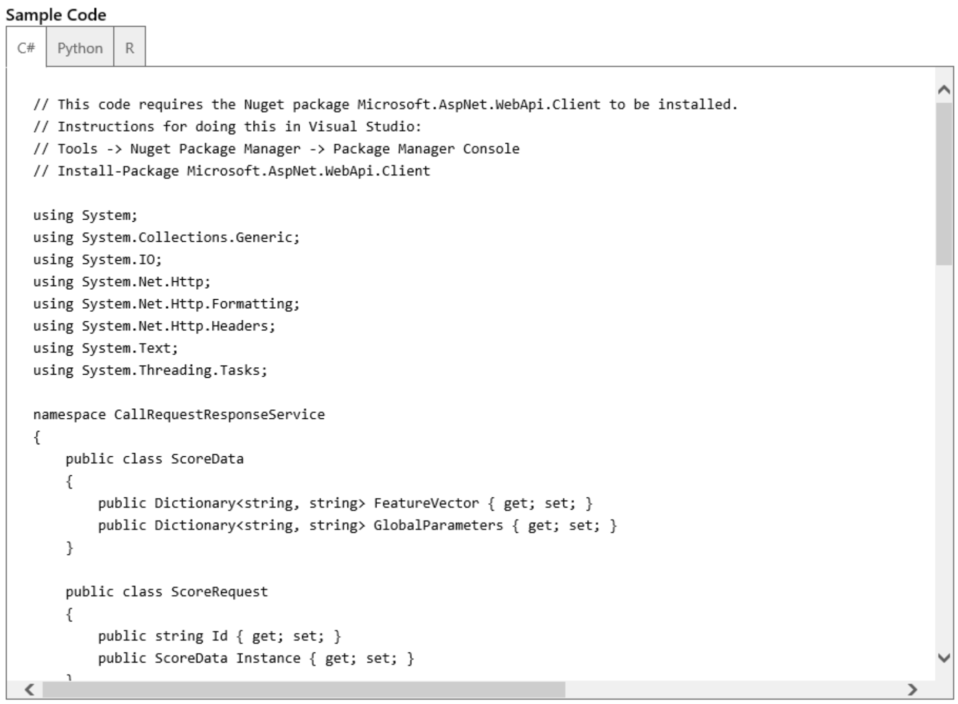


Sample Response

```
{
  "Results": {
    "output1": {
      "type": "DataTable",
      "value": {
        "ColumnNames": [
          "Age",
          "Workclass",
          "Fnlwgt",
          "Education",
          "Education-num",
          "Marital-status",
          "Occupation",
          "Relationship",
          "Race",
          "Sex",
          "Capital-gain",
          "Capital-loss",
          "Hours-per-week",
          "Native-country",
          "Scored Labels",
          "Scored Probabilities"
        ],
        "ColumnTypes": [
          "Numeric",
          "String",
          "Numeric",
          "String",
          "Numeric",
          "String",
          "String",
          "String",
          "String",
          "String",
          "Numeric",
          "Numeric",
          "Numeric",
          "String",
          "String",
          "Numeric"
        ],
        "Values": [
          [
            "0",
            "value",
            "0",
            "value",
            "0",
            "value",
            "value",
            "value",
            "value",
            "value",
            "0",
            "0",
            "0",
            "value",
            "value",
            "0"
          ],
```

FIGURE 3-48 A subsection of the sample Response Body format.

- **Sample Code** Here's the best part: The Azure Machine Learning API webpage even provides sample code provided in C#, Python, or the R programming languages. This is extremely convenient for creating quick client testing applications. Just insert your API key value and populate the input data with valid values from the input dataset. Once those minor inputs are added, you will have a complete working client application that can call the Azure Machine Learning predictive analytics web service! See Figure 3-49.



```
// This code requires the Nuget package Microsoft.AspNet.WebApi.Client to be installed.
// Instructions for doing this in Visual Studio:
// Tools -> Nuget Package Manager -> Package Manager Console
// Install-Package Microsoft.AspNet.WebApi.Client

using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;

namespace CallRequestResponseService
{
    public class ScoreData
    {
        public Dictionary<string, string> FeatureVector { get; set; }
        public Dictionary<string, string> GlobalParameters { get; set; }
    }

    public class ScoreRequest
    {
        public string Id { get; set; }
        public ScoreData Instance { get; set; }
```

FIGURE 3-49 Sample code section for the Azure Machine Learning sample webpage.

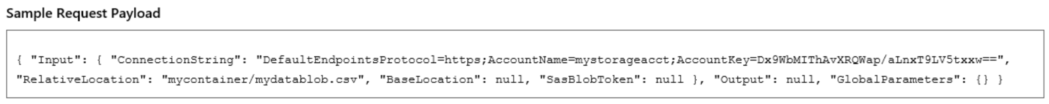
Azure Machine Learning web service BATCH execution

The API for Azure Machine Learning web services call for BATCH processing is similar to the Request/Response API web service call, but, as the name implies, it is meant for passing along a reference to a file location in Azure Storage Blob (Binary Large Object) service and then monitoring the batch execution process until it has completed.

The Azure Machine Learning BATCH web service processes predictions according to the following high-level logic:

1. Create a local file CSV that contains multiple rows of input data for the prediction model inputs.
2. Upload the file to an Azure blob (you will need an Azure Storage account).
3. Call the Batch Execution web service to process the input data file in the blob.
4. The web service prediction results get written to another Azure blob output file.
5. Download the output blob to a local file.

Figure 3-50 shows a sample BATCH web service request payload. Note the location of the input file is passed as a reference in Azure Storage Blob service.


Sample Request Payload

```
{ "Input": { "ConnectionString": "DefaultEndpointsProtocol=https;AccountName=mystorageacct;AccountKey=Dx9WbMIThAvXRQWap/aLnxT9LV5txxw==",
"RelativeLocation": "mycontainer/mydatablob.csv", "BaseLocation": null, "SasBlobToken": null }, "Output": null, "GlobalParameters": {} }
```

FIGURE 3-50 A sample Azure Machine Learning BATCH web service call payload.

After successful invocation of an initial batch web service API call, a Job ID parameter is returned to the caller.

While the batch job is processing, you can issue requests to the Azure Machine Learning web service using the unique Job ID to request the status of the executing web service batch job. Note that it is also possible to issue a Delete command using the Job ID parameter.

If you select the API Help Page link for Batch Execution you will get a similar webpage that will display information about how to perform the following operations to process a batch job of requested predictions:

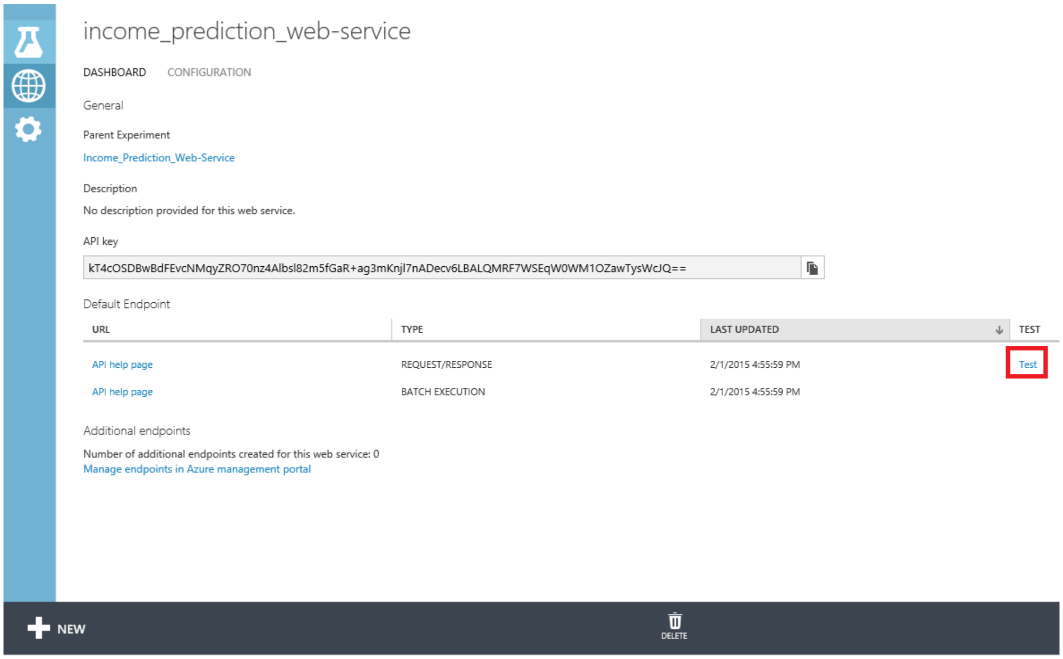
- Submit a job.
- Get job status.
- Delete a job.

The possible status codes returned from the executing web service batch job are as follows:

- 0 Not started
- 1 Running
- 2 Failed
- 3 Cancelled
- 4 Finished

Testing the Azure Machine Learning web service

Now, it's time to test our new Azure Machine Learning web service. Azure ML Studio provides a quick and easy way to interactively test a new web service as part of the publishing process. Simply click the Test link for the Request Response API link shown in Figure 3-51.



income_prediction_web-service

DASHBOARD CONFIGURATION

General

Parent Experiment

Income_Prediction_Web-Service

Description

No description provided for this web service.

API key

kT4cOSDBwBdFEvcNMqyZRO70nz4Albsl82m5fGaR+ag3mKnjl7nADecv6LBALQMRF7WSEqW0WM1OZawTysWcJQ==

Default Endpoint

URL	TYPE	LAST UPDATED	TEST
API help page	REQUEST/RESPONSE	2/1/2015 4:55:59 PM	Test
API help page	BATCH EXECUTION	2/1/2015 4:55:59 PM	

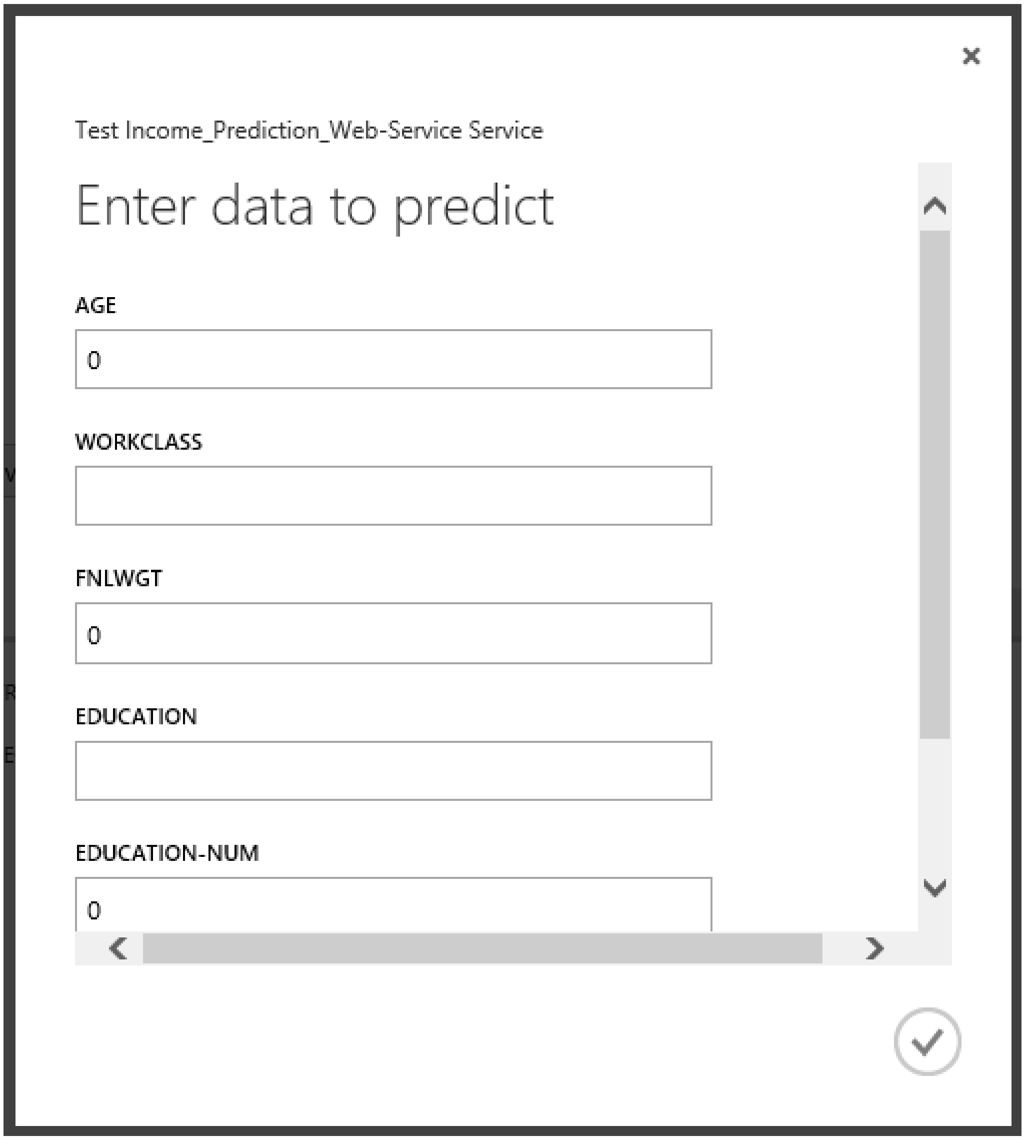
Additional endpoints

Number of additional endpoints created for this web service: 0
Manage endpoints in Azure management portal

FIGURE 3-51 The link for testing a REQUEST/RESPONSE web service API.

The Test link opens a window where you can interactively enter the values to be used to call the web service and then see resulting response. You can enter valid input values by referencing the "Attribute Information" section from the origination dataset download link at http://archive.ics.uci.edu/ml/datasets/Census+Income.

Figure 3-52 shows the interactive web service test pop-up window.


Test Income_Prediction_Web-Service Service

Enter data to predict

AGE

0

WORKCLASS

FNLWGT

0

EDUCATION

EDUCATION-NUM

0

FIGURE 3-52 The Azure Machine Learning web service interactive testing window.

Once you have entered the sample data inputs, click the check mark icon to submit your web service request. The Azure Machine Learning service will begin processing your request and will update the status area at the bottom of your screen.

Click DETAILS, highlighted in Figure 3-53, to see the results of the test web service call, as shown in Figure 3-54.

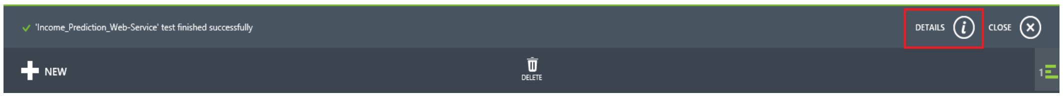
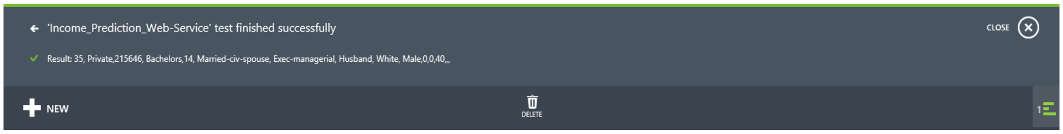

'Income_Prediction_Web-Service' test finished successfully

FIGURE 3-53 Processing the Azure Machine Learning interactive web service request.


'Income_Prediction_Web-Service' test finished successfully

Result: 35, Private,215646, Bachelors,14, Married-civ-spouse, Exec-managerial, Husband, White, Male,0,0,40,,,

FIGURE 3-54 The Details view of the returned results from the Azure Machine Learning web service call.

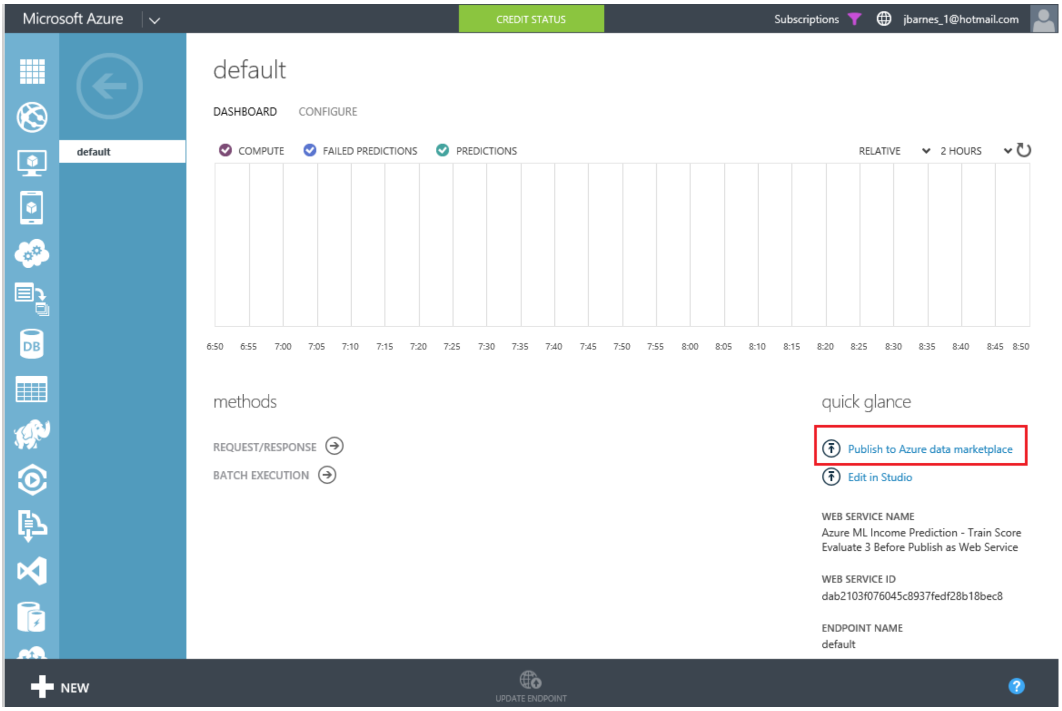
Note that when you expand the Details Information icon, it will nicely display all the input parameters passed to the web service call, along with all the corresponding outputs, which are the two columns on the far right. In this example, the output from our Azure Machine Learning web service call displays the (correct) prediction of ">50K" with a probability factor of 0.529029%.

Publish to Azure Data Marketplace

In addition to being able to publish an Azure Machine Learning prediction web service for your own consumption, you can also submit your prediction model for consumption through the Azure Data Marketplace.

The Azure Data Marketplace provides the ability to publish Azure Machine Learning web services as paid or free services for consumption by external customers. Using this process, you can start making your web services available for other developers to consume in their applications.

Figure 3-55 displays the location of the link to publish to the Azure Data Marketplace.



default

WEB SERVICE NAME
Azure ML Income Prediction - Train Score Evaluate 3 Before Publish as Web Service

WEB SERVICE ID
dab2103f076045c8937fedf28b18bec8

ENDPOINT NAME
default

FIGURE 3-55 The link to publish to the Azure Data Marketplace.

Overview of the publishing process

The following are the steps in publishing an Azure Machine Learning web service to Azure Data Marketplace.

1. Create and publish an Azure Machine Learning Request/Response service (RRS) web service.
2. From the Azure Management Portal, deploy the prediction service into production.
3. Use the URL of the published web service to publish to the Azure Data Marketplace.
4. Once submitted, your offer is reviewed and needs to be approved before your customers can start purchasing it. Be aware that the publishing process can take a few business days.

Guidelines for publishing to Azure Data Marketplace

Follow these links to get started with publishing your Azure Machine Learning web service to the Azure Data Marketplace:

- http://azure.microsoft.com/en-gb/documentation/articles/machine-learning-publish-web-servi ce-to-azure-marketplace/

- http://datamarket.azure.com/home/
- https://publish.windowsazure.com/

Summary

So there you have it! A complete working Azure Machine Learning experiment that shows the entire workflow related to developing, training, and deploying a predictive analytics model over the Web in a matter of minutes! Be amazed, for this feat is truly a marvel of modern cloud alchemy at its best.

As we have seen in this chapter, the Microsoft Azure Machine Learning environment "democratizes" the mysterious art of data science. It exposes this art to the masses by means of an easy-to-use visual designer interface that requires literally no programming and only a browser to use.

We have seen how it enables virtually anyone to create sophisticated and powerful predictive analytic solutions, and how it can then quickly share and expose them for further development, refinement, and implementation. With the option to publish a predictive model right into the Azure Data Marketplace, you can even quickly and easily monetize your next big breakthrough in machine learning!

Modern commercial predictive analytics packages available today will cost you hundreds of thousands of dollars and take many man-months to implement and master. With Azure Machine Learning, you are up and running in minutes. It's free to get started and then pay-as-you-go with all of your solutions. It doesn't get any better than this!

Chapter 4

Creating Azure Machine Learning client and server applications

In this chapter, we build on the example provided in Chapter 3, "Using Azure ML Studio," where we created our first Azure Machine Learning experiment for predicting income levels using an end-to-end example. As a brief refresher, we started with a raw data set downloaded from the UCI data repository, then loaded and saved it into our Azure Machine Learning workspace. Next, we created a workflow to train a new predictive model using the Two-Class Boosted Decision Tree module. When it was ready to publish, we set the publish inputs and outputs and exposed the Azure Machine Learning prediction model as a new web service. We then briefly tested our service by using the "test" link to pop up a window to interactively enter the values to be used to call the web service, and see the resulting response.

The purpose of this chapter is to further expand on a working Azure Machine Learning predictive model and explore the types of client and server applications that you can create to consume Azure Machine Learning web services. Developing a cool and innovative machine learning predictive model is really only half the fun. The other half comes when you can quickly expose and incorporate a new Azure Machine Learning predictive model web service into your core business applications to expedite testing or production scenarios.

In this chapter, we walk through building various client and server applications meant to consume Azure Machine Learning predictive models exposed as web services. We start with some very simple basic client applications and then explore a full-blown ASP.NET Web API web service implementation that implements the REST protocol and provides Azure Machine Learning predictions using JSON payloads for inputs and outputs. This approach allows you to quickly and easily expose Azure Machine Learning web services to a wide variety of modern web and mobile applications for mass consumption.

Why create Azure Machine Learning client applications?

One of the drawbacks with using the built-in, quick tester, pop-up application provided with Azure Machine Learning Studio is that you must know exactly what all the valid input options are for each of the 14 input parameters to successfully make a prediction.

The following is a listing of the attributes and allowed values from the UCI data repository site at http://archive.ics.uci.edu/ml/datasets/Census+Income.

- **age** Continuous
- **workclass** Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
- **fnlwgt** Continuous
- **education** Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
- **education-num** Continuous
- **marital-status** Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
- **occupation** Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
- **relationship** Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
- **race** White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
- **sex** Female, Male
- **capital-gain** Continuous
- **capital-loss** Continuous
- **hours-per-week** Continuous
- **native-country** United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holland-Netherlands

Think about how time-consuming and impractical it now becomes to test a new Azure Machine Learning prediction model. Having to memorize or look up all the allowable parameter options for each variable for each iteration of testing is not the most efficient way to test a model, not to mention the fact that these parameters are also case-sensitive. Input parameter accuracy (even to the degree of proper case sensitivity) becomes a crucial element to providing accurate machine learning predictions.

Creating a client application that simplifies the process and providing drop-down list boxes of valid choices allows the user to quickly and easily submit various combinations of prediction inputs. The user then gets immediate feedback by seeing the corresponding results in real time.

Azure Machine Learning web services sample code

Let's start by looking at the default sample code that is provided for calling Azure Machine Learning web services. After you have exposed an Azure Machine Learning web service, you can reach a code sample page, by navigating to your Azure Machine Learning workspace from the main Azure menu. Follow these navigation steps:

1. Navigate to Azure Machine Learning/<YourWorkSpace>. Select the Web Services tab as shown in Figure 4-1.

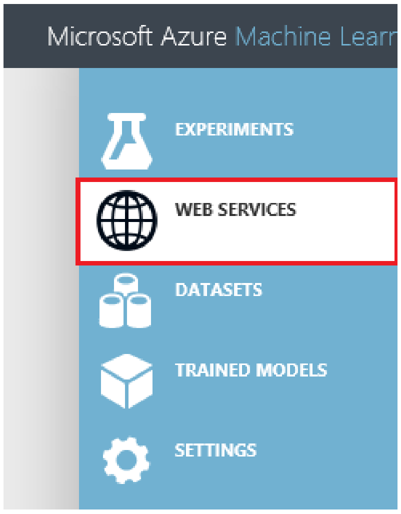


FIGURE 4-1 Navigating to the Azure Machine Learning workspace and selecting the Web Services tab.

2. Select the specific implementation version for the Azure Machine Learning web service as shown in Figure 4-2.

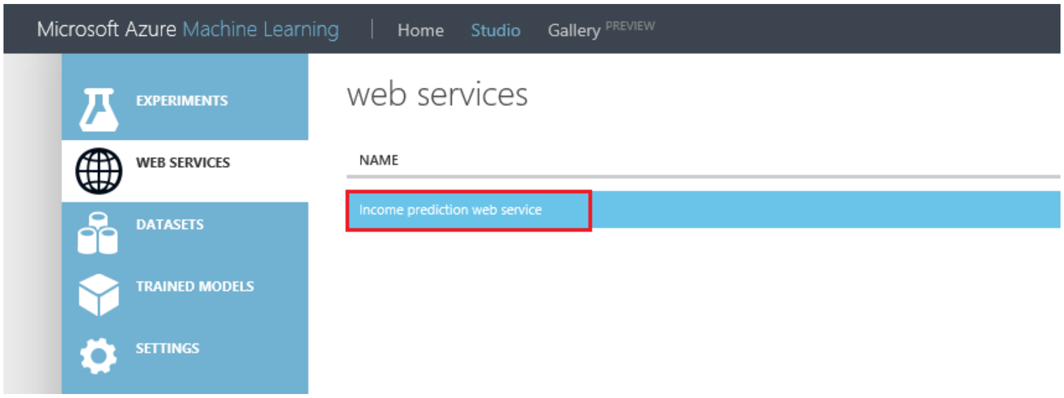
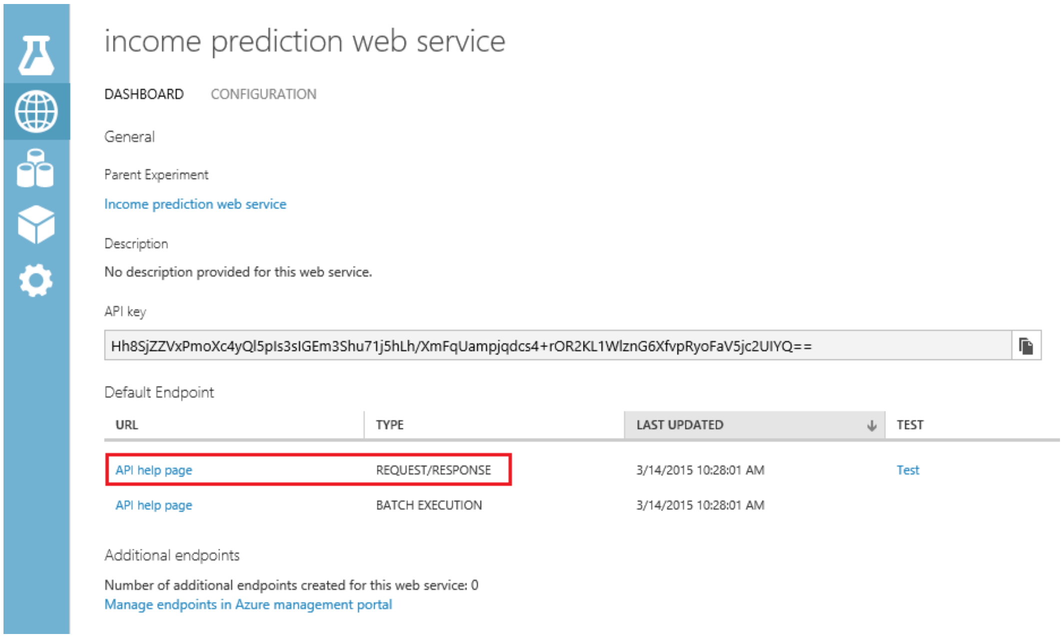


FIGURE 4-2 Select the specific implementation version for the Azure Machine Learning web service.

3. Select the API Help Page for the Request/Response option as shown in Figure 4-3.


income prediction web service

DASHBOARD CONFIGURATION

General

Parent Experiment

Income prediction web service

Description

No description provided for this web service.

API key

Hh8SjZZVxPmoXc4yQl5pls3sIGEm3Shu71j5hLh/XmFqUampjqdcs4+rOR2KL1WlznG6XfvpRyoFaV5jc2UIYQ==

Default Endpoint

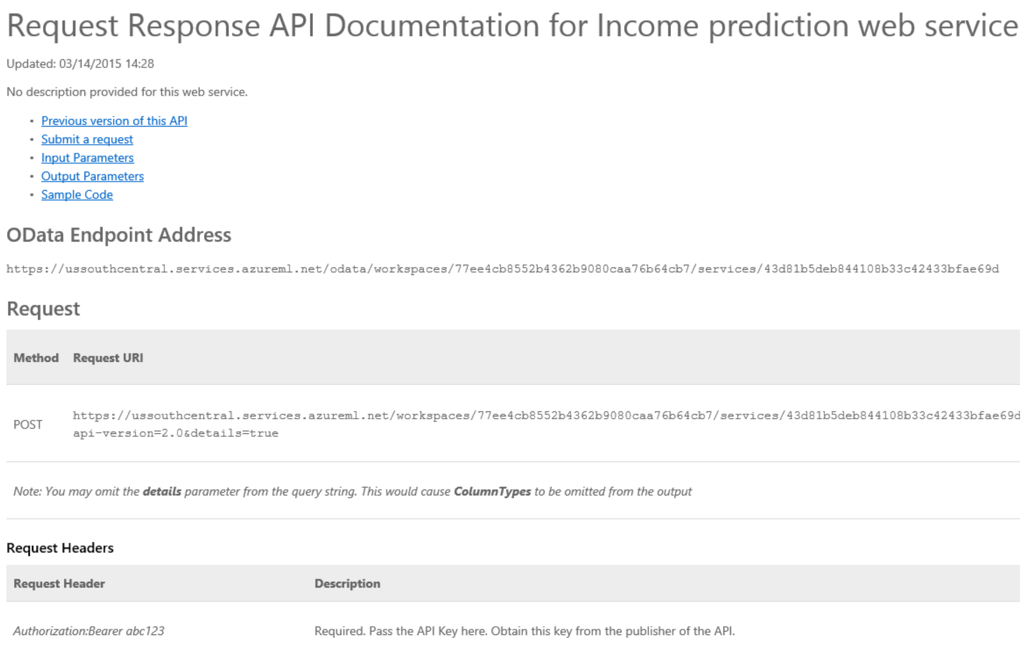
URL	TYPE	LAST UPDATED	TEST
API help page	REQUEST/RESPONSE	3/14/2015 10:28:01 AM	Test
API help page	BATCH EXECUTION	3/14/2015 10:28:01 AM	

Additional endpoints

Number of additional endpoints created for this web service: 0
Manage endpoints in Azure management portal

FIGURE 4-3 Select the API Help Page for the Request/Response methods.

4. You will then be directed to a webpage where you can easily view all the technical details necessary to successfully invoke this Azure Machine Learning web service via an HTTP POST request as seen in Figure 4-4.



Request Response API Documentation for Income prediction web service

Updated: 03/14/2015 14:28

No description provided for this web service.

- Previous version of this API
- Submit a request
- Input Parameters
- Output Parameters
- Sample Code

OData Endpoint Address

```
https://ussouthcentral.services.azureml.net/odata/workspaces/77ee4cb8552b4362b9080caa76b64cb7/services/43d81b5deb844108b33c42433bfae69d
```

Request

Method	Request URI
POST	`https://ussouthcentral.services.azureml.net/workspaces/77ee4cb8552b4362b9080caa76b64cb7/services/43d81b5deb844108b33c42433bfae69d` `api-version=2.0&details=true`

Note: You may omit the ***details*** *parameter from the query string. This would cause* ***ColumnTypes*** *to be omitted from the output*

Request Headers

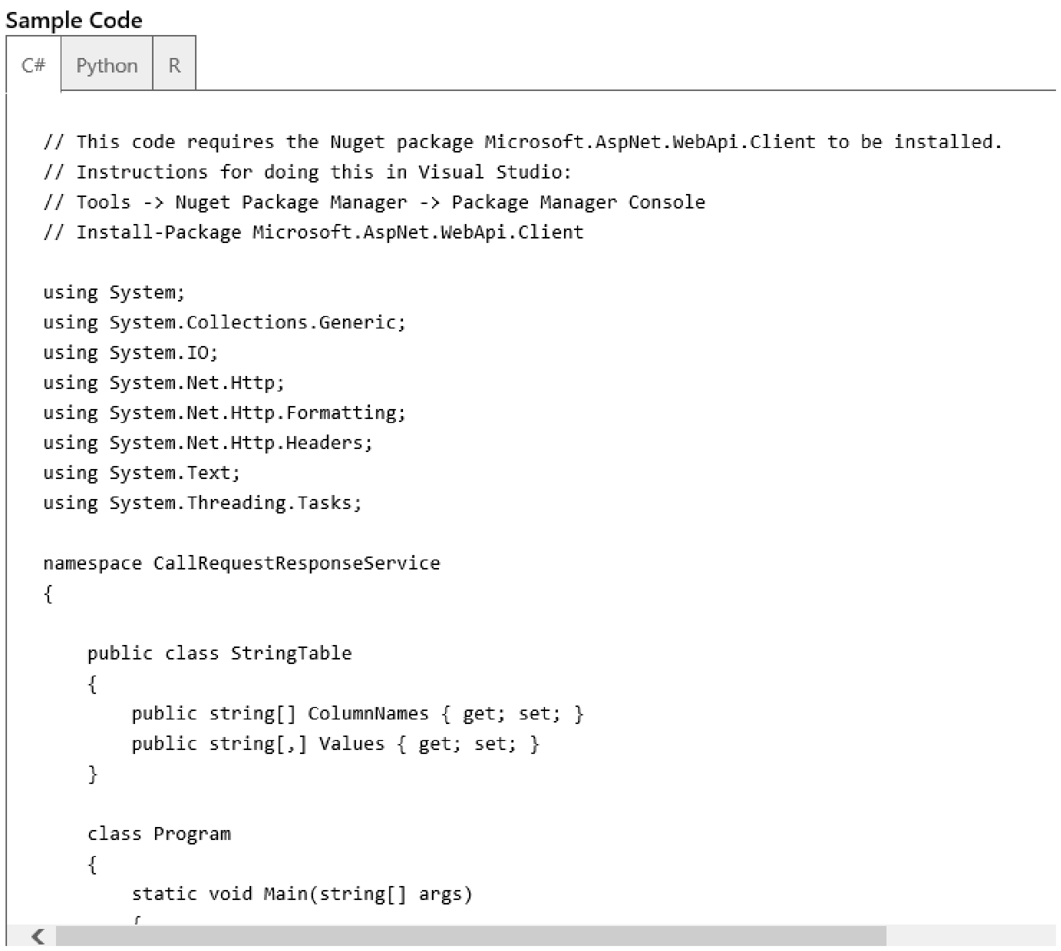
Request Header	Description
Authorization:Bearer abc123	Required. Pass the API Key here. Obtain this key from the publisher of the API.

FIGURE 4-4 The Azure Machine Learning web service API Help page.

This Help page for the Azure Machine Learning API web service provides specific implementation details about the following aspects of making an Azure Machine Learning web service via an HTTP POST request:

- OData End Point Address
- Method Request URI Format
- POST Request Headers
- Sample REQUEST body
- RESPONSE – Status Codes
- RESPONSE – Headers
- RESPONSE – Body
- RESPONSE – Sample Reply

If you scroll all the way down the page, you will see a Sample Code section. This will help you get a jump-start on creating new clients that can call your new Azure Machine Learning web service. Note in Figure 4-5 that implementation samples are provided for the programming languages C#, Python, and R.



```
// This code requires the Nuget package Microsoft.AspNet.WebApi.Client to be installed.
// Instructions for doing this in Visual Studio:
// Tools -> Nuget Package Manager -> Package Manager Console
// Install-Package Microsoft.AspNet.WebApi.Client

using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;

namespace CallRequestResponseService
{

    public class StringTable
    {
        public string[] ColumnNames { get; set; }
        public string[,] Values { get; set; }
    }

    class Program
    {
        static void Main(string[] args)
```

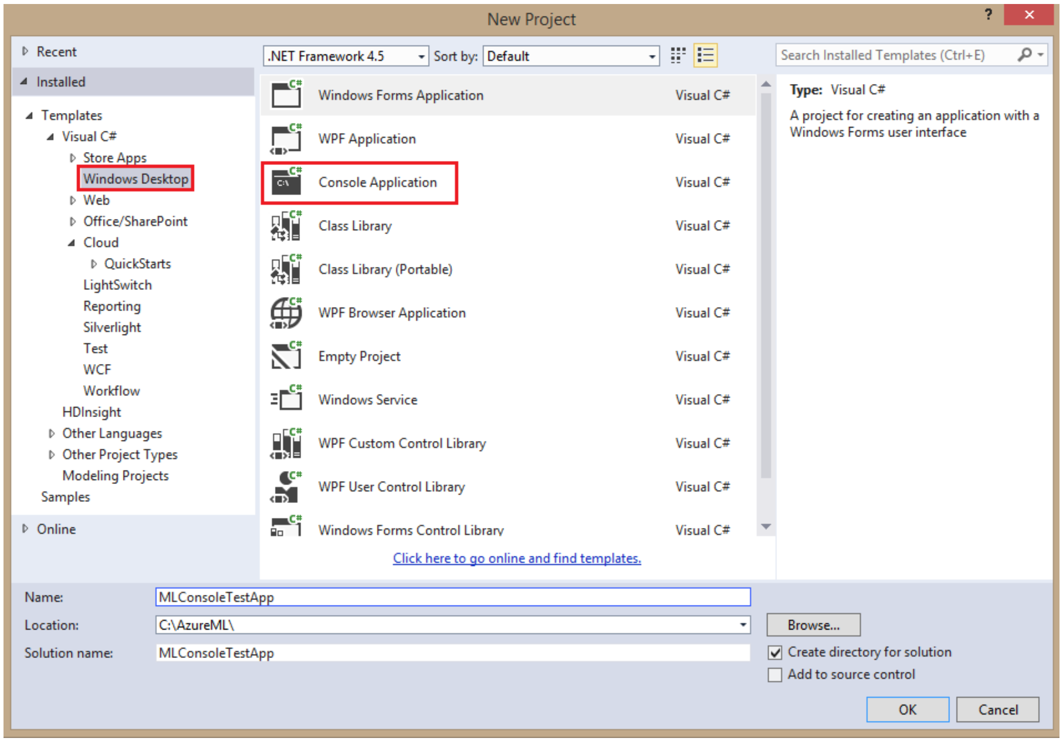
FIGURE 4-5 Sample code for invoking an Azure Machine Learning web service.

C# console app sample code

Let's start by looking at the default sample code that is provided for calling our Azure Machine Language web service from C#. To get started, you will need a copy of Visual Studio. You can download a free copy of Visual Studio Express at the following link. Be sure to scroll down the page to the link for Express 2013 for Windows Desktop:

http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx

Start by opening Visual Studio 2013. Then select File/New Project/Visual C#/Windows Desktop/Console Application. You will see a screen similar to Figure 4-6.


New Project

Recent
Installed
Templates
Visual C#
Store Apps
Windows Desktop
Web
Office/SharePoint
Cloud
QuickStarts
LightSwitch
Reporting
Silverlight
Test
WCF
Workflow
HDInsight
Other Languages
Other Project Types
Modeling Projects
Samples
Online

.NET Framework 4.5 Sort by: Default

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Class Library	Visual C#
Class Library (Portable)	Visual C#
WPF Browser Application	Visual C#
Empty Project	Visual C#
Windows Service	Visual C#
WPF Custom Control Library	Visual C#
WPF User Control Library	Visual C#
Windows Forms Control Library	Visual C#

Click here to go online and find templates.

Search Installed Templates (Ctrl+E)

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: MLConsoleTestApp
Location: C:\AzureML\
Solution name: MLConsoleTestApp

Browse...
☑ Create directory for solution
☐ Add to source control

OK Cancel

FIGURE 4-6 Creating a Desktop console application in C#.

Choose MLConsoleTestApp as the project name. Pick a location folder for your project. Once the solution has been generated, cut, copy, and paste the entire sample C# code from the Azure Machine Learning web service API Help page into the Program.cs module. Be sure to delete all of the original code in the module.

The next step is to install a required NuGet package to allow the application to make HTTP POST requests. Open the NuGet Package Manager Console by selecting Tools/NuGet Package Manager/Package Manager Console as shown in Figure 4-7.

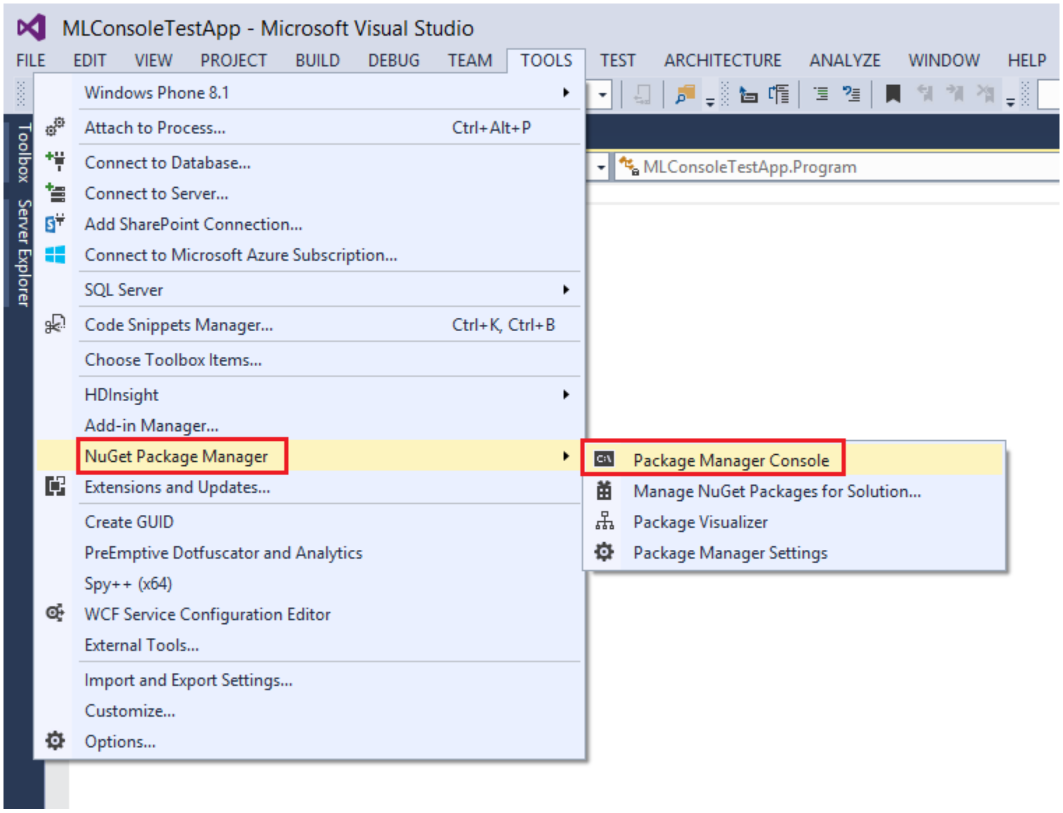


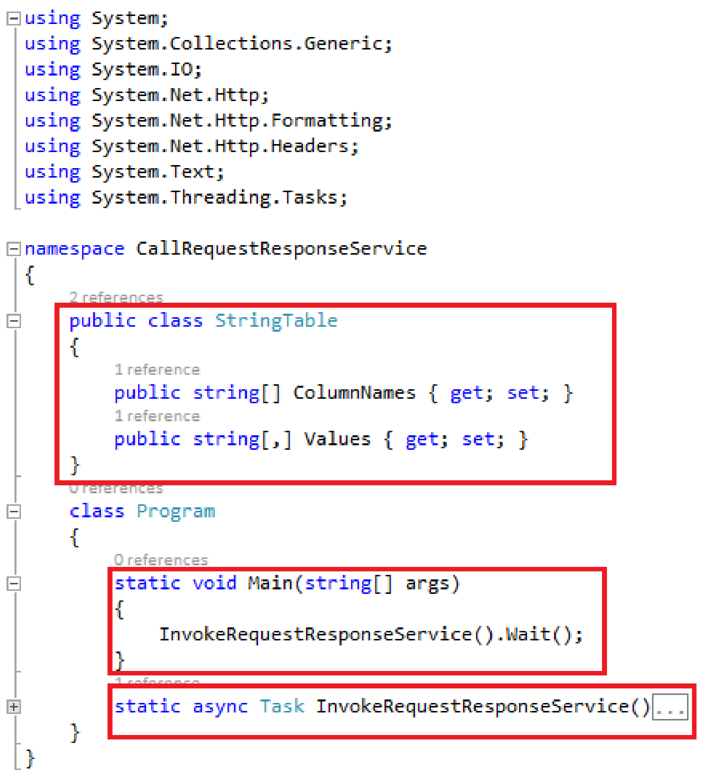
FIGURE 4-7 Invoking the NuGet Package Manager Console.

Once you have a NuGet Package Manager Console window, enter the following command to start the installation of the Microsoft.AspNet.WebApi.Client NuGet packages and their dependencies:

```
Install-Package Microsoft.AspNet.WebApi.Client
```

After the command has finished, you should be able to successfully build the application with no compiler errors.

Now, let's examine the sample C# code in Figure 4-8. Notice that there is a class called StringTable that is used to define a Request to our Azure Machine Learning web service.



```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;

namespace CallRequestResponseService
{
    2 references
    public class StringTable
    {
        1 reference
        public string[] ColumnNames { get; set; }
        1 reference
        public string[,] Values { get; set; }
    }
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            InvokeRequestResponseService().Wait();
        }
        1 reference
        static async Task InvokeRequestResponseService()...
    }
}
```

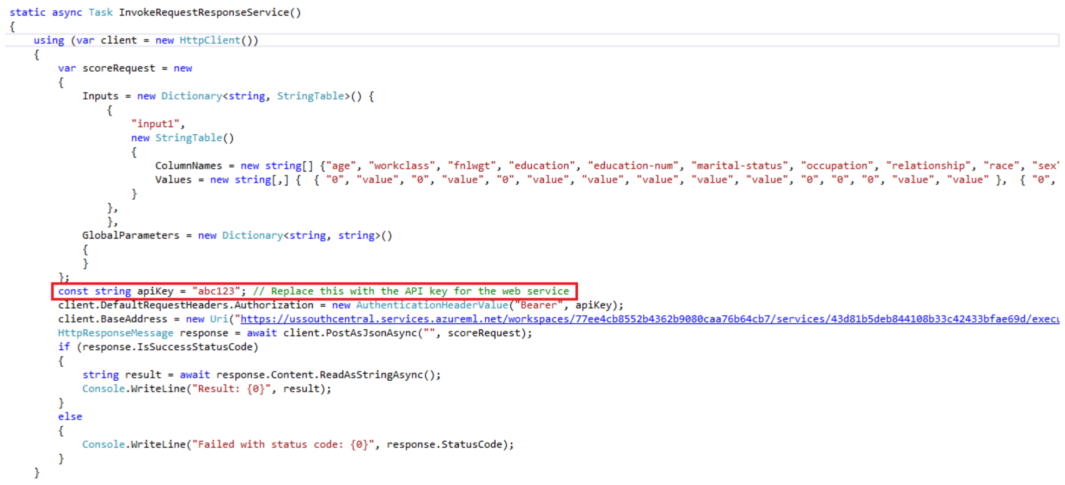
FIGURE 4-8 Sample Visual Studio 2013 C# console application.

The StringTable class contains two string arrays used for invoking the web service.

- The ColumnNames string array contains the column names for the input parameters.
- The Values string array will contain the individual values for each corresponding parameter passed into the web service call.

All the magic really happens in the Main routine, where the program invokes the InvokeRequestResponseService()method call and then displays the results.

Figure 4-9 shows the InvokeRequestResponseService()method call where we can see the actions required to make a successful web service call.



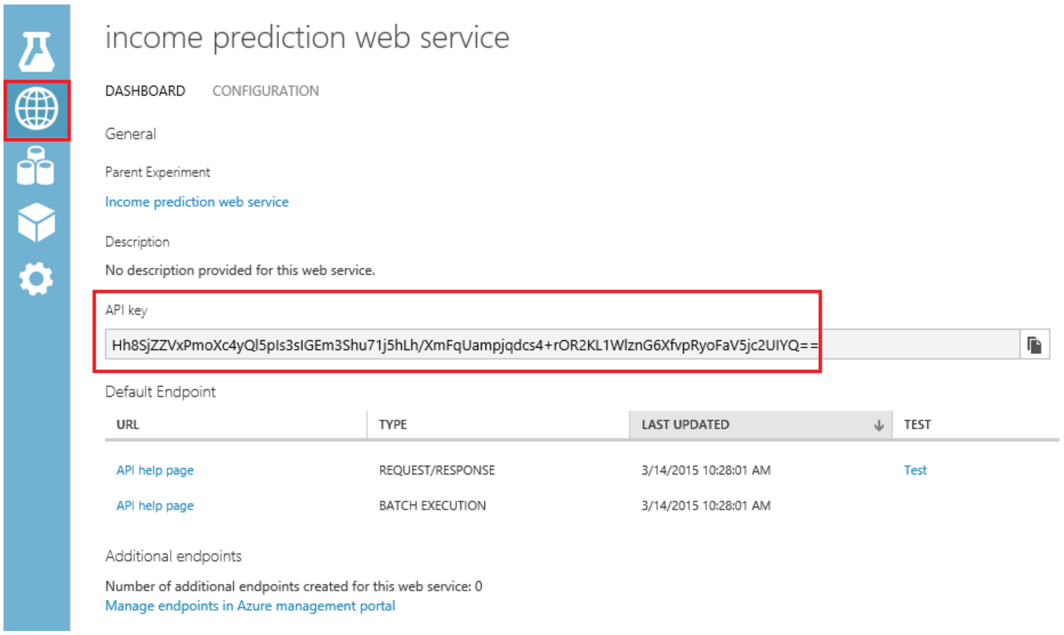
```
static async Task InvokeRequestResponseService()
{
    using (var client = new HttpClient())
    {
        var scoreRequest = new
        {
            Inputs = new Dictionary<string, StringTable>() {
                {
                    "input1",
                    new StringTable()
                    {
                        ColumnNames = new string[] {"age", "workclass", "fnlwgt", "education", "education-num", "marital-status", "occupation", "relationship", "race", "sex'
                        Values = new string[,] {  { "0", "value", "0", "value", "0", "value", "value", "value", "value", "value", "0", "0", "0", "value", "value" },  { "0",
                    }
                },
                },
            GlobalParameters = new Dictionary<string, string>()
            {
            }
        };
        const string apiKey = "abc123"; // Replace this with the API key for the web service
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
        client.BaseAddress = new Uri("https://ussouthcentral.services.azureml.net/workspaces/77ee4cb8552b4362b9080caa76b64cb7/services/43d81b5deb844108b33c42433bfae69d/execu
        HttpResponseMessage response = await client.PostAsJsonAsync("", scoreRequest);
        if (response.IsSuccessStatusCode)
        {
            string result = await response.Content.ReadAsStringAsync();
            Console.WriteLine("Result: {0}", result);
        }
        else
        {
            Console.WriteLine("Failed with status code: {0}", response.StatusCode);
        }
    }
```

FIGURE 4-9 Sample C# code to invoke Azure Machine Learning web services.

There are some noteworthy aspects to point out in the preceding sample code.

- A dictionary object named Inputs is populated with two string arrays.
- The first string array named ColumnNames is prepopulated with the names for the web service input parameters.
- The second string array named Values is prepopulated with default numeric and string values for the web service input parameters.
- All the initial code sample input values are set to 0 or value to invoke a successful web service call.
- The correct input values would come from selecting appropriate options for each of the 14 parameters for the UCI data repository site at http://archive.ics.uci.edu/ml/datasets/Census+Income.
- An HTTP Request header for Authorization is set based on the Azure Machine Learning web service API key field for this web service invocation.
- You will need to provide your own value for the apikey field from your Azure Machine Learning web service.
- The client.BaseAddress variable is set to the specific URL instance of your Azure Machine Learning web service.

You can find the API key for your web service by navigating to the Web Services tab of your Azure Machine Learning Studio workspace. The API key can be found by clicking the specific web service version for your experiment as shown in Figure 4-10.



income prediction web service

DASHBOARD CONFIGURATION

General

Parent Experiment

Income prediction web service

Description

No description provided for this web service.

API key

Hh8SjZZVxPmoXc4yQl5pIs3sIGEm3Shu71j5hLh/XmFqUampjqdcs4+rOR2KL1WlznG6XfvpRyoFaV5jc2UIYQ==

Default Endpoint

URL	TYPE	LAST UPDATED	TEST
API help page	REQUEST/RESPONSE	3/14/2015 10:28:01 AM	Test
API help page	BATCH EXECUTION	3/14/2015 10:28:01 AM	

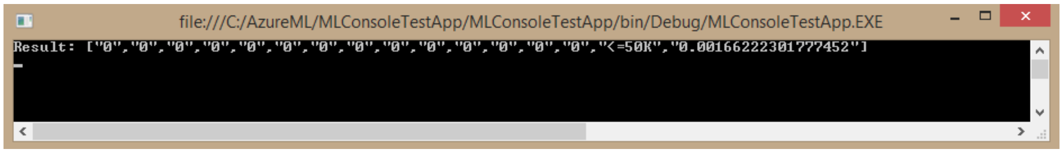
Additional endpoints

Number of additional endpoints created for this web service: 0

Manage endpoints in Azure management portal

FIGURE 4-10 The Azure Machine Learning Web Services page showing the location of the API key.

Note that if we run the console app at this point, we will get the results shown in Figure 4-11, based on providing all zeroes as inputs.



```
file:///C:/AzureML/MLConsoleTestApp/MLConsoleTestApp/bin/Debug/MLConsoleTestApp.EXE
Result: ["0","0","0","0","0","0","0","0","0","0","0","0","0","0","<=50K","0.00166222301777452"]
```

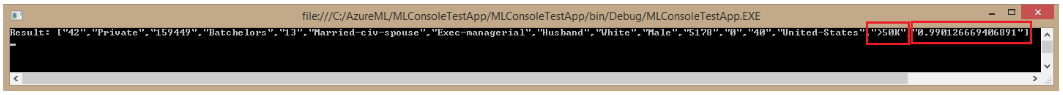
FIGURE 4-11 Results returned from the C# console application after invoking the Azure Machine Learning web service.

In this example, the response back from our call to the Azure Machine Learning web service indicates that this input candidate likely makes less than $50,000 a year. The corresponding confidence level is 0.0010072038276121.

Now watch what happens if we replace the Values input string array parameters with valid data that is likely to have a higher score as in the following code fragment:

```
 Values = new string[,] {  {
                                              "52",
                                              "Self-emp-not-inc",
                                              "209642",
                                              "HS-grad",
                                              "12",
                                              "Married-civ-spouse",
                                              "Exec-managerial",
                                              "Husband",
                                              "White",
                                              "Male",
                                              "0",
                                              "0",
                                              "45",
                                              "United-States",
                                              "0"
                                           }
                                        }
                                     }
                                  },
```

Figure 4-12 shows the results of a more accurate prediction. Based on the updated, nonzero, and valid input parameters given, the candidate is likely to make more than $50,000 a year with a confidence level of 0.549394130706787.


file:///C:/AzureML/MLConsoleTestApp/MLConsoleTestApp/bin/Debug/MLConsoleTestApp.EXE

```
Result: ["42","Private","159449","Batchelors","13","Married-civ-spouse","Exec-managerial","Husband","White","Male","5178","0","40","United-States",">50K","0.990126669406891"]
```

FIGURE 4-12 Results from the Azure Machine Learning web service call after adjusting the input parameters.

R sample code

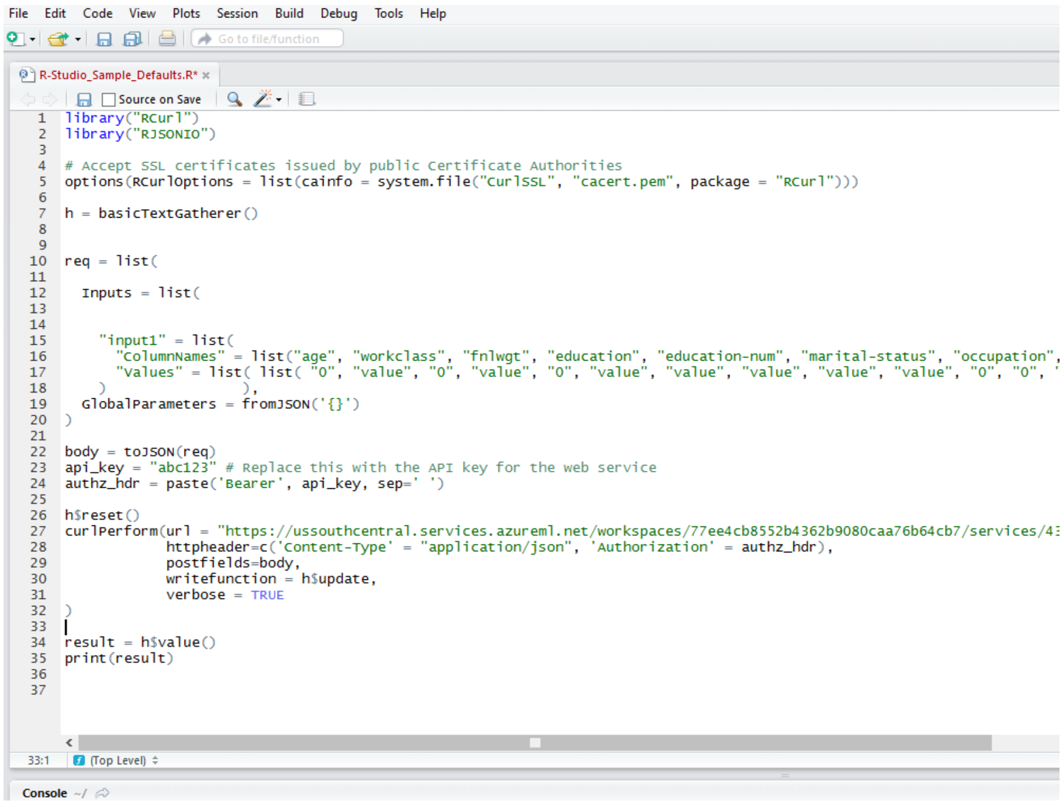
In addition to C#, one of the additional code samples provided for invoking an Azure Machine Learning web service is in the R programming language. In the world of data science, R is a very popular programming language for developing data analytics, statistical modeling, and graphics applications. You can find an excellent tutorial on using the R programming language with Azure Machine Learning at http://azure.microsoft.com/en-us/documentation/articles/machine-learning-r-quickstart/.

To get started with the R code sample, you will need to download a few prerequisites, including the following:

- Open source R (free) downloads are available at the Comprehensive R Archive Network or CRAN at http://www.r-project.org/.

- RStudio is a powerful and productive user interface for R. It's free and open source, and works great on Windows, Mac, and Linux platforms. Download the desktop version at http://www.rstudio.com/products/rstudio/.
- A tutorial introduction to RStudio is available at https://support.rstudio.com/hc/en-us/sections/200107586-Using-RStudio.

Once you have installed all these prerequisites, start the RStudio application and paste in the R sample code from the Azure Machine Learning web services page. Figure 4-13 shows a screenshot of the RStudio environment and the pasted sample Azure Machine Learning R code to invoke our web service.



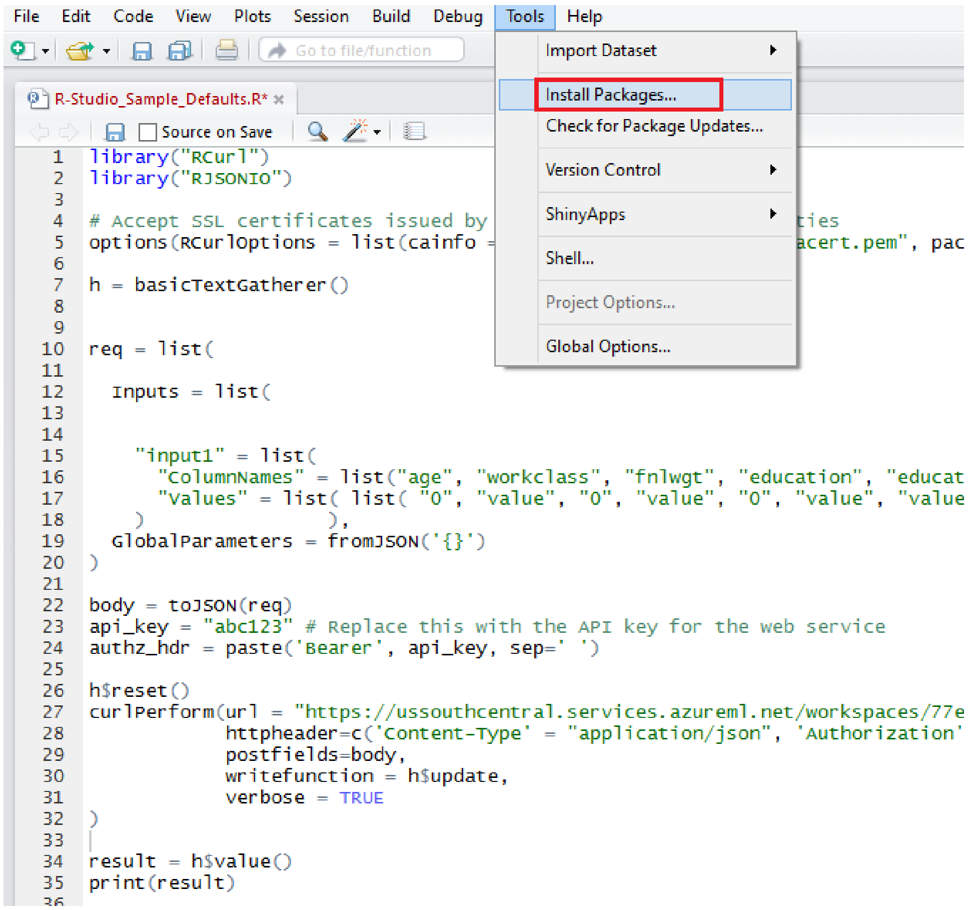
```r
library("RCurl")
library("RJSONIO")

# Accept SSL certificates issued by public Certificate Authorities
options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))

h = basicTextGatherer()


req = list(

  Inputs = list(


    "input1" = list(
      "ColumnNames" = list("age", "workclass", "fnlwgt", "education", "education-num", "marital-status", "occupation",
      "Values" = list( list( "0", "value", "0", "value", "0", "value", "value", "value", "value", "value", "0", "0", '
    )                 ),
  GlobalParameters = fromJSON('{}')
)

body = toJSON(req)
api_key = "abc123" # Replace this with the API key for the web service
authz_hdr = paste('Bearer', api_key, sep=' ')

h$reset()
curlPerform(url = "https://ussouthcentral.services.azureml.net/workspaces/77ee4cb8552b4362b9080caa76b64cb7/services/43
            httpheader=c('Content-Type' = "application/json", 'Authorization' = authz_hdr),
            postfields=body,
            writefunction = h$update,
            verbose = TRUE
)

result = h$value()
print(result)


```

FIGURE 4-13 Sample R code in RStudio to invoke the Azure Machine Learning web service.

Before we can run the R code sample, we must first install two required packages in RStudio that appear at the top of the Azure Machine Learning web service R code sample:

- library("RCurl")
- library("RJSONIO")

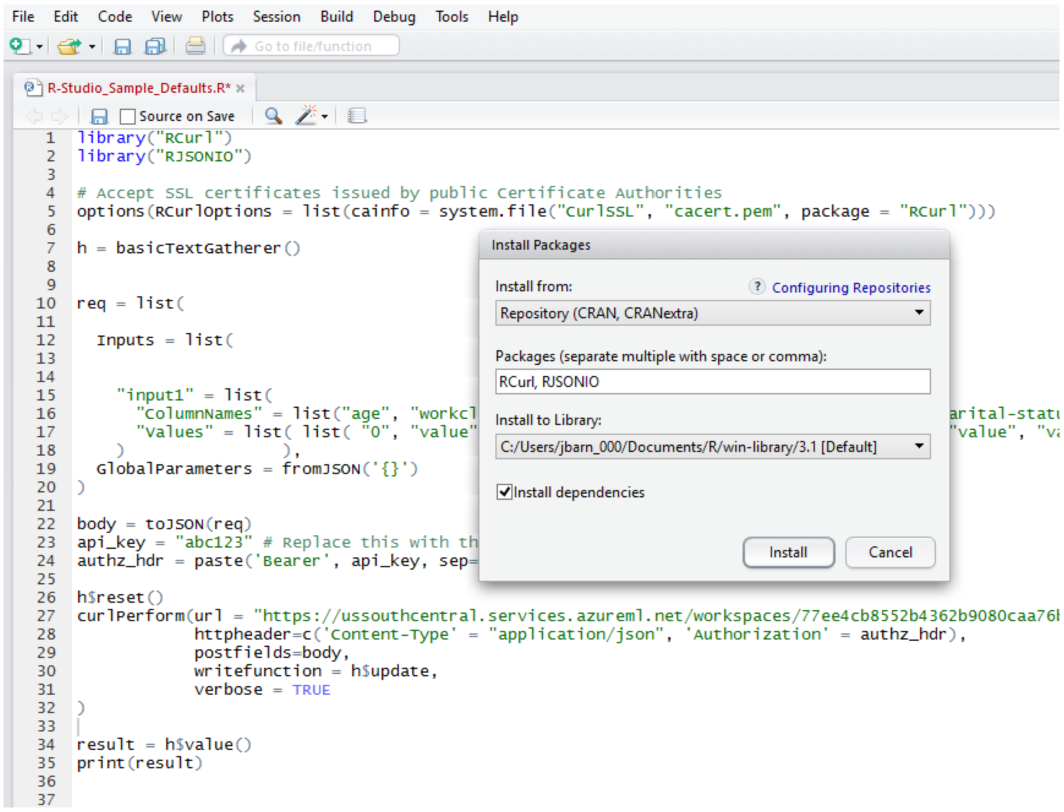
To do this, simply click the menu option for Tools, then click Install Packages, as shown in Figure 4-14.



```
library("RCurl")
library("RJSONIO")

# Accept SSL certificates issued by                                         ties
options(RCurlOptions = list(cainfo =                                        acert.pem", pac

h = basicTextGatherer()


req = list(

  Inputs = list(


    "input1" = list(
      "ColumnNames" = list("age", "workclass", "fnlwgt", "education", "educat
      "Values" = list( list( "0", "value", "0", "value", "0", "value", "value
    )                    ),
  GlobalParameters = fromJSON('{}')
)

body = toJSON(req)
api_key = "abc123" # Replace this with the API key for the web service
authz_hdr = paste('Bearer', api_key, sep=' ')

h$reset()
curlPerform(url = "https://ussouthcentral.services.azureml.net/workspaces/77e
            httpheader=c('Content-Type' = "application/json", 'Authorization'
            postfields=body,
            writefunction = h$update,
            verbose = TRUE
)

result = h$value()
print(result)
```

FIGURE 4-14 Invoking the Install Packages option in the RStudio environment.

Next, type the following package names: **RCurl, RJSONIO.** Click Install as shown in Figure 4-15.



```
library("RCurl")
library("RJSONIO")

# Accept SSL certificates issued by public Certificate Authorities
options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))

h = basicTextGatherer()


req = list(

  Inputs = list(


    "input1" = list(
      "ColumnNames" = list("age", "workcl                                           arital-statu
      "Values" = list( list( "0", "value"                                          "value", "va
    )                 ),
  GlobalParameters = fromJSON('{}')
)

body = toJSON(req)
api_key = "abc123" # Replace this with th
authz_hdr = paste('Bearer', api_key, sep=

h$reset()
curlPerform(url = "https://ussouthcentral.services.azureml.net/workspaces/77ee4cb8552b4362b9080caa76b
            httpheader=c('Content-Type' = "application/json", 'Authorization' = authz_hdr),
            postfields=body,
            writefunction = h$update,
            verbose = TRUE
)

result = h$value()
print(result)
```

Install Packages

Install from: Repository (CRAN, CRANextra)

Configuring Repositories

Packages (separate multiple with space or comma): RCurl, RJSONIO

Install to Library: C:/Users/jbarn_000/Documents/R/win-library/3.1 [Default]

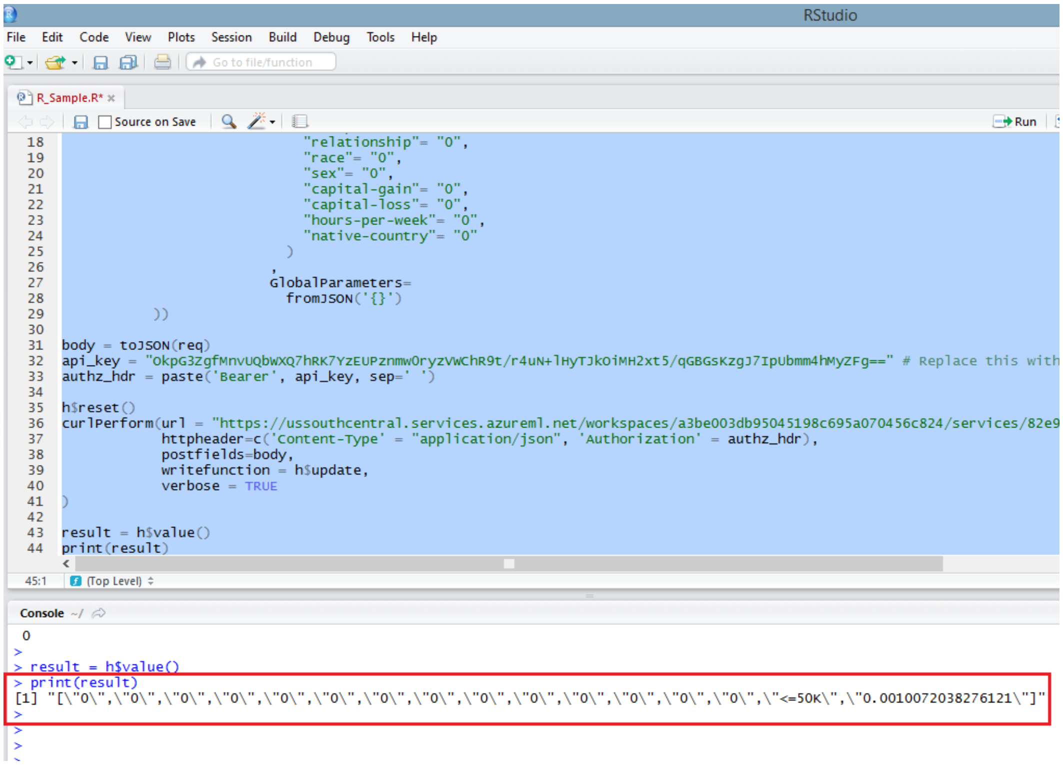
[✓] Install dependencies

FIGURE 4-15 Installing the R packages for RCurl and RJSONIO.

After the packages have been downloaded and installed, we are almost ready to run the R code sample. First, though, we need to update the required parameters for the Azure Machine Learning web service API key.

You can find the API key for your web service by navigating to the Web Services tab of your Azure Machine Learning Studio workspace. The API key can be found by clicking on the specific web service version for your experiment. It is then displayed on the web services dashboard.

Once you have updated the apikey parameter in the R code sample, the code will be ready to run. You can now run through the code by selecting or highlighting all the code in the main window and then clicking Run in the top left quadrant of the RStudio screen. As you step thru each line of the R code, you will see the code being executed on the right side of the RStudio environment. The bottom left portion of the screen contains the results of each call. The results of the call to our Azure Machine Learning web service are shown in Figure 4-16.



```
                            "relationship"= "0",
                            "race"= "0",
                            "sex"= "0",
                            "capital-gain"= "0",
                            "capital-loss"= "0",
                            "hours-per-week"= "0",
                            "native-country"= "0"
                          )
                        ,
                        GlobalParameters=
                          fromJSON('{}')
          ))

body = toJSON(req)
api_key = "OkpG3ZgfMnvUQbWXQ7hRK7YzEUPznmw0ryzVWChR9t/r4uN+lHyTJkOiMH2xt5/qGBGsKzgJ7IpUbmm4hMyZFg==" # Replace this with
authz_hdr = paste('Bearer', api_key, sep=' ')

h$reset()
curlPerform(url = "https://ussouthcentral.services.azureml.net/workspaces/a3be003db95045198c695a070456c824/services/82e9
            httpheader=c('Content-Type' = "application/json", 'Authorization' = authz_hdr),
            postfields=body,
            writefunction = h$update,
            verbose = TRUE
)

result = h$value()
print(result)
```

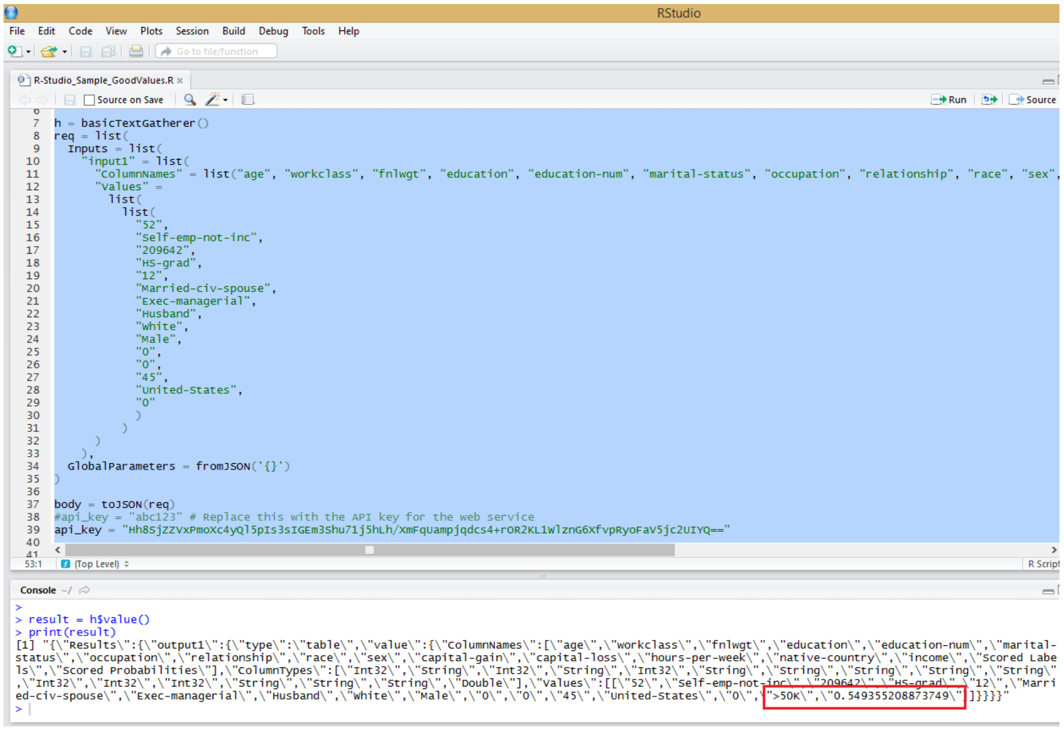
```
 0
>
> result = h$value()
> print(result)
[1] "[\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"0\",\"<=50K\",\"0.0010072038276121\"]"
>
>
>
```

FIGURE 4-16 Invoking the Azure Machine Learning web service from within RStudio.

Because all the input parameters were set to 0 in the R code sample, the result of the call is a prediction of <=50K and a very low confidence factor of 0.0010072038276121. Just as in the previous C# console code sample, if we change the input parameters to contain valid inputs, we will get a different outcome. You can do this by updating the Values list parameters as in the following code sample.

```
"Values" =
  list(
    list(
      "52",
      "Self-emp-not-inc",
      "209642",
      "HS-grad",
      "12",
      "Married-civ-spouse",
      "Exec-managerial",
      "Husband",
      "White",
      "Male",
      "0",
            "0",
            "45",
            "United-States",
            "0"
            )
         )
      )
```

Now, when we rerun the updated R code sample in RStudio with valid (and more reasonable) input parameters, we get a payload back with all the original input parameters, along with a prediction of >50 with a confidence factor of 0.549394130706787, as shown in Figure 4-17.



```
h = basicTextGatherer()
req = list(
  Inputs = list(
    "input1" = list(
      "ColumnNames" = list("age", "workclass", "fnlwgt", "education", "education-num", "marital-status", "occupation", "relationship", "race", "sex",
      "Values" =
        list(
          list(
            "52",
            "Self-emp-not-inc",
            "209642",
            "HS-grad",
            "12",
            "Married-civ-spouse",
            "Exec-managerial",
            "Husband",
            "White",
            "Male",
            "0",
            "0",
            "45",
            "United-States",
            "0"
            )
          )
      )
    ),
  GlobalParameters = fromJSON('{}')
)

body = toJSON(req)
#api_key = "abc123" # Replace this with the API key for the web service
api_key = "Hh8SjZZVxPmoXc4yQl5pIs3sIGEm3Shu71j5hLh/XmFqUampjqdcs4+rOR2KL1WlznG6XfvpRyoFaV5jc2UIYQ=="
```

```
>
> result = h$value()
> print(result)
[1] "{\"Results\":{\"output1\":{\"type\":\"table\",\"value\":{\"ColumnNames\":[\"age\",\"workclass\",\"fnlwgt\",\"education\",\"education-num\",\"marital-
status\",\"occupation\",\"relationship\",\"race\",\"sex\",\"capital-gain\",\"capital-loss\",\"hours-per-week\",\"native-country\",\"income\",\"Scored Labe
ls\",\"Scored Probabilities\"],\"ColumnTypes\":[\"Int32\",\"String\",\"Int32\",\"String\",\"Int32\",\"String\",\"String\",\"String\",\"String\",\"String\"
,\"Int32\",\"Int32\",\"Int32\",\"String\",\"String\",\"String\",\"Double\"],\"Values\":[[\"52\",\"Self-emp-not-inc\",\"209642\",\"HS-grad\",\"12\",\"Marri
ed-civ-spouse\",\"Exec-managerial\",\"Husband\",\"White\",\"Male\",\"0\",\"0\",\"45\",\"United-States\",\"0\",\">50K\",\"0.549355208873749\"]]}}}}"
>
```

FIGURE 4-17 Invocation of the Azure Machine Learning web service from the R sample code with different input parameters.

Moving beyond simple clients

The last two code samples in C# and R were provided as part of the Azure Machine Learning web service environment for use as basic guidance for invoking an Azure Machine Learning web service.

These code samples are effective in helping you get started with building simple clients in various programming languages such as C#, Python, and R, to invoke Azure Machine Learning web services.

They fall short, however, in terms of providing an optimized user experience for entering the many various parameter input combinations required to fully exercise and test an Azure Machine Learning predictive model.

To that end, we explore the options for creating more user-friendly clients. We extend the web services so that the widest variety of desktop, web, and mobile clients can consume the Azure Machine Learning predictive models developed within Azure Machine Learning Studio.

Cross-Origin Resource Sharing and Azure Machine Learning web services

Cross-Origin Resource Sharing (CORS) is a mechanism that allows web objects (e.g., JavaScript) from one domain to request objects (web service requests) from another domain. Cross-origin requests initiated from scripts have been subject to restrictions primarily for security reasons. CORS provides a way for web servers to support cross-site access controls, which enable secure cross-site data transfers.

At the time of this writing, it should be noted that Azure Machine Learning web services do not currently support CORS requests. See the following link for more information:

https://social.msdn.microsoft.com/Forums/en-US/b6ddeb77-30e1-45b2-b7c1-eb4492142c0a/azure-ml-published-web-services-cross-origin-requests?forum=MachineLearning.

This CORS restriction really means that if you wish to fully exploit Azure Machine Learning web services for deployment, testing, and production for a wide variety of (web) clients, you will need to host your own server-side applications. You basically have two choices.

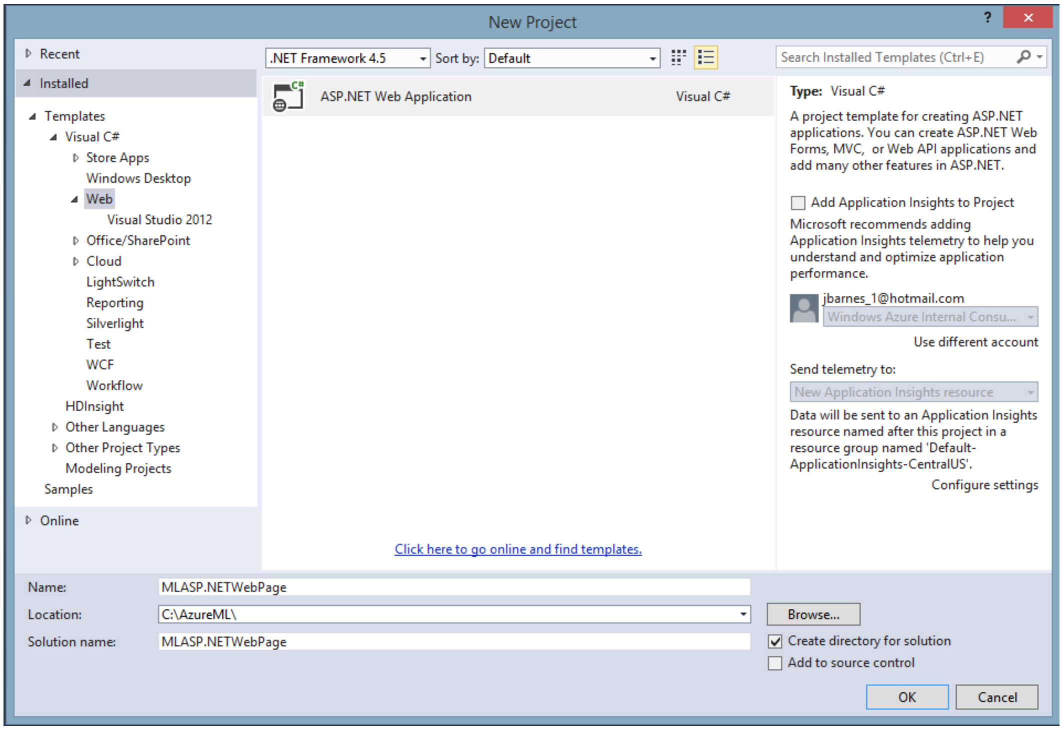
- Host a web application, such as an ASP.NET webpage, and invoke the Azure Machine Learning web service server-side to conform to the current Azure Machine Learning CORS restrictions.
- Host your own web service that does provide CORS support and can in turn invoke the Azure Machine Learning web service on behalf of a wide variety of web and mobile clients via modern protocols and data formats like REST and JSON.

We explore each of these options in the next sections.

Create an ASP.NET Azure Machine Learning web client

To provide a better experience for testing our new Azure Machine Learning income prediction web service, we will build an ASPX webpage. This webpage is based on the sample C# code provided with our Azure Machine Learning web service. If you don’t already have a copy of Visual Studio, you can download a free copy of Visual Studio 2013 Express at http://www.visualstudio.com/en-us/products/

visual-studio-express-vs.aspx. Let's start by creating a new ASP.NET Web Application project as shown in Figure 4-18.



Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

☐ Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

jbarnes_1@hotmail.com

Windows Azure Internal Consu...

Use different account

Send telemetry to:

New Application Insights resource

Data will be sent to an Application Insights resource named after this project in a resource group named 'Default-ApplicationInsights-CentralUS'.

Configure settings

Name: MLASP.NETWebPage

Location: C:\AzureML\

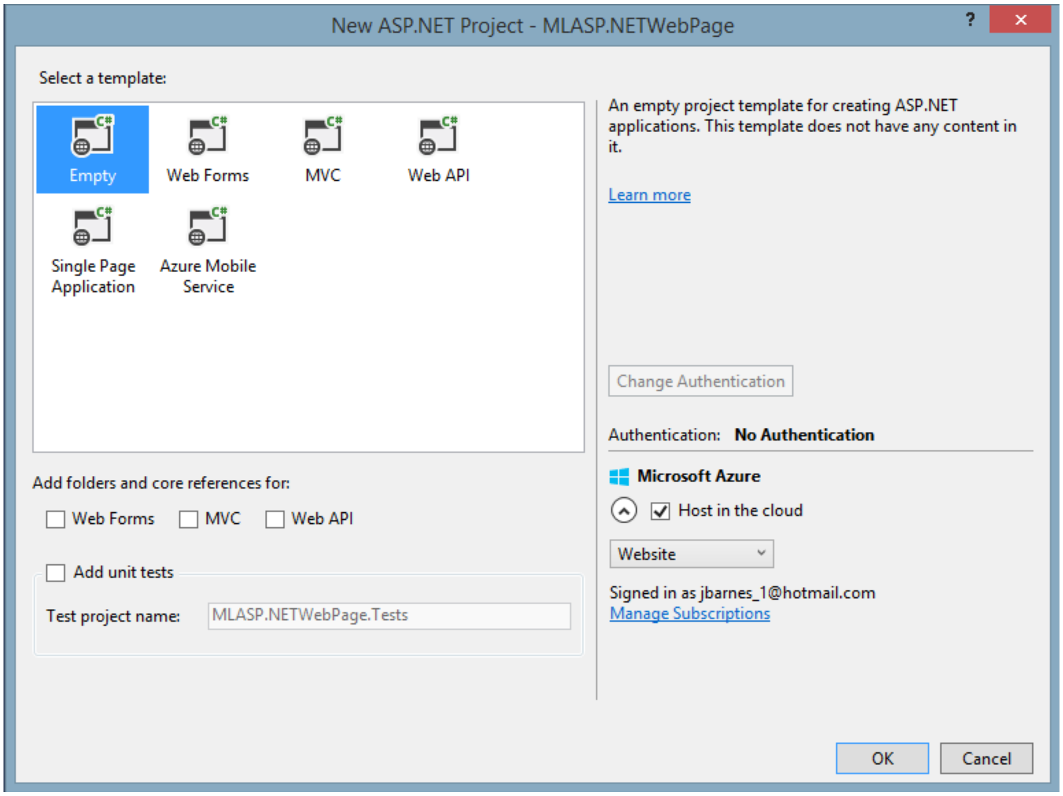
Solution name: MLASP.NETWebPage

☑ Create directory for solution

☐ Add to source control

FIGURE 4-18 Create a new ASP.NET web application in Visual Studio 2013.

Next, select an Empty web project template as shown in Figure 4-19.


New ASP.NET Project - MLASP.NETWebPage

Select a template:

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Authentication: **No Authentication**

Microsoft Azure

[✓] Host in the cloud

Website

Signed in as jbarnes_1@hotmail.com

Add folders and core references for:

[ ] Web Forms [ ] MVC [ ] Web API

[ ] Add unit tests

Test project name: MLASP.NETWebPage.Tests

FIGURE 4-19 Selecting an Empty ASP.NET project template in Visual Studio 2013.

Visual Studio then create an empty ASP.NET Web solution for you. Next, right-click the project and select Add from the shortcut menu as shown in Figure 4-20.

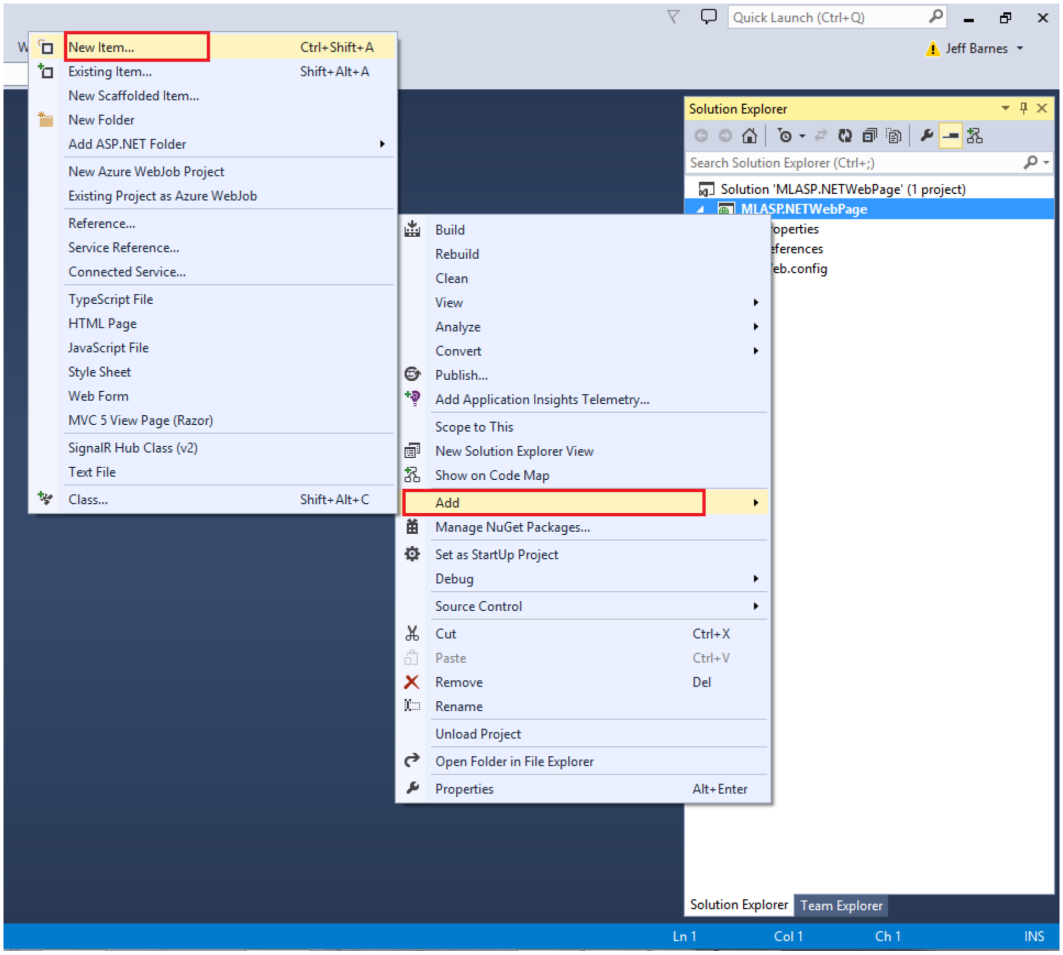
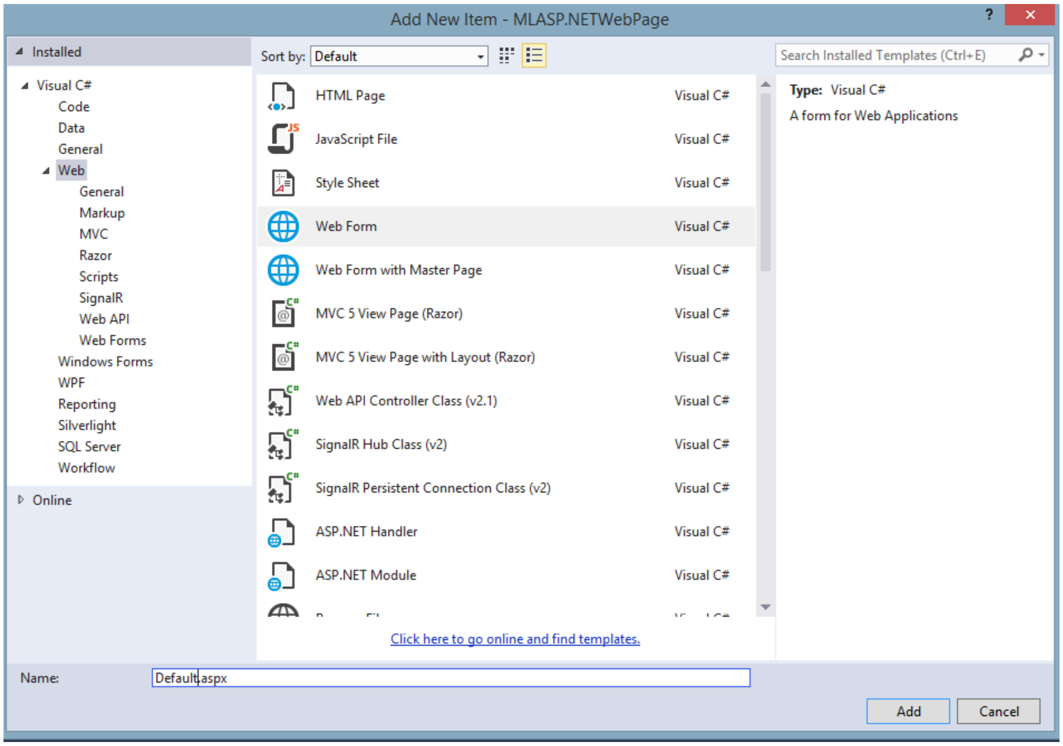


FIGURE 4-20 Adding a new item to the Visual Studio 2013 project.

Choose a Web Form and name the page Default.aspx as shown in Figure 4-21.


Add New Item - MLASP.NETWebPage

Installed

- Visual C#
  - Code
  - Data
  - General
  - Web
    - General
    - Markup
    - MVC
    - Razor
    - Scripts
    - SignalR
    - Web API
    - Web Forms
  - Windows Forms
  - WPF
  - Reporting
  - Silverlight
  - SQL Server
  - Workflow
- Online

Sort by: Default

Search Installed Templates (Ctrl+E)

HTML Page	Visual C#
JavaScript File	Visual C#
Style Sheet	Visual C#
Web Form	Visual C#
Web Form with Master Page	Visual C#
MVC 5 View Page (Razor)	Visual C#
MVC 5 View Page with Layout (Razor)	Visual C#
Web API Controller Class (v2.1)	Visual C#
SignalR Hub Class (v2)	Visual C#
SignalR Persistent Connection Class (v2)	Visual C#
ASP.NET Handler	Visual C#
ASP.NET Module	Visual C#

Type: Visual C#

A form for Web Applications

Click here to go online and find templates.

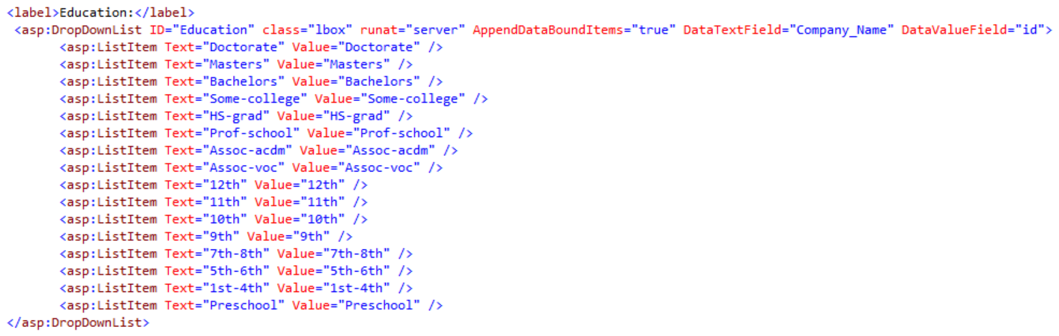
Name: Default.aspx

FIGURE 4-21 Selecting a Web Form item to add to the Visual Studio project.

Making it easier to test our Azure Machine Learning web service

Next, we modify the ASPX page to create a webpage layout that uses input controls like drop-down list boxes, as shown in Figure 4-22. This provides the user a valid list of options for each parameter required for the inputs to the Azure Machine Learning web service.

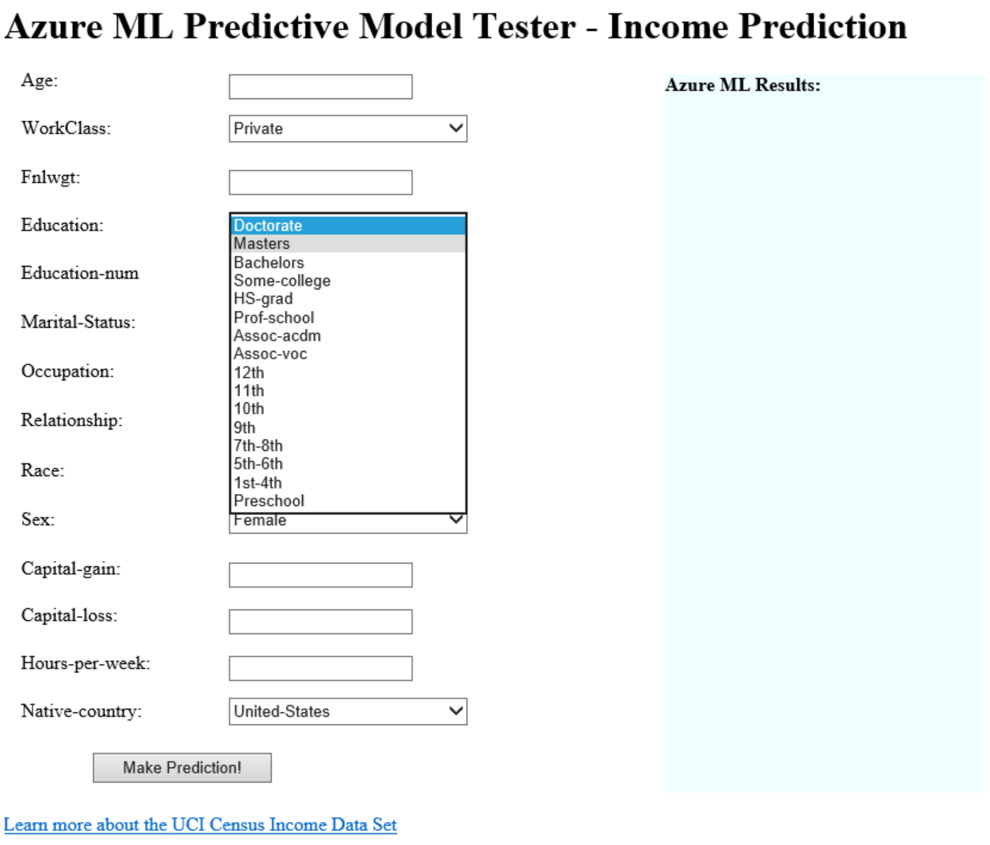
You can get a listing of all of the allowable values for each parameter from the original location where we obtained our sample dataset, at the UCI Machine Learning Repository from the link at http://archive.ics.uci.edu/ml/datasets/Census+Income.



```
<label>Education:</label>
 <asp:DropDownList ID="Education" class="lbox" runat="server" AppendDataBoundItems="true" DataTextField="Company_Name" DataValueField="id">
       <asp:ListItem Text="Doctorate" Value="Doctorate" />
       <asp:ListItem Text="Masters" Value="Masters" />
       <asp:ListItem Text="Bachelors" Value="Bachelors" />
       <asp:ListItem Text="Some-college" Value="Some-college" />
       <asp:ListItem Text="HS-grad" Value="HS-grad" />
       <asp:ListItem Text="Prof-school" Value="Prof-school" />
       <asp:ListItem Text="Assoc-acdm" Value="Assoc-acdm" />
       <asp:ListItem Text="Assoc-voc" Value="Assoc-voc" />
       <asp:ListItem Text="12th" Value="12th" />
       <asp:ListItem Text="11th" Value="11th" />
       <asp:ListItem Text="10th" Value="10th" />
       <asp:ListItem Text="9th" Value="9th" />
       <asp:ListItem Text="7th-8th" Value="7th-8th" />
       <asp:ListItem Text="5th-6th" Value="5th-6th" />
       <asp:ListItem Text="1st-4th" Value="1st-4th" />
       <asp:ListItem Text="Preschool" Value="Preschool" />
</asp:DropDownList>
```

FIGURE 4-22 Default.aspx file showing the markup for a drop-down list box for selecting education level.

We end up with a much more user-friendly interface that facilitates the quick input of various (and valid) parameter input values to test our Azure Machine Learning web service. Figure 4-23 shows a screenshot of the new user interface.



Azure ML Predictive Model Tester - Income Prediction

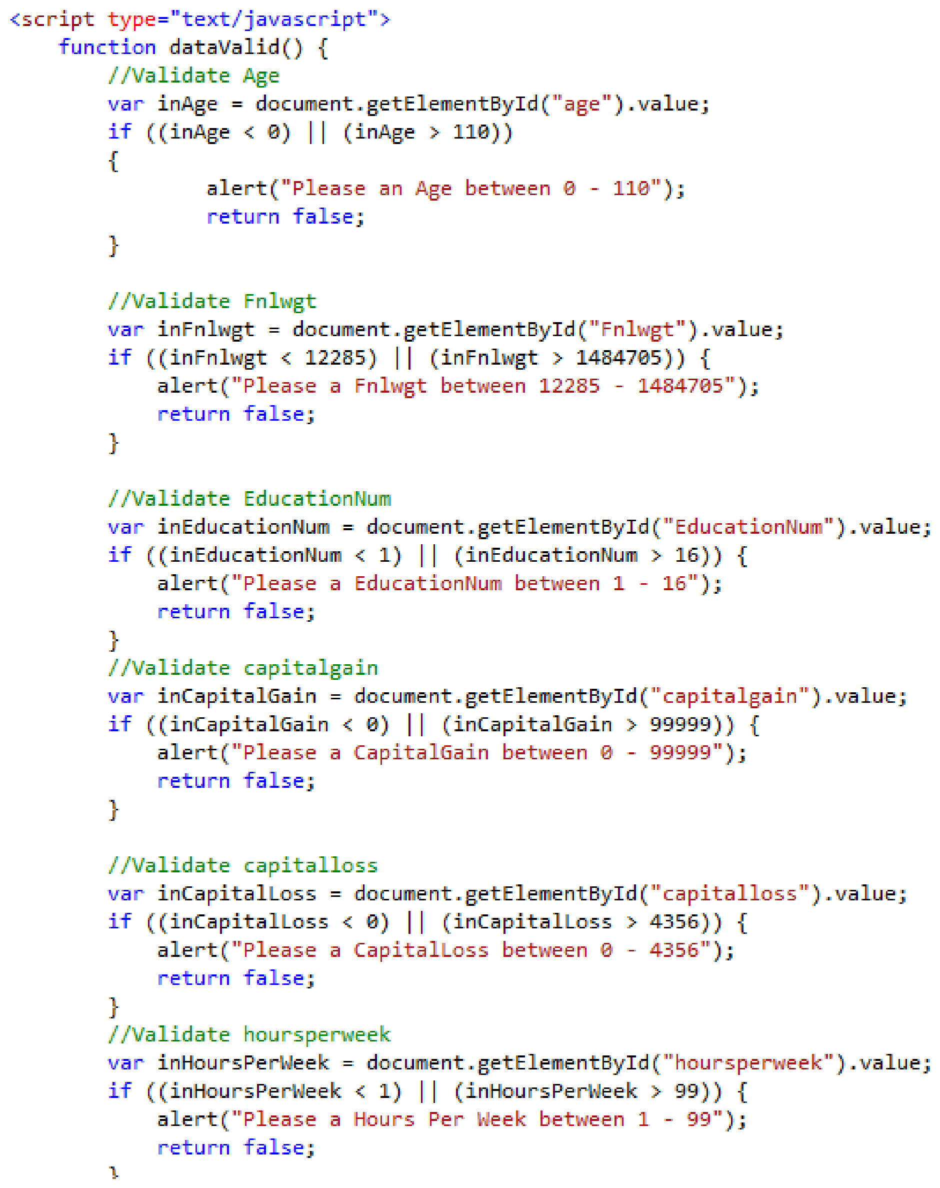
Age:
WorkClass: Private
Fnlwgt:
Education: Doctorate
Education-num
Marital-Status:
Occupation:
Relationship:
Race:
Sex: Female
Capital-gain:
Capital-loss:
Hours-per-week:
Native-country: United-States

Azure ML Results:

FIGURE 4-23 ASP.NET web form showing a drop-down list box for selecting education levels.

Validating the user input

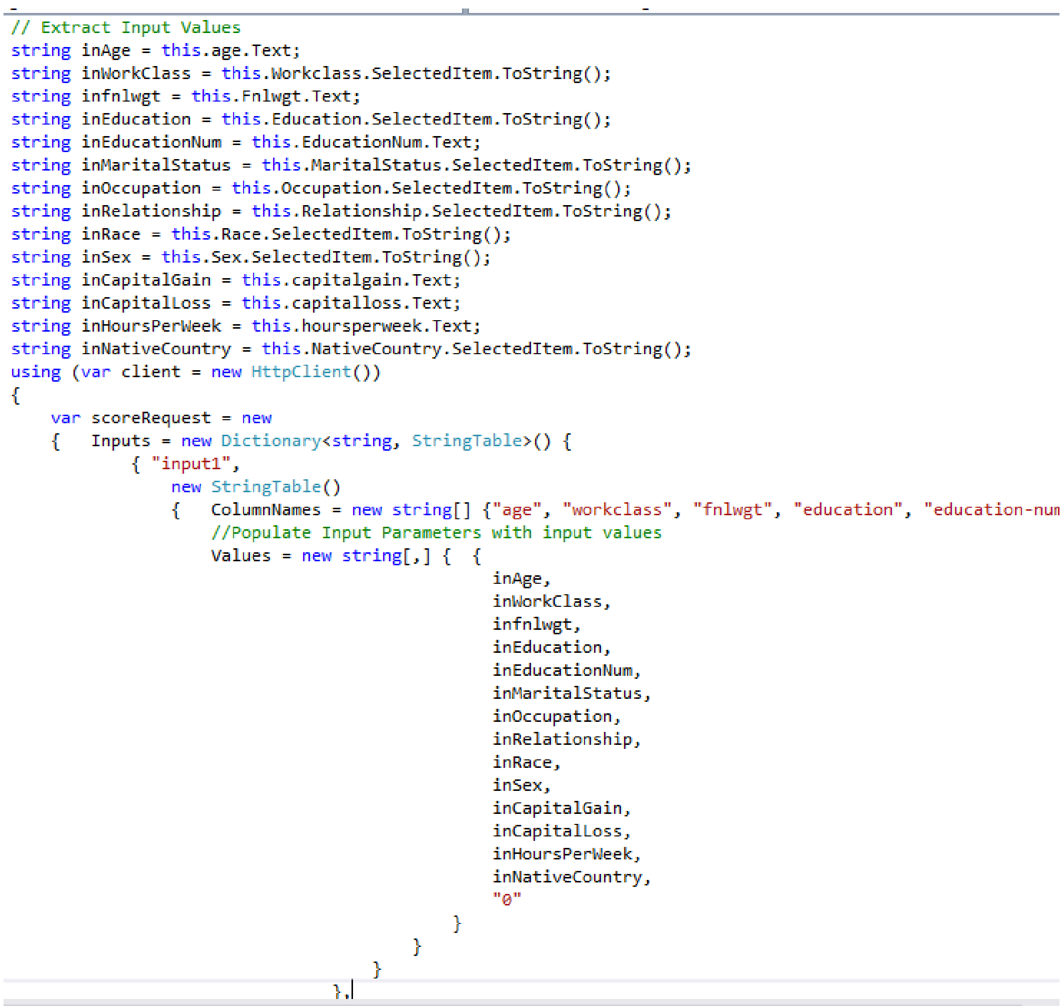
To further validate our user-provided input, we can also insert client-side JavaScript code, as shown in Figure 4-24, to further validate the input values, before we call our Azure Machine Learning web service.



```
<script type="text/javascript">
    function dataValid() {
        //Validate Age
        var inAge = document.getElementById("age").value;
        if ((inAge < 0) || (inAge > 110))
        {
                alert("Please an Age between 0 - 110");
                return false;
        }

        //Validate Fnlwgt
        var inFnlwgt = document.getElementById("Fnlwgt").value;
        if ((inFnlwgt < 12285) || (inFnlwgt > 1484705)) {
            alert("Please a Fnlwgt between 12285 - 1484705");
            return false;
        }

        //Validate EducationNum
        var inEducationNum = document.getElementById("EducationNum").value;
        if ((inEducationNum < 1) || (inEducationNum > 16)) {
            alert("Please a EducationNum between 1 - 16");
            return false;
        }
        //Validate capitalgain
        var inCapitalGain = document.getElementById("capitalgain").value;
        if ((inCapitalGain < 0) || (inCapitalGain > 99999)) {
            alert("Please a CapitalGain between 0 - 99999");
            return false;
        }

        //Validate capitalloss
        var inCapitalLoss = document.getElementById("capitalloss").value;
        if ((inCapitalLoss < 0) || (inCapitalLoss > 4356)) {
            alert("Please a CapitalLoss between 0 - 4356");
            return false;
        }
        //Validate hoursperweek
        var inHoursPerWeek = document.getElementById("hoursperweek").value;
        if ((inHoursPerWeek < 1) || (inHoursPerWeek > 99)) {
            alert("Please a Hours Per Week between 1 - 99");
            return false;
        }
```

FIGURE 4-24 Client-side JavaScript to validate the input fields for the ASP.NET web form page.

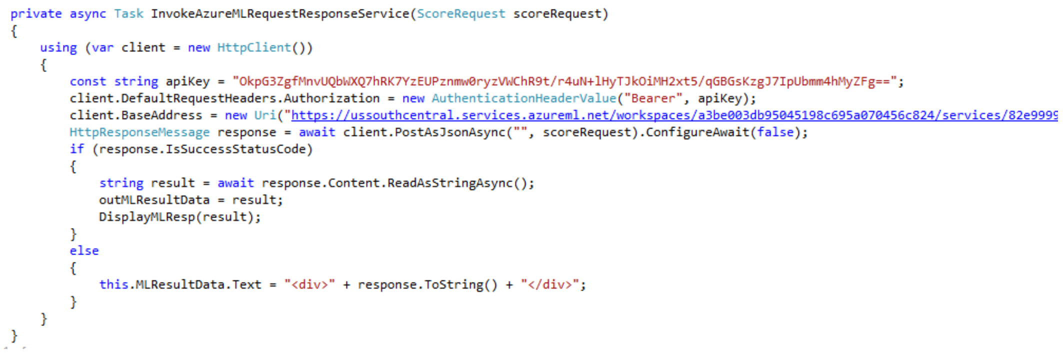
When the user clicks the Make Prediction command button on the ASP.NET web form page, we simply populate the Values string array with the various input values from the webpage as inputs, as shown in Figure 4-25.



```
// Extract Input Values
string inAge = this.age.Text;
string inWorkClass = this.Workclass.SelectedItem.ToString();
string infnlwgt = this.Fnlwgt.Text;
string inEducation = this.Education.SelectedItem.ToString();
string inEducationNum = this.EducationNum.Text;
string inMaritalStatus = this.MaritalStatus.SelectedItem.ToString();
string inOccupation = this.Occupation.SelectedItem.ToString();
string inRelationship = this.Relationship.SelectedItem.ToString();
string inRace = this.Race.SelectedItem.ToString();
string inSex = this.Sex.SelectedItem.ToString();
string inCapitalGain = this.capitalgain.Text;
string inCapitalLoss = this.capitalloss.Text;
string inHoursPerWeek = this.hoursperweek.Text;
string inNativeCountry = this.NativeCountry.SelectedItem.ToString();
using (var client = new HttpClient())
{
    var scoreRequest = new
    {   Inputs = new Dictionary<string, StringTable>() {
            { "input1",
                new StringTable()
                {   ColumnNames = new string[] {"age", "workclass", "fnlwgt", "education", "education-num
                    //Populate Input Parameters with input values
                    Values = new string[,] {  {
                                                 inAge,
                                                 inWorkClass,
                                                 infnlwgt,
                                                 inEducation,
                                                 inEducationNum,
                                                 inMaritalStatus,
                                                 inOccupation,
                                                 inRelationship,
                                                 inRace,
                                                 inSex,
                                                 inCapitalGain,
                                                 inCapitalLoss,
                                                 inHoursPerWeek,
                                                 inNativeCountry,
                                                 "0"
                                             }
                                         }
                                     }
                                 },
```

FIGURE 4-25 Capturing the user inputs in the ASP.NET web form application and assembling the payload to call the Azure Machine Learning web service.

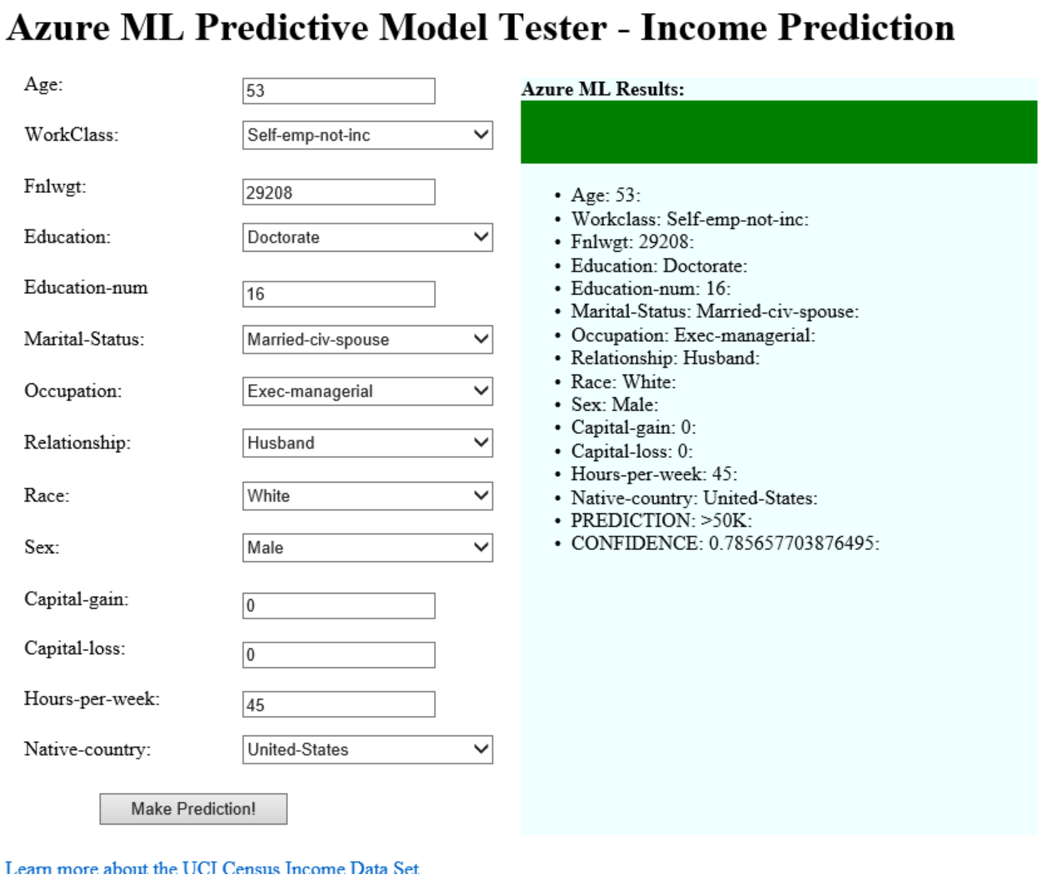
We can then invoke our call to the Azure Machine Learning web service using a similar code block as the one from the C# console application, as shown in Figure 4-26.



```
private async Task InvokeAzureMLRequestResponseService(ScoreRequest scoreRequest)
{
    using (var client = new HttpClient())
    {
        const string apiKey = "OkpG3ZgfMnvUQbWXQ7hRK7YzEUPznmw0ryzVWChR9t/r4uN+lHyTJkOiMH2xt5/qGBGsKzgJ7IpUbmm4hMyZFg==";
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
        client.BaseAddress = new Uri("https://ussouthcentral.services.azureml.net/workspaces/a3be003db95045198c695a070456c824/services/82e999
        HttpResponseMessage response = await client.PostAsJsonAsync("", scoreRequest).ConfigureAwait(false);
        if (response.IsSuccessStatusCode)
        {
            string result = await response.Content.ReadAsStringAsync();
            outMLResultData = result;
            DisplayMLResp(result);
        }
        else
        {
            this.MLResultData.Text = "<div>" + response.ToString() + "</div>";
        }
    }
}
```

FIGURE 4-26 Invoking the Azure Machine Learning web service from an ASP.NET client.

Figure 4-27 shows how our completed Azure Machine Learning web service Tester ASP.NET webpage now looks with all the input parameters and easy-to-use drop-down lists on the left side. The right side of the screen shows the results returned from the Azure Machine Learning web service call.


Azure ML Predictive Model Tester - Income Prediction

Age:	53
WorkClass:	Self-emp-not-inc
Fnlwgt:	29208
Education:	Doctorate
Education-num	16
Marital-Status:	Married-civ-spouse
Occupation:	Exec-managerial
Relationship:	Husband
Race:	White
Sex:	Male
Capital-gain:	0
Capital-loss:	0
Hours-per-week:	45
Native-country:	United-States

Azure ML Results:

- Age: 53:
- Workclass: Self-emp-not-inc:
- Fnlwgt: 29208:
- Education: Doctorate:
- Education-num: 16:
- Marital-Status: Married-civ-spouse:
- Occupation: Exec-managerial:
- Relationship: Husband:
- Race: White:
- Sex: Male:
- Capital-gain: 0:
- Capital-loss: 0:
- Hours-per-week: 45:
- Native-country: United-States:
- PREDICTION: >50K:
- CONFIDENCE: 0.785657703876495:

Learn more about the UCI Census Income Data Set

FIGURE 4-27 The finished ASP.NET webpage showing a sample Azure Machine Learning web service call and response.

You can see how this type of simple ASP.NET webpage application can make it far easier to interact with our Azure Machine Learning web service to perform basic testing operations via the use of basic UI controls like list boxes and client-side validation. This approach also gets around the current support restriction for CORS capabilities in Azure Machine Learning web services as the ASP.NET implementation is all done on the server.

The other advantage of this approach is that it allows you to quickly create and deploy an ASP.Net web application to the cloud to help test your Azure Machine Learning prediction model over the Internet. This means your teams can help evaluate your predictive models even faster.

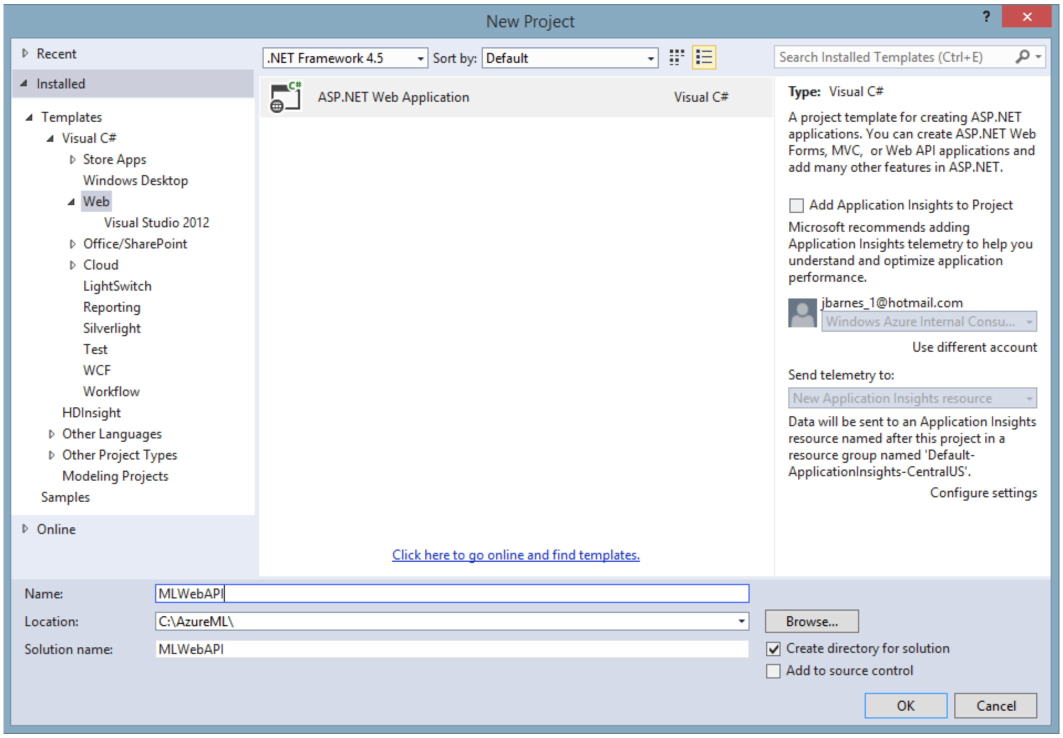
Create a web service using ASP.NET Web API

The HTTP protocol can be used for much more than just serving up webpages. It is also a powerful platform for building APIs that can expose services and data. Almost every development platform today has an HTTP library, so HTTP services can reach a broad range of clients, including browsers, mobile devices, and traditional desktop applications.

Consequently, the ultimate solution to unlocking your brand new Azure Machine Learning prediction web service is to expose it via your own web service. You can maximize the usage across a wide variety of mobile and web clients.

We accomplish the goal of exposing our new Azure Machine Learning prediction model to the largest number of clients possible by creating a new web service using the ASP.NET Web API. The ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers, desktops, and mobile devices. The ASP.NET Web API makes it very easy, and it is an ideal platform for building HTTP REST applications with the .NET Framework.

Let's get started on our new web service by starting Visual Studio, clicking the File menu, selecting New, and then selecting Project. In the Templates pane, select Installed Templates and expand the Visual C# node. Under Visual C#, select Web. In the list of project templates, select ASP.NET Web Application. Name the project MLWebAPI and click OK. See Figure 4-28.



Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

☐ Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

jbarnes_1@hotmail.com

Windows Azure Internal Consu...

Use different account

Send telemetry to:

New Application Insights resource

Data will be sent to an Application Insights resource named after this project in a resource group named 'Default-ApplicationInsights-CentralUS'.

Configure settings

Name: MLWebAPI

Location: C:\AzureML\

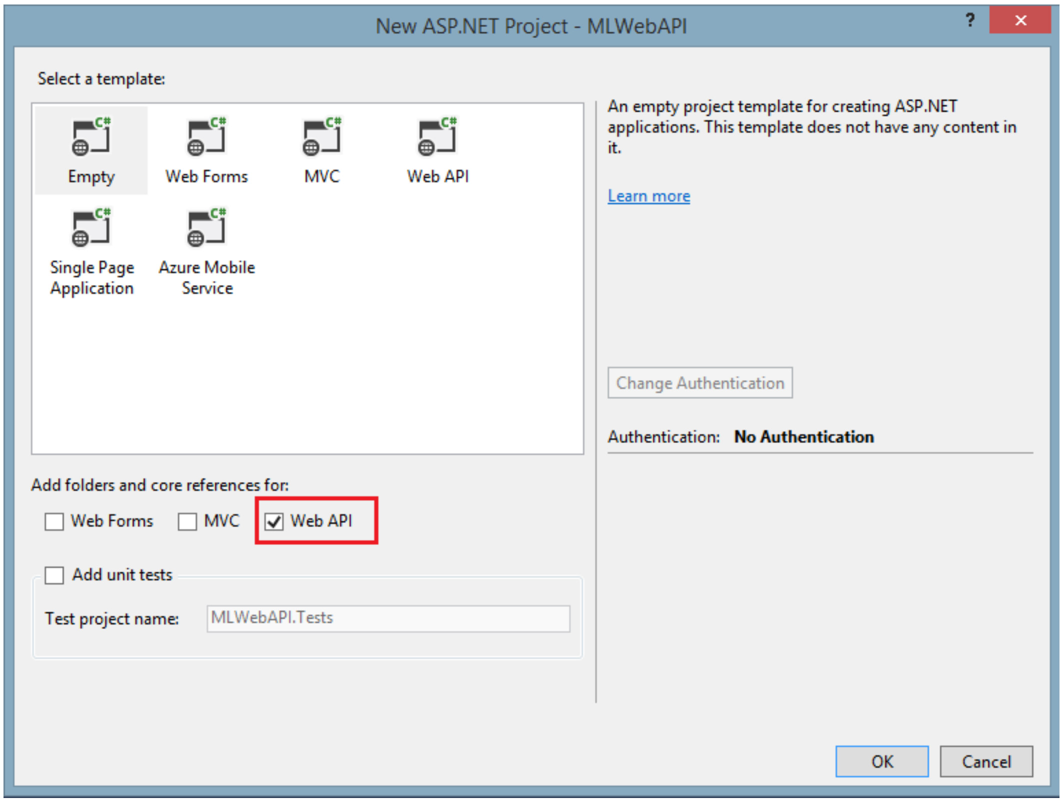
Solution name: MLWebAPI

☑ Create directory for solution

☐ Add to source control

FIGURE 4-28 Creating a new ASP.NET Web API project in Visual Studio 2013.

Next, in the New ASP.NET Project dialog box, select the Empty project template. Under Add Folders And Core References For", select the Web API check box, as shown in Figure 4-29. Click OK.


New ASP.NET Project - MLWebAPI

Select a template:

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Authentication: **No Authentication**

Add folders and core references for:

☐ Web Forms ☐ MVC [✓ Web API]

☐ Add unit tests

Test project name: MLWebAPI.Tests

FIGURE 4-29 Selecting the Visual Studio 2013 options for an empty template and including support for Web API.

Visual Studio 2013 will then create for you a new, blank solution with folders for Controllers and Models along with a WebAPIConfig.cs file in the App_Start folder (see Figure 4-30).

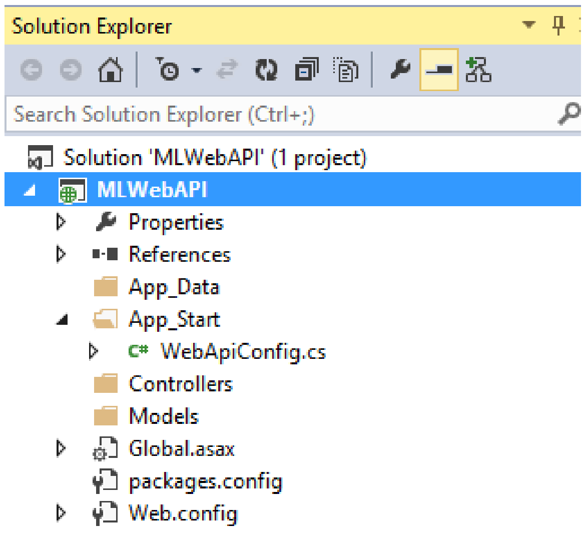


FIGURE 4-30 The Visual Studio project folder structure for a Web API solution.

Let's start by first creating a new model to define the output of our new Web API web service call that will invoke our Azure Machine Learning web service. Right-click the Models folder and then select Add/New Item, as shown in Figure 4-31.

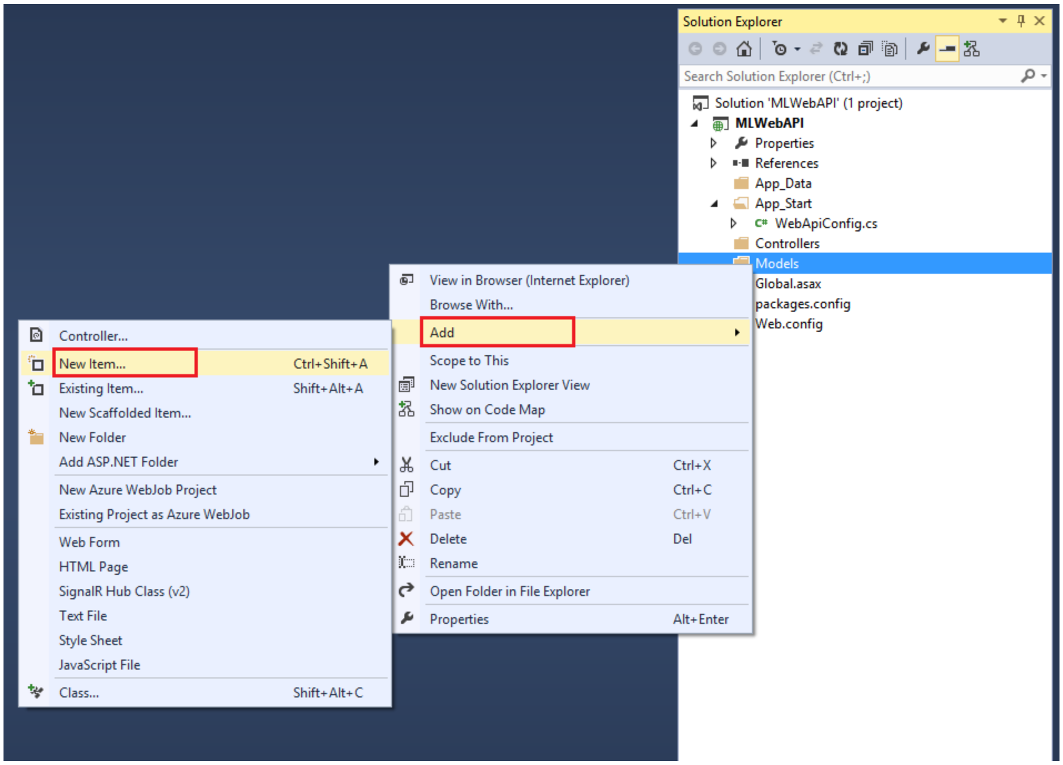


FIGURE 4-31 Adding a new item to the Visual Studio 2013 project.

Next, select Class, as shown in Figure 4-32, and name the file IncomePredictionResults.cs.

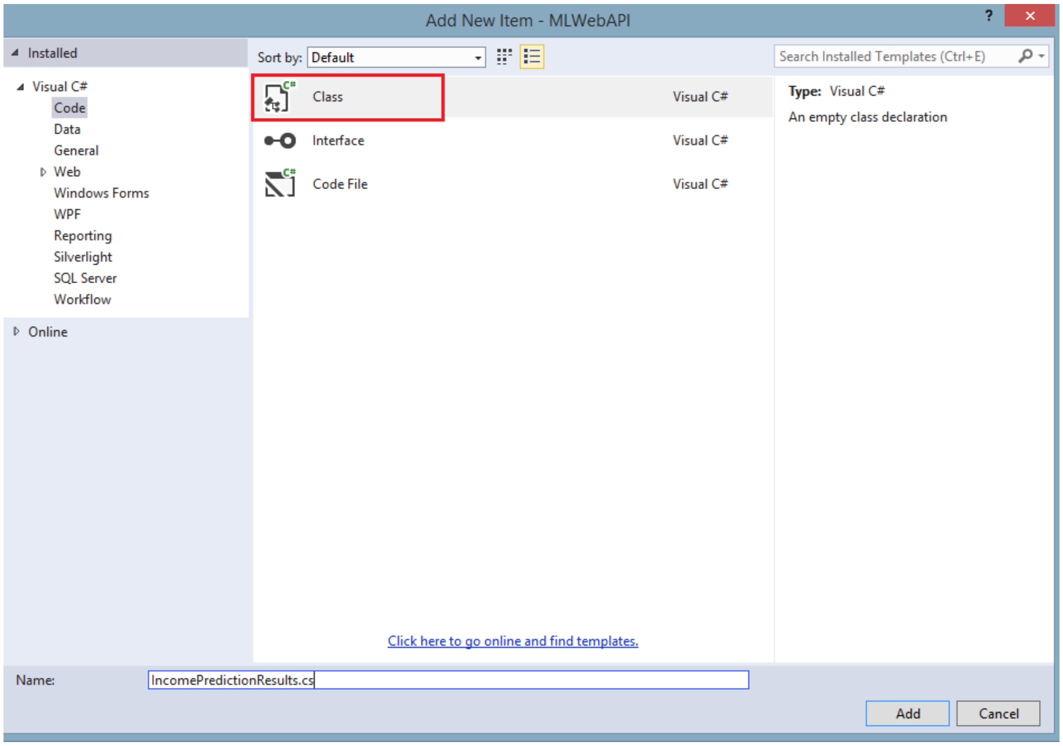
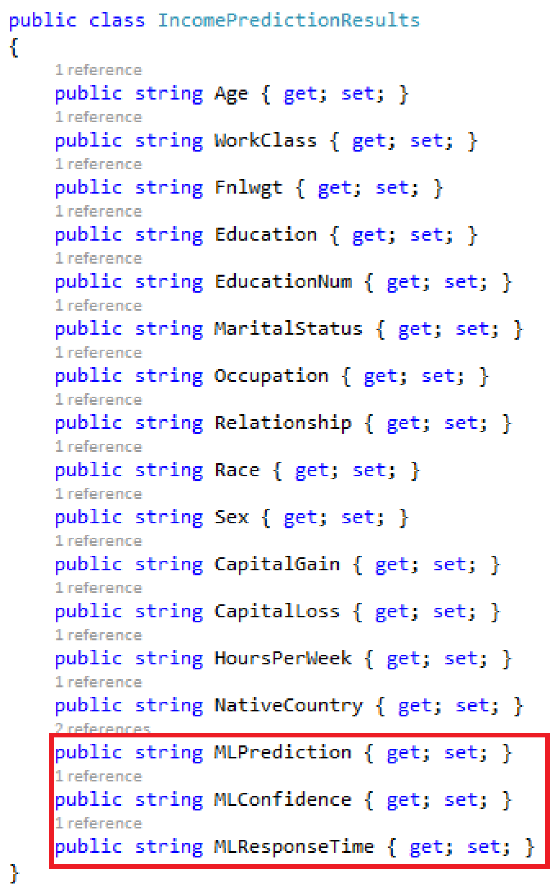


FIGURE 4-32 Adding a new class to the Visual Studio 2013 Web API project.

This creates a new, empty class that we can use to define the result fields of our Web API web service call in JSON format. We will populate this new class with the fields for all of the existing input parameters for our Azure Machine Learning web service call plus three additional fields for the following (see Figure 4-33):

- **MLPrediction** This field contains the prediction results (<=50k or >50K) returned from the call to the Azure Machine Learning web service.
- **MLConfidence** This field contains the confidence factor returned from the call to the Azure Machine Learning web service.
- **MLResponseTime** This is an additional field that we will populate as part of our web service to provide performance metrics on how long the actual call to the Azure Machine Learning web service took to return the prediction results.



```
public class IncomePredictionResults
{
    1 reference
    public string Age { get; set; }
    1 reference
    public string WorkClass { get; set; }
    1 reference
    public string Fnlwgt { get; set; }
    1 reference
    public string Education { get; set; }
    1 reference
    public string EducationNum { get; set; }
    1 reference
    public string MaritalStatus { get; set; }
    1 reference
    public string Occupation { get; set; }
    1 reference
    public string Relationship { get; set; }
    1 reference
    public string Race { get; set; }
    1 reference
    public string Sex { get; set; }
    1 reference
    public string CapitalGain { get; set; }
    1 reference
    public string CapitalLoss { get; set; }
    1 reference
    public string HoursPerWeek { get; set; }
    1 reference
    public string NativeCountry { get; set; }
    2 references
    public string MLPrediction { get; set; }
    1 reference
    public string MLConfidence { get; set; }
    1 reference
    public string MLResponseTime { get; set; }
}
```

FIGURE 4-33 C# class data structure for the returned results from the Web API web service.

In ASP.NET Web API terms, a controller is an object that handles HTTP requests. We'll add a new Controller class to our project by right-clicking the Controller folder in the project solution and then selecting Add and then Controller as shown in Figure 4-34.

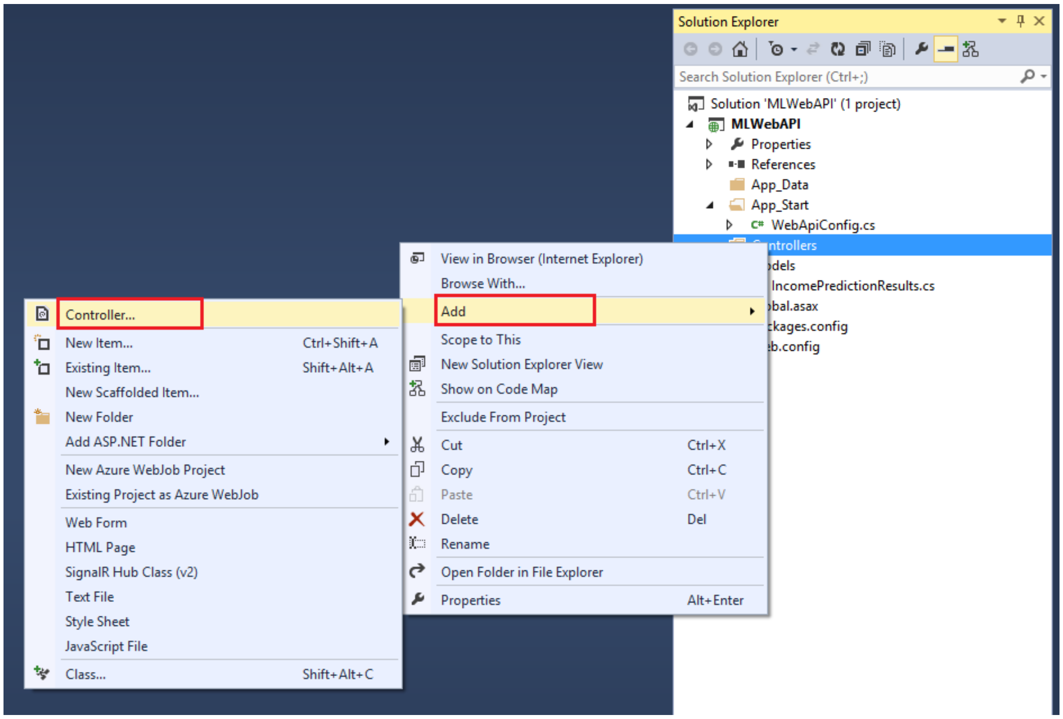
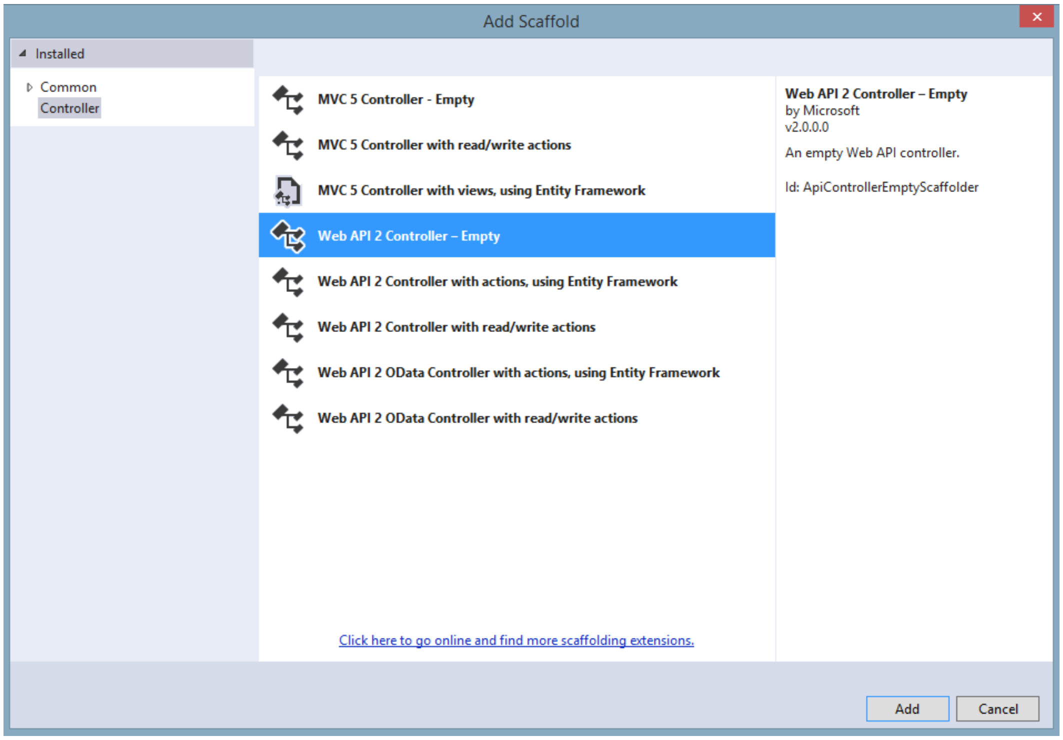


FIGURE 4-34 Adding a new Controller class to the Visual Studio 2013 Web API project.

Next, select the Web API 2 Controller – Empty option as shown in Figure 4-35.


Add Scaffold

Installed
Common
Controller

MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData Controller with actions, using Entity Framework
Web API 2 OData Controller with read/write actions

Web API 2 Controller – Empty
by Microsoft
v2.0.0.0
An empty Web API controller.

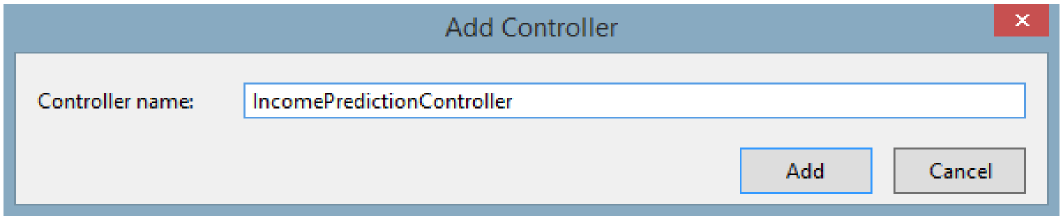
Id: ApiControllerEmptyScaffolder

Click here to go online and find more scaffolding extensions.

Add Cancel

FIGURE 4-35 Adding the empty scaffolding for a Web API 2 controller.

When prompted for the new controller name, enter **IncomePredictionController** and then click Add as shown in Figure 4-36.


Add Controller

Controller name: IncomePredictionController

Add Cancel

FIGURE 4-36 Adding and naming the new IncomePredictionController controller to the project.

This creates a new, empty Web API controller class that we can use for implementing the processing logic and functions required to call the Azure Machine Learning web service. Before we populate that code, let's take care of a few prerequisites.

Enabling CORS support

Browser security prevents a webpage from making AJAX requests to another domain. This restriction is called the same-origin policy, and it prevents a malicious site from reading sensitive data from another site. However, sometimes you might want to let other sites call your web API.

CORS is a W3C web standard that allows a server to relax the same-origin policy. Using CORS, a server can explicitly allow some cross-origin requests and reject others.

One of the key reasons that we are creating our own web service to invoke the Azure Machine Learning web service is to bypass the current Azure Machine Learning restrictions on the use of CORS. As stated earlier, the Azure Machine Learning web services currently only allow for the services to be invoked from a desktop app or server-side web application. To expose and use CORS in our own web service, we need to first install a NuGet package into our project.

To do this, start by right-clicking References for the project and selecting Manage NuGet Packages as shown in Figure 4-37.

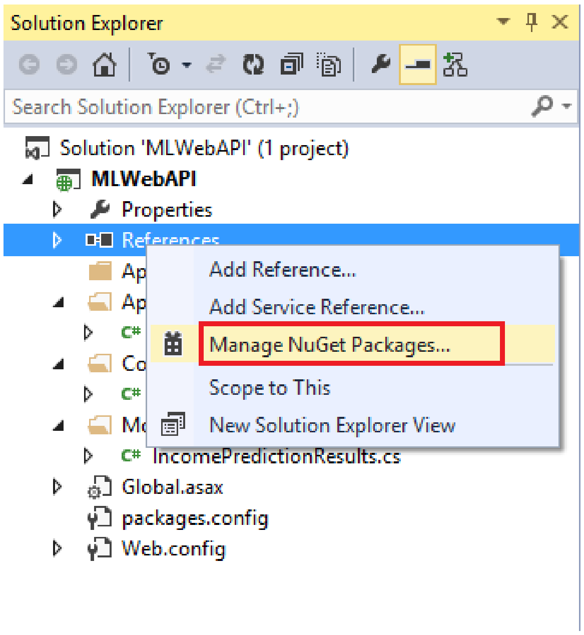
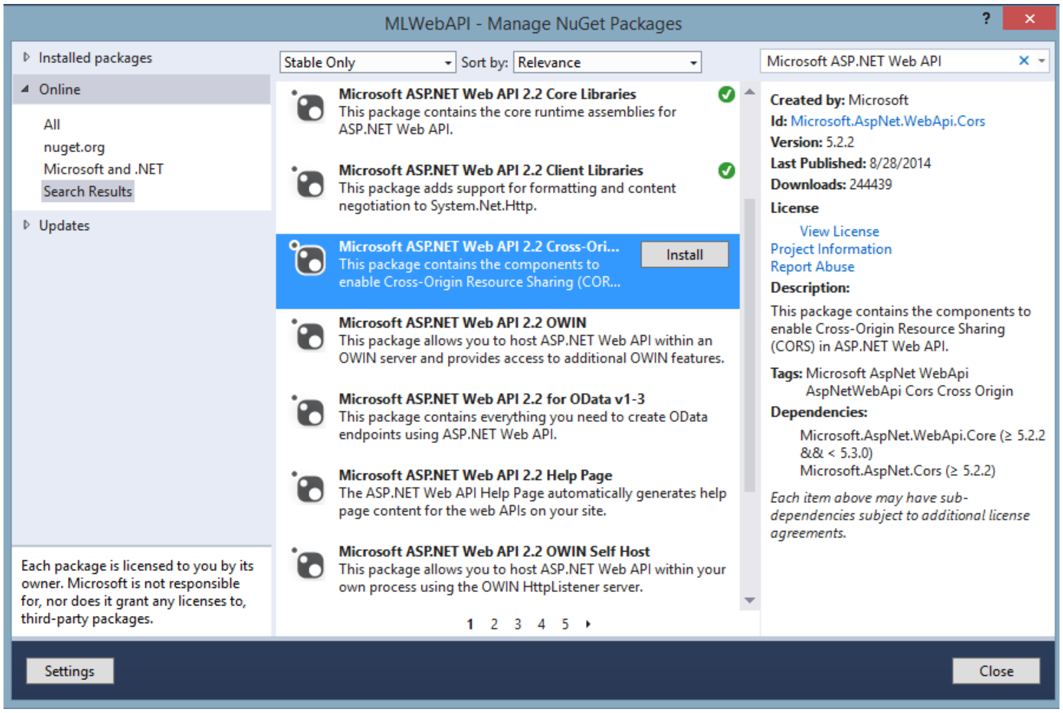


FIGURE 4-37 Selecting the Manage NuGet Packages option in Visual Studio 2013.

Once the NuGet Package Manager starts, select Online on the left side and then type **Microsoft ASP.NET Web API** into the search box in the top right corner. Scroll down through the results until you see the package for Microsoft ASP.NET 2.2 Web API Cross-Origin Resource Sharing (CORS) In ASP.NET Web API as shown in Figure 4-38.



Microsoft ASP.NET Web API 2.2 Core Libraries
This package contains the core runtime assemblies for ASP.NET Web API.

Microsoft ASP.NET Web API 2.2 Client Libraries
This package adds support for formatting and content negotiation to System.Net.Http.

Microsoft ASP.NET Web API 2.2 Cross-Ori...
This package contains the components to enable Cross-Origin Resource Sharing (COR...

Microsoft ASP.NET Web API 2.2 OWIN
This package allows you to host ASP.NET Web API within an OWIN server and provides access to additional OWIN features.

Microsoft ASP.NET Web API 2.2 for OData v1-3
This package contains everything you need to create OData endpoints using ASP.NET Web API.

Microsoft ASP.NET Web API 2.2 Help Page
The ASP.NET Web API Help Page automatically generates help page content for the web APIs on your site.

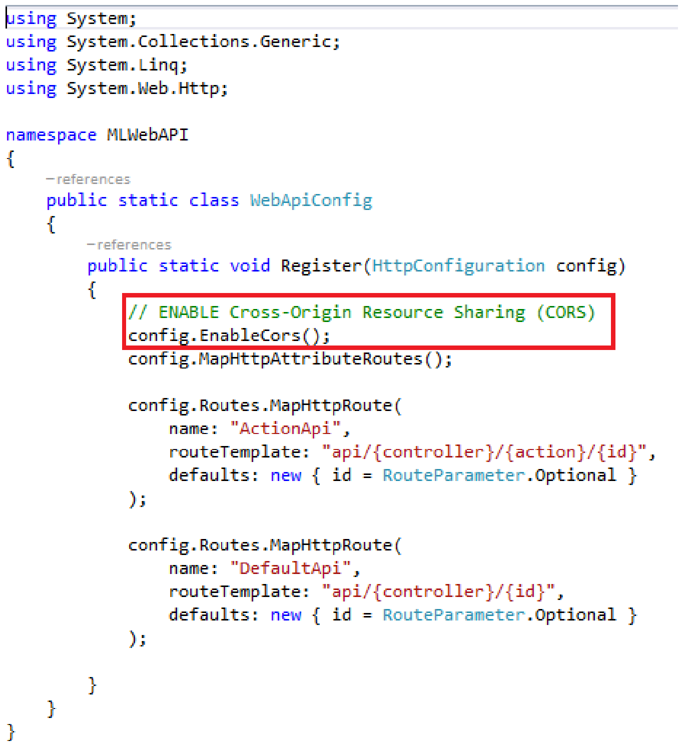
Microsoft ASP.NET Web API 2.2 OWIN Self Host
This package allows you to host ASP.NET Web API within your own process using the OWIN HttpListener server.

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Created by: Microsoft
Id: Microsoft.AspNet.WebApi.Cors
Version: 5.2.2
Last Published: 8/28/2014
Downloads: 244439
License
View License
Project Information
Report Abuse
Description:
This package contains the components to enable Cross-Origin Resource Sharing (CORS) in ASP.NET Web API.
Tags: Microsoft AspNet WebApi AspNetWebApi Cors Cross Origin
Dependencies:
Microsoft.AspNet.WebApi.Core (≥ 5.2.2 && < 5.3.0)
Microsoft.AspNet.Cors (≥ 5.2.2)
Each item above may have sub-dependencies subject to additional license agreements.

FIGURE 4-38 Installing the NuGet package for ASP.NET WEB API CORS support.

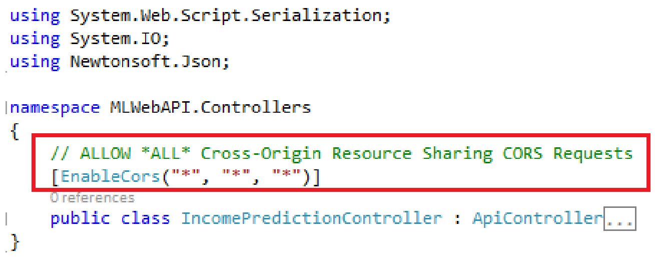
Select the appropriate NuGet package, read and accept the license terms, and install the package into the Visual Studio project. This is a key piece of the solution to allow support for CORS in our web service. Next, in the App_Start folder, open the WebApiConfig.cs file and insert the code snippet to enable CORS support in our web service as shown in Figure 4-39.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace MLWebAPI
{
    -references
    public static class WebApiConfig
    {
        -references
        public static void Register(HttpConfiguration config)
        {
            // ENABLE Cross-Origin Resource Sharing (CORS)
            config.EnableCors();
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "ActionApi",
                routeTemplate: "api/{controller}/{action}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

        }
    }
}
```

FIGURE 4-39 Enabling CORS support in the WebAPIConfig class.

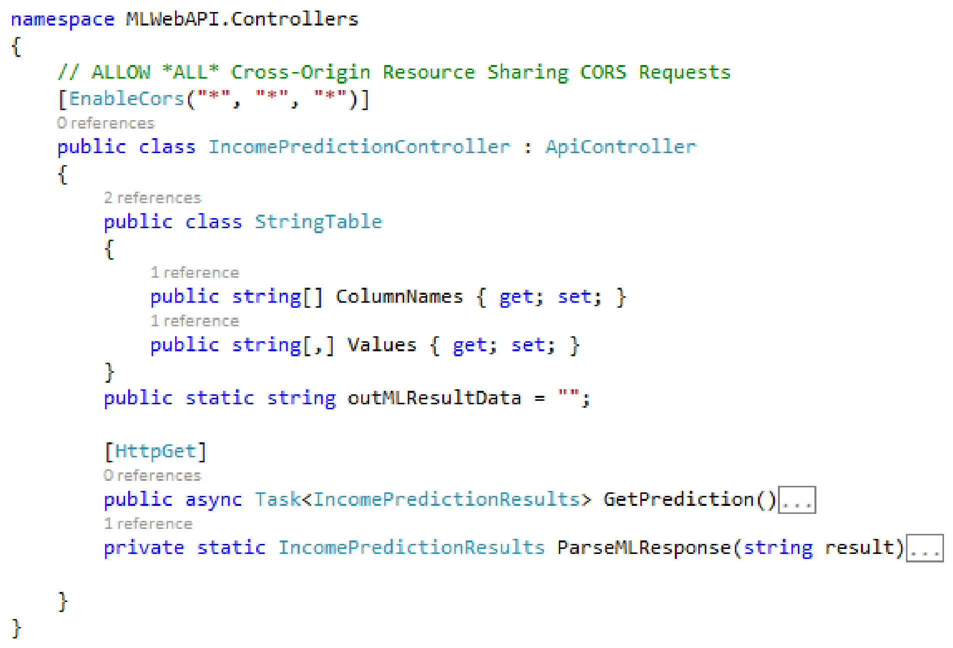
The other task necessary to enable CORS support in our web service is to decorate the IncomePredictionController class with the attributes shown in Figure 4-40.



```
using System.Web.Script.Serialization;
using System.IO;
using Newtonsoft.Json;

namespace MLWebAPI.Controllers
{
    // ALLOW *ALL* Cross-Origin Resource Sharing CORS Requests
    [EnableCors("*", "*", "*")]
    0 references
    public class IncomePredictionController : ApiController...
}
```

FIGURE 4-40 Enabling CORS support in the Web API Controller class.

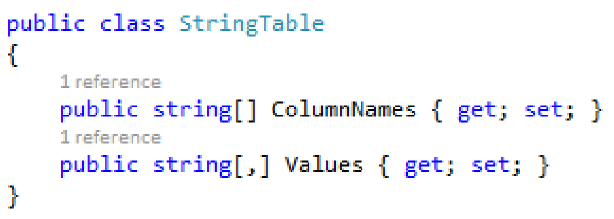
Figure 4-41 shows the high-level structure of our Web API Controller class to handle web requests from our clients.



```
namespace MLWebAPI.Controllers
{
    // ALLOW *ALL* Cross-Origin Resource Sharing CORS Requests
    [EnableCors("*", "*", "*")]
    0 references
    public class IncomePredictionController : ApiController
    {
        2 references
        public class StringTable
        {
            1 reference
            public string[] ColumnNames { get; set; }
            1 reference
            public string[,] Values { get; set; }
        }
        public static string outMLResultData = "";

        [HttpGet]
        0 references
        public async Task<IncomePredictionResults> GetPrediction()...
        1 reference
        private static IncomePredictionResults ParseMLResponse(string result)...

    }
}
```

FIGURE 4-41 The IncomePredictionController C# class.

The StringTable class is used to package the input data parameters into string arrays of name–value pairs before invoking the Azure Machine Learning web service for income predictions, as shown in Figure 4-42.



```
public class StringTable
{
    1 reference
    public string[] ColumnNames { get; set; }
    1 reference
    public string[,] Values { get; set; }
}
```

FIGURE 4-42 The StringTable class.

Processing logic for the Web API web service

In our sample Web API web service, all the work is done in the asynchronous method call named GetPrediction(). Here is the high-level logic for processing our Azure Machine Learning web service request:

- Prepare a new IncomePredictionResults data structure for returning to the caller in JSON format.

- Read and parse the parameters passed in from the HTTP GET request.
- Set up a new StringTable data structure for invoking the Azure Machine Learning web service.
- Populate the StringTable data structure with the input parameters passed in to our web service from the HTTP GET request.
- Start a stopwatch object to start timing the Azure Machine Learning web service request.
- Make the Azure Machine Learning web service request (asynchronously).
- When the Azure Machine Learning web service call returns, stop the stopwatch object and save the elapsed time (in milliseconds) that it took to call the Azure Machine Learning web service and return the results.
- Populate the output data structure with the results of the Azure Machine Learning web service call.
- Return the results to the caller for the Web API web service request.

The following is the C# code for our Income Prediction Web API Controller class.

```
public async Task<IncomePredictionResults> GetPrediction()
{
//Prepare a new ML Response Data Structure for the results
IncomePredictionResults incomePredResults = new IncomePredictionResults();

//Parse the input parameters from the request
NameValueCollection nvc = HttpUtility.ParseQueryString(Request.RequestUri.Query);

//Validate Number of Input parameters (TODO: Add more validations)
if (nvc.Count < 14) { }

// Extract Input Values
string inAge = nvc[0];
string inWorkClass = nvc[1];
string infnlwgt = nvc[2];
string inEducation = nvc[3];
string inEducationNum = nvc[4];
string inMaritalStatus = nvc[5];
string inOccupation = nvc[6];
string inRelationship = nvc[7];
string inRace = nvc[8];
string inSex = nvc[9];
string inCapitalGain = nvc[10];
string inCapitalLoss = nvc[11];
string inHoursPerWeek = nvc[12];
string inNativeCountry = nvc[13];

using (var client = new HttpClient())
{
var scoreRequest = new
```

```
{
Inputs = new Dictionary<string, StringTable>() {
   { "input1",
      new StringTable()
      {  ColumnNames = new string[] {"age", "workclass", "fnlwgt", "education", "education-num",
"marital-status", "occupation", "relationship", "race", "sex", "capital-gain", "capital-loss",
"hours-per-week", "native-country", "income"},
          //Populate Input Parameters with input values
          Values = new string[,] {  {
                                 inAge,
                                 inWorkClass,
                                 infnlwgt,
                                 inEducation,
                                 inEducationNum,
                                 inMaritalStatus,
                                 inOccupation,
                                 inRelationship,
                                 inRace,
                                 inSex,
                                 inCapitalGain,
                                 inCapitalLoss,
                                 inHoursPerWeek,
                                 inNativeCountry,
                                 "0"
                              }
                           }
                        }
                     },
                     },
GlobalParameters = new Dictionary<string, string>()
{
}
};
var sw = new Stopwatch();
const string apiKey =
"Hh8SjZZVxPmoXc4yQl5pIs3sIGEm3Shu71j5hLh/XmFqUampjqdcs4+rOR2KL1WlznG6XfvpRyoFaV5jc2UIYQ==";
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
client.BaseAddress = new
Uri("https://ussouthcentral.services.azureml.net/workspaces/77ee4cb8552b4362b9080caa76b64cb7/ser
vices/43d81b5deb844108b33c42433bfae69d/execute?api-version=2.0&details=true");

//Time the ML Web Service Call
sw.Start();
HttpResponseMessage response = await client.PostAsJsonAsync("",
scoreRequest).ConfigureAwait(false);
sw.Stop();
string elapsed = sw.Elapsed.TotalSeconds.ToString();

//Check Status of Azure ML Web Service Call
if (response.IsSuccessStatusCode)
{
//Read the HTTP response
string MLResp = await response.Content.ReadAsStringAsync();//.ConfigureAwait(false);
//Parse ML Web Service Response and return a populated IncomePredictionResults response record
incomePredResults = ParseMLResponse(MLResp);

//Update for ML Service Response Time
incomePredResults.MLResponseTime = elapsed;
}
else
{
incomePredResults.MLPrediction = response.ReasonPhrase.ToString();
}

client.Dispose();
//return Ok(incomePredResults);
return incomePredResults;
}
}
```

The following code is for a helper method called ParseMLResponse() that will parse results from the Azure Machine Learning web service call and populate the IncomePredictionResults data structure with the returned results.

```
private static IncomePredictionResults ParseMLResponse(string result)
{
   var cleaned = result.Replace("\"", string.Empty);
   cleaned = cleaned.Replace("[", string.Empty);
   cleaned = cleaned.Replace("]", string.Empty);
   string[] mlResultsArr = cleaned.Split(",".ToCharArray());

   IncomePredictionResults incomePredResult = new IncomePredictionResults();
   for (int i = 0; i < mlResultsArr.Length; i++)
   {
      switch (i)
      {
         case 0:
            incomePredResult.Age = mlResultsArr[i].ToString(); break;
         case 1:
            incomePredResult.WorkClass = mlResultsArr[i].ToString(); break;
         case 2:
            incomePredResult.Fnlwgt = mlResultsArr[i].ToString(); break;
         case 3:
            incomePredResult.Education = mlResultsArr[i].ToString(); break;
         case 4:
            incomePredResult.EducationNum = mlResultsArr[i].ToString(); break;
         case 5:
            incomePredResult.MaritalStatus = mlResultsArr[i].ToString(); break;
         case 6:
            incomePredResult.Occupation = mlResultsArr[i].ToString(); break;
         case 7:
            incomePredResult.Relationship = mlResultsArr[i].ToString(); break;
         case 8:
            incomePredResult.Race = mlResultsArr[i].ToString(); break;
         case 9:
                incomePredResult.Sex = mlResultsArr[i].ToString(); break;
            case 10:
                incomePredResult.CapitalGain = mlResultsArr[i].ToString(); break;
            case 11:
                incomePredResult.CapitalLoss = mlResultsArr[i].ToString(); break;
            case 12:
                incomePredResult.HoursPerWeek = mlResultsArr[i].ToString(); break;
            case 13:
                incomePredResult.NativeCountry = mlResultsArr[i].ToString(); break;
            case 14:
                incomePredResult.MLPrediction = mlResultsArr[i].ToString(); break;
            case 15:
                incomePredResult.MLConfidence = mlResultsArr[i].ToString(); break;
            default:
                break;
        }
    }
    return incomePredResult;
}
```

Now we are ready to build and run our new ASP.NET Web API web service. When you first run the solution, you might notice that the browser returns a bad request. ASP.NET Web API services are invoked via the HTTP URL requests. To call our service, we will append the following API parameters to the end of the base URL:

```
/api/IncomePrediction/GetPrediction?age=53….[the rest of the parameters]
```

The following is a complete sample HTTP call to our ASP.NET Web API web service with all the input parameters populated with valid sample data:

```
http://localhost:24462/api/IncomePrediction/GetPrediction?age=53&workclass=Self-emp-not-inc&fnlw
gt=209642&education=HS-grad&education-num=9&marital-status=Married-civ-spouse&occupation=Exec-ma
nagerial&relationship=Husband&race=White&sex=Male&capital-gain=0&capital-loss=0&hours-per-week=4
5&native-country=United-States
```

Now if we test our new web service with a simple browser request, we will be prompted to download a JSON result file, as shown in Figure 4-43.

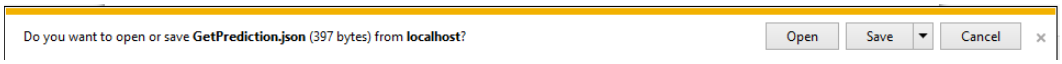

Do you want to open or save **GetPrediction.json** (397 bytes) from **localhost**?

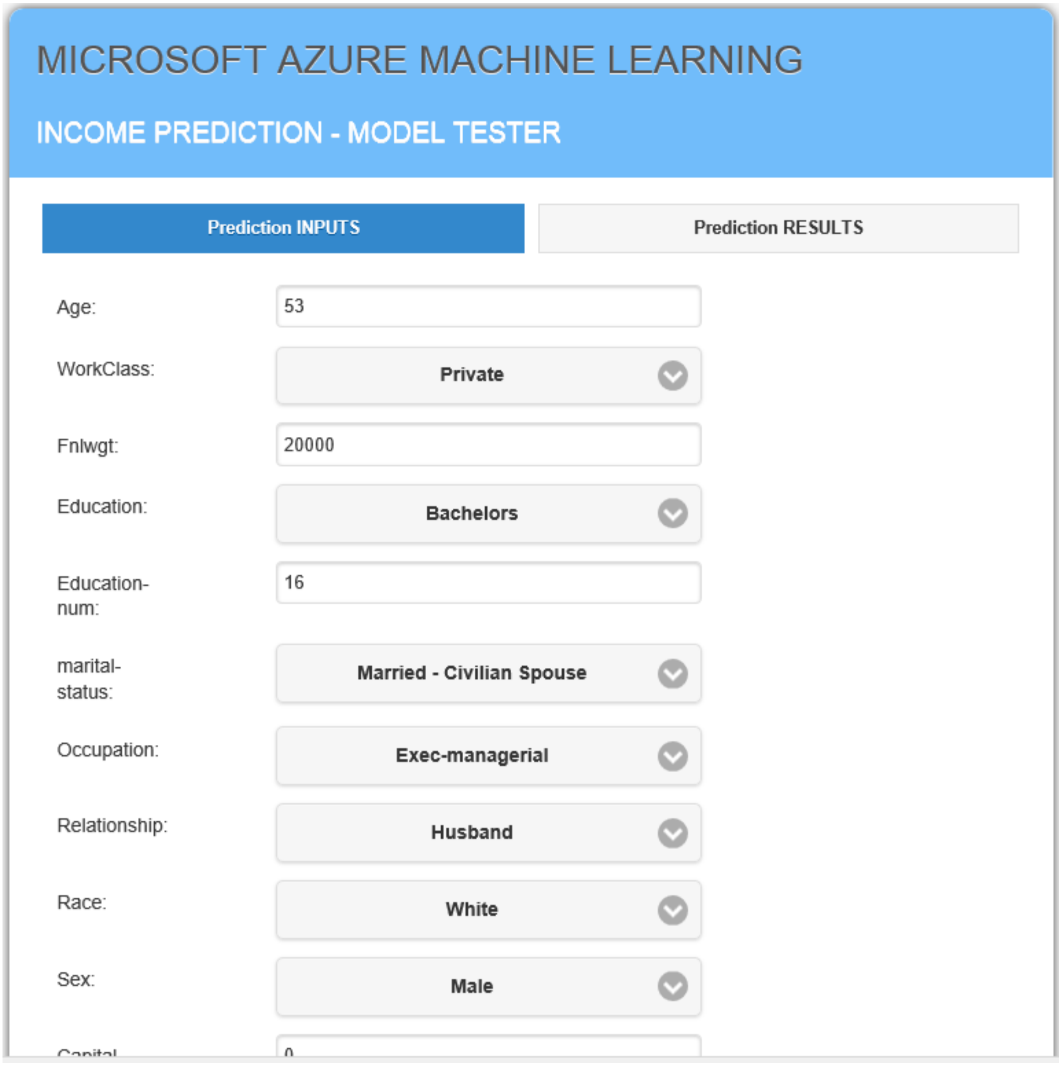
FIGURE 4-43 A JSON file is returned to the browser after a web request to the ASP.NET Web API web service with the income prediction results.

If we open the returned JSON file, we see a nicely formatted JSON data payload with all the input parameters along with the results for the Azure Machine Learning web service call, including the amount of time the Azure Machine Learning web service call actually took:

```
{"Age":"53",
"WorkClass":"Self-emp-not-inc",
"Fnlwgt":"209642",
"Education":"HS-grad",
"EducationNum":"9",
"MaritalStatus":"Married-civ-spouse",
"Occupation":"Exec-managerial",
"Relationship":"Husband",
"Race":"White",
"Sex":"Male",
"CapitalGain":"0",
"CapitalLoss":"0",
"HoursPerWeek":"45",
"NativeCountry":"United-States",
"MLPrediction":">50K",
"MLConfidence":"0.549394130706787",
"MLResponseTime":"1.1816294"
}
```

Now that we have a working ASP.NET Web API web service that can accept and process HTTP GET requests, we can create a simple web client to consume our new ASP.NET Web API web service. One easy way to do this is by creating a simple webpage that uses jQuery Mobile. The jQuery Mobile framework allows you to design a single responsive website or web application that will work on all popular smartphone, tablet, and desktop platforms. You can learn more about jQuery Mobile at http://jquerymobile.com.

Figure 4-44 shows a sample webpage that was built using jQuery Mobile to help facilitate the testing and usage of Azure Machine Learning web services across a multitude of devices.

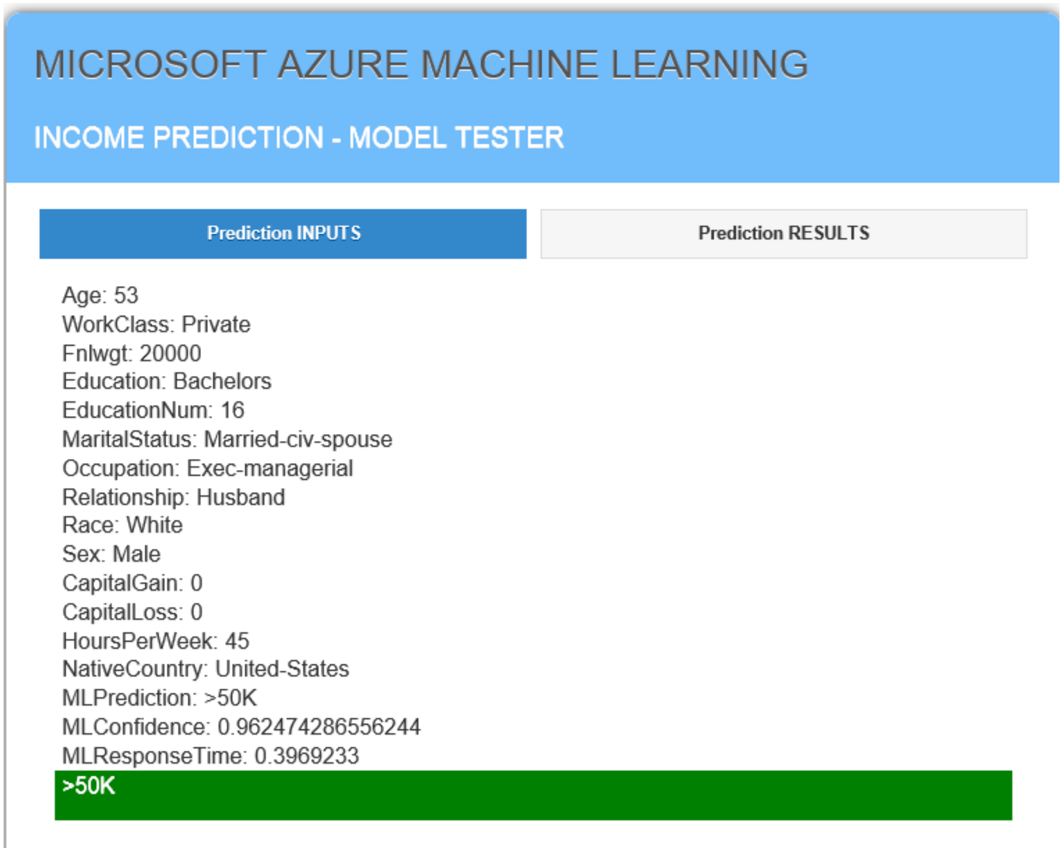

MICROSOFT AZURE MACHINE LEARNING

INCOME PREDICTION - MODEL TESTER

Age:	53
WorkClass:	Private
Fnlwgt:	20000
Education:	Bachelors
Education-num:	16
marital-status:	Married - Civilian Spouse
Occupation:	Exec-managerial
Relationship:	Husband
Race:	White
Sex:	Male
Capital	0

FIGURE 4-44 A jQuery Mobile web application to call our ASP.NET Web API web service.

In Figure 4-45, we can see the Prediction Results tab showing all the original web service call parameters along with the results of the call to our Azure Machine Learning web service for predicting a person's income level based on basic demographic data.


MICROSOFT AZURE MACHINE LEARNING
INCOME PREDICTION - MODEL TESTER
Prediction INPUTS
Prediction RESULTS

```
Age: 53
WorkClass: Private
Fnlwgt: 20000
Education: Bachelors
EducationNum: 16
MaritalStatus: Married-civ-spouse
Occupation: Exec-managerial
Relationship: Husband
Race: White
Sex: Male
CapitalGain: 0
CapitalLoss: 0
HoursPerWeek: 45
NativeCountry: United-States
MLPrediction: >50K
MLConfidence: 0.962474286556244
MLResponseTime: 0.3969233
>50K
```

FIGURE 4-45 The results of a call to our new ASP.NET Web API web service displayed in a jQuery Mobile web application.

The following is the JavaScript code behind the Make Prediction button, which makes the call via AJAX to our exposed web service.

```
$("#btnMakePrediction").click(function (e) {
    e.preventDefault();
    jQuery.support.cors = true;

    //Get Input Variables
    var iAge = document.getElementById("Age").value.toString();
    var iWorkclass = document.getElementById("Workclass").value.toString();
    var iFnlwgt = document.getElementById("Fnlwgt").value.toString();
    var iEducation = document.getElementById("Education").value.toString();
    var iEducationnum = document.getElementById("Education-num").value.toString();
    var iMaritalstatus = document.getElementById("marital-status").value.toString();
    var iOccupation = document.getElementById("occupation").value.toString();
    var iRelationship = document.getElementById("relationship").value.toString();
    var iRace = document.getElementById("race").value.toString();
    var iSex = document.getElementById("sex").value.toString();
    var iCapitalgain = document.getElementById("capital-gain").value.toString();
    var iCapitalloss = document.getElementById("capital-loss").value.toString();
    var iHoursperweek = document.getElementById("hours-per-week").value.toString();
    var iNativecountry = document.getElementById("native-country").value.toString();
    // Make AJAX call to the Web Service
    var jqxhr = $.ajax(
       {
          url: 'http://mlwebservice.azurewebsites.net/api/IncomePrediction/GetPrediction',
          type: "GET",
          data: {
             age: iAge,
             workclass: iWorkclass,
             fnlwgt: iFnlwgt,
             education: iEducation,
             educationnum: iEducationnum,
             maritalstatus: iMaritalstatus,
             occupation: iOccupation,
             relationship: iRelationship,
             race: iRace,
             sex: iSex,
             capitalgain: iCapitalgain,
             capitalloss: iCapitalloss,
             hoursperweek: iHoursperweek,
             nativecountry: iNativecountry
          },
          dataType: "json",
          async: false
       }
       )
       // Process the results of the AJAX call
       .done(function (responseData) {
          var MLStatus = "";
          $("#MLRedResults").html('');
          $.each(responseData, function (index, item) {
             $("#MLRedResults").append("<li>" + index + ": " + item + "</li>");
             if (index == 'MLPrediction')
             {
                MLStatus = item;
             }
          });
       //Render the results
          $("#MLRedResults").append("<label id='MLPredStatus' >" + MLStatus +
"</label><br/><br/><br/>");
          if (MLStatus == '>50K') {
             $('#MLPredStatus').css({ background: "green" })
          }
          if (MLStatus == '<=50K') {
             $('#MLPredStatus').css({ background: "red" })
          }
          //Display the Results Panel
          $("#tabs").tabs("option", "active", 1);
          $("#tabs").tabs({ active: 1 });
```

```
            return false;
         })
         .fail(function (xhr, status, error) {
            var err = eval("(" + xhr.responseText + ")");
            alert(xhr.statusText);
         })
         .always(function () {
            //alert("complete");
         });

      // Set another completion function for the request above
      jqxhr.always(function () {
         //alert("second complete");
      });
   });
```

Summary

In this chapter, we explored how to interact with Azure Machine Learning web services for our income prediction model. We saw how the sample Azure Machine Learning web service provides some great starting code samples for C#, Python, and R for creating stand-alone clients. To that end, we demonstrated calling the Azure Machine Learning prediction web services from simple clients such as a C# console application and a stand-alone R program.

We learned how to further expose our Azure Machine Learning web service via an ASP.NET Web client to overcome the current Azure Machine Learning limitation with CORS requests, CORS being the limitation of a webpage loaded from one domain that is restricted from making AJAX requests to another domain.

Ultimately, by making use of the ASP.NET Web API, we were able to expose our own web service and overcome the existing Azure Machine Learning CORS restriction. This API enabled the widest variety of desktop, web, and mobile clients to consume and test our machine learning predictive model over the Web. In the end, developing an Azure Machine Learning solution is really only valuable if you quickly and easily expose it to a larger population of test or production environments. Only then will you be able to facilitate feedback loops that are crucial to allowing this technology to surpass simple machine training into the world of continuous learning for machines.

Chapter 5

Regression analytics

In the last chapter, we covered a complete, end-to-end Azure Machine Learning solution. We started with a simple Excel file as the training dataset to a fully exposed REST/JSON web service that can be consumed by the widest variety of server, desktop, and mobile clients in use today. The solution focused on enabling an income prediction model using a simple binary classification algorithm. The prediction was considered "binary" due to the dual nature of the prediction type, specifically, whether the data inputs signaled a candidate who was likely to make either more or less than $50,000 a year in income. This prediction solution was an example of a classification algorithm, which is a popular type of supervised learning.

Recall that supervised learning can be separated into two general categories of algorithms:

- **Classification** This category of algorithms is used for predicting responses that can have just a few known values, such as "married," "single," or "divorced," based on the other columns in the dataset. Our earlier example was predicting income <= $50K or > $50K.
- **Regression** These are algorithms that can predict one or more continuous variables, such as profit or loss, based on other columns in the dataset.

Now that we have covered a basic Azure Machine Learning example completely end-to-end using a classification algorithm, the next stop on our tour is to take a deeper look at some of the more advanced machine learning algorithms that are exposed in Azure ML Studio. To that end, we now take a look at implementing regression algorithms in Azure ML Studio.

Linear regression

In this chapter, we explore the theory and practical implementations of linear regression algorithms using Azure ML Studio. In its simplest form, regression can be defined as a method of statistical analysis that analyzes the associations and relationships between one key variable and two or more other variables. It is commonly used to find the relationship between two variables to make accurate predictions.

A deeper look reveals that there are actually three flavors of statistical regression models that can be implemented.

- **Simple linear regression** A single key variable is predicted based on one or more other variables in the dataset.

- **Multiple linear regression** More than one key variable is predicted based on one or more other variables in the dataset.
- **Multivariate linear regression** In this variation of the multiple linear regression algorithm, multiple correlated dependent key variables are predicted, rather than a single key variable.

Regression is a type of inferential algorithm, meaning that it that can be used to draw conclusions about data populations based on random samples from those populations.

Some simple, practical examples of using regression algorithms might include making inferences about the following relationships:

- Body height and shoe size.
- Lifetime sun exposure and number of freckles.
- Home elevation and the cost of flood insurance.
- Phase of the moon and crime statistics in major U.S. cities.
- Breakfast cereal brand and political party affiliation.
- Cloudy weather forecasts and groundhog activity levels.

Linear regression was one of the first types of regression analysis to be widely studied and is used extensively in many practical applications today. Linear regression algorithms are commonly used in business today to analyze key internal and external variables along with market trends. These algorithms are used to make better informed business strategies based on prediction estimates and sales forecasts.

A very simple and common business scenario is to chart sales data in a linear fashion along with monthly sales data on the y axis and time on the x axis. This produces a line that that depicts the upward and downward trends in sales. After creating the trending line, the company could use the slope of the line, along with historical sales data, to forecast sales in future months.

Consumer behavior patterns are also a common application of linear regression algorithms to predict scenarios like the effect of pricing on consumer buying patterns. For example, a company changes the price on a certain product to influence the volume of sales. The company would normally start by recording the quantity of sales at each price level. From there, they could perform a linear regression with quantity sold as the dependent variable and price as the causal variable. The result would be a line that depicts the extent to which consumers reduce their consumption of the product as prices increase, which could help guide future pricing decisions and product strategies.

The insurance industry has been leveraging linear regression methodologies for many years to analyze risk. A common scenario might be for a health insurance company to conduct a linear regression analysis by plotting the number of claims per insured against causal factors such as age. Through this model they could discover that older customers tend to make more health insurance claims. It is easily seen how these types of analysis and predictive capabilities can heavily influence strategic business decisions in an effort to avoid risk and minimize major expenses.

Azure Machine Learning linear regression example

In this section, we look at what it takes to implement a linear regression algorithm in Azure Machine Learning. In this case, we implement a predictive model in which a single key variable is predicted based on one or more other variables in the dataset.

Our next scenario involves predicting the price of an automobile based on the various vehicle features and options selected. At last count, there were more than 254.4 million registered passenger vehicles in the United States. When you think about it, the owner of each of these vehicles had to make a conscious decision about each vehicle's feature trade-offs. Each owner had to decide what was mandatory and what was optional. These decisions had to be made to make a purchasing decision that would fit within the desired price range and budget.

In the past, the vehicle purchasing process was highly facilitated by a car salesperson from the local dealership. The salesperson's job was to match vehicle buyers with vehicles on the lot, based on what the buyer was able to afford.

With the power of Azure Machine Learning and linear regression, we see how this process can be taken to the next level of sophistication. To power this example, we incorporate a sample automobile dataset from the University of California, Irvine (UCI) Machine Learning Repository at the following URLs:

- **Home** https://archive.ics.uci.edu/ml/datasets/Automobile
- **Data download**
  https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data
- **Element column names/values**
  https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.names

Note that this dataset uses automobile feature and price data donated from May 1987. It should be considered outdated with regard to today's current vehicle features and associated pricing. Please use caution if you are using this dataset as a basis for predicting your next vehicle purchase!

In our next experiment, we implement a predictive model in Azure Machine Learning using linear regression to predict the price of a vehicle based on the selection of more than 25 possible features and attributes. The sample automobile dataset contains the following data columns and allowable values for each column:

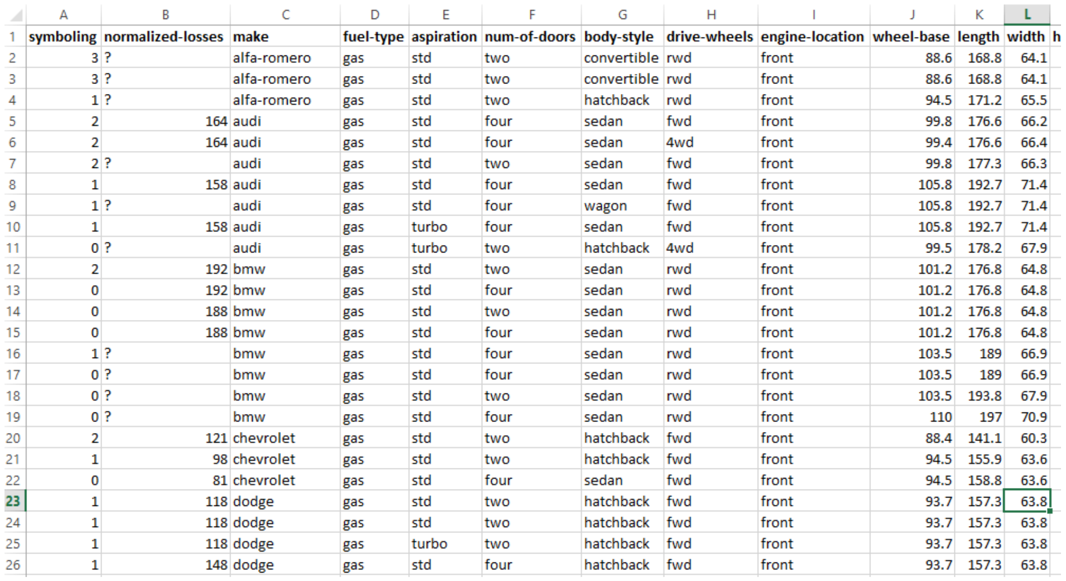
1. **symboling**: -3, -2, -1, 0, 1, 2, 3

2. **normalized-losses**: continuous from 65 to 256
3. **make**: alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugot, plymouth, porsche, renault, saab, subaru, toyota, volkswagen, volvo
4. **fuel-type**: diesel, gas
5. **aspiration**: std, turbo
6. **num-of-doors**: four, two
7. **body-style**: hardtop, wagon, sedan, hatchback, convertible
8. **drive-wheels**: 4wd, fwd, rwd
9. **engine-location**: front, rear
10. **wheel-base**: continuous from 86.6 120.9
11. **length**: continuous from 141.1 to 208.1
12. **width**: continuous from 60.3 to 72.3
13. **height**: continuous from 47.8 to 59.8
14. **curb-weight**: continuous from 1488 to 4066
15. **engine-type**: dohc, dohcv, l, ohc, ohcf, ohcv, rotor
16. **num-of-cylinders**: eight, five, four, six, three, twelve, two
17. **engine-size**: continuous from 61 to 326
18. **fuel-system**: 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi
19. **bore**: continuous from 2.54 to 3.94
20. **stroke**: continuous from 2.07 to 4.17
21. **compression-ratio**: continuous from 7 to 23
22. **horsepower**: continuous from 48 to 288
23. **peak-rpm**: continuous from 4150 to 6600
24. **city-mpg**: continuous from 13 to 49
25. **highway-mpg**: continuous from 16 to 54
26. **price**: continuous from 5118 to 45400

Note that in this case, the Azure Machine Learning linear regression algorithm will calculate a real number, the price of a vehicle. This is a slightly more complicated calculation than the simpler binary classification calculation solution in the last chapter, which predicted whether an individual was likely to have an income level of either <=$50K or >$50K.

Download sample automobile dataset

The first step is to download a sample dataset from the UCI Machine Learning Repository download location at https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data. Save the Imports-85.data file with an extension of .csv and open it in Excel. The next step is to add a new row at the top and insert column headings corresponding to the published data column names form the Imports-85.namefile download. When you are finished, your spreadsheet should look like Figure 5-1.



	A	B	C	D	E	F	G	H	I	J	K	L	
1	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	h
2	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	
3	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	
4	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	
5	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	
6	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	
7	2	?	audi	gas	std	two	sedan	fwd	front	99.8	177.3	66.3	
8	1	158	audi	gas	std	four	sedan	fwd	front	105.8	192.7	71.4	
9	1	?	audi	gas	std	four	wagon	fwd	front	105.8	192.7	71.4	
10	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	192.7	71.4	
11	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	178.2	67.9	
12	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	
13	0	192	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	
14	0	188	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	64.8	
15	0	188	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	64.8	
16	1	?	bmw	gas	std	four	sedan	rwd	front	103.5	189	66.9	
17	0	?	bmw	gas	std	four	sedan	rwd	front	103.5	189	66.9	
18	0	?	bmw	gas	std	two	sedan	rwd	front	103.5	193.8	67.9	
19	0	?	bmw	gas	std	four	sedan	rwd	front	110	197	70.9	
20	2	121	chevrolet	gas	std	two	hatchback	fwd	front	88.4	141.1	60.3	
21	1	98	chevrolet	gas	std	two	hatchback	fwd	front	94.5	155.9	63.6	
22	0	81	chevrolet	gas	std	four	sedan	fwd	front	94.5	158.8	63.6	
23	1	118	dodge	gas	std	two	hatchback	fwd	front	93.7	157.3	63.8	
24	1	118	dodge	gas	std	two	hatchback	fwd	front	93.7	157.3	63.8	
25	1	118	dodge	gas	turbo	two	hatchback	fwd	front	93.7	157.3	63.8	
26	1	148	dodge	gas	std	four	hatchback	fwd	front	93.7	157.3	63.8	

FIGURE 5-1 Automobile feature price data with column headers, ready for upload.

Save the new Excel spreadsheet as Imports-85.data-hdrs.csv to denote that we have added column headings to this dataset.

Upload sample automobile dataset

Now that we have downloaded our sample automobile dataset and added column headings, the next step is to upload it into our Azure ML Studio workspace.

Start by clicking Datasets in the left navigation bar of the Azure ML Studio screen as shown in Figure 5-2.

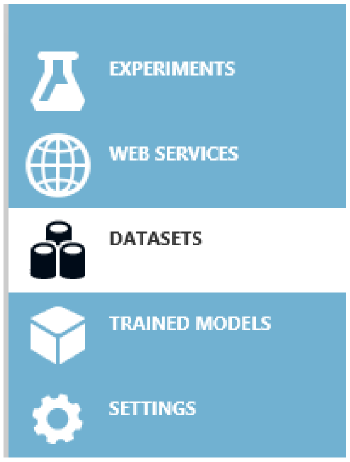


FIGURE 5-2 The Datasets section of Azure ML Studio.

Next click + New on the bottom left of the screen. Then select Dataset in the navigation bar on the left to upload a new dataset from a local file, as shown in Figure 5-3.

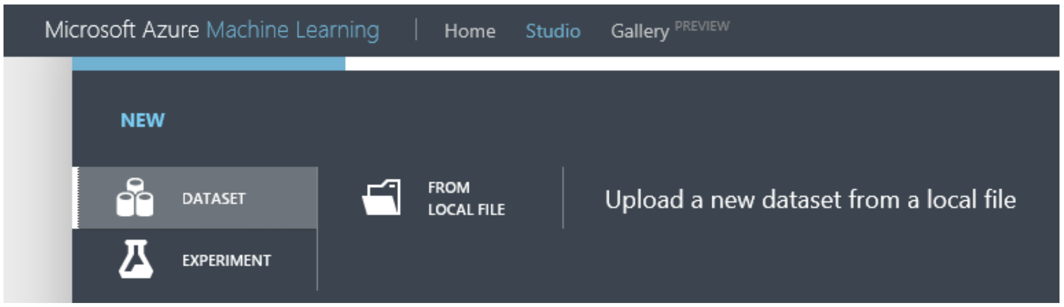
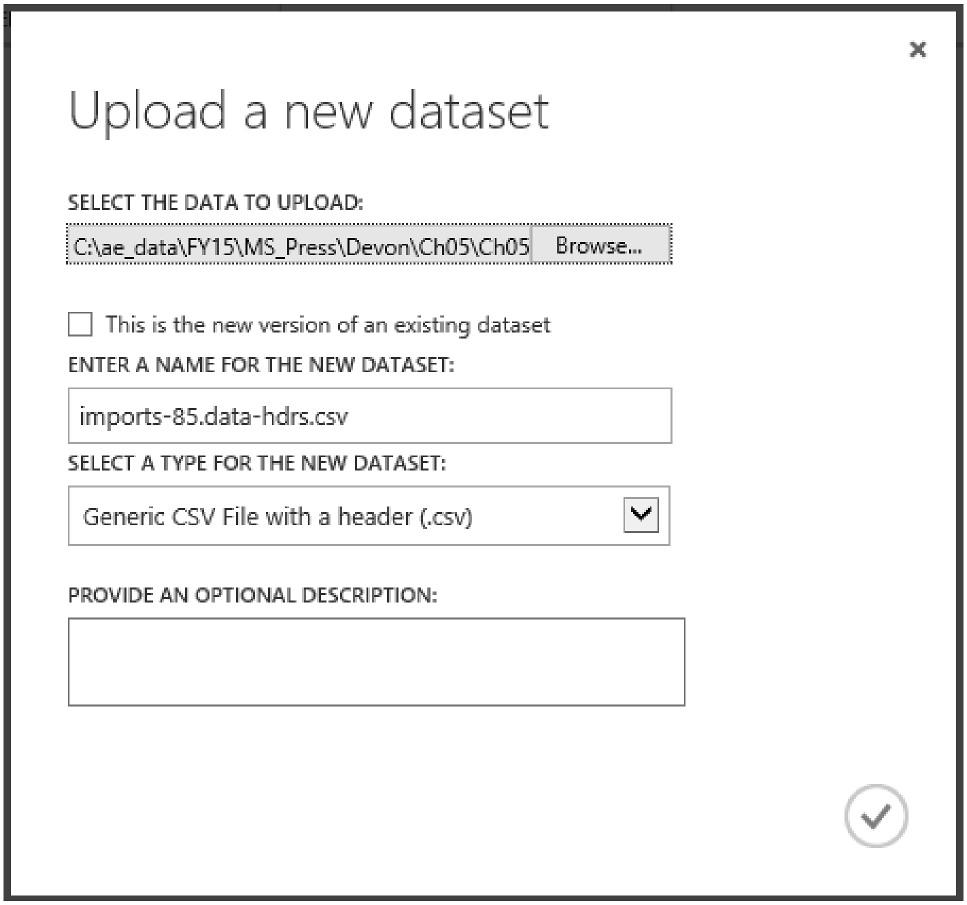

Microsoft Azure Machine Learning
Home
Studio
Gallery PREVIEW
NEW
DATASET
EXPERIMENT
FROM LOCAL FILE
Upload a new dataset from a local file

FIGURE 5-3 The upload dataset FROM LOCAL FILE section of Azure ML Studio.

Next, click From Local File. Browse to the location where you saved the Excel spreadsheet we modified for column headings and saved as Imports-85.data-hdrs.csv.

Select the Generic CSV File With A Header (.csv) type as shown in Figure 5-4.



Upload a new dataset

SELECT THE DATA TO UPLOAD:

C:\ae_data\FY15\MS_Press\Devon\Ch05\Ch05

This is the new version of an existing dataset

ENTER A NAME FOR THE NEW DATASET:

imports-85.data-hdrs.csv

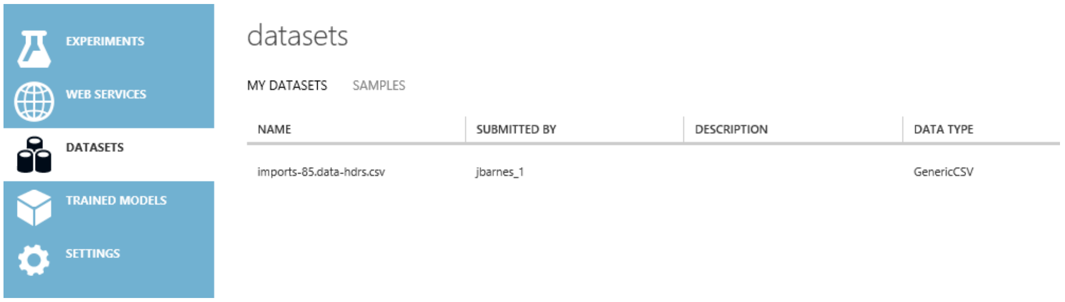
SELECT A TYPE FOR THE NEW DATASET:

Generic CSV File with a header (.csv)

PROVIDE AN OPTIONAL DESCRIPTION:

FIGURE 5-4 The Upload A New Dataset options within Azure ML Studio.

Click the check mark when ready and your dataset will be uploaded to the Azure ML Studio Datasets repository as shown in Figure 5-5.



NAME	SUBMITTED BY	DESCRIPTION	DATA TYPE
imports-85.data-hdrs.csv	jbarnes_1		GenericCSV

FIGURE 5-5 The uploaded Imports-85.data-hdrs.csv dataset in Azure ML Studio.

Create automobile price prediction experiment

Now that we have uploaded our sample automobile dataset, it's time to create our linear regression sample experiment to predict the price of an automobile based on the selected features.

To get started with creating a new experiment, select the Experiments tab on the left navigation bar. Then click the + New icon on the bottom left app bar as shown in Figure 5-6.

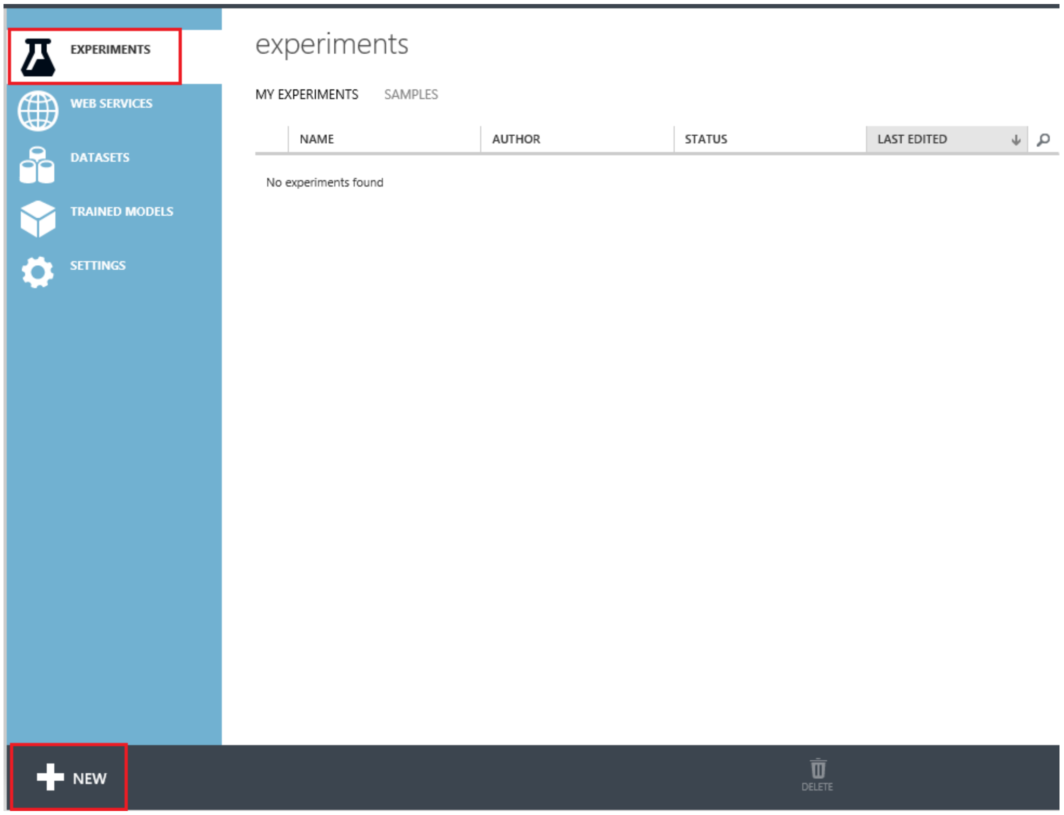


FIGURE 5-6 The Experiments tab of Azure ML Studio ready to add a new experiment.

You will then be presented with the option to start with a blank experiment or chose from a gallery of tutorials and other sample experiments. For our automobile price prediction experiment, start with a blank experiment as shown in Figure 5-7.

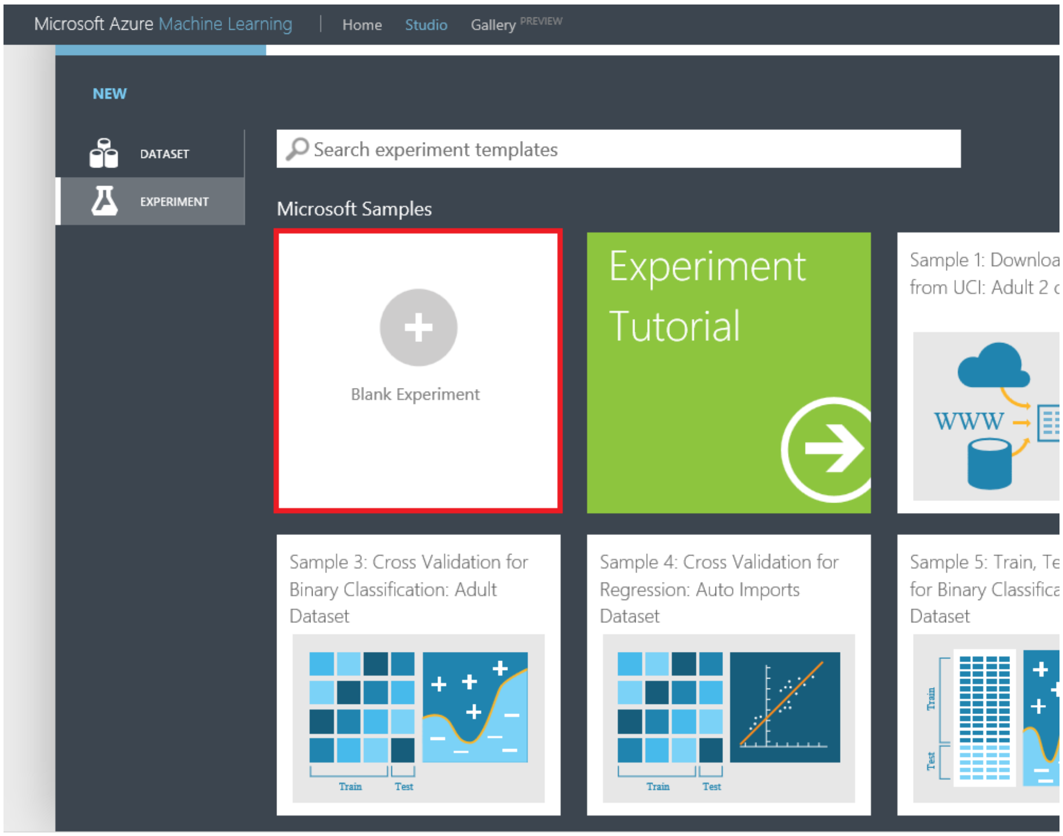


FIGURE 5-7 Azure ML Studio: Selecting a blank experiment.

Azure ML Studio will then create a new blank experiment for you. The blank experiment opens with a drag-and-drop template for filling in the steps of your experiment processing logic. Once created, you can name your experiment Automobile Price Prediction by clicking the experiment's name and entering the new name. Figure 5-8 shows our new blank experiment with the experiment name highlighted.

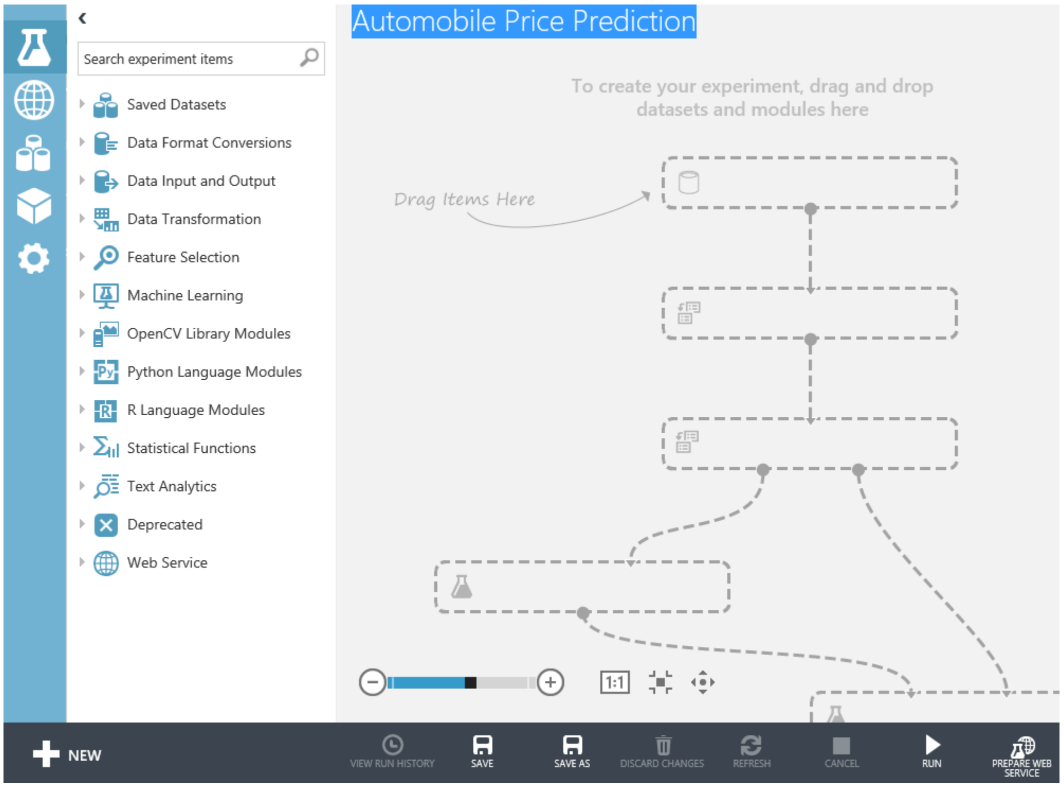


FIGURE 5-8 Azure ML Studio: A new blank experiment for automobile price prediction.

The first step is to drag our new dataset onto the Azure ML Studio canvas. Click the Saved Datasets tab on the left navigation bar. Look for our freshly uploaded dataset named Imports-85.data-hdrs.csv. Drag it onto the top of the designer surface.

Let's use the Visualize utility to examine our dataset by hovering over the bottom connector. Right-click and then select the Visualize command, as shown in Figure 5-9.

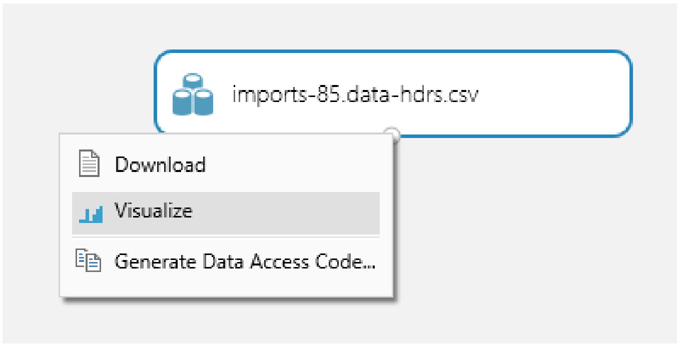
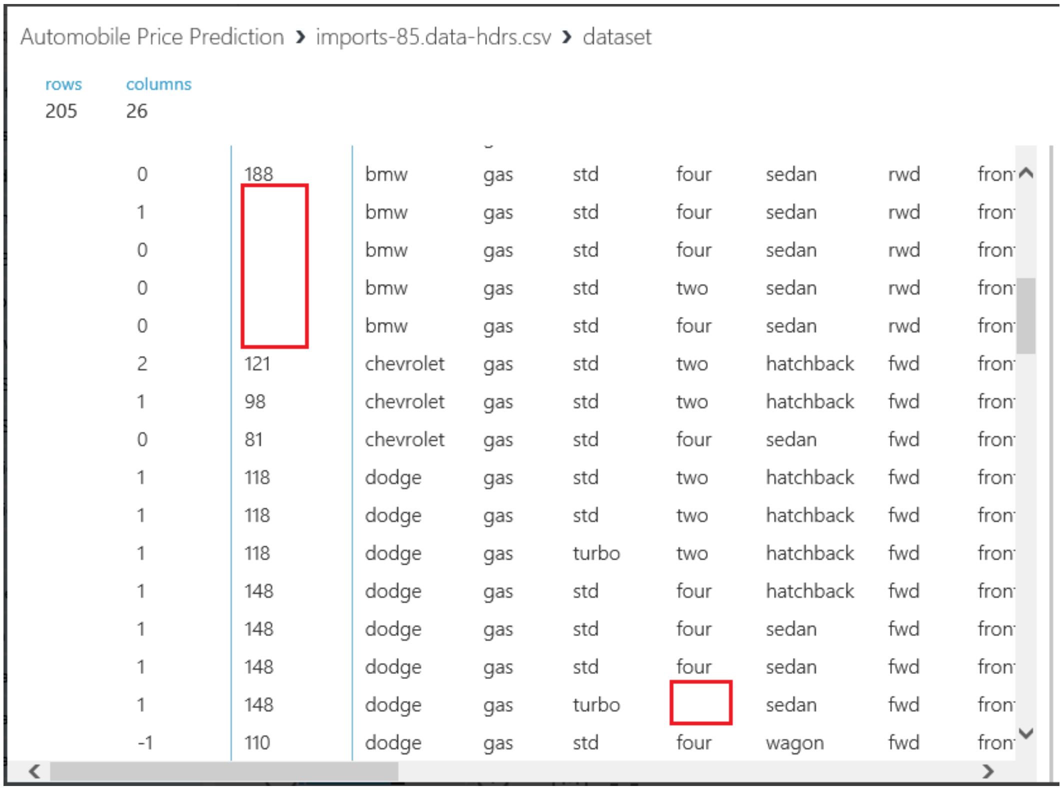


FIGURE 5-9 Selecting the option to visualize the automobile price prediction dataset.

A cursory inspection of the automobile pricing dataset using the Visualize tool reveals that we have a few missing values in our dataset, as shown in Figure 5-10.



Automobile Price Prediction › imports-85.data-hdrs.csv › dataset

rows 205

columns 26

0	188	bmw	gas	std	four	sedan	rwd	fron
1		bmw	gas	std	four	sedan	rwd	fron
0		bmw	gas	std	four	sedan	rwd	fron
0		bmw	gas	std	two	sedan	rwd	fron
0		bmw	gas	std	four	sedan	rwd	fron
2	121	chevrolet	gas	std	two	hatchback	fwd	fron
1	98	chevrolet	gas	std	two	hatchback	fwd	fron
0	81	chevrolet	gas	std	four	sedan	fwd	fron
1	118	dodge	gas	std	two	hatchback	fwd	fron
1	118	dodge	gas	std	two	hatchback	fwd	fron
1	118	dodge	gas	turbo	two	hatchback	fwd	fron
1	148	dodge	gas	std	four	hatchback	fwd	fron
1	148	dodge	gas	std	four	sedan	fwd	fron
1	148	dodge	gas	std	four	sedan	fwd	fron
1	148	dodge	gas	turbo		sedan	fwd	fron
-1	110	dodge	gas	std	four	wagon	fwd	fron

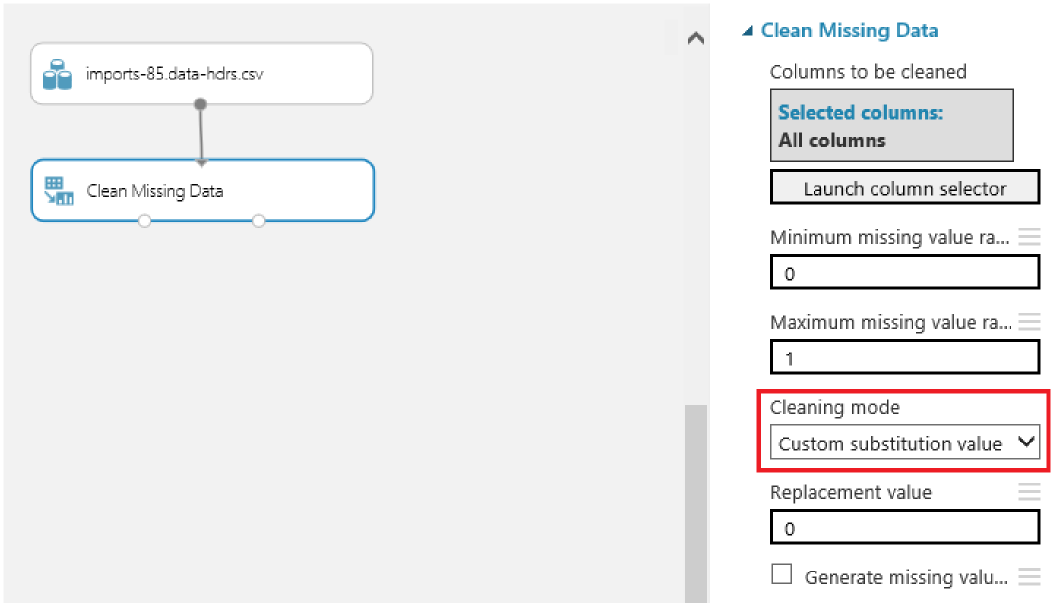
FIGURE 5-10 Missing values in the automobile price prediction dataset.

The answer to this problem is to use an Azure ML Studio module named Clean Missing Data to detect missing values. This module will supply default values instead.

To find and implement the Clean Missing Data module in Azure ML Studio, simply type the name into the search bar on the top of the left navigation bar. Let the Azure ML Studio search function find it for you.

The other method to find this module is to traverse down through the various modules in the left navigation bar. Find and expand the Data Transformation tab. Then expand the Manipulation tab to see the Clean Missing Data module. Once you have found the module, simply drag and drop it underneath the Iimports-85.data-hdrs.csv dataset module.

Next, we configure the Clean Missing Data module to fill in the missing columns of data by using the properties for the module. Figure 5-11 depicts our experiment with the Clean Missing Data module attached.



Clean Missing Data

Columns to be cleaned

Selected columns:
All columns

Minimum missing value ra...
0

Maximum missing value ra...
1

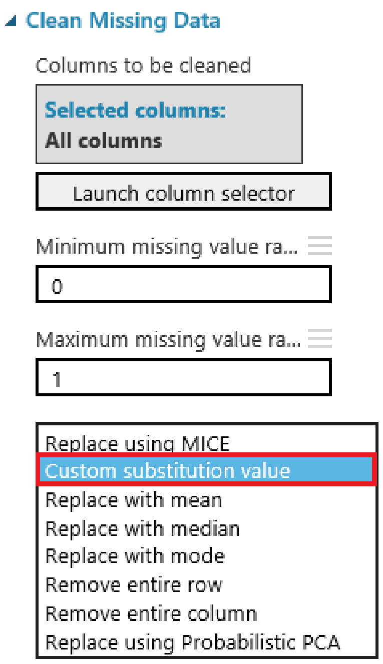
Cleaning mode
Custom substitution value

Replacement value
0

☐ Generate missing valu...

FIGURE 5-11 The Clean Missing Data module configuration parameters.

This is a very handy module in the Azure ML Studio toolbox and allows us to easily handle cases where we have missing or invalid data in our initial dataset. We start the configuration by launching the column selector and including all columns. We can then set the minimum and maximum missing value ranges. Then set the Cleaning Mode. When you click the Cleaning Mode drop-down list, you will see that there are actually eight different cleaning mode options we could use to scrub our data, as shown in Figure 5-12.


Clean Missing Data

Columns to be cleaned

Selected columns:
All columns

Launch column selector

Minimum missing value ra...

0

Maximum missing value ra...

1

Replace using MICE
Custom substitution value
Replace with mean
Replace with median
Replace with mode
Remove entire row
Remove entire column
Replace using Probabilistic PCA

FIGURE 5-12 The Clean Missing Data module configuration parameters showing the options for the Cleaning Mode. parameter.

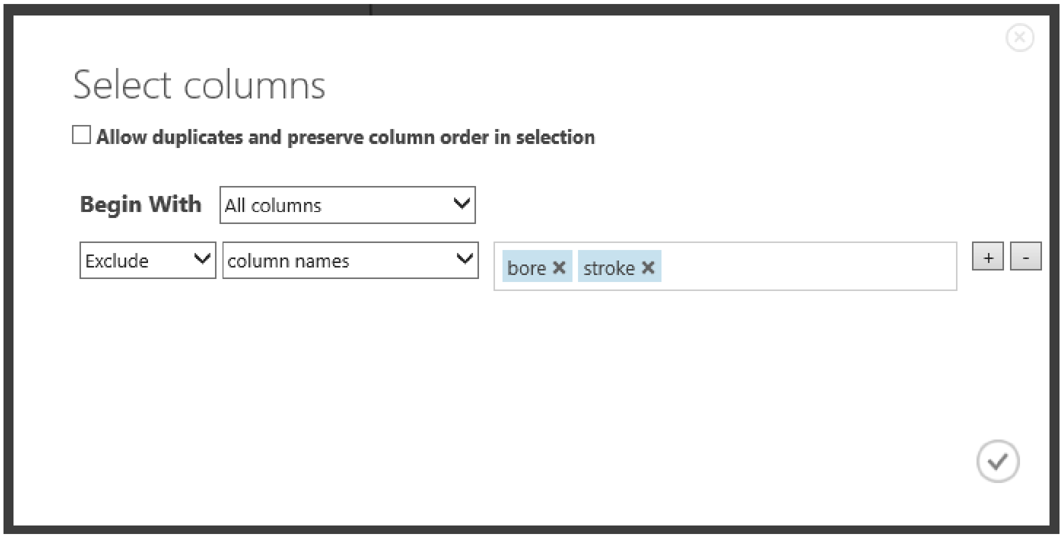
For our purposes in the automobile price prediction experiment, select the Custom Substitution option and designate the replacement value to be 0. This means any column in the initial dataset that has missing data will be automatically updated to have a 0 substituted for the missing column at run time.

The next step in building our automobile price prediction experiment will be to insert a Project Columns module to allow us to selectively include or exclude columns of data from our initial dataset. The Project Columns module is probably one of the most useful and widely used modules in the entire Azure ML Studio module collection.

As before, there are basically two ways to find modules in the Azure ML Studio designer. Search for them or manually traverse through the list of modules on the left side.

To find and implement the Project Columns module in Azure ML Studio, simply type the name into the search bar on the top of the left navigation bar. Let the Azure ML Studio search function find it for you. If you want to find it manually, click the Data Transformation tab and then click the Manipulation subtab.

Once you find the Project Columns module, drag and drop it below the Clean Missing Data module and attach it to one of the bottom two connectors on the Clean Missing Data module. You will note that there is a red exclamation mark displayed in the Project Columns module because the module needs to have its properties configured for including or excluding columns. To do this, simply click the Project Columns module and launch the column selector. Set the Begin With drop-down list to include All Columns. Then select Exclude and Column Names. Select the bore and stroke columns for exclusion. Figure 5-13 displays our configuration options for the Project Columns module.


Select columns

☐ Allow duplicates and preserve column order in selection

Begin With: All columns

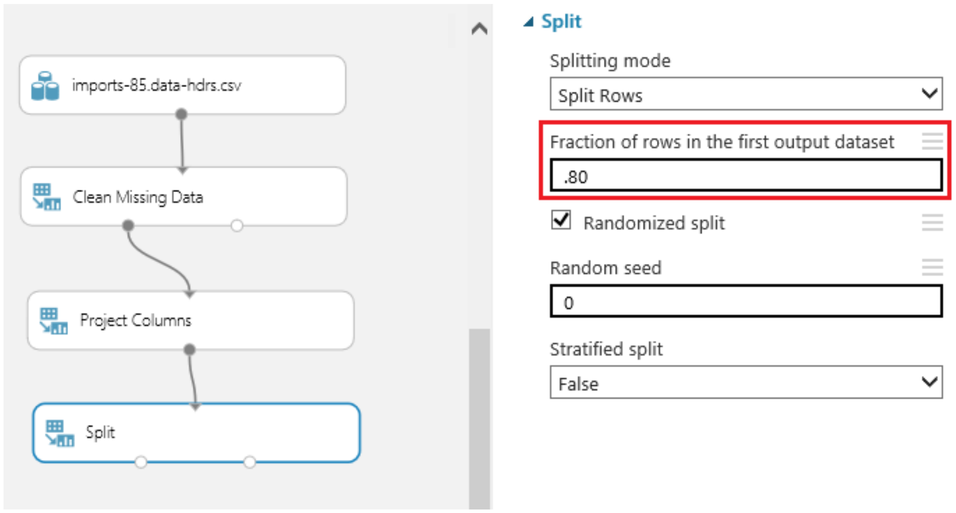
Exclude | column names | bore ✕ stroke ✕

FIGURE 5-13 The configuration parameters for the Project Columns module.

After adding the Project Columns module and configuring the module options, we have now chosen to effectively exclude two columns from our initial dataset. The assumption here is that the columns named bore and stroke will have no material impact on the prediction model and therefore do not need to be included in the input dataset. The addition and configuration of the Project Columns module will also have an impact when we configure a web service for our experiment. The input endpoint for our web service will respect this setting for the desired input columns. The endpoint will therefore not require these two columns when we submit an incoming web service request.

The next step is to add a Split module to our experiment so we can divert a certain percentage of our dataset for training purposes. The Split module will also designate another percentage for scoring and evaluating our new prediction model. As before, you can find this module by typing **Split** into the search bar on the top of the left navigation bar and let the Azure ML Studio search function find it for you. Or you can find it manually by clicking the Data Transformation tab and then the Sample And Split subtab.

Once you find the Split module, drag it onto the Azure ML Studio design surface and place it under the Project Columns module. Next, drag and connect the Split module to the Project Columns module. Now, click the Split module and set the parameter for the Fraction Of Rows In The First Output Dataset to .80. In this context, .80 means that we use 80 percent of the data in the Imports-85.data-hdrs.csv dataset for training our model, as shown in Figure 5-14.



Split

Splitting mode: Split Rows

Fraction of rows in the first output dataset: .80

☑ Randomized split

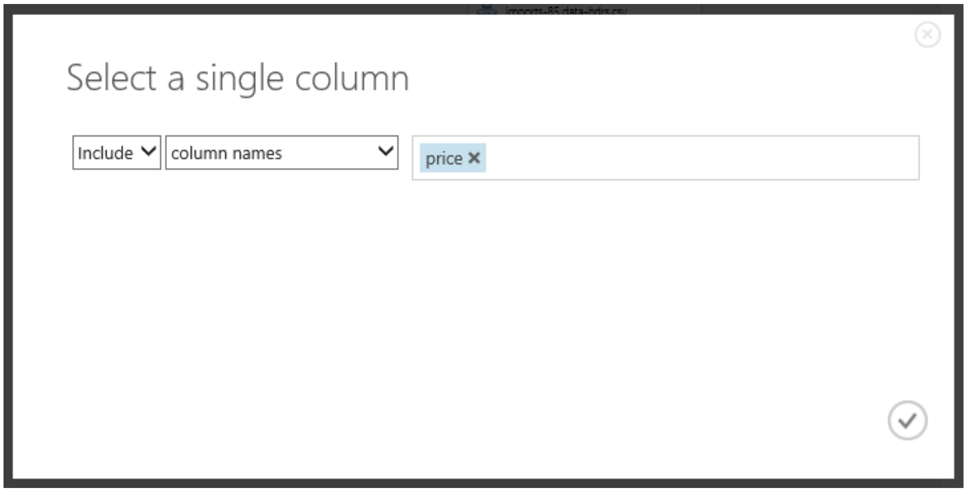
Random seed: 0

Stratified split: False

FIGURE 5-14 The configuration parameters for the Project Columns module.

The next step is to find a Train Model module and add it below the Split module. You can add this module by either searching for it or manually traversing the left navigation bar to the Azure Machine Learning tab and then the Train subtab. Connect the bottom left connector of the Split modules to the top right connector of the Train module.

Again, you will notice a red exclamation mark on the Train module, which means we need to finish configuring the module. Do this by clicking the module and then launching the column selector in the properties section on the right side of the Azure ML Studio designer. The Train module needs to know which column in the dataset is to be used for the prediction. Configure the column selector by selecting Include, selecting Column Names, and adding the price column for training, as shown in Figure 5-15.


Select a single column

Include | column names | price

FIGURE 5-15 The configuration parameters for the Project Columns module.

The next step is to find and add the Linear Regression module to the designer surface. You can find this model by using the search function or locating it under Machine Learning/Initialize Model/Regression. Drag and drop this module above and to the left of the Train module. Connect the bottom of the Linear Regression module to the top right of the Train module.

Now you are almost done. Click Run on the bottom app bar. Watch as a green check mark appears to the right of each module in our experiment as it is processed in turn. Figure 5-16 depicts how our experiment looks at this point in the Azure ML Studio environment.

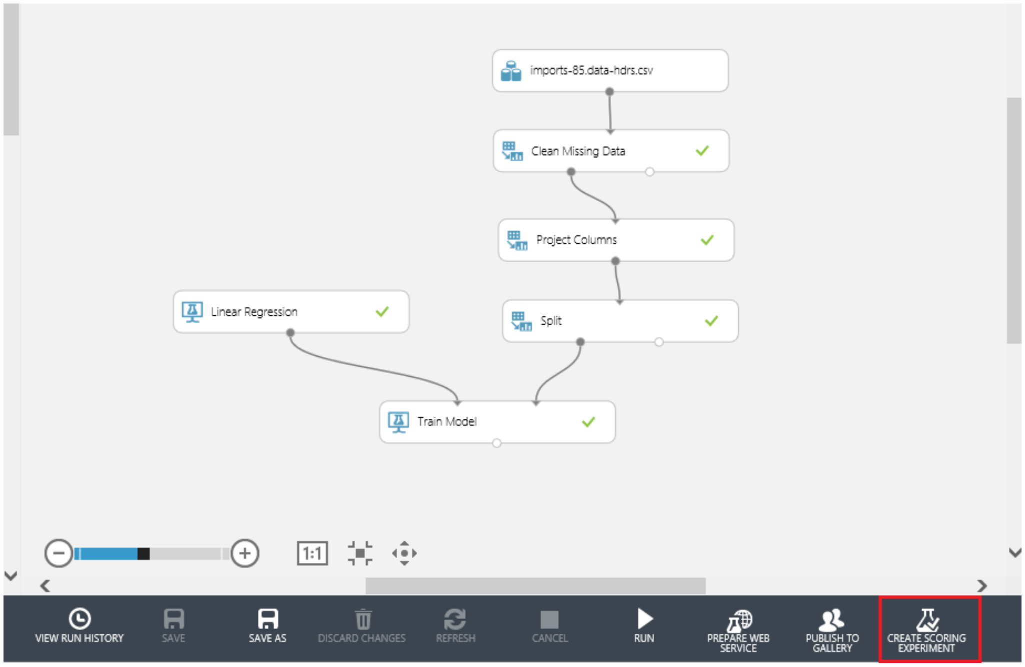


FIGURE 5-16 The automobile price prediction experiment successfully configured in training mode. The Create Scoring Experiment option is ready to be run next.

Now that we have successfully created our training experiment for automobile price prediction and run it to process all the modules, we are ready to switch from training mode to operational mode. Before we do this would be a good time to click Save As to preserve our experiment before we modify it for the next step. After saving, you will need to click Run again on the bottom app bar to process the experiment modules and thereby enable the Create Scoring Experiment option.

To complete the transformation of our automobile price prediction experiment using linear regression from training to operational mode, simply click Create Scoring Experiment on the bottom app bar. Azure ML Studio then takes over and transforms your experiment by automatically making the following changes:

- Saves the newly trained model under the Trained Models tab on the left navigation menu of Azure ML Studio.
- Removes the Linear Regression module.
- Removes the Train module.
- Adds the newly trained automobile price prediction model back into the experiment.
- Adds a Score Model module into the experiment.

Now, click Run to process the newly transformed experiment. Figure 5-17 shows our operationalized automobile prediction experiment.

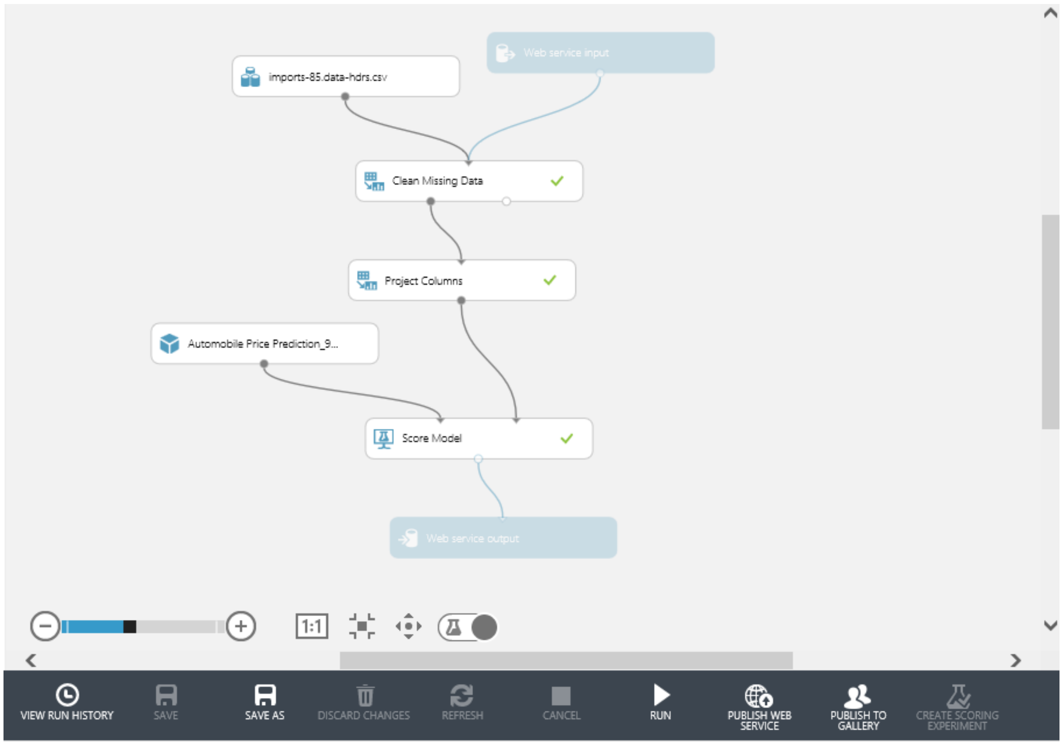


FIGURE 5-17 The automobile price prediction experiment after running the Create Scoring Experiment command.

Now that we have created our automobile price prediction experiment using the linear regression algorithm in Azure ML Studio, it's time to see the results of our efforts. Click the bottom connector of the Score Model module and select Visualize, as shown in Figure 5-18.

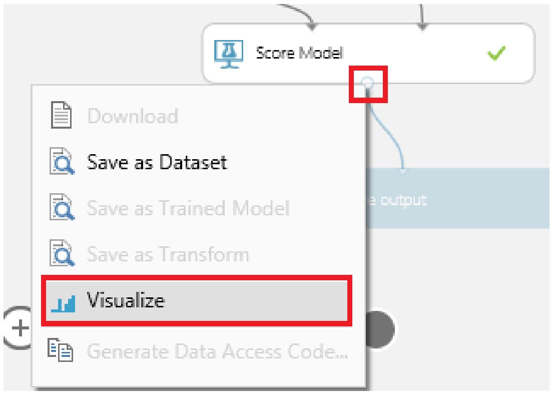
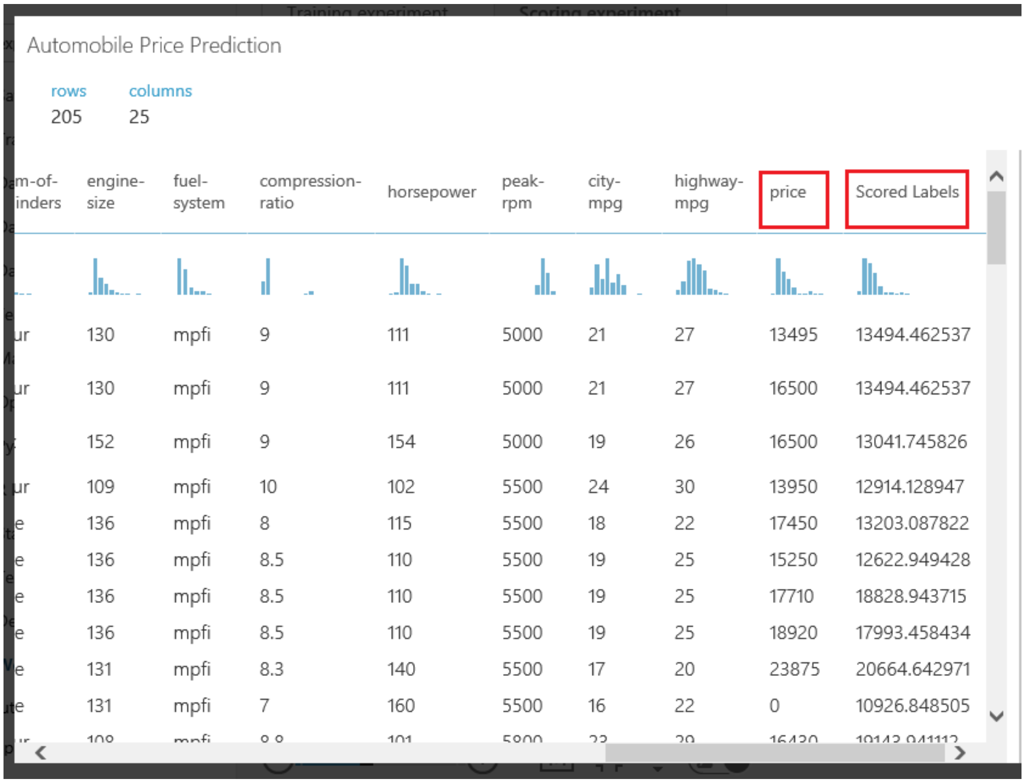


FIGURE 5-18 The option to Visualize the Score Model results of the automobile price prediction experiment.

After you select Visualize, you will see a pop-up window that displays the columns of our initial dataset. Scroll all the way over to the right and you will see a screen similar to Figure 5-19.



Automobile Price Prediction

rows 205 columns 25

m-of-inders	engine-size	fuel-system	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	Scored Labels
ur	130	mpfi	9	111	5000	21	27	13495	13494.462537
ur	130	mpfi	9	111	5000	21	27	16500	13494.462537
	152	mpfi	9	154	5000	19	26	16500	13041.745826
ur	109	mpfi	10	102	5500	24	30	13950	12914.128947
e	136	mpfi	8	115	5500	18	22	17450	13203.087822
e	136	mpfi	8.5	110	5500	19	25	15250	12622.949428
e	136	mpfi	8.5	110	5500	19	25	17710	18828.943715
e	136	mpfi	8.5	110	5500	19	25	18920	17993.458434
e	131	mpfi	8.3	140	5500	17	20	23875	20664.642971
e	131	mpfi	7	160	5500	16	22	0	10926.848505
	108	mpfi	8.8	101	5800	23	29	16430	19143.941112

FIGURE 5-19 Visualization of the Score Model results of the automobile price prediction experiment.

In this example, an additional column named Scored Labels is displayed next to the price column to indicate the numerical prediction results of our newly trained model. As you compare these two columns, you will note that the numerical results of the Scored Labels column are generally close to the values in the price column. You will also note that the model is fairly accurate for some rows; others have a greater numerical discrepancy and the associated accuracy is not as great.

Recall that this experiment is a demonstration of a simple linear regression algorithm. The end goal of regression algorithms is to predict one or more continuous variables, such as profit or loss, based on other columns in the dataset. In this case, our goal was to create a model that could predict the price of an automobile based on the selected features.

To take our experiment one step further, let's create a web service. We can then use that with the Excel tester template to help evaluate our new predictive model. To get started, click Publish Web Service on the bottom app bar, as shown in Figure 5-20.

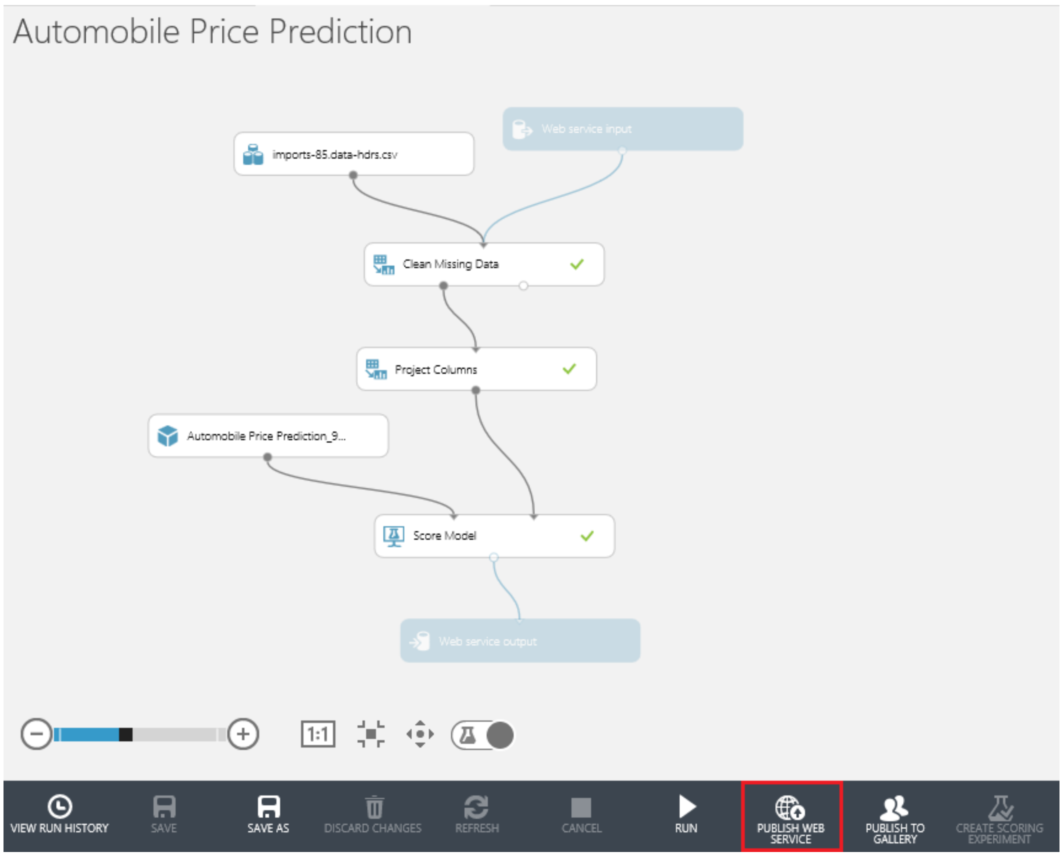
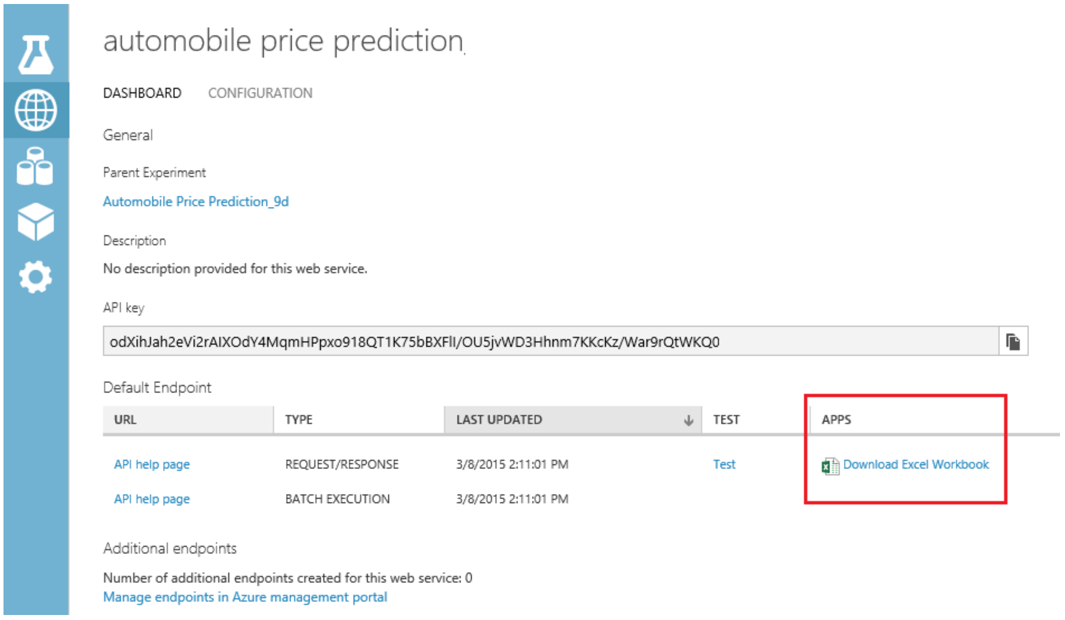


FIGURE 5-20 Ready to run the Publish Web Service command for the automobile price prediction experiment.

After clicking Publish Web Service on the bottom app bar you will see a screen similar to the one shown in Figure 5-21.


automobile price prediction

DASHBOARD CONFIGURATION

General

Parent Experiment

Automobile Price Prediction_9d

Description

No description provided for this web service.

API key

odXihJah2eVi2rAIXOdY4MqmHPpxo918QT1K75bBXFlI/OU5jvWD3Hhnm7KKcKz/War9rQtWKQ0

Default Endpoint

URL	TYPE	LAST UPDATED	TEST	APPS
API help page	REQUEST/RESPONSE	3/8/2015 2:11:01 PM	Test	Download Excel Workbook
API help page	BATCH EXECUTION	3/8/2015 2:11:01 PM		

Additional endpoints

Number of additional endpoints created for this web service: 0

Manage endpoints in Azure management portal

FIGURE 5-21 The Azure ML Studio web service dashboard after running the Publish Web Service command for the automobile price prediction experiment.

To make it easy to test our new Azure ML Studio experiment for predicting automobile prices using a linear regression algorithm, all we have to do is download the sample Excel workbook that is provided as part of the web service dashboard for our experiment. Simply click the Download Excel Workbook link, and you will have the option to either open it or save it locally to your system.

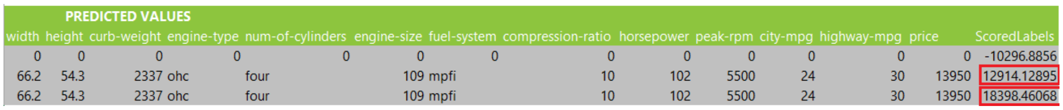
Once you open the Excel workbook, you will be prompted to enable the content. This is due to the fact that the workbook contains macros that can call our new Azure Machine Learning web service on our behalf. You can view these macros by pressing Alt + F8 and then selecting Edit to view the code. To exit from viewing the macro code, simply click File. Then, select Close And Return To Microsoft Excel, and you will be returned to the normal Excel view.

Here’s how it works: The Excel spreadsheet is constructed so that the input data elements are displayed in the blue columns marked with the heading Parameters. If you scroll over to the right, you will see the out parameters displayed in the green columns marked with the heading Predicted Values. As you enter a new row of parameter input information, the Excel macro will automatically trigger a call to the Azure Machine Learning web service to return the values in the Predicted Values section.

To quickly test our new Azure Machine Learning predictive model using the Excel workbook, you have two options. The first option is to enter new input values manually in the Parameters section column by column, being careful to only enter valid values for each column. Valid values are based on the initial input values found in the Imports-85.data.csv dataset.

A much faster approach is to simply copy an entire row of data from the original Imports-85.data.csv dataset. Paste the copied row into the Parameters section of the Excel workbook. As you move off the row, the Azure Machine Learning web service will be invoked by a macro in the downloaded workbook. A few seconds later, the results will appear in the Predicted Values section of the workbook. You can then compare the actual price provided in the blue Parameters section with the Scored Labels that is returned in the green Predicted Values section of the workbook.

Once you have populated a known good row of data, based on data from our initial dataset, you can then vary the individual parameter columns. As you make changes, you will see the impact that they have on the predicted price. For example, Figure 5-22 illustrates the effect of changing the value of just one field in the input parameters on the predicted price.



PREDICTED VALUES

width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	ScoredLabels
0	0	0	0	0	0	0	0	0	0	0	0	0	-10296.8856
66.2	54.3	2337	ohc	four	109	mpfi	10	102	5500	24	30	13950	12914.12895
66.2	54.3	2337	ohc	four	109	mpfi	10	102	5500	24	30	13950	18398.46068

FIGURE 5-22 The Excel workbook that calls the Azure Machine Learning web service for testing the new automobile price prediction experiment, after changing the make value from audi to bmw.

In this example, we added a duplicate row of information. We then only changed the value of the make parameter input column from audi to bmw. The Excel workbook then invoked the Azure Machine Learning web service and returned a new (higher) predicted price for this vehicle, amounting to an increase of $5,484.00.

Predicting a numerical value, such as an automobile price, demonstrates a much more complicated and advanced example of the power of Azure ML Studio. This is especially true when linear regression is contrasted to the much simpler binary classification experiment we examined in Chapter 3, “Using Azure ML Studio.” It also demonstrates just how easy it is to harness that power using Azure ML Studio. This is just one more step in our journey as we start to dive deeper into the other various predictive analysis algorithms that come standard in Azure ML Studio.

In this chapter, we took a look at what it takes to implement a simple linear regression algorithm using Azure ML Studio. Currently, there are eight different variations of linear regression algorithms available, as you can see in Figure 5-23.

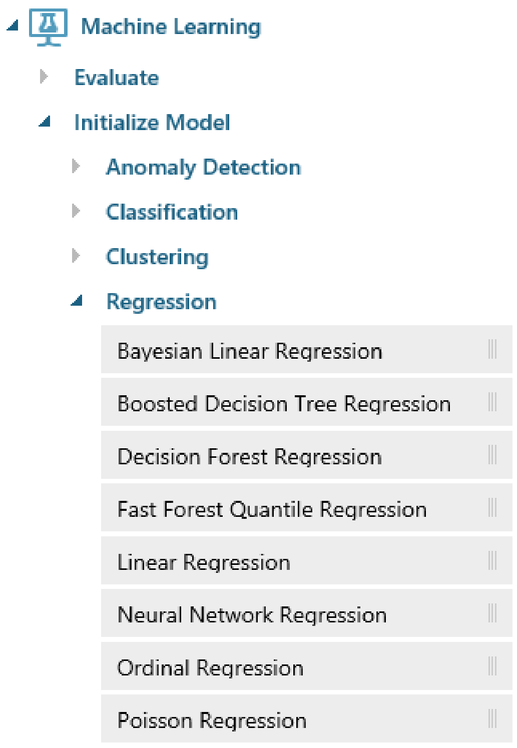


FIGURE 5-23 The Azure ML Studio Regression modules.

Here is a brief summary of the regression algorithms available in Azure ML Studio.

- **Bayesian Linear Regression** In this approach to linear regression, the statistical analysis is undertaken within the context of Bayesian inference. The Bayesian approach uses linear regression supplemented by additional information in the form of a prior probability distribution. Prior information about the input parameters is combined with a prediction function to generate estimates for the parameters.
- **Boosted Decision Tree Regression** This approach uses an efficient implementation of a gradient boosting algorithm. Gradient boosting is a machine learning technique for regression problems. It builds each regression tree in a stepwise fashion, using a predefined loss function to measure the error in each step and correct for it in the next. Thus the prediction model is actually a collection or ensemble of weaker prediction models. In regression problems, boosting builds a series of trees in a stepwise fashion. The model then selects the optimal tree using an arbitrary differentiable loss function.
- **Decision Forest Regression** Decision trees are models that perform a sequence of simple tests for each instance, traversing a binary tree data structure until a leaf node (decision) is reached.

This regression model consists of a collection of decision trees, which have these advantages:

- They are efficient in both computation and memory usage during training and prediction.
- They can represent nonlinear decision boundaries.
- They perform integrated feature selection and classification.
- They are resilient in the presence of noisy features.

- **Fast Forest Quantile Regression** This approach creates a regression model that can predict values for a specified number of quantiles. Quantile regression is useful if you want to understand more about the distribution of the predicted value, rather than get a single mean prediction value. This method has many applications, including the following:
  - Predicting product prices.
  - Estimating student performance or applying growth charts to assess development stages of children.
  - Discovering predictive relationships in cases where there are only weak relationships between other variables.
- **Linear Regression** This approach is widely used for modeling the relationship between a scalar dependent variable Y and one or more explanatory variables denoted X. It can be, and typically is, applied for prediction, forecasting, or reduction. Linear regression models are typically fitted using the least squares approach, but other options also exist to measure which fit is best.
- **Neural Network Regression** These algorithms are widely known for use in deep learning and modeling complex problems such as image recognition. They are easily adapted to regression problems. Any class of statistical models can be termed a neural network, if they use adaptive weights and can approximate nonlinear functions of their inputs. Thus neural network regression is suited to problems where a more traditional regression model cannot fit a solution.
- **Ordinal Regression** In this regression model, the labels or target values have a natural ordering or ranking among them. The real distance between categories is unknown. Only the relative ordering between different values is significant. Because the labels or target values have a natural ordering or ranking, any numeric column can be used as an ordinal target.
- **Poisson Regression** This is a special type of regression analysis that is typically used to model prediction counts. For example, Poisson regression would be useful for the following functions:
  - Modeling the number of colds associated with airplane flights.
  - Estimating the number of calls related to an event or promotion.
  - Creating contingency tables.

Summary

In this chapter, we took a deeper look at linear regression, one of the more popular prediction algorithms used in the data science field today. To illustrate the processing logic concepts, we created a new Azure Machine Learning experiment. We used this experiment to predict the price of automobiles based on a dataset from the UCI Machine Learning Repository.

We also added a few new highly useful modules to the mix, such as the Clean Missing Data module and the Project Columns module. We used the Clean Missing Data module to replace missing values in our initial dataset. The Project Columns module excluded columns that were not deemed material to the prediction analytics calculation.

Resources

For more information about Azure Machine Learning, please see the following resources.

Documentation

- Azure Machine Learning Regression Modules Reference
- Azure Machine Learning Gallery Regression samples
- Azure Machine Learning Sample datasets
- Run a Linear Regression Model on the Azure platform

Videos

- Machine Learning on Azure

Chapter 6

Cluster analytics

In this chapter, we continue our exploration of the three primary data science algorithms used in machine learning today:

- **Classification algorithms** These are used to classify data into different categories, which can then be used to predict one or more discrete variables, based on the other attributes in the dataset.
- **Regression algorithms** These are used to predict one or more continuous variables, such as profit or loss, based on other attributes in the dataset.
- **Clustering algorithms** These determine natural groupings and patterns in datasets. They are typically used to predict grouping classifications for a given variable.

At this point in our machine learning journey, we have explored the classification algorithms with a look at binary classification by producing a model that can predict whether a person is likely to have an income level of greater than or less than $50,000 per year.

In the last chapter, we covered the topic of regression analytics in Azure ML Studio. We ran through a complete example of linear regression analytics using Azure ML Studio to predict the price of an automobile using data elements that described the various features of an automobile. In that example, we were able to train our model by providing a dataset with known inputs (automobile features) and known outputs (automobile prices) to produce a working model that could predict a new price based on a set of features.

The focus of this chapter is on the third type of algorithm in our series, clustering algorithms. The first two types of algorithms we explored, classification and regression, can be categorized as types of supervised machine learning. This is due to the fact that in both cases, we provided training datasets that allowed the Azure Machine Learning algorithms the ability to learn from the data. The supervised learning in these algorithms determines what the correct prediction should be.

Clustering algorithms are highly interesting for use in situations where we want the machine to conduct its own analysis on the dataset, determine relationships, infer logical groupings, and generally attempt to make sense of chaos by determining the forests from the trees.

Unsupervised machine learning

As we discussed back in Chapter 2, "Getting started with Azure Machine Learning," with the case of unsupervised machine learning, the success of the new predictive model depends entirely on the ability of the machine to correctly infer and identify patterns, structures, and relationships in the incoming data set.

The goal of inferring these patterns and relationships is that the objects within a group be similar to one another. They should also be different from other objects in other groups.

One of the most common unsupervised learning algorithms is known as cluster analysis, which is used to find hidden patterns or groupings within data sets. Clustering can be useful when there is enough data to form clusters to logically delineate the data. The delineated data then make inferences about the groups and individuals in the cluster.

Cluster analysis

In this chapter, we explore the theory and practical implementations of cluster algorithms using Azure ML Studio. In its simplest form, clustering can be defined as a method of statistical analysis to help determine natural groupings and patterns in datasets. The ultimate goal is then to develop a model that can successfully predict grouping classifications for a given variable.

To help illustrate the concept, let's look at a simple example. Figure 6-1 depicts a common scenario where we have a large population of ungrouped and unsegmented data elements.

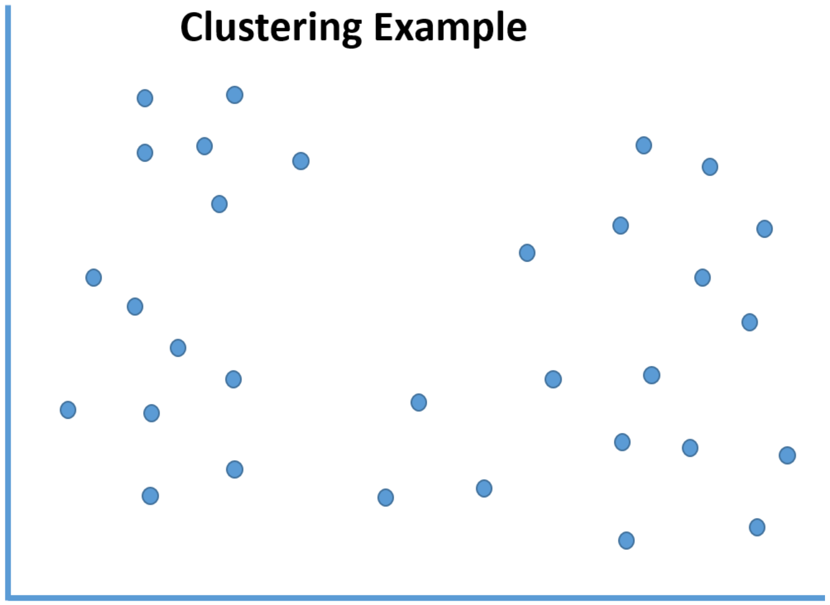


FIGURE 6-1 A basic unclustered dataset.

After applying machine learning clustering algorithms, we are able to infer certain groupings, relationships, and patterns within the data to determine logical data boundaries and demarcations.

Figure 6-2 illustrates the effect of applying clustering algorithms to this set of seemingly random data elements.

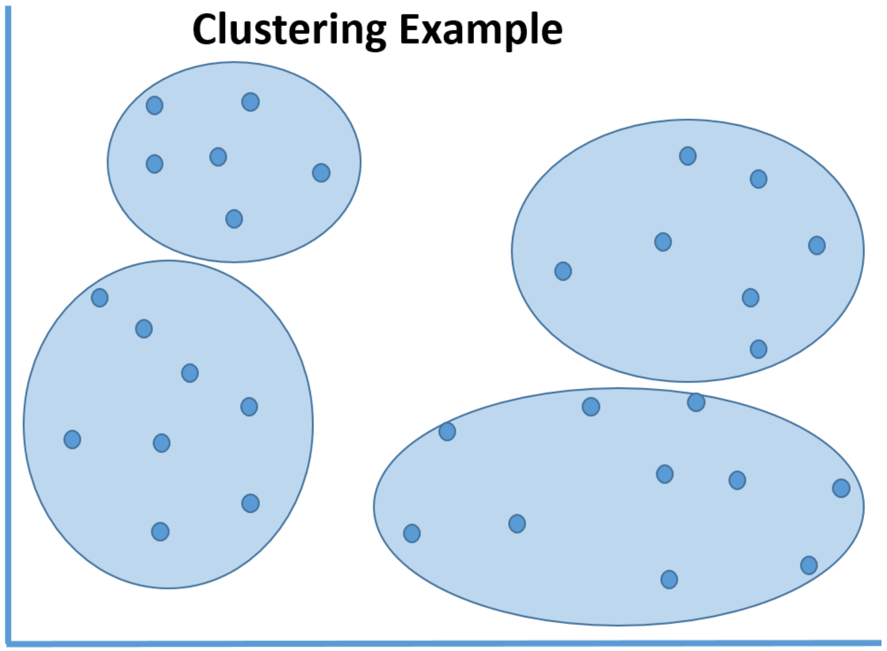


FIGURE 6-2 Visualizing a random dataset after applying clustering algorithms.

As you can see from Figure 6-2, these logical borders help to isolate and characterize the data elements so that general inferences can be made about the members of these groups.

KNN: K nearest neighbor algorithm

One of the underlying principles of cluster analysis is the ability to classify data variables by placing them in a category with the most similar, or nearest neighbors.

Cluster analysis is the task of grouping a set of objects in such a way that objects in the same group (or cluster) are more similar (based on common characteristics) to each other than to those in other groups or clusters. Cluster analysis is a primary output for data mining and a common technique for statistical data analysis. Its use can be documented in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bio-informatics.

Cluster analysis is not necessarily defined as the use of one specific algorithm, but rather a method of tackling the general problem to be solved. Clustering can be achieved by using multiple algorithms. The algorithms might differ significantly in their interpretation of what constitutes a member of a particular cluster. The way to efficiently find and segregate those members might differ as well.

The common implementation of clustering algorithms includes the following steps:

1. Defining groups with small distances among the cluster members.
2. Highlighting the denser areas of the dataset continuum.
3. Determining intervals or statistical distributions.

Clustering can therefore be described as a multifaceted approach to solving a common optimization problem.

Clustering modules in Azure ML Studio

Azure ML Studio contains three built-in modules to work with cluster algorithms in your experiments. These three modules all work together to help produce a predictive model based on clustering.

- **K-Means Clustering module** This is one of the most popular clustering algorithms in use in the field of data science today. The K-means method finds a specified number of clusters for a set of D-dimensional data points. It starts with an initial set of K centroids, and then uses algorithms to iteratively refine the locations of the centroids. The algorithm then terminates when the centroids stabilize or when a specified number of iterations are completed.
- **Train Clustering Model module** This module takes an untrained clustering model, such as that produced by the K-Means Clustering module, and an unlabeled data set. It returns a trained clustering model that can be passed to the Assign to Clusters module. It also returns labels for the training data.
- **Assign to Clusters module** This module takes a trained clustering model, produced by the Train Clustering Model module, and an unlabeled data set. The module then returns the cluster assignments (indexes) for the input data.

Note in Figure 6-3 that, unlike the other machine learning categories of algorithms, there is only one clustering algorithm available. The K-Means Clustering algorithm module can be found under the Machine Learning/Initialize Model/Clustering tab in Azure ML Studio.

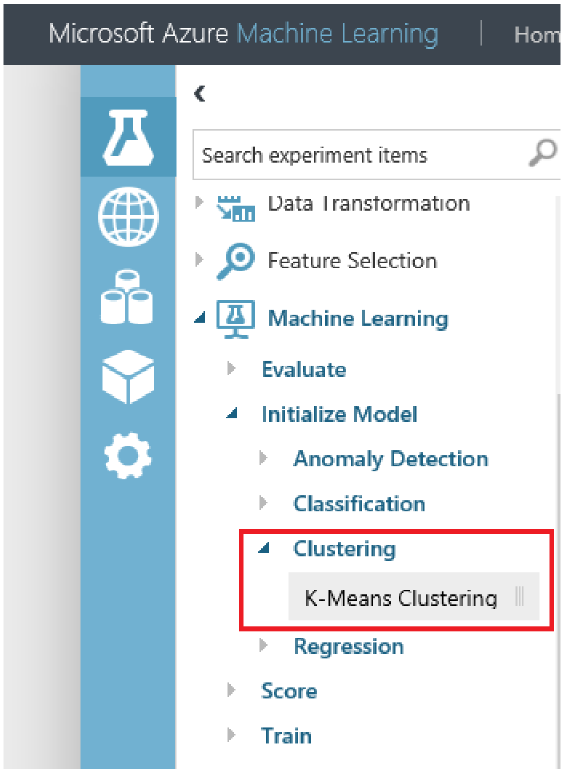


FIGURE 6-3 The K-Means Clustering algorithm in Azure ML Studio.

Clustering sample: Grouping wholesale customers

To illustrate the usage of the K-means Clustering algorithm in Azure ML Studio, we explore a sample experiment scenario that uses the clustering algorithm to determine the customer segmentation of wholesale customers.

In this Azure ML Studio experiment, we use the Wholesale customers dataset from the UCI Machine Learning Repository located at https://archive.ics.uci.edu/ml/datasets/Wholesale+customers.

This dataset refers to clients of a wholesale distributor. It includes the annual spending on the following product categories:

- **Channel** Retail channel types: hotel/restaurant/café
- **Region** Customer region code
- **Fresh** Annual spending on fresh products

- **Milk** Annual spending on milk products
- **Grocery** Annual spending on grocery products
- **Frozen** Annual spending on frozen products
- **Detergents_Paper** Annual spending on detergents and paper products
- **Delicatessen** Annual spending on delicatessen products

The goal of this Azure Machine Learning experiment is to analyze the dataset and infer logical relationships to classify wholesale customers by logical groupings.

This sample experiment could easily be extrapolated and applied to many similar types of modern customer stratification applications to determine marketing, product, and pricing strategies.

Let's get started. The first step is to acquire the Wholesale customers dataset from the UCI Machine Learning Repository. In the previous Azure Machine Learning experiments, we downloaded the datasets to our local file system to then upload them directly into the Azure ML Studio Saved Datasets. In this example, we use a Reader module to directly access the dataset over the Internet.

Start by creating a new, blank Azure Machine Learning experiment and naming it Wholesale Customers. Next, find and implement the Reader module in Azure ML Studio. You can find it by simply typing the name into the search bar on the top of the left navigation bar and letting the Azure ML Studio search function find it for you. Alternatively, you can find this module under Data Input And Output as shown in Figure 6-4.

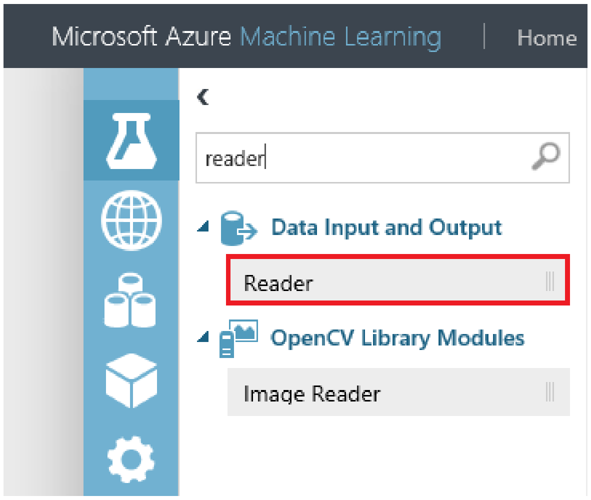


FIGURE 6-4 The Azure ML Studio reader module.

After you drag this module onto the Azure ML Studio design surface, click the module to display the

Properties window on the right side of the designer surface. Click the Data Source drop-down list in the Properties window to set the Reader's data source. You will see the various options for selecting data sources as shown in Figure 6-5.

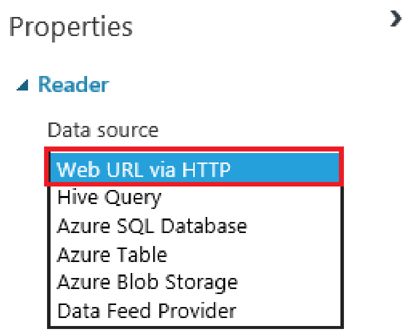


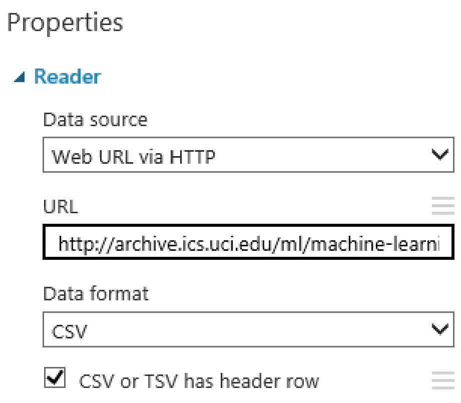
FIGURE 6-5 The Data Source drop-down list options for the data Reader module.

For this experiment, select the Web URL Via HTTP option to denote that we will have Azure ML Studio read the dataset directly from this location over the Web.

In the URL field, insert the address of our wholesale customer dataset from the UCI Machine Learning Repository:

```
http://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20customers%20data.cs
v
```

Set the Data Format drop-down list to CSV to denote that the dataset is comprised of comma-separated values. Select the CSV Or TSV Has Header Row check box to denote that the dataset contains headers in the first row of data. The reader module should now look like Figure 6-6.



Properties

Reader

Data source: Web URL via HTTP

URL: http://archive.ics.uci.edu/ml/machine-learni

Data format: CSV

☑ CSV or TSV has header row

FIGURE 6-6 The Reader module properties for the wholesale customers experiment.

Now that we have the Reader module fully configured, you can test it out by clicking Run on the bottom app bar. Wait for the green check mark to appear in the reader module that signifies that the module has run successfully. Once the module has been processed, click the bottom connector and select Visualize as shown in Figure 6-7.

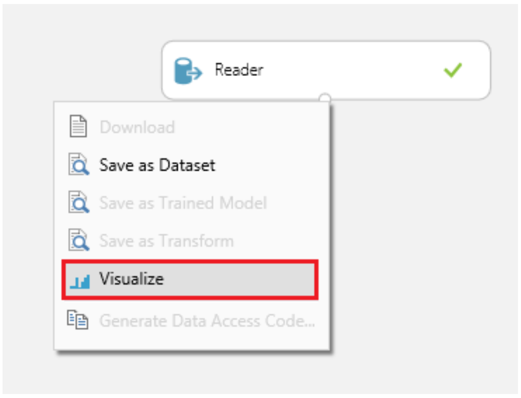
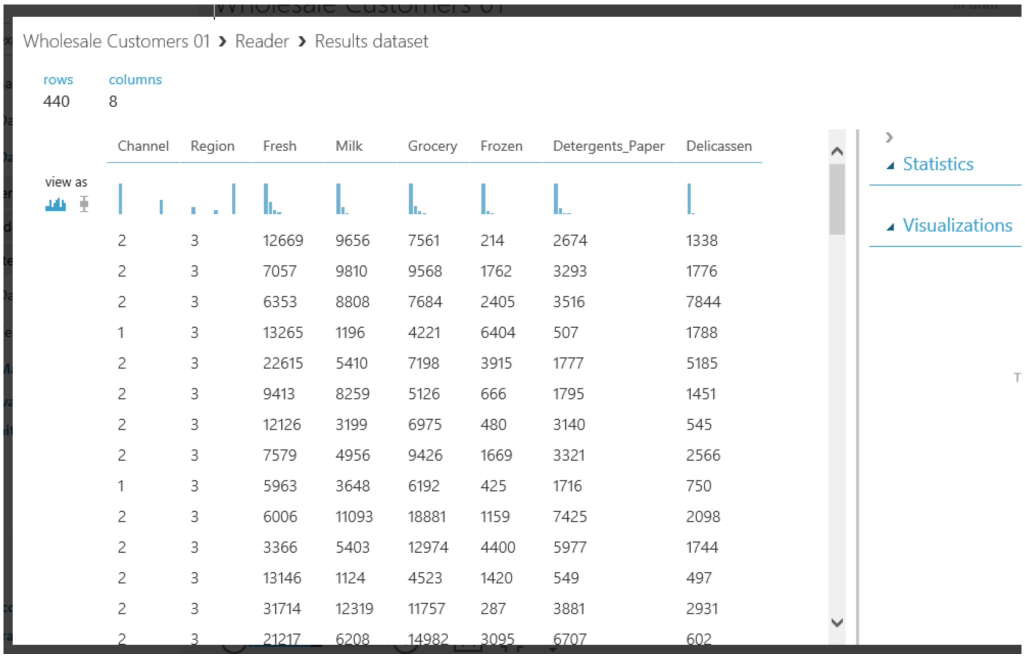


FIGURE 6-7 The Reader module properties for the wholesale customers experiment.

After you select the Visualize option, you will see a screen similar to Figure 6-8, displaying the eight columns of data from the UCI wholesale customer dataset.



Wholesale Customers 01 › Reader › Results dataset

rows 440 columns 8

Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
2	3	12669	9656	7561	214	2674	1338
2	3	7057	9810	9568	1762	3293	1776
2	3	6353	8808	7684	2405	3516	7844
1	3	13265	1196	4221	6404	507	1788
2	3	22615	5410	7198	3915	1777	5185
2	3	9413	8259	5126	666	1795	1451
2	3	12126	3199	6975	480	3140	545
2	3	7579	4956	9426	1669	3321	2566
1	3	5963	3648	6192	425	1716	750
2	3	6006	11093	18881	1159	7425	2098
2	3	3366	5403	12974	4400	5977	1744
2	3	13146	1124	4523	1420	549	497
2	3	31714	12319	11757	287	3881	2931
2	3	21217	6208	14982	3095	6707	602

FIGURE 6-8 The Visualize option for the Reader module displaying the wholesale customer dataset column values.

In viewing the wholesale customers dataset via the Visualize option of the Reader module, we can see that the dataset contains 440 rows of data with eight columns per row. The objective for Azure ML Studio will be to analyze this wholesale customer dataset. In this experiment, Azure ML Studio will infer logical groupings of similar customer segments based on the eight exposed attributes. The end goal in this scenario will be to accomplish the following business goals:

- **Identify logical customer segments** These segments can then be used in the creation of product, marketing, and pricing campaigns.
- **Forecast product demand** This is based on relative spending habits of other similar wholesale customers.
- **Create a new predictive model** This model can be used in categorizing future wholesale customers according to similar patterns.

Now that we have the objectives out of the way, the next step is to add a Train Clustering Model module to the Azure ML Studio design surface. You can find this module by simply typing the name into the search box on the top of the left navigation bar and let the Azure ML Studio search function find it for you. Alternatively, you can find this module under the Machine Learning/Train tabs as shown in Figure 6-9.

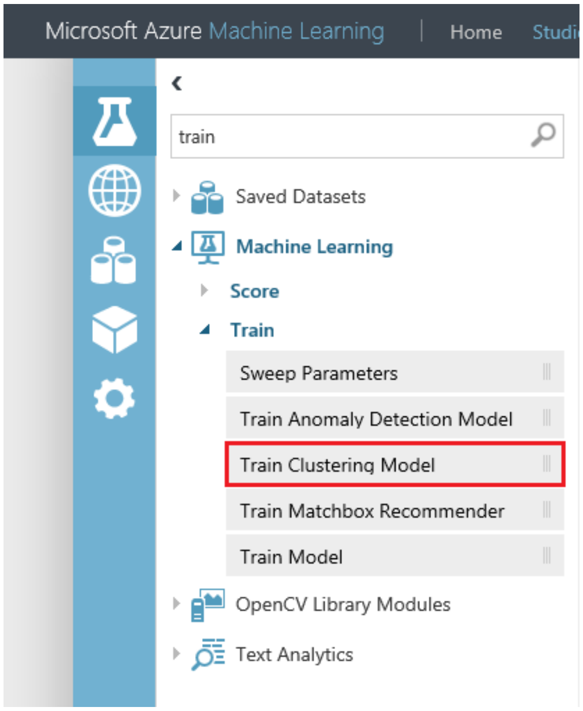
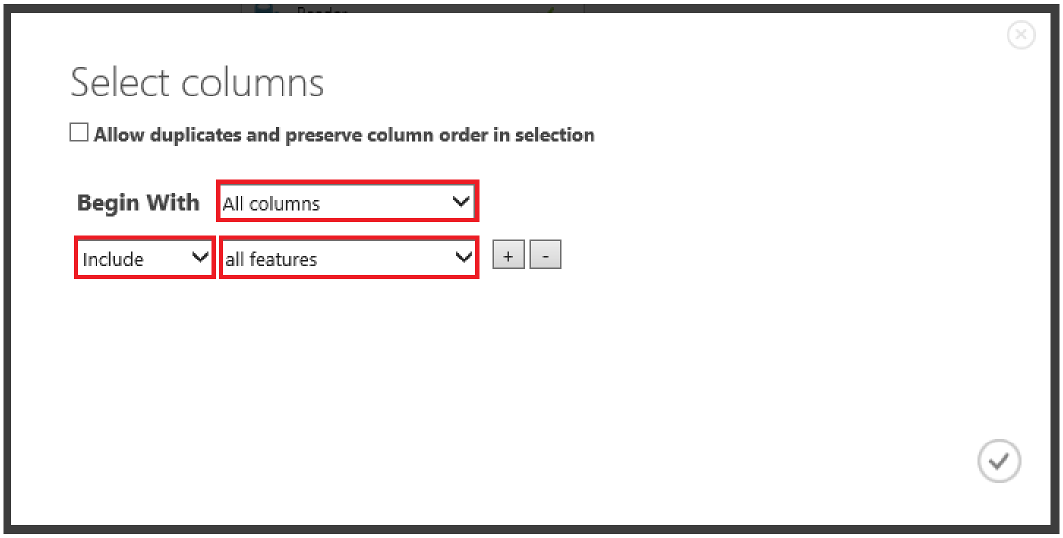


FIGURE 6-9 The Train Clustering Model module in Azure ML Studio.

After placing the Train Clustering Model module on the design surface, connect the bottom of the reader module to the top right connector of the Train Clustering Model module. At this point, you will notice a red exclamation mark on the right side of the module indicating that the properties need to be specified. To do this, click the Train Clustering Model module and then click the Launch Column Selector button located on the right Properties pane of the Azure ML Studio designer.

In the Select Columns interface, specify the options shown in Figure 6-10.

- Begin With: All Columns
- Include/All Features



Select columns

☐ Allow duplicates and preserve column order in selection

Begin With: All columns

Include | all features

FIGURE 6-10 The Select Columns options for the Train Clustering Model module.

Now that we have our Train Clustering Model module configured, the next step is to locate the K-Means Clustering module and drag it onto the Azure ML Studio designer surface. As before, you can either use the module search feature in the left navigation bar or traverse through the available modules and select Machine Learning/Initialize Model/Clustering to find the K-Means Clustering module as shown in Figure 6-11.

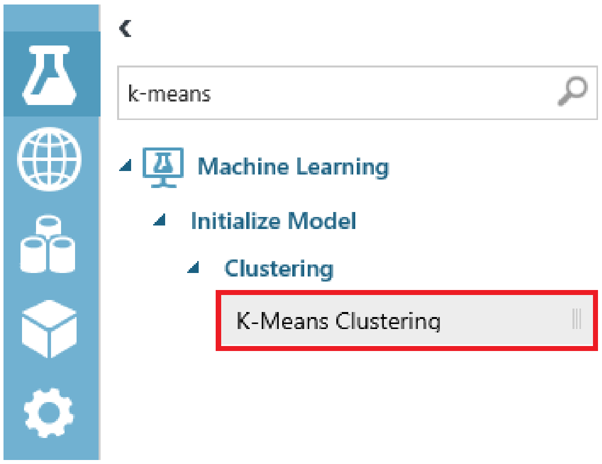
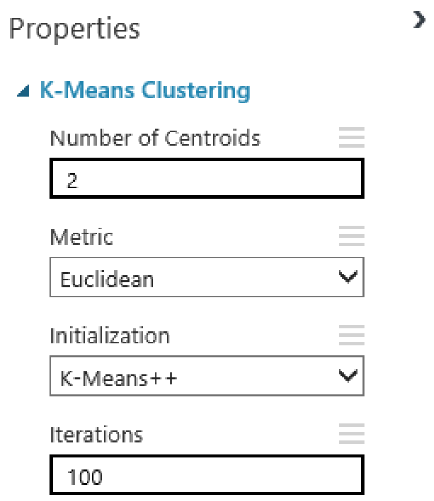


FIGURE 6-11 The K-Means Clustering module in Azure ML Studio.

After dragging the K-Means Clustering module to the designer surface, connect the bottom of the module to the top left connector on the Train Cluster Model module. Next, click the K-Means Clustering module to examine the default properties of the module as shown in Figure 6-12.


Properties

K-Means Clustering

Number of Centroids: 2

Metric: Euclidean

Initialization: K-Means++

Iterations: 100

FIGURE 6-12 The properties of the K-Means Clustering module in Azure ML Studio.

Note that the Number Of Centroids parameter refers to the desired number of clustered groupings for the data in our dataset. For the purposes of this example, we set the Number Of Centroids parameter to 4 to denote that we want Azure Machine Learning to infer and create four distinct logical clusters from our dataset values.

he Metric parameter offers two options:

- **Euclidean** This, the default, refers to the distance between two points and is the length of the line segment connecting them.
- **Cosine** This refers to the measure of similarity between two vectors of an inner space that measures the cosine of the angles between them.

Now we are almost done. The last step to set up our demo clustering experiment in Azure ML Studio is to add the Metadata Editor module to our experiment. To add this module, you can either use the module search feature in the left navigation bar, or traverse through the available modules to select Data Transformation/Manipulation to find the Metadata Editor module as shown in Figure 6-13.

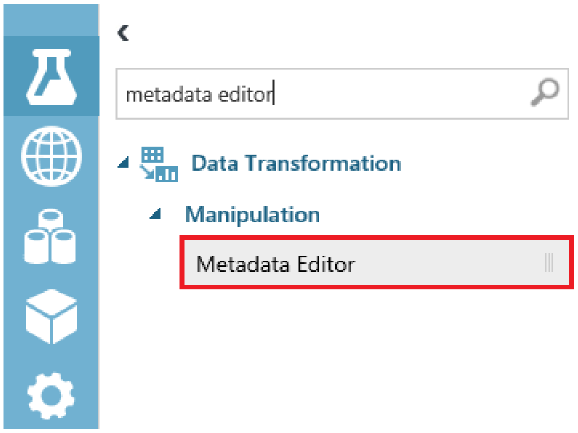


FIGURE 6-13 The Metadata Editor module in Azure ML Studio.

Now, drag and drop the Metadata Editor module onto the Azure Machine Learning designer surface and connect the bottom right connector of the Train Clustering Model module to the top of the Metadata Editor module.

At this point, you will notice a red exclamation mark on the right side of the Metadata Editor module indicating that the properties need to be specified. To do this, click the Metadata Editor module and then click the Launch Column Selector button located in the right Properties pane of the Azure ML Studio designer.

In the Select Columns interface, specify the following options:

- Begin With: All Columns
- Include/All Features

Save your project and then click Run on the bottom app bar. At this point, Figure 6-14 depicts our completed clustering experiment for the wholesale customer dataset.

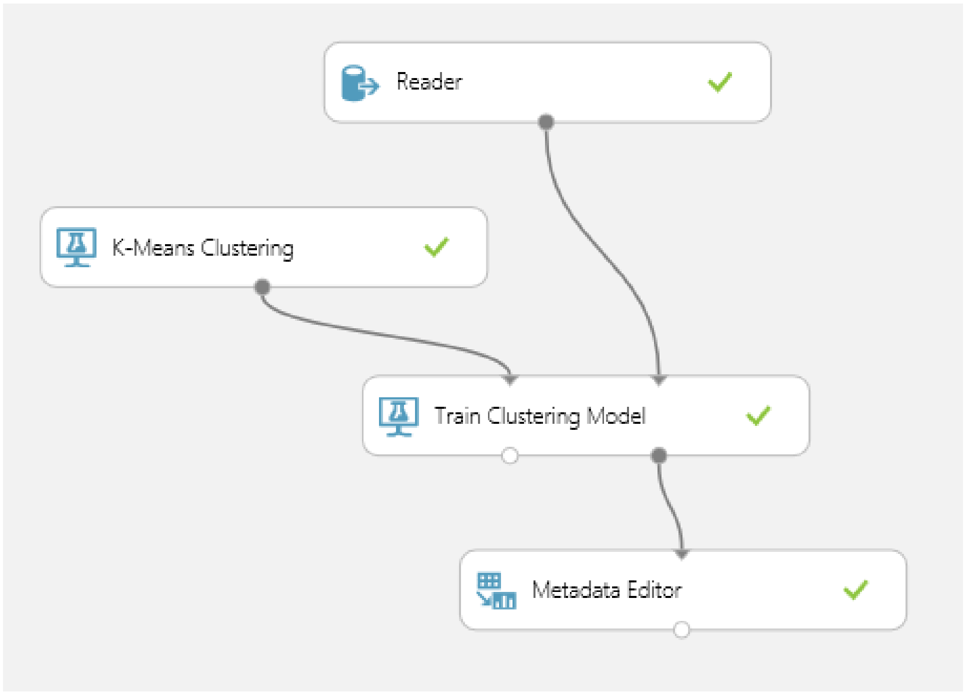


FIGURE 6-14 The K-means clustering experiment for analyzing wholesale customers in Azure ML Studio.

After running the K-means clustering experiment for wholesale customers, we can view the results of the cluster groupings by clicking the bottom connector of the Metadata Editor module and then selecting the Visualize command as shown in Figure 6-15.

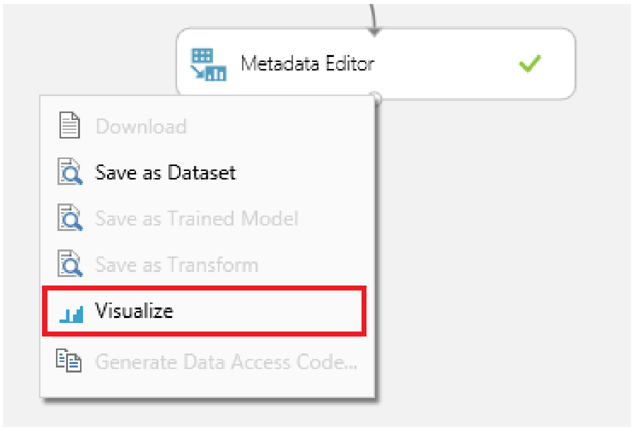
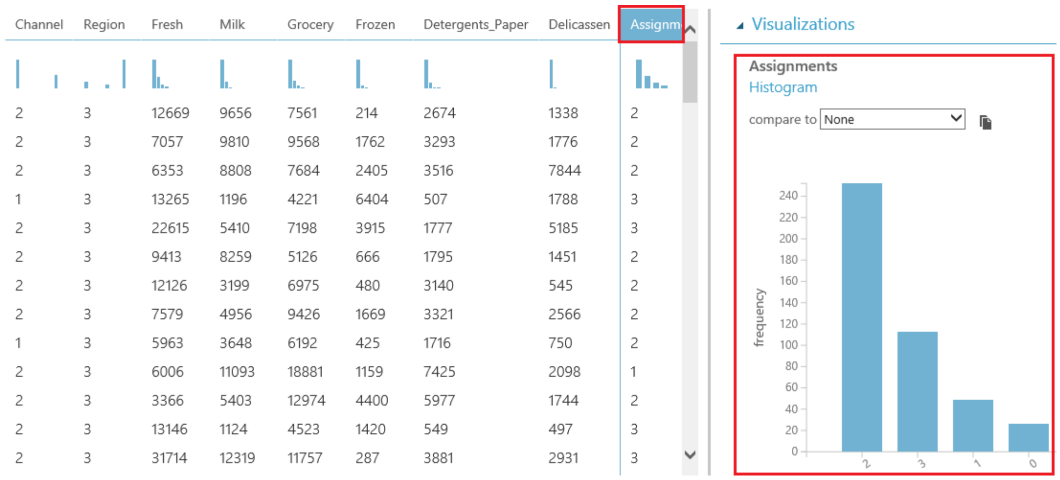


FIGURE 6-15 Accessing the Visualize command from the Metadata Editor module.

After you select the Visualize command, Azure ML Studio renders a visualization window depicting the results of our K-means clustering experiment for wholesale customers. Note that there is now an additional column at the end named Assignments that has been added to our dataset.

The new Assignments column denotes the cluster group number that has been assigned to each row in our dataset based on the parameter settings and the K-Means Clustering algorithm processing logic we used in our experiment. If you click the Assignments column, you can then see a histogram visualization that depicts the distribution of the dataset records across four distinct grouping clusters. Recall that we set the parameter to four for the Number Of Centroids in our K-Means Clustering module. Figure 6-16 illustrates the visualization window for our experiment.



Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Assignm
2	3	12669	9656	7561	214	2674	1338	2
2	3	7057	9810	9568	1762	3293	1776	2
2	3	6353	8808	7684	2405	3516	7844	2
1	3	13265	1196	4221	6404	507	1788	3
2	3	22615	5410	7198	3915	1777	5185	3
2	3	9413	8259	5126	666	1795	1451	2
2	3	12126	3199	6975	480	3140	545	2
2	3	7579	4956	9426	1669	3321	2566	2
1	3	5963	3648	6192	425	1716	750	2
2	3	6006	11093	18881	1159	7425	2098	1
2	3	3366	5403	12974	4400	5977	1744	2
2	3	13146	1124	4523	1420	549	497	3
2	3	31714	12319	11757	287	3881	2931	3

FIGURE 6-16 Visualizing the results of the K-means clustering experiment for wholesale customer data in Azure ML Studio.

As you can see from the histogram displayed in the Azure ML Studio visualization, the chart shows us that our model has effectively “clustered” our 440 wholesale customer records into four discrete groups numbered 0 to 3. Clustering is a powerful machine learning tool that allows us to easily infer relationships, stratify huge datasets, and find the “forests in the trees.”

Operationalizing a K-means clustering experiment

To operationalize our Azure ML Studio clustering experiment for wholesale customers, we will need to modify our experiment slightly before we can transform it into a working web service.

To start the transformation of our experiment, we need to add two more modules to our experiment and rewire the existing connections.

The first step is to locate a Split module and drag it onto the design surface right under the Reader module. To add this module, you can either use the module search feature in the left-hand module menu, or traverse through the available modules to select Data Transformation/Sample And Split tabs to find the Split module as shown in Figure 6-17.

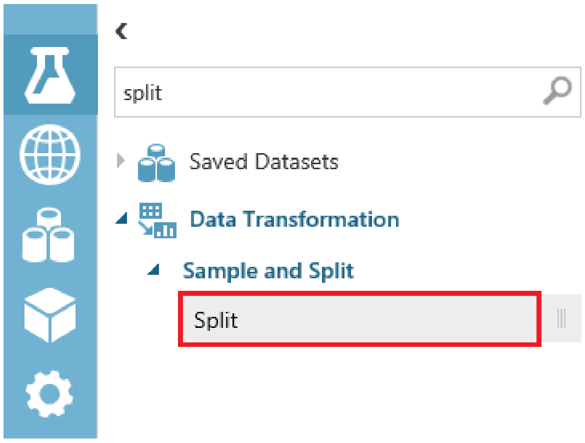
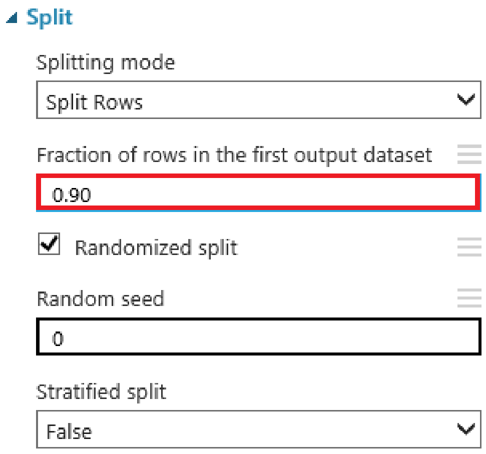


FIGURE 6-17 The Split module in the Azure ML Studio module menu.

After you drag and drop the Split module under the Reader module, connect it to the bottom of the Reader module. Then click the Split module to display the module properties in the right Properties pane in Azure ML Studio. Next, set the value of the Fraction Of Rows In The First Output Dataset parameter to 0.90 as shown in Figure 6-18 to denote that we want to use 90 percent of our wholesale customer dataset to train our predictive cluster model.



Split

Splitting mode: Split Rows

Fraction of rows in the first output dataset: 0.90

☑ Randomized split

Random seed: 0

Stratified split: False

FIGURE 6-18 Setting the properties of the Split module to train the model.

After configuring the Split module, the next step is to connect the bottom left connector of the Split module to the top right connector of the Train Clustering Model module. The top left connector on the

Train Clustering Model module remains connected to the bottom of the "K-Means Clustering Model module.

The next step is to locate and add an Assign to Clusters module to the experiment design surface. To add the Assign to Clusters module, you can either use the module search feature in the left module menu, or traverse through the available modules to select the Machine Learning/Score modules to find the Assign to Clusters module as shown in Figure 6-19.

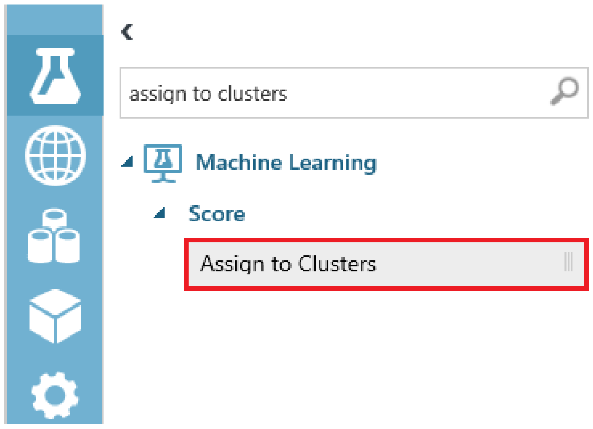
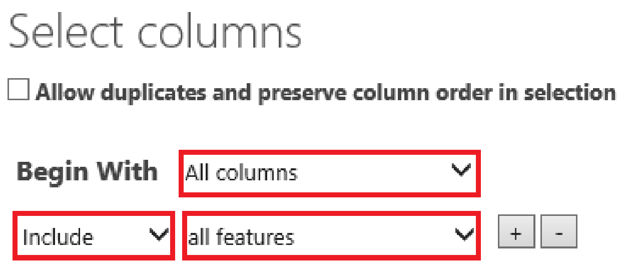


FIGURE 6-19 Locating the Assign to Clusters module in Azure ML Studio.

Place the Assign to Clusters module under the Train Clustering Model module and a little to the right. Next, connect the bottom right connector on the Split module to the top right connector on the Assign to Clusters module. Then connect the top left of the Assign to Clusters module to the bottom left connector on the Train Clustering Model module.

At this point, the Assign to Clusters module might show a red exclamation mark on the right side of the module. To fix this, click the Assign to Clusters module and click the Launch Column Selector button in the right Properties pane. Set the properties as shown in Figure 6-20.


Select columns

☐ Allow duplicates and preserve column order in selection

Begin With: All columns

Include: all features

FIGURE 6-20 Setting the column properties of the Assign to Clusters module in Azure ML Studio.

The last step to complete our experiment transformation is to connect the bottom left connector on the Assign to Clusters module to the top right connector on the Metadata Editor module. Figure 6-21 shows how our experiment looks after the modifications have been made to operationalize our experiment.

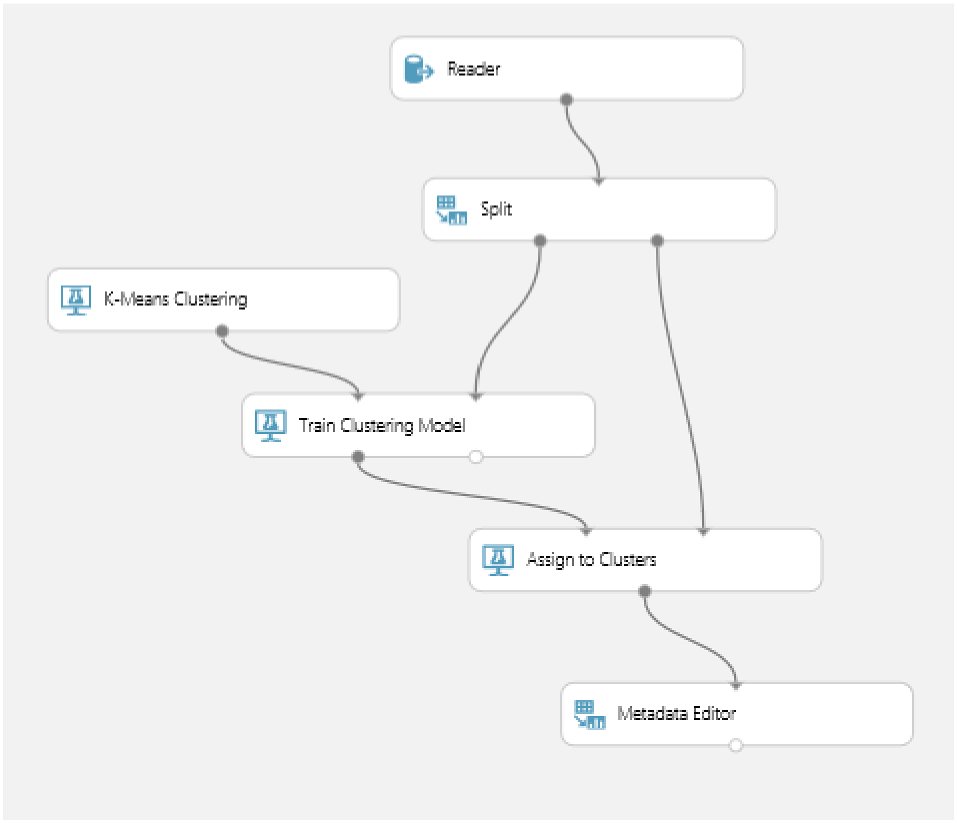


FIGURE 6-21 The wholesale customers cluster experiment in Azure ML Studio after modifications to prepare it for becoming a web service.

Next, click Run on the bottom app bar to process all the modules in the experiment. As each module is processed, a green check mark will appear on the right side of the module.

Now click Save As on the bottom app bar and save this experiment as a new version before completing the final transformation to a web service. After creating a new version, click Run again on the bottom app bar to prepare for the next step.

Next, click the Create Scoring Experiment command on the bottom app bar, shown in Figure 6-22, to complete the transformation before we can publish our experiment as a web service. This command will make the following modifications to the experiment:

- Saves our trained model to the Trained Models repository.

- Removes the Train Clustering Model, K-Means Clustering, Split, and Metadata Editor modules.
- Inserts the newly trained model into the experiment.
- Adds the Web Service Input and Web Service Output modules.

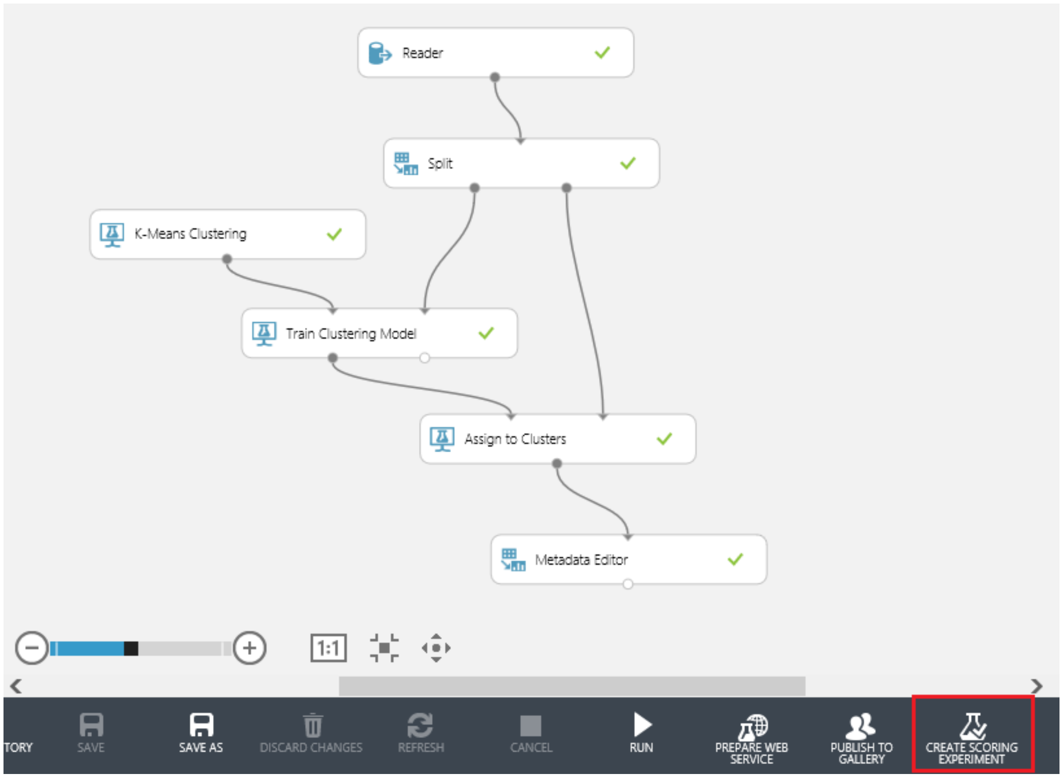


FIGURE 6-22 The wholesale customers cluster experiment in Azure ML Studio before running the Create Scoring Experiment command.

Next, click Run to process the modifications made to our experiment by the Create Scoring Experiment command. By clicking Run, we enable the next command we need to run, which is Publish Web service on the bottom app bar, as shown in Figure 6-23.

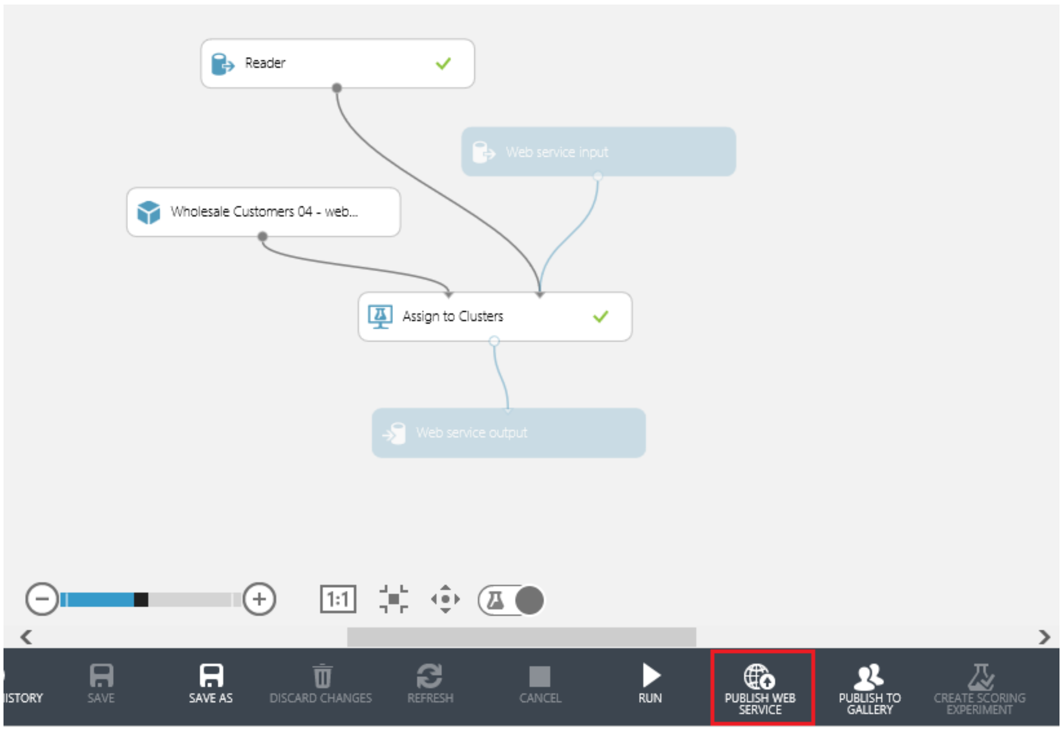
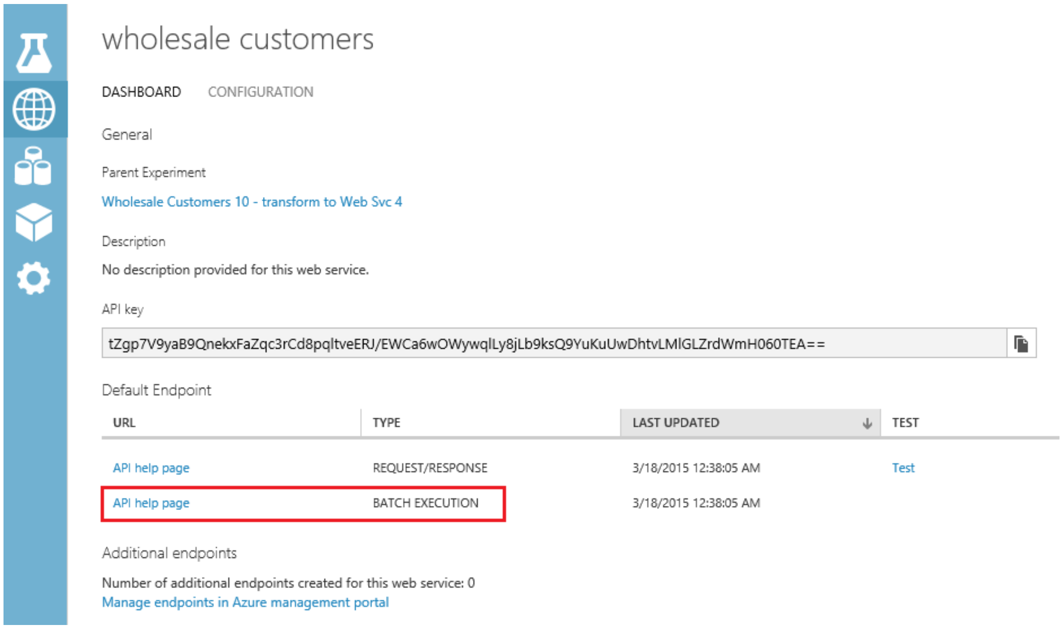


FIGURE 6-23 The wholesale customers cluster experiment in Azure ML Studio before running the PUBLISH WEB SERVICE command.

Now it's time to finally create our wholesale customers clustering web service. Click Publish Web Service and watch as Azure ML Studio automatically creates a new set of web services. Figure 6-24 shows the web services dashboard that is rendered after the new web services have been created.


wholesale customers

DASHBOARD CONFIGURATION

General

Parent Experiment

Wholesale Customers 10 - transform to Web Svc 4

Description

No description provided for this web service.

API key

tZgp7V9yaB9QnekxFaZqc3rCd8pqltveERJ/EWCa6wOWywqlLy8jLb9ksQ9YuKuUwDhtvLMlGLZrdWmH060TEA==

Default Endpoint

URL	TYPE	LAST UPDATED	TEST
API help page	REQUEST/RESPONSE	3/18/2015 12:38:05 AM	Test
API help page	BATCH EXECUTION	3/18/2015 12:38:05 AM	

Additional endpoints

Number of additional endpoints created for this web service: 0
Manage endpoints in Azure management portal

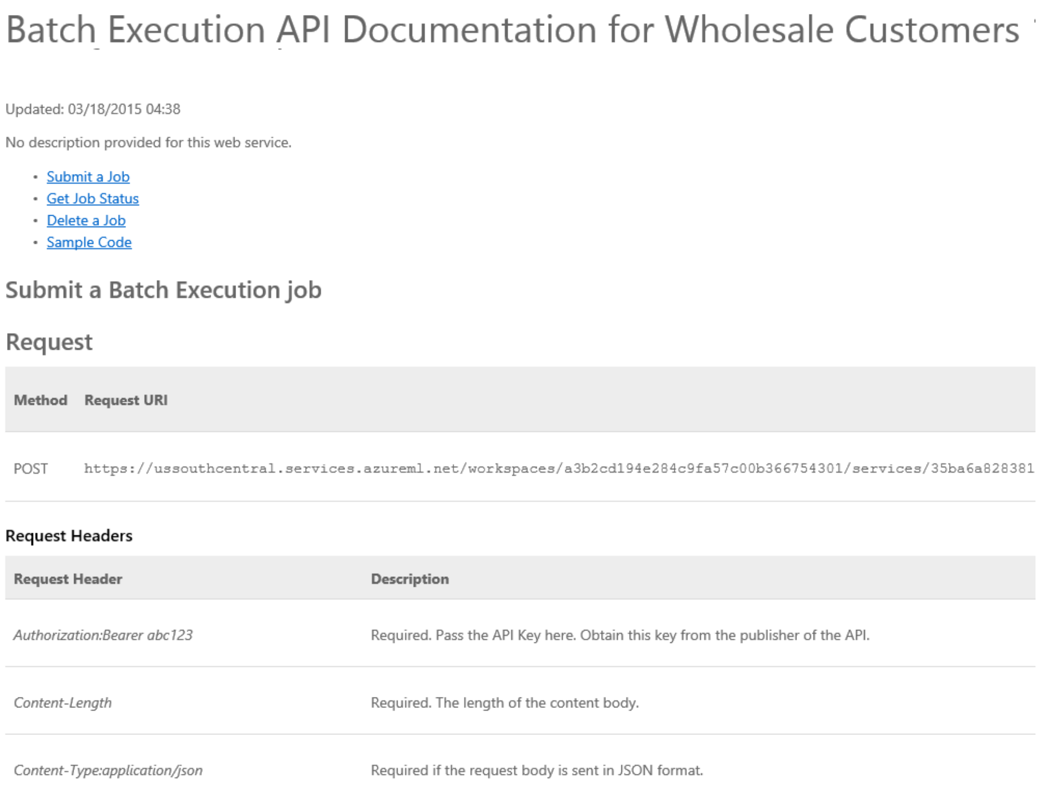
FIGURE 6-24 The wholesale customers cluster experiment web services dashboard.

It is important to note that in the previous client applications that we have developed so far in this book to demonstrate the consumption of Azure Machine Learning web services, we always used the Request/Response method for handling a single prediction transaction.

In the case of implementing web services for clustering experiments in Azure ML Studio, it makes sense that these transactions all need to occur in batches, as the ultimate goal is to assign a grouping value for each row of data.

The Batch Execution web service model provides the perfect use case for processing large sets of incoming dataset records and determining logical groupings. The K-Means Clustering algorithm relies heavily on determining the mathematical distance relationships or nearest neighbors to infer logical groupings. This would be impossible to accomplish if we only passed in one record at a time. Therefore, we conclude our web service publishing exercise by implementing a batch execution scenario to maximize the usefulness of our new predictive model.

Start by clicking API Help page, as shown in Figure 6-24. You will see a new page that provides you all the implementation details to invoke the web service in Batch Execution mode as shown in Figure 6-25.


Batch Execution API Documentation for Wholesale Customers

Updated: 03/18/2015 04:38

No description provided for this web service.

- Submit a Job
- Get Job Status
- Delete a Job
- Sample Code

Submit a Batch Execution job

Request

Method	Request URI
POST	`https://ussouthcentral.services.azureml.net/workspaces/a3b2cd194e284c9fa57c00b366754301/services/35ba6a828381`

Request Headers

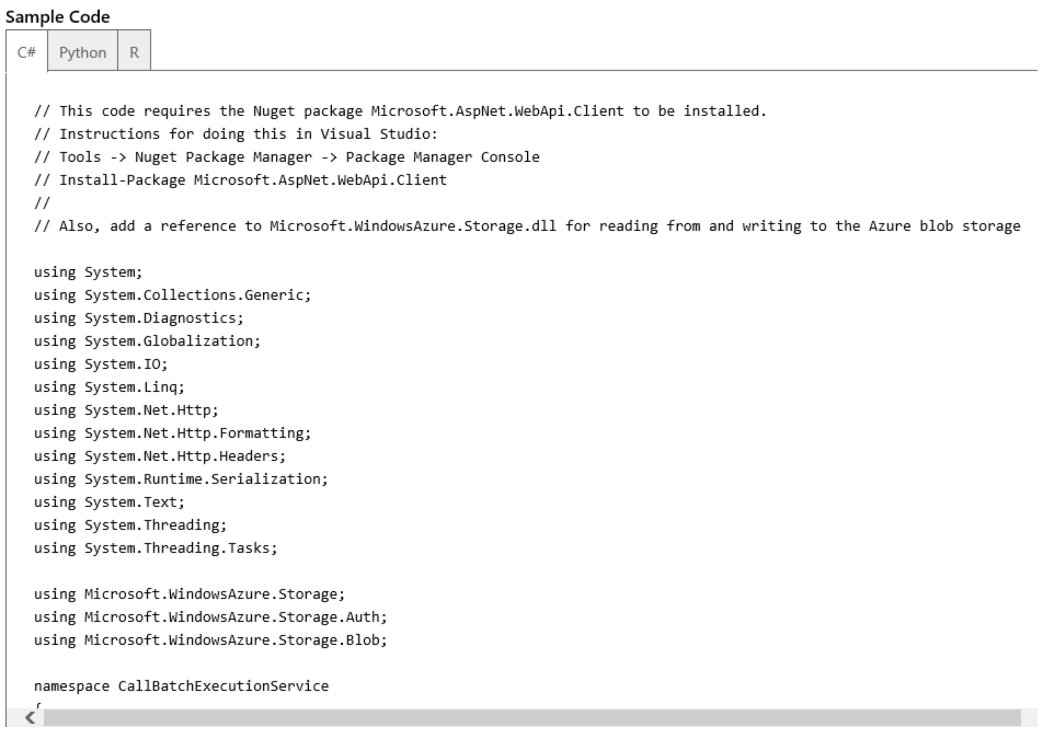
Request Header	Description
Authorization:Bearer abc123	Required. Pass the API Key here. Obtain this key from the publisher of the API.
Content-Length	Required. The length of the content body.
Content-Type:application/json	Required if the request body is sent in JSON format.

FIGURE 6-25 The Batch Execution web service documentation page for the wholesale customers experiment.

As we saw in the previous web service example in Chapter 4, "Creating Azure Machine Learning client and server applications," this help page for the Azure Machine Learning API web service provides specific implementation details about the following aspects of making a Batch Execution Azure Machine Learning web service via an HTTP POST request for the following operations:

- Submit a Job.
- Get Job Status.
- Delete a Job.

If you scroll all the way down the webpage, you will see a Sample Code section. This will help us get a jump start on creating new clients that can call our new Azure Machine Learning web service and execute batch jobs. Note in Figure 6-26 that implementation samples are provided for the programming languages C#, Python, and R.



```
// This code requires the Nuget package Microsoft.AspNet.WebApi.Client to be installed.
// Instructions for doing this in Visual Studio:
// Tools -> Nuget Package Manager -> Package Manager Console
// Install-Package Microsoft.AspNet.WebApi.Client
//
// Also, add a reference to Microsoft.WindowsAzure.Storage.dll for reading from and writing to the Azure blob storage

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Runtime.Serialization;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Auth;
using Microsoft.WindowsAzure.Storage.Blob;

namespace CallBatchExecutionService
```

FIGURE 6-26 The API help sample code for creating Batch Execution web services.

To create a C# console client to test our batch web service, simply start Visual Studio 2013 and click File/New/Project, and then specify Windows Desktop and Console Application to create a new solution.

After the solution has been created, click TOOLS option on the top menu. Then select NuGet Package Manager and Package Manager Console. Next, run the following command in the NuGet console to install the Microsoft Web API client module:

```
Install-Package Microsoft.AspNet.WebApi.Client
```

The next step is to right-click the project references and select the Add Reference option. Add a reference to the Microsoft.WindowsAzure.Storage.dll.

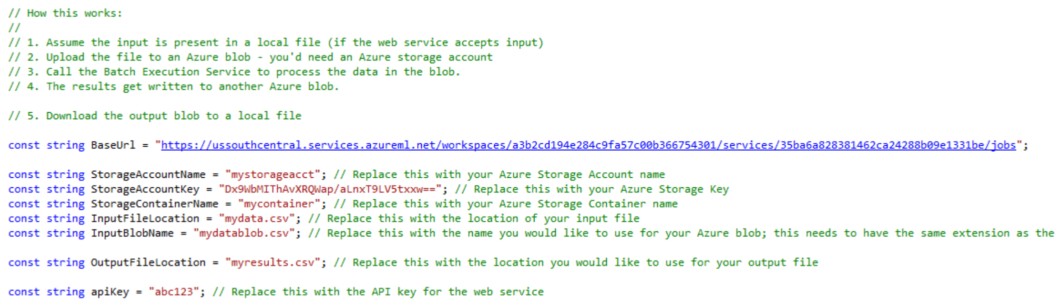
The last steps are to follow the instructions in the sample code comments to create a storage account and storage container in Azure to hold the contents of the batch input file.

You will need to replace the values for the following variables in the sample C# console application based on your specific Azure implementations:

- StorageAccountName

- StorageAccountKey
- StorageContainerName
- InputFileLocation
- InputBlobName
- OutputFileLocation
- apiKey

Figure 6-27 shows the code fragments where you will need to update these values in the C# sample code.



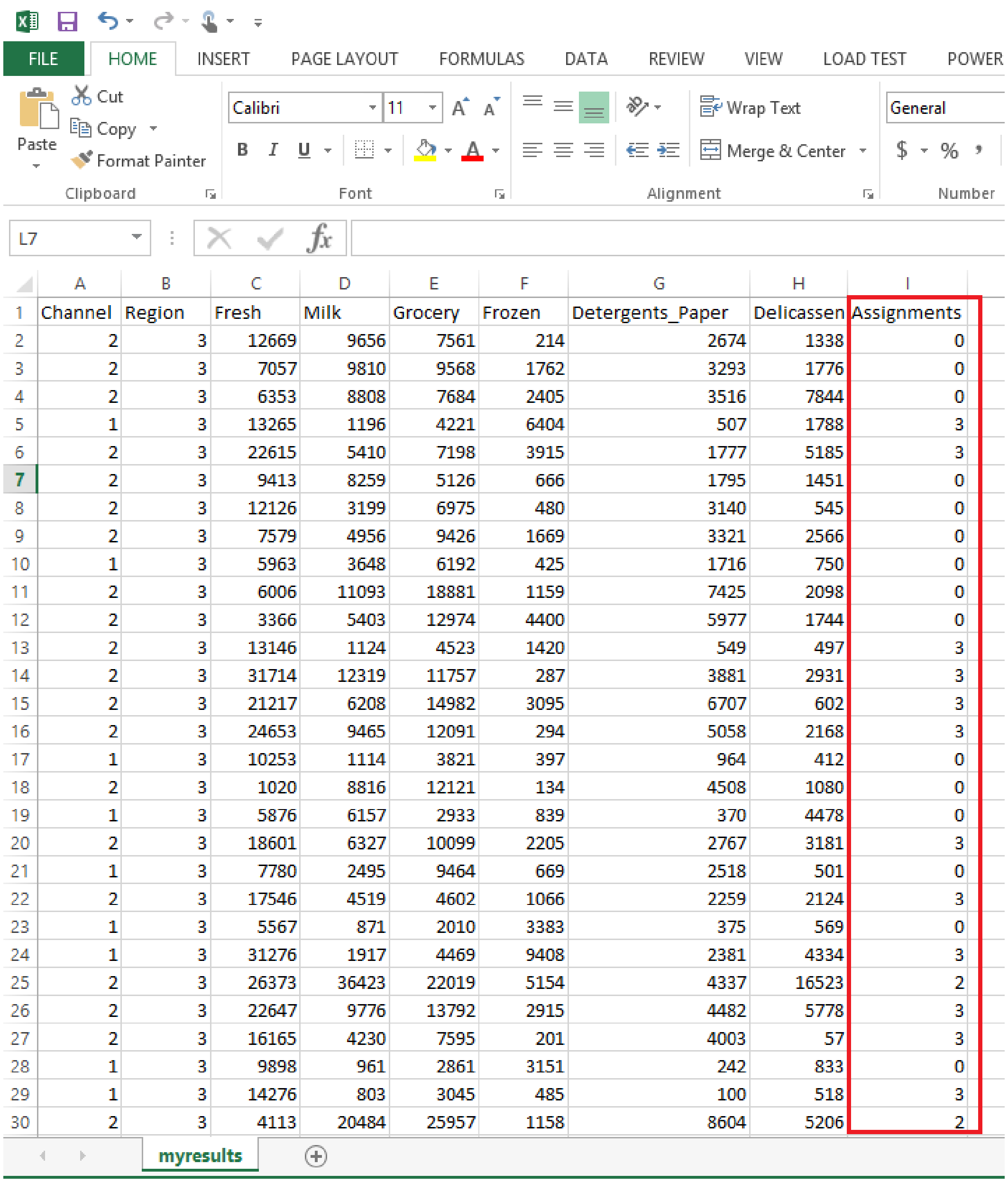
```
// How this works:
//
// 1. Assume the input is present in a local file (if the web service accepts input)
// 2. Upload the file to an Azure blob - you'd need an Azure storage account
// 3. Call the Batch Execution Service to process the data in the blob.
// 4. The results get written to another Azure blob.

// 5. Download the output blob to a local file

const string BaseUrl = "https://ussouthcentral.services.azureml.net/workspaces/a3b2cd194e284c9fa57c00b366754301/services/35ba6a828381462ca24288b09e1331be/jobs";

const string StorageAccountName = "mystorageacct"; // Replace this with your Azure Storage Account name
const string StorageAccountKey = "Dx9WbMIThAvXRQWap/aLnxT9LV5txxw=="; // Replace this with your Azure Storage Key
const string StorageContainerName = "mycontainer"; // Replace this with your Azure Storage Container name
const string InputFileLocation = "mydata.csv"; // Replace this with the location of your input file
const string InputBlobName = "mydatablob.csv"; // Replace this with the name you would like to use for your Azure blob; this needs to have the same extension as the

const string OutputFileLocation = "myresults.csv"; // Replace this with the location you would like to use for your output file

const string apiKey = "abc123"; // Replace this with the API key for the web service
```

FIGURE 6-27 The C# sample console application for invoking the Batch Execution web service.

For submitting a test input file for batch processing, you can simply download the wholesale customers dataset from the UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/Wholesale+customers.

This dataset file is already in .csv format and is actually the same file as we pointed to the parameters for the Reader module in our wholesale customers experiment.

After you run the C# sample console application, an output file is generated as a result of invoking the batch web service. This file is placed in your designated output folder. Open the file in Excel and you will see that a new column named Assignments was added to the dataset. This new column contains the Group or Clustering number that was generated for each row of data based on the relationship to the other rows of data in the dataset. Figure 6-28 shows the Excel output file results.



	A	B	C	D	E	F	G	H	I
1	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Assignments
2	2	3	12669	9656	7561	214	2674	1338	0
3	2	3	7057	9810	9568	1762	3293	1776	0
4	2	3	6353	8808	7684	2405	3516	7844	0
5	1	3	13265	1196	4221	6404	507	1788	3
6	2	3	22615	5410	7198	3915	1777	5185	3
7	2	3	9413	8259	5126	666	1795	1451	0
8	2	3	12126	3199	6975	480	3140	545	0
9	2	3	7579	4956	9426	1669	3321	2566	0
10	1	3	5963	3648	6192	425	1716	750	0
11	2	3	6006	11093	18881	1159	7425	2098	0
12	2	3	3366	5403	12974	4400	5977	1744	0
13	2	3	13146	1124	4523	1420	549	497	3
14	2	3	31714	12319	11757	287	3881	2931	3
15	2	3	21217	6208	14982	3095	6707	602	3
16	2	3	24653	9465	12091	294	5058	2168	3
17	1	3	10253	1114	3821	397	964	412	0
18	2	3	1020	8816	12121	134	4508	1080	0
19	1	3	5876	6157	2933	839	370	4478	0
20	2	3	18601	6327	10099	2205	2767	3181	3
21	1	3	7780	2495	9464	669	2518	501	0
22	2	3	17546	4519	4602	1066	2259	2124	3
23	1	3	5567	871	2010	3383	375	569	0
24	1	3	31276	1917	4469	9408	2381	4334	3
25	2	3	26373	36423	22019	5154	4337	16523	2
26	2	3	22647	9776	13792	2915	4482	5778	3
27	2	3	16165	4230	7595	201	4003	57	3
28	1	3	9898	961	2861	3151	242	833	0
29	1	3	14276	803	3045	485	100	518	3
30	2	3	4113	20484	25957	1158	8604	5206	2

FIGURE 6-28 The CSV output file opened in Excel after calling the Batch Execution web service for wholesale customer cluster predictions.

Summary

In this chapter, we took a deeper look at the K-Means Clustering algorithm in Azure ML Studio. We saw how this type of unsupervised learning can be used to infer logical groups or clusters of similar data elements to infer patterns and stratifications in large datasets.

In this chapter, we also worked with a few new Azure Machine Learning modules such as the powerful Reader module. This module can be pointed directly at files or datasets on the Web to provide easy inputs for our Azure Machine Learning experiments. We also explored the various additional modules used in conjunction with the K-Means Clustering module such as the Train Clustering module and Assign to Cluster modules.

Finally, we implemented a C# console client application to consume the Batch Execution web service for our wholesale customer experiment. The use of cluster algorithms in Azure ML Studio represents a new paradigm and requires a different approach to processing predictions in a web service environment. We also learned how to easily handle the specific requirements for processing clustering predictions, based on processing larger populations of input datasets.

Resources

For more information about Azure Machine Learning, please consult the following resources.

Documentation

- Azure ML Studio K-Means Clustering Model
- Azure Machine Learning Module reference Cluster Model
- Azure Machine Learning Module reference Train Clustering Model
- Azure Machine Learning Module reference Assign to Clusters
- Gallery Sample: Customer Segmentation of Wholesale Customers

Chapter 7

The Azure ML Matchbox recommender

Did you know that people who read the first chapter of this book also read the last chapter? Or, that people who read this book also registered for a free trial of Azure Machine Learning? In this chapter, we take a look at the Azure Machine Learning recommendation service. Recommendation engines are one of the most powerful and pervasive implementations of predictive analytics today. Recommendation engines can be found in many of today's most popular web applications and are crucial to success in many consumer industries.

Whether you are looking for a good movie, book, music, home contractor, or soul mate, the use of recommendations is a time-tested and extremely powerful force when it comes to eliciting the power of human emotion in making a purchasing decision. One of the most successful and early pioneers of this selling technique would be certainly be Amazon.com.

Recommendation engines in use today

In the case of Amazon.com, the product suggestions and recommendations are all based on aggregated data composed of your browsing, purchasing, and reading habits. This data is extrapolated to determine which books or products you might like to order next. The recommendations are then all based on similar customer transaction histories.

There are several forms of recommendations that are generated. One of the simplest and most effective is the "Also Bought" category of recommendations. In this case, the Amazon.com product detail web page displays similar titles that other customers have purchased along with your selection.

It has been rumored that Amazon has been able to achieve a 10 to 20 percent rate of sales increases through the use of this highly effective recommendation technique. One extremely brilliant, yet simple, truth is that (generally speaking) humans are naturally interested in what other humans are doing. We all have a strong internal desire to "follow the herd" in terms of making purchasing decisions.

Another highly effective form of recommendations is through the use of millions of personalized emails that Amazon generates and sends to its customers every day with personalized purchase recommendations.

The real beauty of the Amazon.com approach is that the recommendation accuracy continues to improve every time you browse or purchase more products from the website. In effect, they have implemented a very sophisticated and effective feedback loop that continually hones the accuracy of product recommendations to create one of the best web-based, virtual salespersons ever!

In terms of data sources, many of today's modern websites have many different data sources to leverage for producing recommendations. Some of these include the following:

- **Purchase history** This data is from actual sales transactions from real people with sometimes very rich demographics collected as part of a membership profile.
- **Abandoned shopping carts** This records items (and groupings of items) added to shopping carts but later abandoned.
- **Pricing and packaging experiments** These consist of online A/B testing scenarios where the same products are offered at different prices or presented in combinations to determine the effects on consumer purchasing behavior.
- **Wish lists** These are highly desired products that are placed on virtual "layaway," perhaps until the price drops or the shopper has the ways and means to make a real purchase.
- **"Park" times** This measures how long a user remains on a particular webpage before making a browsing or purchasing decision.
- **Demographics** This consists of associated personal data gleaned from membership profile information, billing and shipping address data, or social network graphs.
- **Referral sites** Additional insights about the shopper or product can be gleaned by intercepting the URL from which a shopper has originated.
- **Ratings** Purchase patterns and behaviors of products with low, medium, or high social network ratings can be analyzed.
- **Email click-through data** This data tracks user behavior as a result of email marketing campaigns, which can reveal how compelling product offers are and the likelihood that responses can be turned into purchases.
- **Browser sessions** Analyzing user navigation flows and patterns within a website can provide invaluable feedback for making recommendations when translated into usage and behavior analytics.
- **Number of times viewed** Understanding how many times a user views a product before making a final purchase can reveal powerful insights into what converts a shopper into a buyer.

In addition to retail applications such as Amazon.com, recommendation algorithms are also extremely popular in the entertainment industry. In certain applications, such as movie rentals, Netflix has revealed that as much as 75 percent of the content watched on the service comes from its recommendation engine.

Back in the good old days of video and DVD rentals, movie analytics and recommendations were extremely important on the supply side of the business. These types of data insights were absolutely critical when it came to knowing exactly how many copies of a title to have in inventory to make sure every customer went home happy. Too few copies of popular titles caused immediate stock-outs, waiting lists, and customer satisfaction issues. Too many copies of the wrong titles caused fire sales and immediate product markdowns to make shelf space for the next batch of titles.

With the advent of instant video streaming technologies, the use of recommendation engines has become even more crucial in determining success. Companies such as Netflix seek to continuously optimize the member experience. They have measured significant gains in member satisfaction whenever they improve the personalization experience for their members.

In the case of Netflix, careful analysis is performed to present the member with a completely personalized home page representing the top 10 videos that are recommended for viewing. Nothing is left to chance. The specific titles, genres, and order of presentation are all carefully calculated to maximize the experience to drive entertainment value and satisfaction.

Many popular online dating sites have also recognized the benefits of leveraging recommendation engines and algorithms to find a perfect match for users. In many cases, this becomes a highly complex and challenging design consideration. A prerequisite to success is having members complete an in-depth personality questionnaire. This is both a chore and the secret ingredient to superior soul-mate recommendations.

The trick becomes how to keep a user engaged while completing a tedious questionnaire that is typically the mission-critical component of a site's recommendation engine. The more honestly and thoroughly a user answers these questions, the more compatible the suggested matches will be.

In addition to retail and entertainment applications, recommendation systems are now common in insurance and finance, where they can suggest coverage plans and investment opportunities, and they are also showing up in business-to-business (B2B) settings, recommending prospects and strategies to salespeople.

Mechanics of recommendation engines

Many of today's modern recommendation engines use a number of different technologies to achieve results. These techniques can be divided into two primary methodologies to make intelligent predictions.

- **Collaborative filtering** This method is used to analyze a mass of past choices, from one user or many users, to make new suggestions. Collaborative filtering applications typically involve very large data sets. The items recommended to a user are those preferred by similar users. This is the classic Amazon.com "Also Bought" category of recommendations.

- **Content-based filtering** This method takes attributes of one item to suggest new items with similar properties. For example, if a Netflix user has watched many science fiction movies, then the system would most likely recommend a movie classified in the database from the same genre.

In reality, the two methods are often combined and even incorporated with other predictive algorithms such as clustering to maximize the recommendation engine accuracy. These highly tuned algorithms soon become mission-critical intellectual property (IP). They might even become the cornerstone for many organization's success or failure.

Azure Machine Learning Matchbox recommender background

Historically, advanced recommendation engines and predictive services were the exclusive domain of only the largest online web properties. This was due to the fact that the development, care, and feeding of such systems required a dedicated team of highly trained professionals. Now, with the advent of Microsoft Azure Machine Learning and the democratization of data science, recommendation engine capabilities are accessible to a much wider audience of individuals and businesses. Now more people can use it for the benefit of their own customers.

The early origins of the Azure Machine Learning Matchbox recommender modules can be found in the earlier work of the Microsoft Research team. The focus of the team was to discover how best to connect people with other people, content, or products they care about via the Web. Their research explored tasks like product recommendations, social matchmaking, targeted advertising, and content filtering. Ultimately, they developed a large-scale Bayesian recommender system called Matchbox. More information on the project can be found at http://research.microsoft.com/apps/pubs/default.aspx?id=79460

The Microsoft Research Matchbox recommender engine was designed to learn about people's preferences from observing how they rated items such as movies, content, or other products. Based on those observations, the Matchbox recommender was then able to recommend new items to users on request.

The Matchbox recommender was designed to use the available data for each user as efficiently as possible. Its learning algorithm is designed specifically for the large streams of data typical for web-scale applications. One of the main features is that the Matchbox recommender takes advantage of metadata available for both users and items. This means that things learned about one user or item can then be extrapolated to other users or items.

Basically, there are two types of entities involved in the Azure Matchbox recommendation engine. These can be thought of as users and items. Users are the people to whom you would like to make recommendations. Items are the things you would like to recommend to them such as movies, books, webpages, recipes, or even other people.

The Azure Machine Learning Matchbox recommender engine makes use of content information in the form of user and item metadata. It then combines that information with the collaborative filtering information from previous user behavior to predict the value of an item for a user.

In Azure Machine Learning, the recommendation engine is enabled when you build a recommendation model based on the following data elements:

- A catalog or repository of the items you want to recommend.
- Data representing the usage of items per user or session. This data can be acquired over time via data acquisition.

After a recommendation model is built, you can use it to predict items that a user might be interested in, according to a set of items (or a single item).

One critical problem for a recommendation engine is how to progress from a cold start. In most cases, new users might not have rated enough items, and new items might not have been rated by enough users to make accurate predictions. To mitigate this problem, the Azure Machine Learning Matchbox recommender makes it possible to represent users and items not just by their ID, but by a feature vector constructed from metadata. For users, this could include profile information such as age or geolocation. For items such as movies, it might include information such as genre, actors, director, year of production, and so on. In this way, Azure Machine Learning can generalize across users and items by making use of common attributes in the metadata to overcome the cold start problem.

When constructing Azure Machine Learning experiments using the Matchbox recommender algorithm, the following modules will typically be incorporated into the design.

- **Train Matchbox Recommender** This module trains the model for the Matchbox recommender engine. The recommendation algorithm is based on the Matchbox model developed by Microsoft Research. This module takes a dataset of user-item-rating triples and returns a trained Matchbox recommender.
- **Score Matchbox Recommender** This module supports four different kinds of predictions:
    - Predict ratings for a given user and item.
    - Recommend items to a given user.
    - Find users related to a given user.
    - Find items related to a given item.

  The latter three kinds of predictions can operate in two modes: one that considers all entities (typically used in production), and one that operates on a reduced set of entities (typically used in experimentation or model evaluation).

- **Evaluate Recommender** This module can measure the accuracy of four different kinds of predictions made by a recommendation model:
  - Ratings predicted for a given user and item.
  - Items recommended for a given user.
  - A list of users potentially related to a given user. The users are predicted to have similar preferences.
  - A list of items potentially related to a given item.

  To evaluate a set of predictions, you need two datasets as inputs:
  - **A test dataset** This set contains user-item-rating triples. If you have an existing dataset that is used to train a model, you can create a test dataset by using Split and choosing Recommender Split as the splitting mode.
  - **A scored dataset** You can create this dataset by using Score Model and selecting one of the options in the Recommender Scorer.

Azure Machine Learning Matchbox recommender: Restaurant ratings

To demonstrate the Azure Machine Learning Matchbox recommendation engine in action, we use a familiar scenario. We create an experiment that will predict restaurant ratings.

In this example, we would like to recommend a restaurant to a given user based on the five-star ratings that this user and other users have provided for some of the restaurants in the neighborhood. This recommendation task can be broken down into two discrete steps:

1. Predict for each restaurant, on a rating scale of one to five stars, how the user would rate the restaurant.
2. Recommend from a list of eligible restaurants which ones would receive the highest rating by that user.

The problem then becomes predicting how a specific user would rate all of those restaurants that he or she has not visited and actually rated. You can think of this as a large matrix, with users as rows, items as columns, and entries as ratings. Figure 7-1 illustrates the problem at hand where we have a matrix of restaurants (items) and diners (users). The matrix is considered “sparse” due to the fact that not every intersection between restaurants and diners has been populated with a corresponding rating.

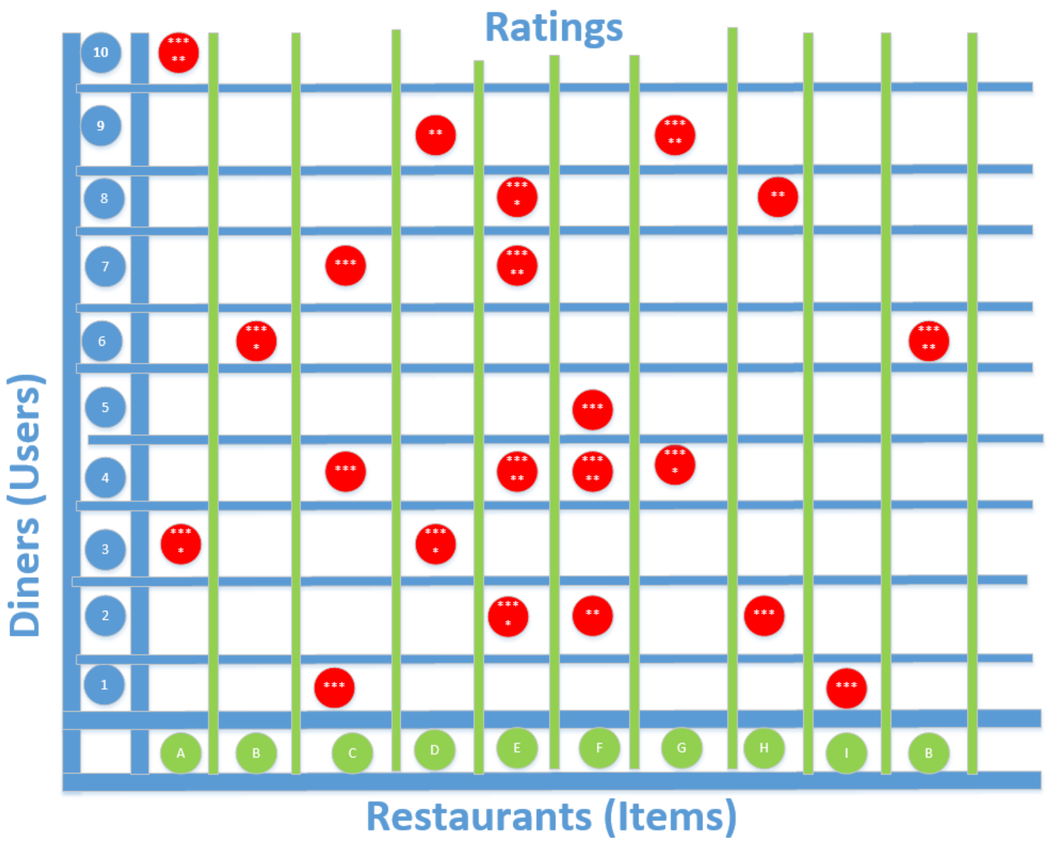


FIGURE 7-1 The restaurant ratings sparse matrix problem domain.

This problem domain is where Azure Machine Learning and the Matchbox recommender engine come into play. To build a machine learning model that can predict arating, we need to collect data in the form of userID, itemID, and ratings. This data works against a given user–item combination and predicts how the user would rate the individual item.

The basic workflow to generate a predictive solution using the Azure Machine Learning Matchbox recommender engine consists of the following high-level steps:

1. **Create a model** A model is a container of your usage data, catalog data, and the recommendation model.
2. **Import catalog data** This is an optional step. A catalog contains metadata information on the items. If you do not upload catalog data, the recommendation's services will learn about your catalog implicitly from the usage data.

3. **Import usage data** Usage data can be uploaded in one of two ways (or both):
   - By uploading a file that contains the usage data.
   - By sending data acquisition events. Usually you upload a usage file to be able to create an initial recommendation model (bootstrap). You would use this usage file until the system gathers enough data by using the data acquisition format.
4. **Build a recommendation model** This is an asynchronous operation in which the recommendation system takes all the usage data and creates a recommendation model. This operation can take several minutes or several hours depending on the size of the data and the build configuration parameters. When triggering the build, you will get a build ID. Use it to check when the build process has ended before starting to consume recommendations.

Building the restaurant ratings recommender

Let's get started building a restaurant ratings recommender experiment. The first step after signing into the Azure portal and creating an Azure Machine Learning workspace is to sign in to Azure Machine Learning Studio and create a new, blank experiment.

For our experiment, we use three separate datasets as input to train the Azure Machine Learning Matchbox recommender consisting of the following files:

- **Restaurant features** This includes information about each restaurant, such as placeID, location, name, address, state, whether alcohol is served, smoking policy, dress code, accessibility, and price.
- **Restaurant customers** This includes a wide variety of personal attributes including the following: userID, smoker, drink level, dress preference, marital status, birth year, interests, personality traits, religion, favorite color, weight, height, and budget.
- **Restaurant ratings** This incudes key fields like userID, placeID, and rating.

These three data files are included in the Azure Machine Learning sample Saved Datasets repository. To easily locate the files, simply type **restaurant** into the Search bar on the left side of the Azure Machine Learning Studio designer surface. Alternatively, you can find them under the Saved Datasets tab and locate them alphabetically below the tab.

Once you have located the files, drag and drop them onto the Azure Machine Learning designer surface as shown in Figure 7-2.

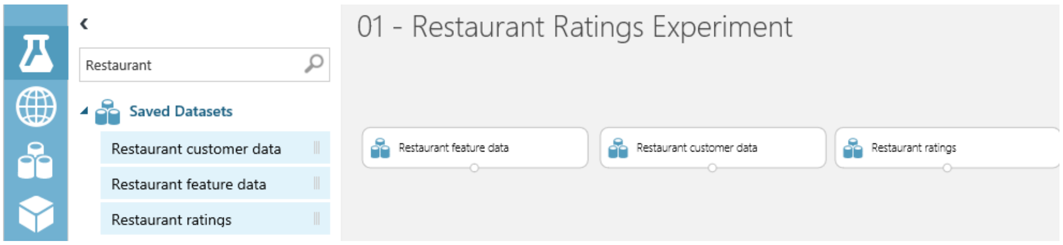
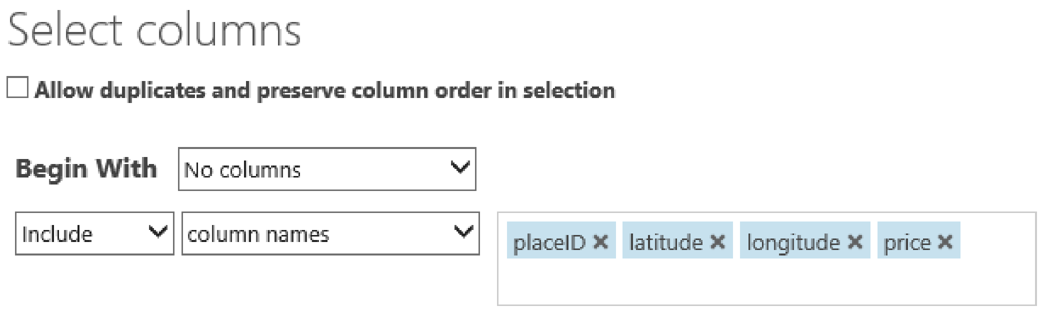


FIGURE 7-2 Populating the three restaurant ratings experiment data files from the sample Azure Saved Datasets repository.

The next step is to add two of the Project Columns modules to the experiment so that we can filter out unnecessary columns in the Restaurant Feature Data and Restaurant Customer Data datasets. Locate the module by typing **Project Columns** into the search bar on the left side of the Azure Machine Learning Studio designer surface. You can also manually find the module under the Data Transformation/Manipulation tab.

Drag and drop two copies of the Project Columns module directly underneath the Restaurant Feature Data and Restaurant Customer Data datasets on the designer surface. Next, connect the bottom of the dataset modules to the top of the Project Columns modules. You will notice a red exclamation mark indicating that we need to set the properties of the Project Columns modules.

Start by clicking the Project Columns module that is underneath and connected to the Restaurant Feature Data module. Click Launch Column Selector and set the selector to only include the following columns: placeID, latitude, longitude, and price, as shown in Figure 7-3.


Select columns

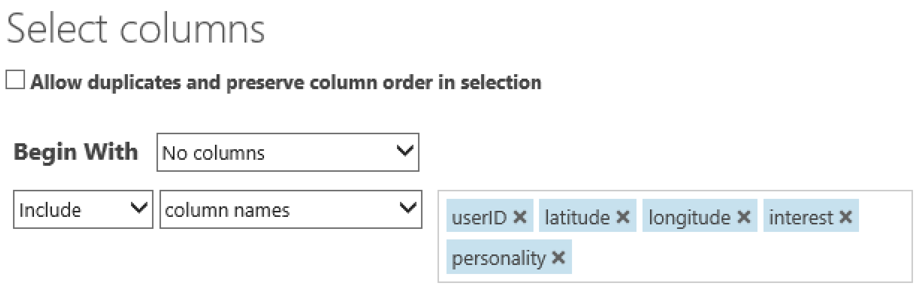
☐ Allow duplicates and preserve column order in selection

Begin With: No columns

Include | column names | placeID ✕ latitude ✕ longitude ✕ price ✕

FIGURE 7-3 Column selection values for the Project Columns module associated with the Restaurant Feature Dataset module.

Next, configure the columns to include from the Restaurant Customer Dataset module by clicking the Project Columns module that is underneath the module. Click Launch Column Selector and set the selector to only include the following columns: placeID, latitude, longitude, interest, and personality, as shown in Figure 7-4.


Select columns

☐ Allow duplicates and preserve column order in selection

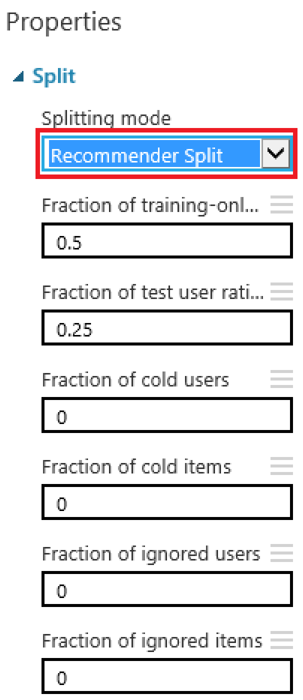
Begin With: No columns

Include | column names | userID, latitude, longitude, interest, personality

FIGURE 7-4 Column selection values for the Project Columns module associated with the Restaurant Customer Dataset module.

Next, we add a Split module underneath the Restaurant ratings dataset to specify that half of this dataset is to be used for training and the other half for scoring the new recommendation model.

Type **Split** into the search bar on the left side of the Azure Machine Learning Studio designer surface to quickly locate the module. Alternatively, you can find it under Data Transformation/Sample And Split. Drag and drop the Split module underneath the Restaurant Ratings dataset and connect the modules. Click the Split module to set the Property pane on the right side of the designer. Click the Splitting Mode drop-down list-box and set the value to Recommender Split, as shown in Figure 7-5.


Properties

Split

Splitting mode: Recommender Split

Fraction of training-onl...: 0.5

Fraction of test user rati...: 0.25

Fraction of cold users: 0

Fraction of cold items: 0

Fraction of ignored users: 0

Fraction of ignored items: 0

FIGURE 7-5 The Properties settings for the Split module for the Restaurant Ratings dataset.

Now it's time to add the Train Matchbox Recommender module to our experiment. Type **Train matchbox** into the search bar on the left side of the Azure Machine Learning Studio designer surface to quickly locate the module. Alternatively, you can find it under Machine Learning/Train, as shown in Figure 7-6.

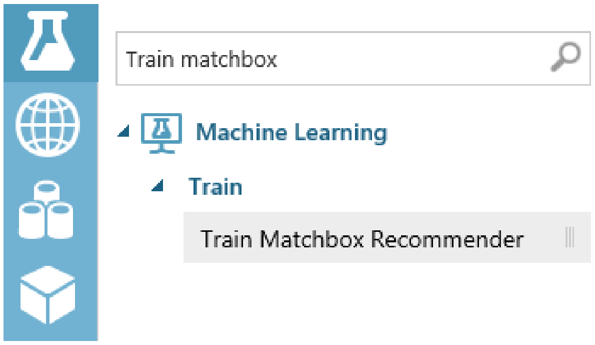


FIGURE 7-6 The Train Matchbox Recommender module in the Azure Machine Learning Studio modules menu.

Drag and drop the Train Matchbox Recommender module underneath the Project Columns module. The Project Columns module should be underneath the Restaurant Customer dataset in the middle of the designer surface.

Note that there are three input connectors and order does matter when it comes to connecting modules to the three top inputs on the Train Matchbox Recommender module. Here are the exact inputs and their ordering.

1. **Training dataset of user-item-rating triples** Ratings of items by users, expressed as a triple (User, Item, Rating).
2. **Training dataset of user features** Dataset containing features that describe users.
3. **Training dataset of item features** Dataset containing features that describe items.

Here's how to make these critical connections to the Train Matchbox Recommender module in our experiment, moving from right to left:

- Connect the Project Columns module under the Restaurant Feature dataset to the **top right connector**.
- Connect the Project Columns module under the Restaurant Customer dataset to the **top middle connector**.
- Connect the bottom left Split module under the Restaurant Ratings dataset to the **top left connector** of the Train Matchbox Recommender module.

Figure 7-7 shows the order of the inbound connectors.

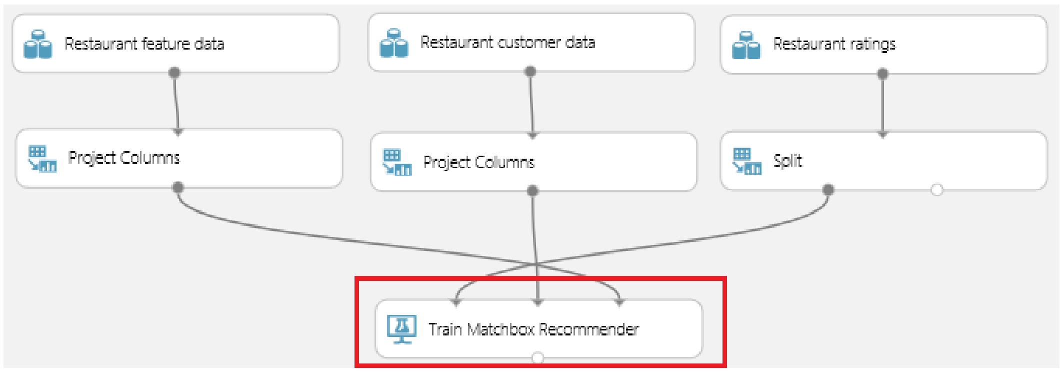


FIGURE 7-7 The inbound connections to the Train Matchbox Recommender module.

The next step is to add a Score Matchbox Recommender module to our experiment. Type **Score Matchbox** into the search bar on the left side of the Azure Machine Learning Studio designer surface to quickly locate the module. Alternatively, you can find it under Machine Learning/Score, as shown in Figure 7-8.

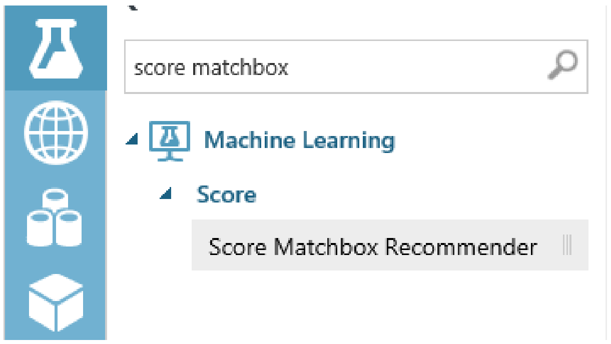


FIGURE 7-8 The Train Matchbox Recommender module in the Azure Machine Learning Studio modules menu.

Drag and drop the Score Matchbox Recommender module directly under the Train Matchbox Recommender module. Note that there are four input connectors for this module. Just like the Train Matchbox Recommender module, there is an explicit order of input connectors required for this module to work correctly. The required connection inputs and the order required (from left-to-right) are as follows.

1. Trained Matchbox recommender.
2. Dataset to score.
3. User features: Dataset containing features that describe users.
4. Item features: Dataset containing features that describe items.

Armed with this knowledge, let's wire up the input connections to the top four connectors of the Score Matchbox Recommender module as follows.

1. Connect the Train Matchbox Recommender module to the top left connector.
2. Connect the bottom right Split module to the top second from left connector.
3. Connect the Project Columns module under the Restaurant Customer Data module to the top third from left connector.
4. Connect the Project Columns module under the Restaurant Feature Data module to the top right connector.

Figure 7-9 shows the order and sequence of the inbound connectors for the Score Matchbox Recommender module in our experiment.

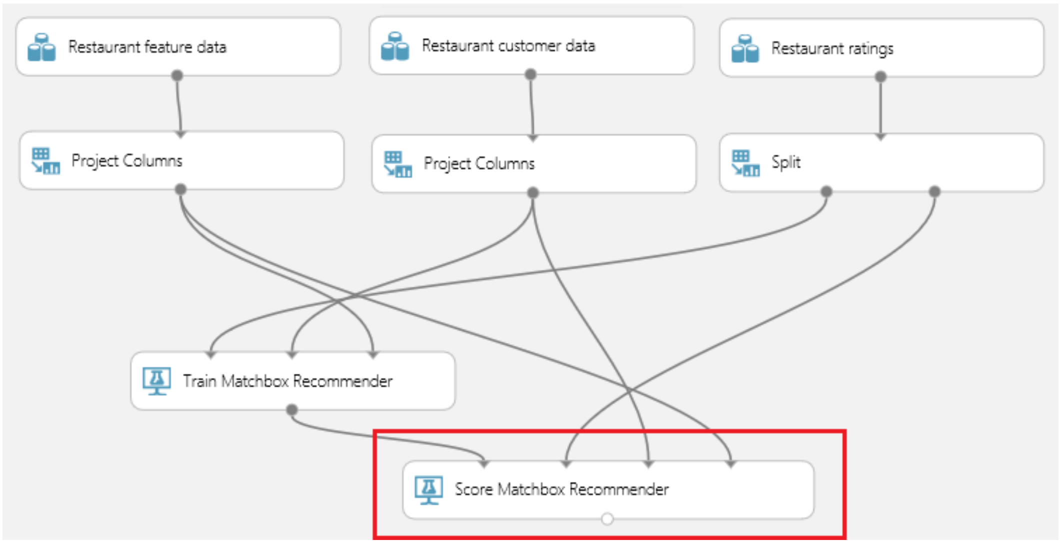
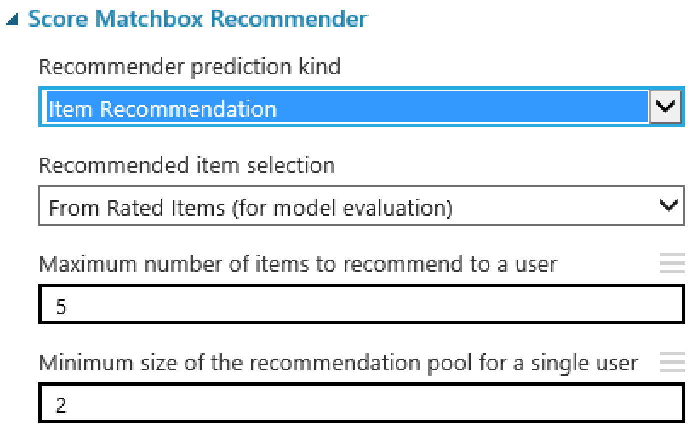


FIGURE 7-9 The inbound connections to the Score Matchbox Recommender module.

Now, click the Score Matchbox Recommender module to see the properties in the right pane of Azure Machine Learning studio. You will notice that there is a drop-down list box that allows us to specify what kind of prediction we want to make. The following are the possible options.

- **Rating Prediction** Predict the rating that a customer will give a particular restaurant.
- **Item Recommendation** Predict which restaurants will be most highly rated by the user.
- **Related Users** Predict which customers (users) are most like this customer.
- **Related Items** Predict which restaurants (items) are most like this restaurant.

Figure 7-10 displays the properties available when the Recommender Prediction Kind parameter is set to Item Recommendation. In this case, there are three additional values that can be set. Later, when we test the Matchbox recommender experiment, we can use these options to control things like how many restaurant recommendations to return to a user.


Score Matchbox Recommender

Recommender prediction kind
Item Recommendation

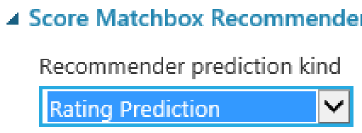
Recommended item selection
From Rated Items (for model evaluation)

Maximum number of items to recommend to a user
5

Minimum size of the recommendation pool for a single user
2

FIGURE 7-10 The Score Matchbox Recommender module parameters for the Recommender Prediction Kind when the value has been set to Item Recommendation.

Note that if we set the Recommender Prediction Kind parameter to Rating Prediction as in Figure 7-11, there are no other parameters to set. This means that the model will simply return a prediction of how many stars a particular restaurant will receive from a particular user.


Score Matchbox Recommender

Recommender prediction kind
Rating Prediction

FIGURE 7-11 The Score Matchbox Recommender module parameters for the Recommender Prediction Kind when the value has been set to Rating Prediction.

At this point, it is important to point out that there is a critical decision to make when configuring the Score Matchbox Recommender module. The property values that are set in this one module will have a direct impact on the type and behavior of the prediction that is returned.

The downstream impacts continue when we convert the model to a web service. Depending on the configuration, the model will predict any one of the four options available in the Recommender Prediction Kind drop-down list. Additionally, the input parameters will also change. The input parameter changes will require reconfiguring the web service inputs to point to the correct input stream for the desired prediction output.

The last module to add to our Matchbox recommender experiment is the Evaluate Recommender module. Again, you can locate this module by typing **Evaluate Recommender** into the search bar on the left side of the Azure Machine Learning Studio designer surface to quickly locate the module. Alternatively, you can also find it under Machine Learning/Evaluate, as shown in Figure 7-12.

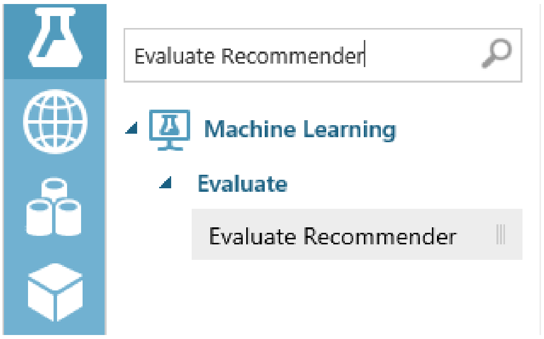


FIGURE 7-12 The Evaluate Recommender module in the Azure Machine Learning Studio modules menu.

After locating the Evaluate Recommender module, drag and drop it onto the designer surface. Place it under the Score Matchbox Recommender module. Next, make the following connections to wire up the module:

- Connect the bottom right connector of the Split module to the top left connector on the Evaluate Recommender module.
- Connect the bottom connector of the Score Matchbox Recommender module to the top right connector on the Evaluate Recommender module.

At this point, we can save and run our experiment to process the model. Figure 7-13 shows our completed Restaurant Matchbox recommender experiment.

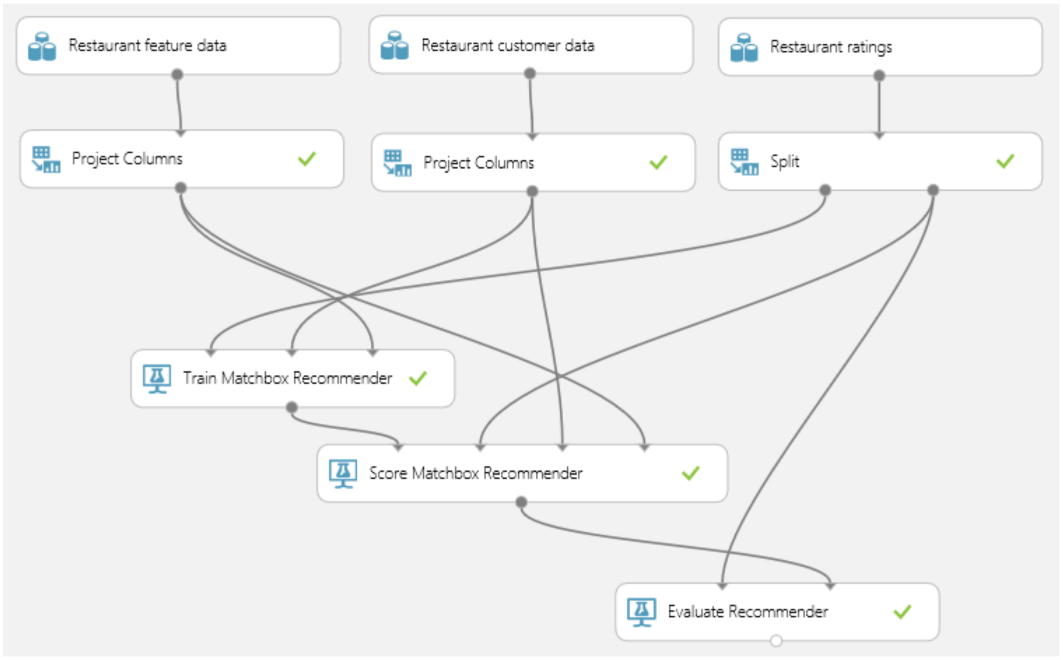


FIGURE 7-13 The completed Restaurant Matchbox recommender experiment in Azure Machine Learning Studio.

To check the prediction data that our new model has generated, click the bottom connector of the Score Matchbox Recommender module and select Visualize, as shown in Figure 7-14.

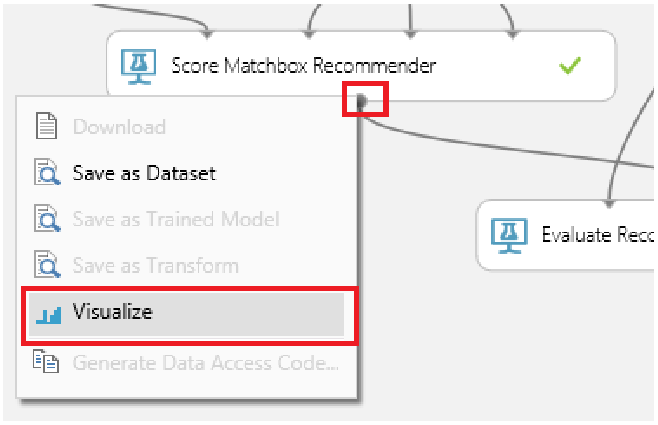
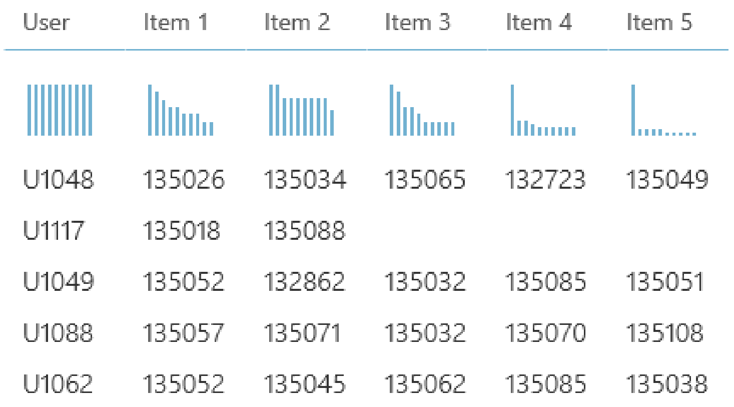


FIGURE 7-14 Selecting the Visualize option for the Score Matchbox Recommender module.

After selecting the Visualize option, you will see a modal window that displays the prediction dataset. Figure 7-15 illustrates the dataset that is produced when the Score Matchbox Recommender module properties have been set to predict using the Item Recommendation option. In this case, the prediction model will return a set of five recommended restaurants for a given user. The list of five restaurants is returned in order by ratings, from highest to lowest.

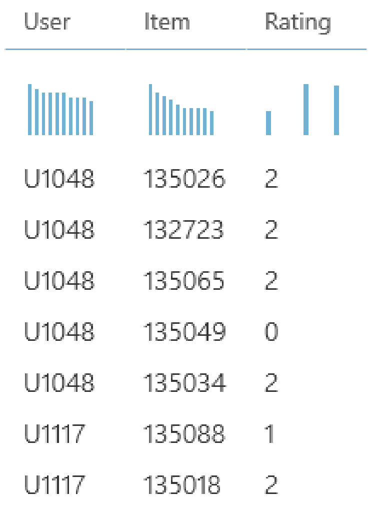


User	Item 1	Item 2	Item 3	Item 4	Item 5
U1048	135026	135034	135065	132723	135049
U1117	135018	135088			
U1049	135052	132862	135032	135085	135051
U1088	135057	135071	135032	135070	135108
U1062	135052	135045	135062	135085	135038

FIGURE 7-15 The generated dataset for the Score Matchbox Recommender module when the properties are set using the Item Recommendation option.

Now, let's reset the properties of the Score Matchbox Recommender module to see what ratings are predicted for a given user and a given restaurant. To reconfigure, click the Score Matchbox Recommender module. Set the Recommender Prediction Kind parameter to Rating Prediction. Then click Run on the bottom app bar to regenerate the Matchbox recommender prediction model.

After the Run command has finished, click the bottom connector of the Score Matchbox Recommender module and select the Visualize option. This time the modal window displays a completely different prediction dataset. This new dataset only contains three columns for User, Item, and Rating as shown in Figure 7-16.



User	Item	Rating
U1048	135026	2
U1048	132723	2
U1048	135065	2
U1048	135049	0
U1048	135034	2
U1117	135088	1
U1117	135018	2

FIGURE 7-16 The generated dataset for the Score Matchbox Recommender module when the properties are set using the Rating Prediction option.

Creating a Matchbox recommender web service

Now it's time to create a web service from our completed experiment. Because this step will dramatically alter our existing experiment, now is a good time to click Save on the bottom app bar to save the current state.

After the save is completed, click Save As on the bottom app bar to save the experiment under a new name, as we will be dramatically altering it to transform it into a web service. Next, click Run to process the new, renamed version of the experiment.

To start the transformation into a web service, click on Create Scoring Experiment on the bottom app bar as shown in Figure 7-17.

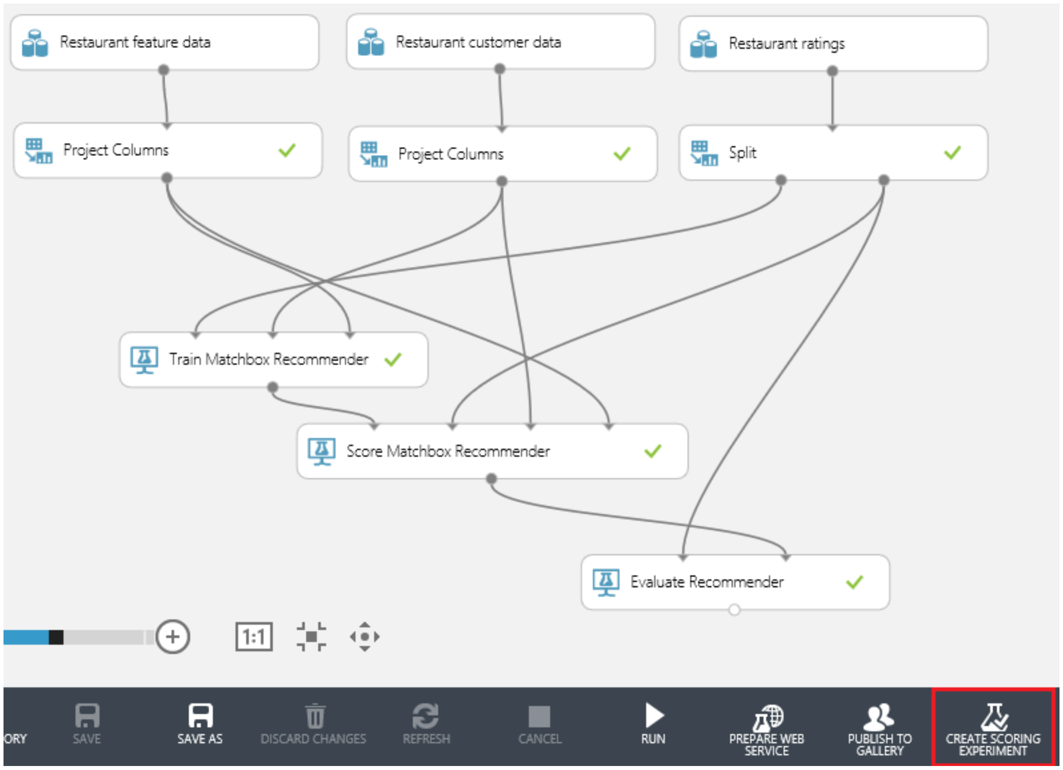


FIGURE 7-17 The completed experiment ready to be processed by clicking Create Scoring Experiment.

After the Create Scoring Experiment command has completed, you will notice that the experiment has been modified to streamline the operations.

The changes made to our experiment reduce the number of modules in the experiment. The Azure Machine Learning Studio designer now contains the following modules:

- Matchbox Recommender training module.
- Evaluate modules have been replaced with the newly “trained” model.
- Two new web service input and output endpoints have been added to the experiment.
- Click Run again to process our newly modified model.

Now we are finally ready to publish our restaurant ratings Matchbox recommender experiment as a web service. Click Publish Web Service as shown in Figure 7-18.

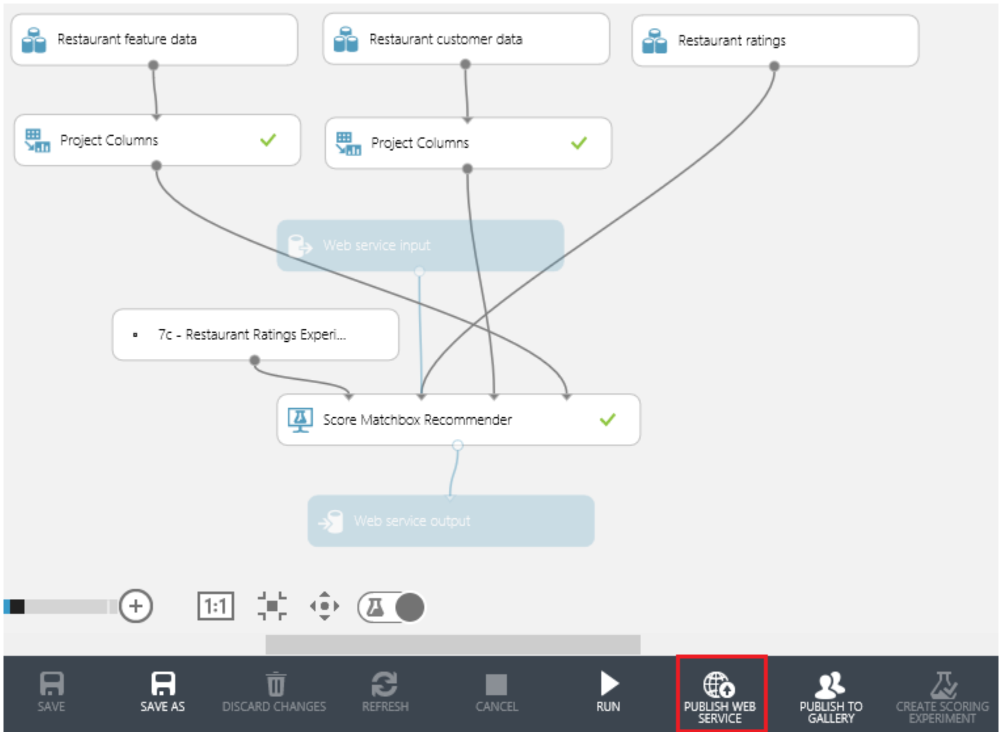
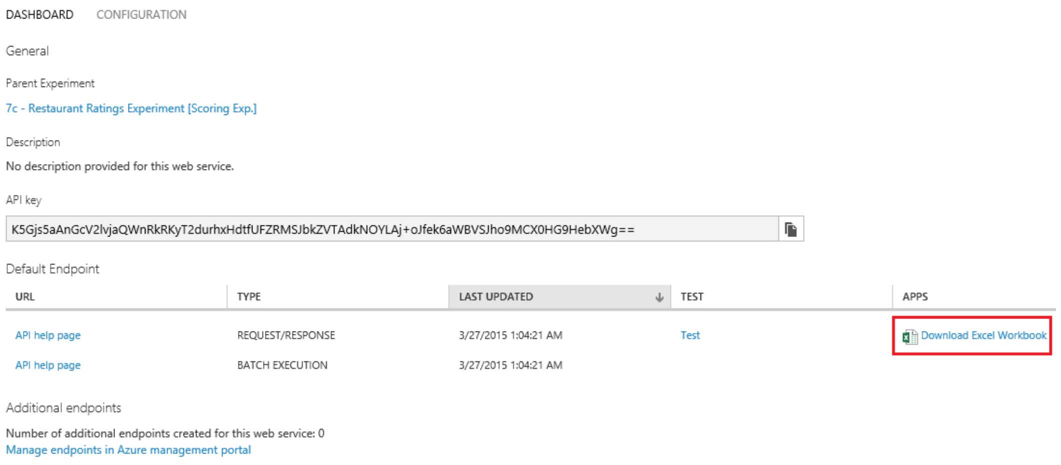


FIGURE 7-18 The completed experiment ready to be transformed into a web service.

After the web service has been generated for our experiment, a new page will be displayed showing the Azure Machine Learning dashboard view. To easily test our new Azure Machine Learning web service from within an Excel client, simply click Download Excel Workbook as shown in Figure 7-19.



DASHBOARD CONFIGURATION

General

Parent Experiment

7c - Restaurant Ratings Experiment [Scoring Exp.]

Description

No description provided for this web service.

API key

K5Gjs5aAnGcV2lvjaQWnRkRKyT2durhxHdtfUFZRMSJbkZVTAdkNOYLAj+oJfek6aWBVSJho9MCX0HG9HebXWg==

Default Endpoint

URL	TYPE	LAST UPDATED	TEST	APPS
API help page	REQUEST/RESPONSE	3/27/2015 1:04:21 AM	Test	Download Excel Workbook
API help page	BATCH EXECUTION	3/27/2015 1:04:21 AM		

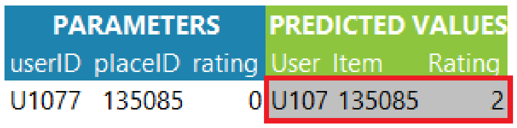
Additional endpoints

Number of additional endpoints created for this web service: 0

Manage endpoints in Azure management portal

FIGURE 7-19 The Azure Machine Learning web service dashboard for the restaurant ratings Matchbox recommender experiment.

After downloading the Excel workbook, we can quickly test our new web service by entering values in the left Parameters section of the worksheet. In the example in Figure 7-20, we entered a valid user ID and a valid restaurant ID along with a value of zero for the rating to see what would be predicted. After leaving the last cell in the Parameters section, the Excel workbook executed a macro to call our new web service. The returned results are updated in the Predicted Values section. In this case, we received a prediction that our designated user would provide a two-star rating for our designated restaurant.



PARAMETERS			PREDICTED VALUES		
userID	placeID	rating	User	Item	Rating
U1077	135085	0	U107	135085	2

FIGURE 7-20 Testing the new Azure Machine Learning restaurant ratings Matchbox recommender web service using an Excel client.

And there you have it! A complete, end-to-end Azure Machine Learning recommendation engine for predicting restaurant ratings using the Matchbox recommender engine. This complete example can be also found in the Azure Machine Learning Sample Gallery. So you can either build it from scratch or work with the completed experiment to help jump start your machine learning journey.

Summary

In this chapter, we explored the exciting world of recommendation engines, which is one of the most popular uses of predictive analytics and machine learning today. We worked through a complete example by building an end-to-end Azure Machine Learning recommendation engine for predicting restaurant ratings using the Matchbox recommender engine.

Resources

For more information about Azure Machine Learning, please consult the following resources.

Microsoft Research

- Matchbox Recommender System

  http://research.microsoft.com/en-us/projects/matchbox/

- Matchbox: Large-Scale Bayesian Recommendations

  http://research.microsoft.com/apps/pubs/default.aspx?id=79460

Documentation

- Train Matchbox Recommender

  http://help.azureml.net/Content/html/FA4AA69D-2F1C-4BA4-AD5F-90EA3A515B4C.htm

- Score Matchbox Recommender

  http://help.azureml.net/Content/html/55544522-9A10-44BD-884F-9A91A9CEC2CD.htm

- Evaluate Recommender

  http://help.azureml.net/Content/html/E9AD68A7-E91B-4AE6-800E-B5EE7E22CD17.htm

- Setting up and using Machine Learning Recommendations API FAQ

  http://azure.microsoft.com/en-us/documentation/articles/machine-learning-recommendation-api-faq/

- Quick start guide for the Machine Learning Recommendations API

  http://azure.microsoft.com/en-us/documentation/articles/machine-learning-recommendation-api-quick-start-guide/

Chapter 8

Retraining Azure ML models

In this chapter, we continue our exploration of the expansive capabilities of Azure Machine Learning by examining the mechanisms for incorporating "continuous learning" into the workflow for our predictive models.

To summarize our journey up to this point, we have examined the basic mechanics of implementing various predictive analytic models in Azure Machine Learning using a representative cross-section of popular data science algorithms. In all these cases, we used training datasets to teach those models in both a supervised and unsupervised capacity.

We then published our new Azure Machine Learning predictive models as web services into test or production environments. Implementing models as web services also allowed us to extend our new predictive functionality to the widest population of clients and applications, to garner crucial feedback about the accuracy and effectiveness of the model.

One of the most important stages of the Azure Machine Learning workflow process is the feedback loop. Feedback is absolutely critical to the initial development, testing, and refinement of a new predictive model. Typically this critical feedback is provided by data scientists or the human "caretakers" of the machine learning solution.

The frequency and quality of the predictive analytics feedback loop can ultimately determine the success or failure of this technology in your organization. There are many shining examples, across many industries today, where machine learning and predictive analytics have been implemented successfully to create a distinct competitive advantage for many organizations.

Alternatively, without the proper care, feeding, and consideration of feedback loops, these technologies can also become a tremendous "black hole" in terms of costs and questionable effectiveness. Figure 8-1 summarizes the Azure Machine Learning workflow steps we covered back in Chapter 2, "Getting started with Azure Machine Learning."

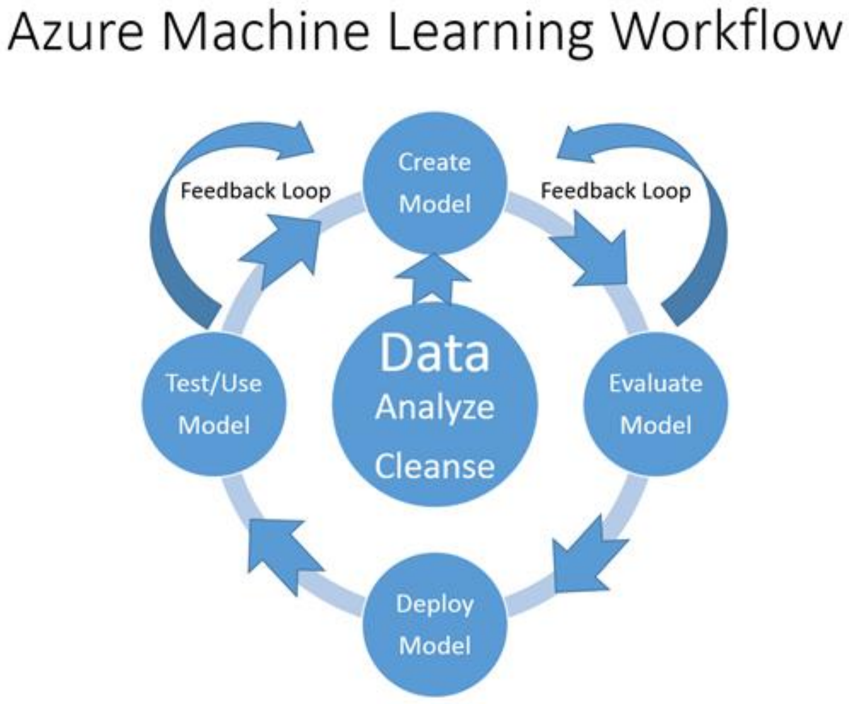


FIGURE 8-1 The steps in the Azure Machine Learning workflow process.

Workflow for retraining Azure Machine Learning models

This chapter extends our Azure Machine Learning workflow process with an additional critical dimension, incorporating automatic retraining of the predictive model. By providing a constant stream of new data to continuously retrain a predictive model, we can successfully introduce the notion of adaptability to the Azure Machine Learning paradigm.

By constantly providing new dataset inputs and automatically retraining our predictive models programmatically, we can then create even more dynamic model implementations. These new models can then adapt and withstand the test of time and evolve rapidly as conditions change in the real world. This also becomes a defining evolutionary moment in machine learning history. Programmatic retraining capabilities provide a method to supersede the standard process of initial model training to a more advanced implementation of continuous learning.

Feedback loops in the existing Azure Machine Learning workflow process are typically implemented by humans, or “carbon-based-browsers.” Alternatively, this new capability of predictive model retraining can be achieved automatically and programmatically without human intervention required.

Figure 8-2 illustrates the two approaches to training Azure Machine Learning prediction models. The upper portion describes the traditional approach we have covered so far in our journey. The lower portion of Figure 8-2 represents the workflow of the new retraining process we explore in this chapter. Note that the basic overall workflows are similar. The major difference is where humans are involved in the process.

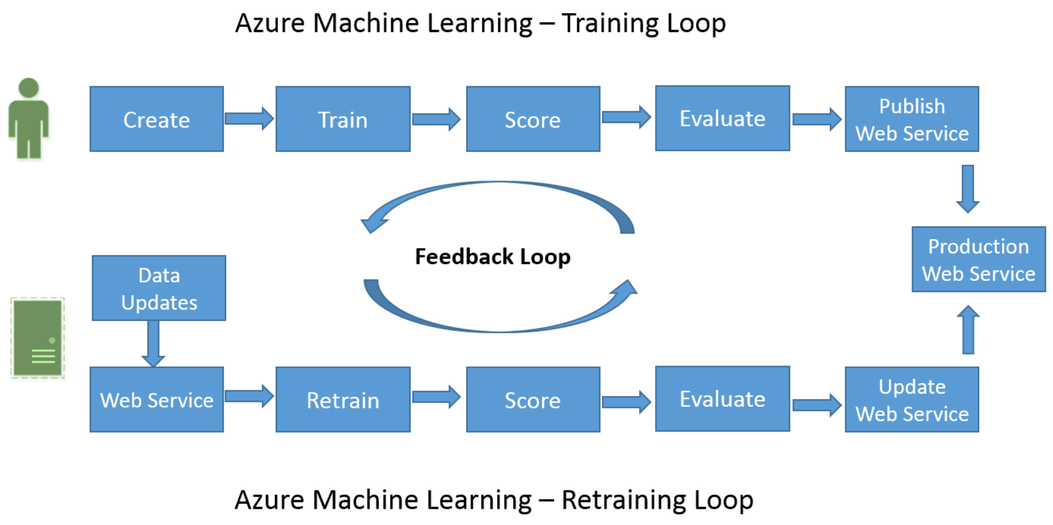


FIGURE 8-2 Azure Machine Learning process workflows for training and retraining predictive models.

Retraining models in Azure Machine Learning Studio

Azure Machine Learning allows users to publish trained machine learning models as web services enabling quick operationalization of their experiments. With the latest release of the Retraining APIs, Azure Machine Learning now enables programmatic retraining of these models with new data. This section provides a high-level overview of that process.

To demonstrate how the retraining process is implemented in Azure Machine Learning Studio, we start with extending the income prediction experiment we first covered back in Chapter 3, "Using Azure Machine Learning Studio." Recall that the income prediction experiment was an example of using a binary classification algorithm to predict whether an individual was likely to have an income level of more or less than $50,000 per year. In this example, we used a dataset from the UCI Machine Learning Repository that represented census data for a subset of the population from a 1994 Census database.

Because the relevance of that initial dataset has undoubtedly changed since 1994, the objective for our retraining example will be to extend that initial Azure Machine Learning experiment. Previously, this was only possible through the Azure Machine Learning studio user interface. With the introduction of the Programmatic Retraining API features in Azure Machine Learning, you now can retrain the model. This will update the web service to use the newly trained model. Each can be done programmatically using the Retraining APIs. In this iteration, we make use of these new APIs that were introduced into the Azure Machine Learning feature set as it went from Preview mode to Production mode in early 2015.

To get started, you can either open a copy of the previous income prediction experiment we created in Chapter 3 or you can import a similar experiment form the Azure Machine Learning Gallery. The link to the Azure Gallery sample is Sample 5: Train, Test, Evaluate for Binary Classification: Adult Dataset, at http://gallery.azureml.net/Details/3fe213e3ae6244c5ac84a73e1b451dc4.

Figure 8-3 illustrates the income prediction experiment at the time it was first developed. It also represents the training mode of the experiment before it was transformed into a web service. The experiment reflects being in training mode because the initial experiment was set up to use the train, binary classification, scoring, and evaluation modules to create a trained model. Recall that these modules are modified or removed at the time we operationalize the experiment into a web service.

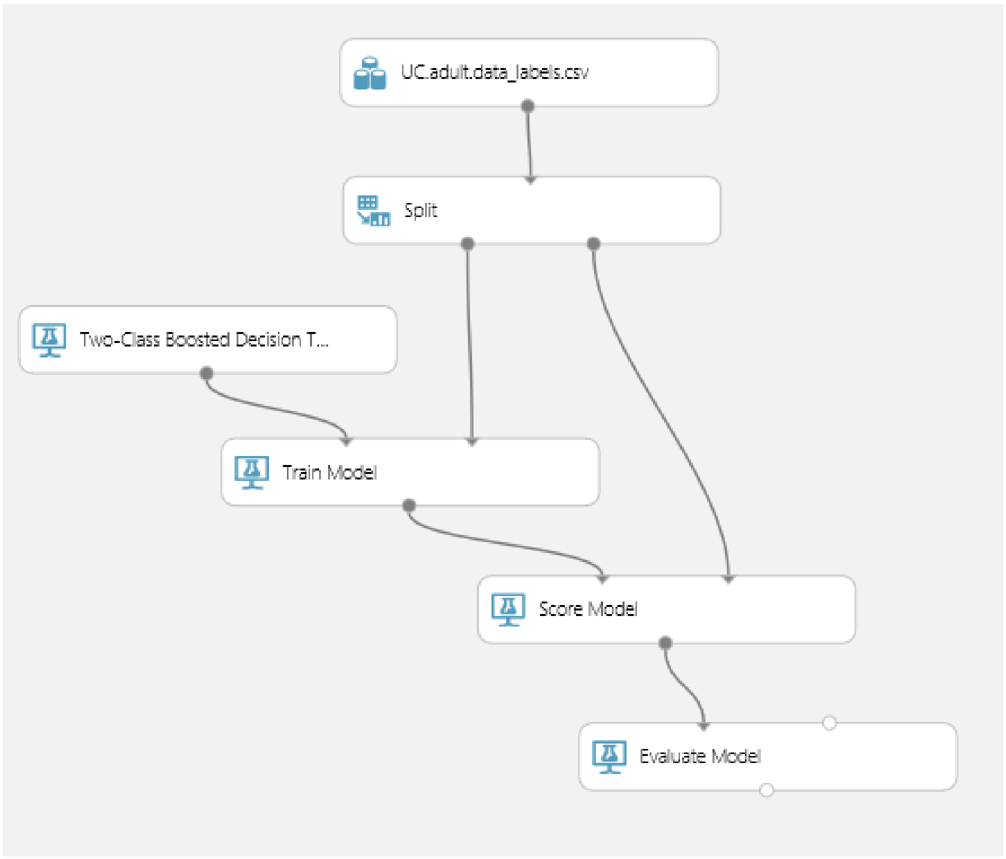


FIGURE 8-3 The Azure Machine Learning income prediction binary classification experiment.

Once you have re-created or located a copy of the income prediction experiment, the first step is to click Run on the bottom app bar in Azure Machine Learning Studio to process all the modules in the experiment.

After all the modules have been processed, the next step is to click Create Scoring Experiment on the bottom app bar as shown in Figure 8-4.

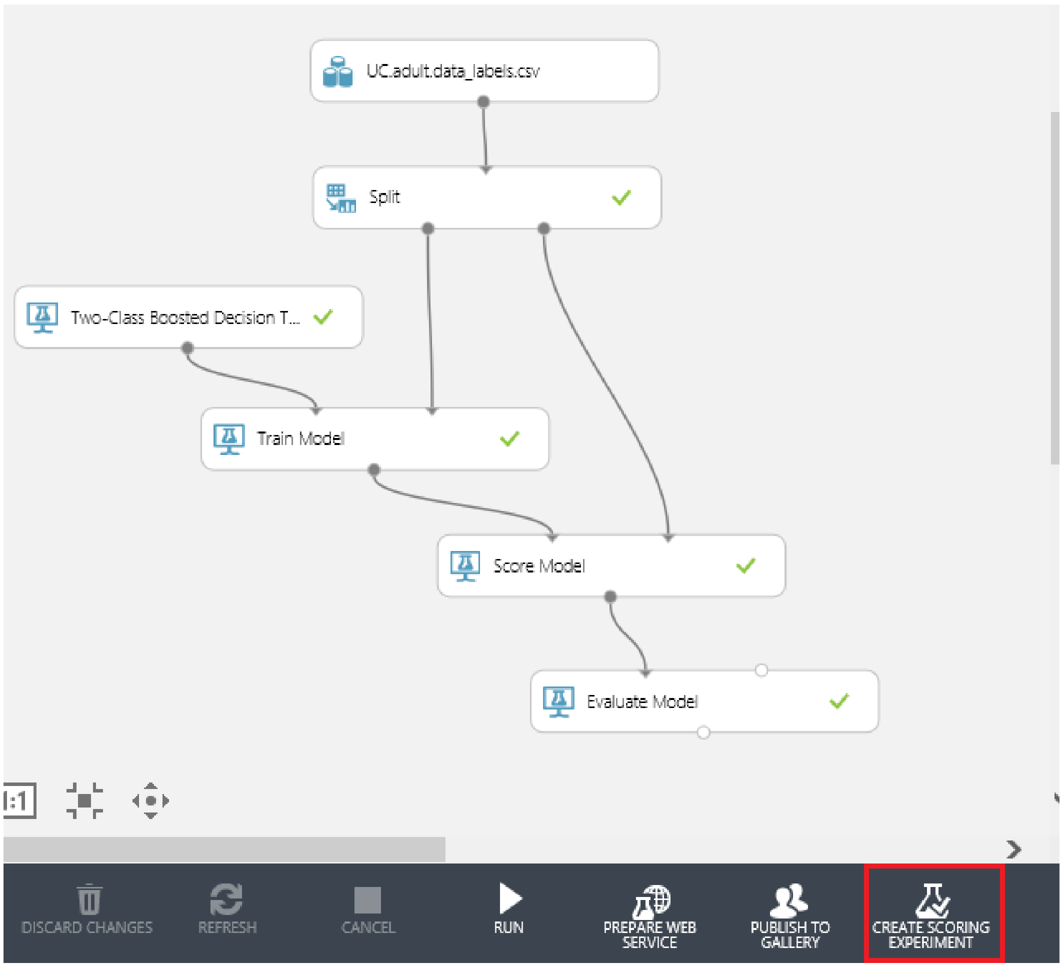


FIGURE 8-4 The Azure Machine Learning income prediction experiment, ready to execute the Create Scoring Experiment command.

Recall from the previous examples in this book that the Create Scoring Experiment command is responsible for transforming the experiment from training mode to an operational mode that is optimized for web service calls. Click Run again to process all the modules in the newly altered experiment. At this point, the experiment should look similar to Figure 8-5.

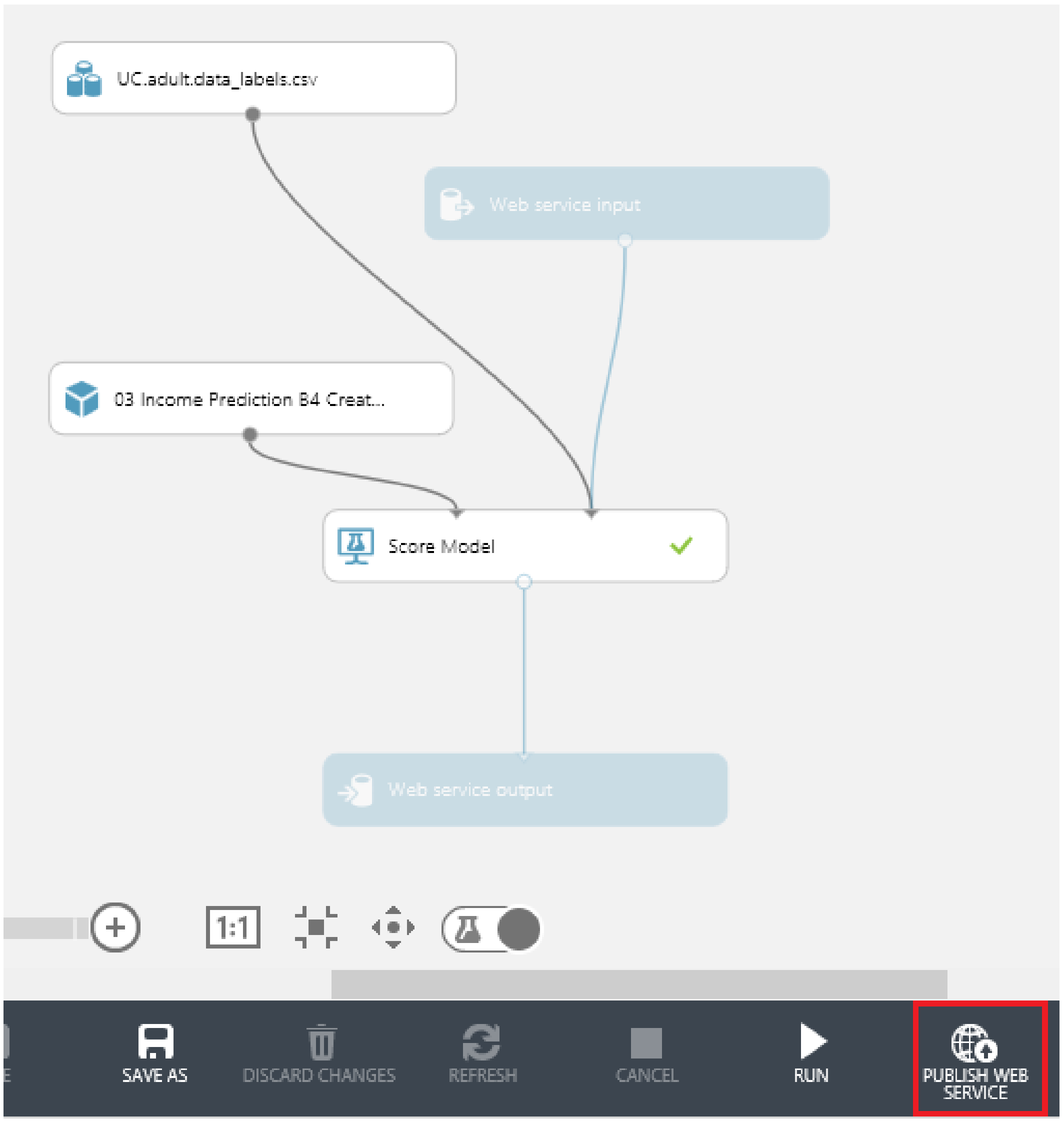


FIGURE 8-5 The income prediction experiment ready to be published as a web service.

Note that the Create Scoring Experiment command modified our income prediction experiment by adding both a web service input endpoint and a web service output endpoint to the design. The modifications required to enable our experiment for retraining will be implemented in exactly this same fashion. It will add additional web service endpoints to insert new data and then update the model based on the new datasets.

Next, click Publish Web Service to complete the process of deploying our income prediction model as a web service. After processing, the Web Service Dashboard will be displayed with the API Key and the API Help page for Request/Response and Batch Execution. Publishing the income prediction experiment as a web service is the first step in creating a model that can then be retrained.

Modify original training experiment

To incorporate retraining capabilities into our trained model, we need to modify and publish the original income prediction training experiment as a web service.

This new retraining web service will need a Web Service Output module connected to the Train Model module to be able to produce new trained models.

To transform the original training experiment, we need to add three Web Service modules to the designer. We will add one Web Service Input and two Web Service Output modules to the workflow. The Web Service Input and Output modules can be found on the bottom left module menu in Azure Machine Learning Studio as shown in Figure 8-6.

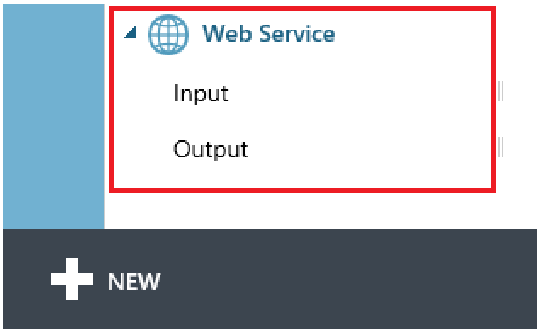


FIGURE 8-6 The Azure Machine Learning Studio Web Service modules.

Drag and drop the following three Web Service modules to the income prediction training experiment.

- **Add a Web Service Input module** Connect this to the top of the Split module. This will provide a means to insert a new dataset into the workflow.
- **Add a Web Service Output module** Connect this to the bottom of the Train module. This web service will provide us the newly trained model.
- **Add a Web Service Output module** Connect this to the bottom of the Evaluate module. This web service will return the module's Evaluate Model output. This is so the newly trained model can be evaluated to analyze the accuracy and effectiveness of the new dataset.

After adding the three additional web service modules to the original income prediction training experiment, click Run on the bottom app bar to process all the modules.

Figure 8-7 illustrates how our modified experiment looks after these changes have been made to introduce retraining capabilities.

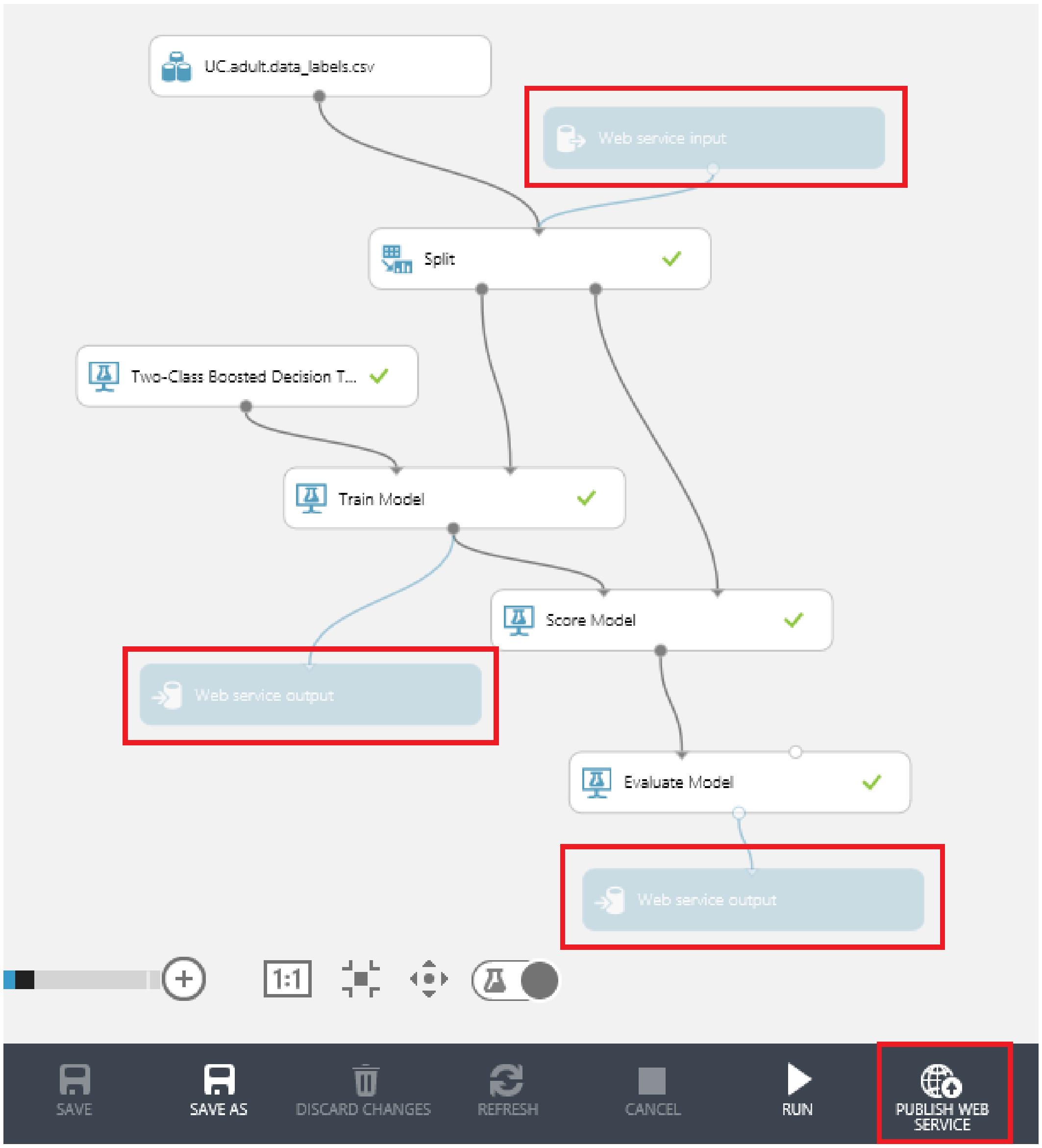
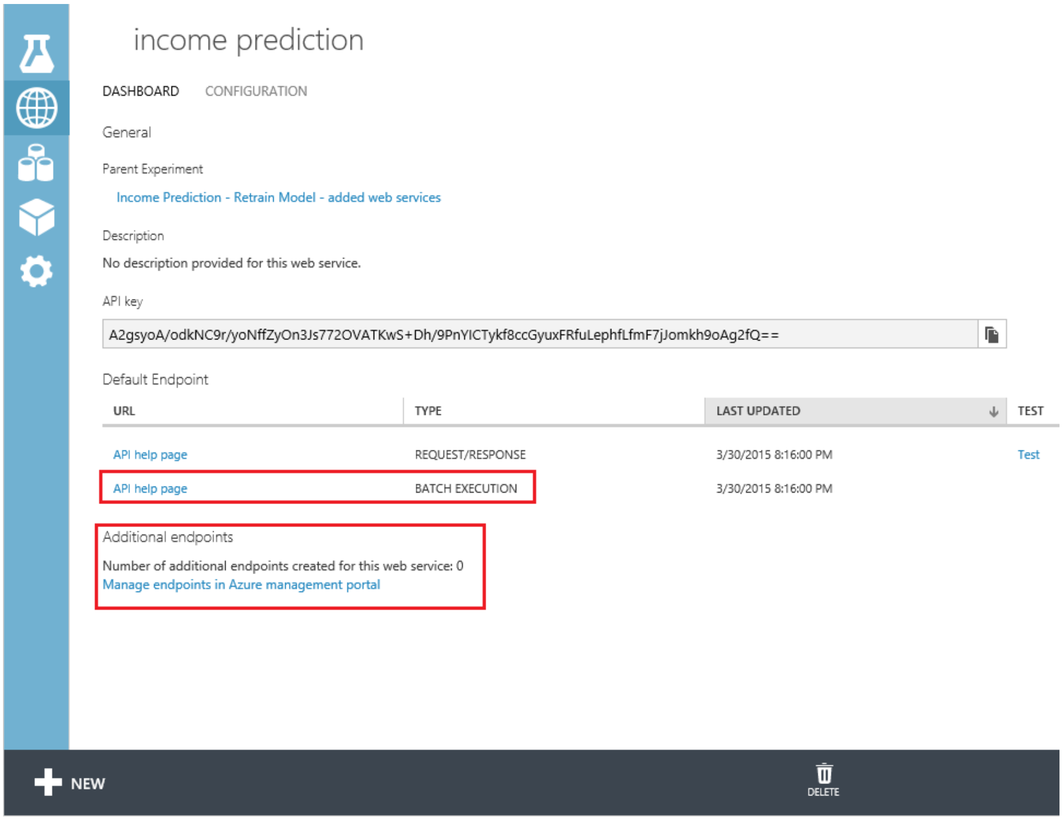


FIGURE 8-7 The income prediction Azure Machine Learning Studio training experiment after updates for adding retraining functionality.

After adding the three additional Web service modules to our original training experiment, and processing the modules via the Run command, the next step is to click Publish Web Service on the bottom app bar. This will publish the income prediction training experiment as a web service that produces trained models and model evaluation results. The Azure Machine Learning web service dashboard will be displayed with the API Key and the API help page for Batch Execution as shown in Figure 8-8.


income prediction

DASHBOARD CONFIGURATION

General

Parent Experiment

Income Prediction - Retrain Model - added web services

Description

No description provided for this web service.

API key

A2gsyoA/odkNC9r/yoNffZyOn3Js772OVATKwS+Dh/9PnYICTykf8ccGyuxFRfuLephfLfmF7jJomkh9oAg2fQ==

Default Endpoint

URL	TYPE	LAST UPDATED	TEST
API help page	REQUEST/RESPONSE	3/30/2015 8:16:00 PM	Test
API help page	BATCH EXECUTION	3/30/2015 8:16:00 PM	

Additional endpoints

Number of additional endpoints created for this web service: 0

Manage endpoints in Azure management portal

FIGURE 8-8 The Azure Machine Learning web service dashboard after publishing the income prediction training experiment.

It is important to note that the only way to retrain Azure Machine Learning models is by using the Batch Execution API method. This makes sense, as the updated training data is most likely to arrive in datasets rather than as individual request/response transactions. That said, in some cases, there is value in updating an Azure Machine Learning model with new retraining dataset rows as fast as they become available.

By having an immediate feedback loop in the form of new training data to retrain the model almost instantly, you can create a highly adaptive predictive engine that is highly responsive to changes in conditions. In a sense, we are creating the ability for Azure Machine Learning solutions to actually learn on the fly.

Add an additional web endpoint

Note that the initial web service that was created for our nontraining experiment was created with a default endpoint. The default endpoints are kept in sync with the originating experiment, and therefore a default endpoint's trained model cannot be replaced. To create an updatable endpoint, there are two options.

- **Manually** Access the Azure Portal and manually add a new endpoint.
- **Programmatically** Construct and execute an HTTP PUT message.

Azure Machine Learning Studio allows you to create multiple endpoints for a published web service. Each endpoint is individually addressed, throttled, and managed, independently of the other endpoints of that web service. Additionally, there is a unique URL and authorization key for each endpoint.

To create an additional endpoint for our income prediction service, start by navigating to the Azure Portal. Select the workspace that contains the experiment for which you want to add an additional web service endpoint. Once your Azure Machine Learning workspace has been selected, click the Web Services tab in the upper left navigation menu, as shown in Figure 8-9.

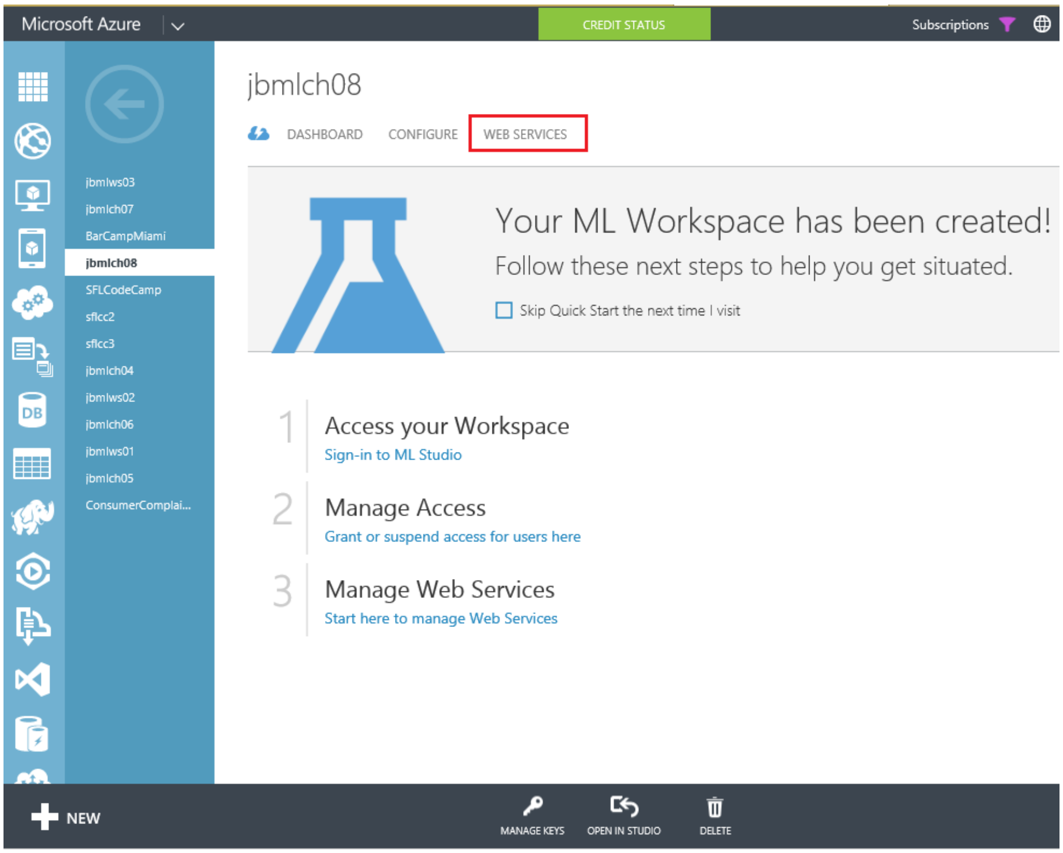
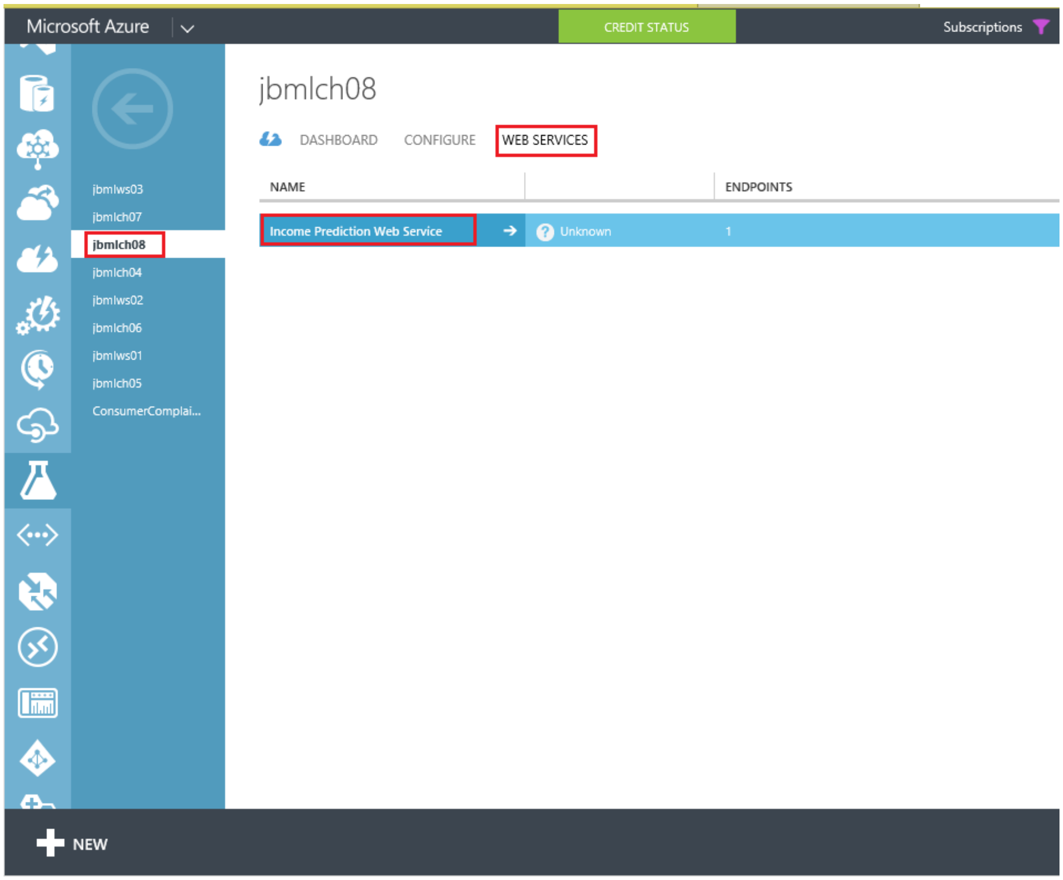


FIGURE 8-9 Azure Machine Learning Workspace menu option for managing web services.

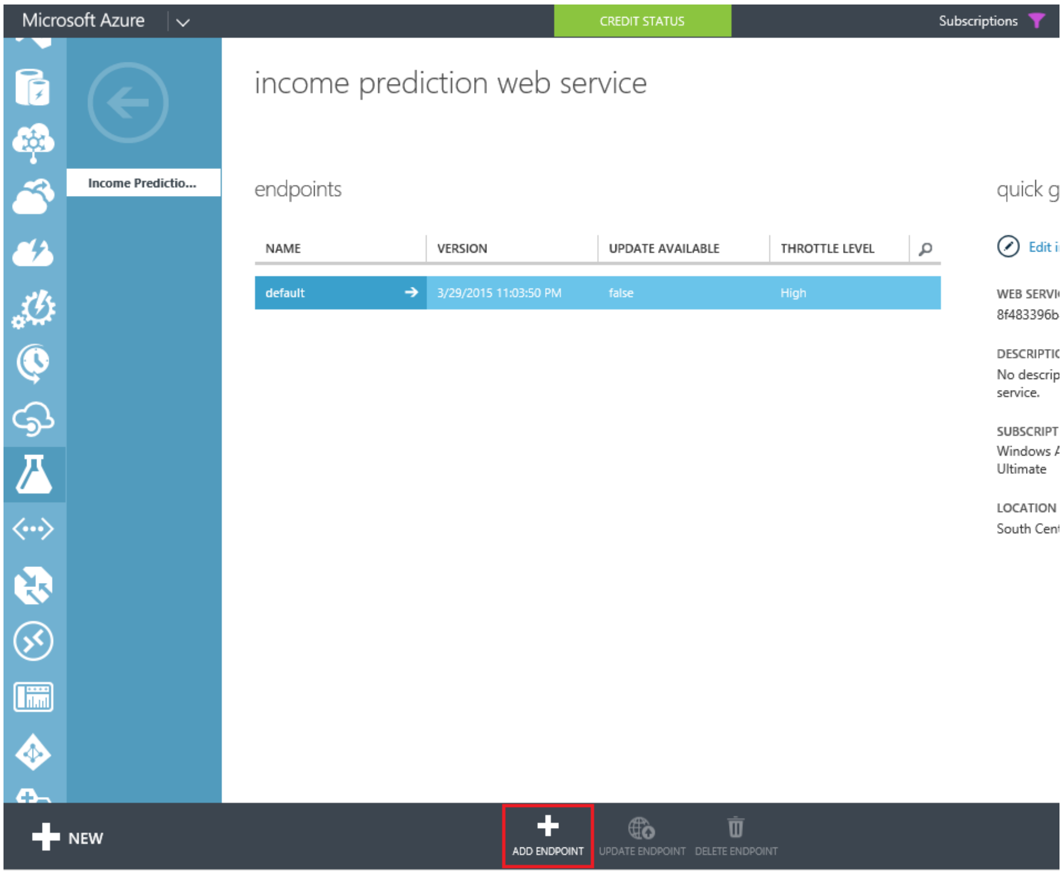
After clicking the Web Services tab, select the specific workspace experiment that contains the income prediction experiment to see the list of available endpoints. Figure 8-10 displays the web service that was created when we published our experiment in the previous steps.



NAME		ENDPOINTS
Income Prediction Web Service	Unknown	1

FIGURE 8-10 The web service available for the Azure Machine Learning workspace that contains the income prediction experiment.

After selecting the appropriate workspace that contains the income prediction experiment, click the web service that was created. This will display an additional Azure Machine Learning webpage that displays the default web service for the experiment along with any additional web service endpoints. Figure 8-11 displays a sample Azure Machine Learning web service endpoint web page.



income prediction web service

endpoints

NAME	VERSION	UPDATE AVAILABLE	THROTTLE LEVEL
default	3/29/2015 11:03:50 PM	false	High

WEB SERVI
8f483396b

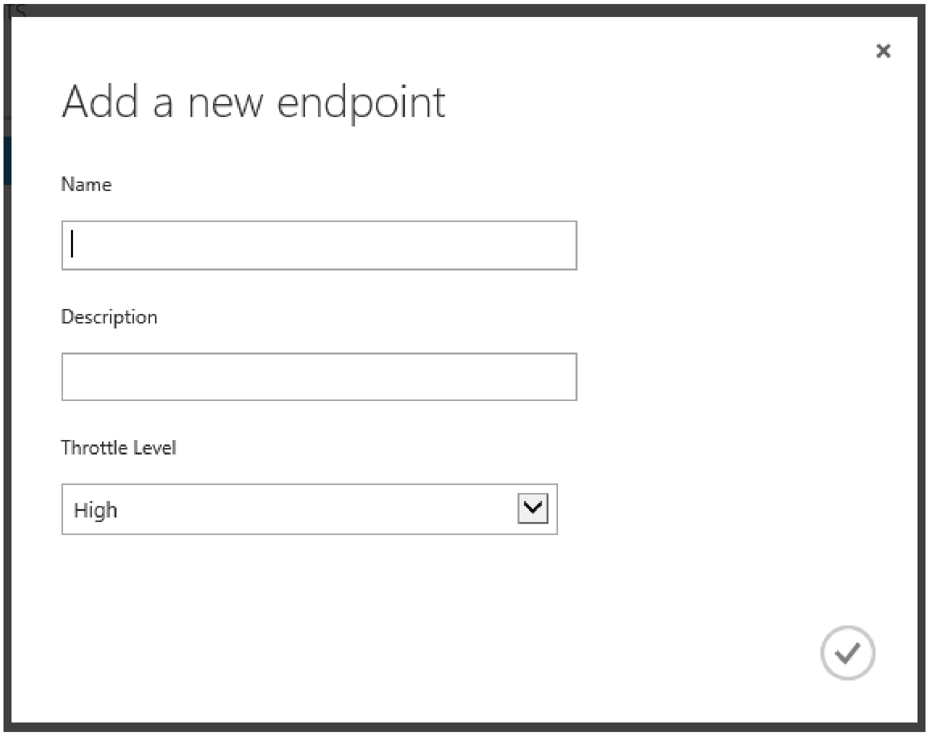
DESCRIPTIC
No descrip
service.

SUBSCRIPT
Windows A
Ultimate

LOCATION
South Cent

FIGURE 8-11 A sample Azure Machine Learning web service endpoint webpage.

Now, click Add Endpoint on the bottom app bar to invoke a pop-up window to add an additional endpoint. Enter a name for your new endpoint along with a description. Note that the endpoint name must be unique for all endpoints in this web service. Leave the throttle level with the default value unless you need to adjust this parameter. Figure 8-12 shows the input parameters for the Add A New Endpoint command.


Add a new endpoint

Name

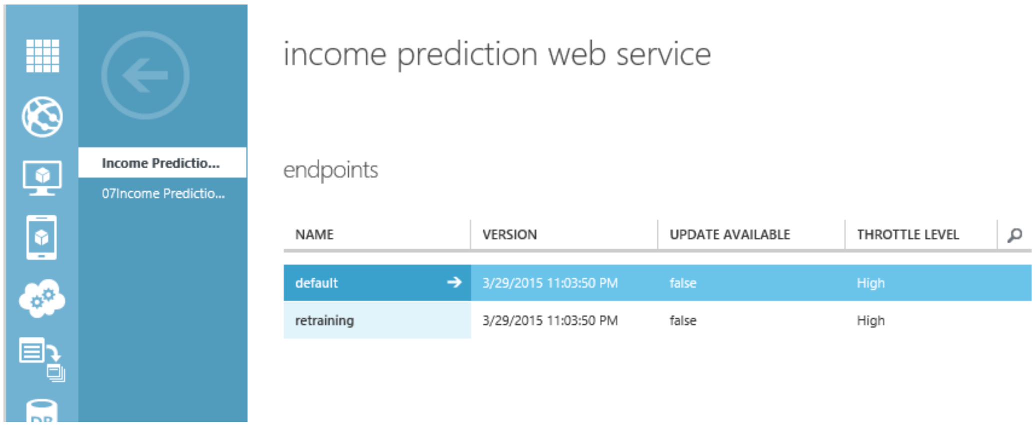
Description

Throttle Level

High

FIGURE 8-12 The input fields for adding a new web service endpoint in Azure Machine Learning Studio.

For our new retraining example, we add a new endpoint with the name retraining and description of retraining endpoint. Figure 8-13 shows our two endpoints for the original income prediction web service. The default endpoint was created when we originally published the income prediction experiment. The retraining endpoint was just added to provide a mechanism for retraining and updating the original model.



NAME	VERSION	UPDATE AVAILABLE	THROTTLE LEVEL
default	3/29/2015 11:03:50 PM	false	High
retraining	3/29/2015 11:03:50 PM	false	High

FIGURE 8-13 The two endpoints for the Azure Machine Learning income prediction web service.

Retrain the model via batch execution service

The next step is to implement a sample C# console application to provide new training data and update our model. The easiest way to accomplish this task is by using the C# sample code that was generated for the training experiment web service.

Recall that the only way to retrain Azure Machine Learning models is by using the Batch Execution API method. Navigate to the API Help page that was generated for our training version of the income prediction experiment and click the API Help page link for Batch Execution. Scroll all the way to the bottom to find the sample C# console code.

To create a C# console application for retraining an Azure Machine Learning model, you can refer to the instructions in Chapter 4, “Creating Azure Machine Learning client and server applications.” The high-level steps are as follows:

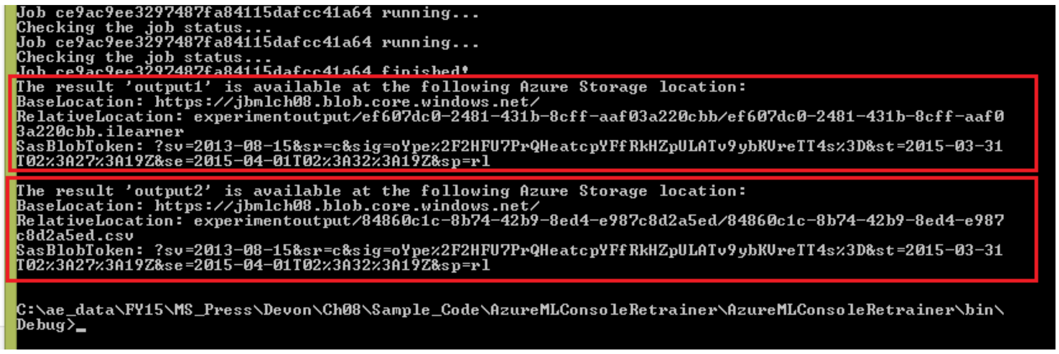
1. Start by opening Visual Studio 2013.
2. Select File/New Project/Visual C#/Windows Desktop/Console Application.
3. Install the Microsoft.AspNet.WebApi.Client NuGet packages.
4. Paste the C# sample code into the Program.cs file. The C# sample code can be found on the API Help page. The Help page can be found in the Azure Machine Learning Studio web service dashboard page. See Figure 8-8 for an illustration of the Azure Machine Learning Studio web service dashboard.
5. Add a reference to Microsoft.WindowsAzure.Storage.

Once you have the basic C# console project created, the next step is to update the variables in the

InvokeBatchExecutionService()method. Specifically, the following variables will need to be updated for your environment:

- **StorageAccountName** Create a new storage account or use an existing one.
- **StorageAccountKey** This is the storage account key from the Azure portal.
- **StorageContainerName** Create a new container in your storage account.
- **InputFileLocation** Specify the location of the new training dataset on your local file system. Remember to use double back-slashes for folder path delimiters.
- **InputBlobName** This is the name for an Azure blob to contain the output results. This name must have the same file extension as the input file in the preceding variable.
- **apiKey** This is the API key for the training experiment web service.

After updating the C# sample console application variables with your information, you can compile the application and then run the application from a command line to see the output window, as shown in Figure 8-14.

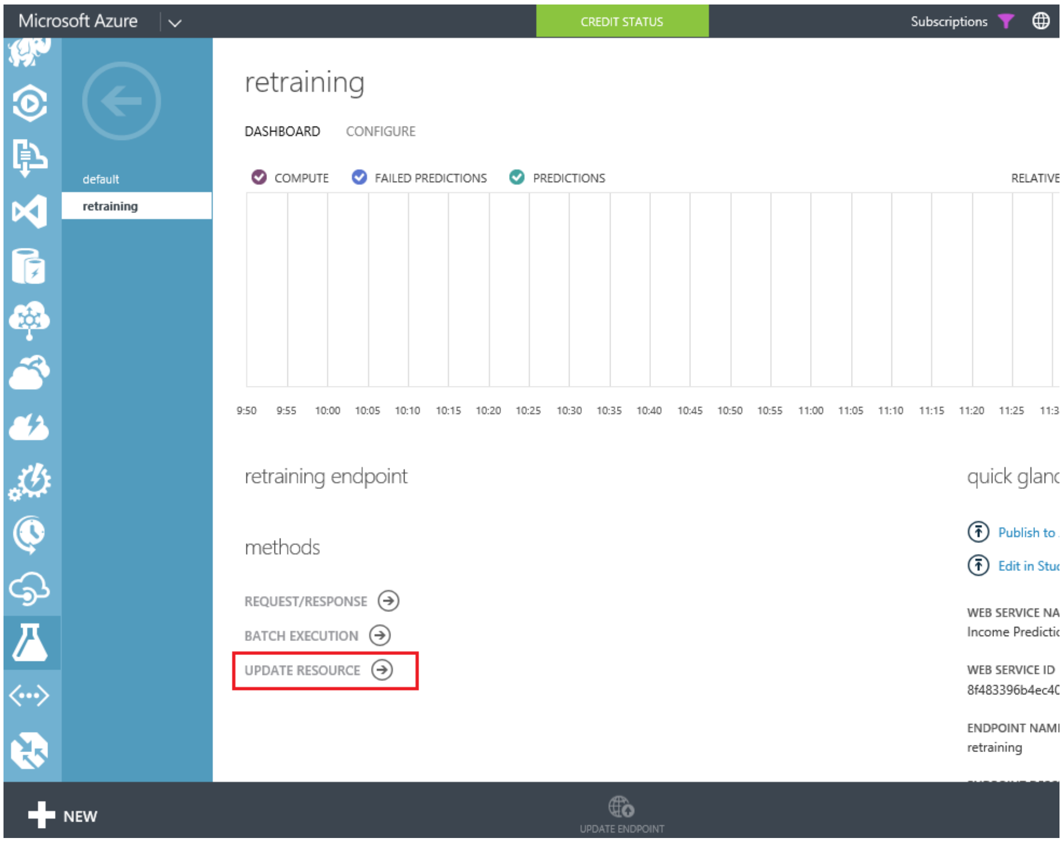


```
Job ce9ac9ee3297487fa84115dafcc41a64 running...
Checking the job status...
Job ce9ac9ee3297487fa84115dafcc41a64 running...
Checking the job status...
Job ce9ac9ee3297487fa84115dafcc41a64 finished!
The result 'output1' is available at the following Azure Storage location:
BaseLocation: https://jbmlch08.blob.core.windows.net/
RelativeLocation: experimentoutput/ef607dc0-2481-431b-8cff-aaf03a220cbb/ef607dc0-2481-431b-8cff-aaf0
3a220cbb.ilearner
SasBlobToken: ?sv=2013-08-15&sr=c&sig=oYpe%2F2HFU7PrQHeatcpYFfRkHZpULATv9ybKVreTT4s%3D&st=2015-03-31
T02%3A27%3A19Z&se=2015-04-01T02%3A32%3A19Z&sp=rl

The result 'output2' is available at the following Azure Storage location:
BaseLocation: https://jbmlch08.blob.core.windows.net/
RelativeLocation: experimentoutput/84860c1c-8b74-42b9-8ed4-e987c8d2a5ed/84860c1c-8b74-42b9-8ed4-e987
c8d2a5ed.csv
SasBlobToken: ?sv=2013-08-15&sr=c&sig=oYpe%2F2HFU7PrQHeatcpYFfRkHZpULATv9ybKVreTT4s%3D&st=2015-03-31
T02%3A27%3A19Z&se=2015-04-01T02%3A32%3A19Z&sp=rl

C:\ae_data\FY15\MS_Press\Devon\Ch08\Sample_Code\AzureMLConsoleRetrainer\AzureMLConsoleRetrainer\bin\
Debug>_
```

FIGURE 8-14 The output from the C# console app to import a new training set and retrain a model.

Notice that the results include storage location information for both of them, as seen below. "Output1" is the output of the Trained Model, and "output2" is the output of the Evaluate Model. The Batch Execution Service (BES) output shows the information for the result of retraining for "output1," which contains the retrained model location information.

To complete the process, we need to update the trained model of the income prediction web service endpoint we created previously. We now need to take this trained model and update the scoring endpoint. The key is to get the API key from the bottom of the retraining web service page and the Request URI from the Update Resource link. The Update Resource link on the retraining endpoint dashboard is shown in Figure 8-15.



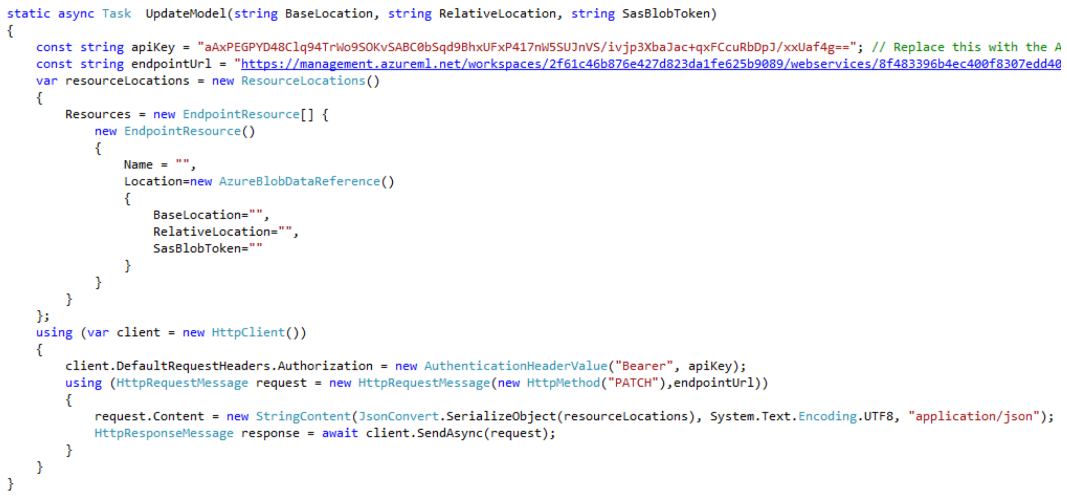
WEB SERVICE NA
Income Predictic

WEB SERVICE ID
8f483396b4ec4C

ENDPOINT NAMI
retraining

FIGURE 8-15 The Azure Machine Learning web service methods for the retraining endpoint.

The final step is to add the additional code shown in Figure 8-16 to the sample C# console application.



```
static async Task  UpdateModel(string BaseLocation, string RelativeLocation, string SasBlobToken)
{
    const string apiKey = "aAxPEGPYD48Clq94TrWo9SOKvSABC0bSqd9BhxUFxP417nW5SUJnVS/ivjp3XbaJac+qxFCcuRbDpJ/xxUaf4g=="; // Replace this with the A
    const string endpointUrl = "https://management.azureml.net/workspaces/2f61c46b876e427d823da1fe625b9089/webservices/8f483396b4ec400f8307edd40
    var resourceLocations = new ResourceLocations()
    {
        Resources = new EndpointResource[] {
            new EndpointResource()
            {
                Name = "",
                Location=new AzureBlobDataReference()
                {
                    BaseLocation="",
                    RelativeLocation="",
                    SasBlobToken=""
                }
            }
        }
    };
    using (var client = new HttpClient())
    {
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
        using (HttpRequestMessage request = new HttpRequestMessage(new HttpMethod("PATCH"),endpointUrl))
        {
            request.Content = new StringContent(JsonConvert.SerializeObject(resourceLocations), System.Text.Encoding.UTF8, "application/json");
            HttpResponseMessage response = await client.SendAsync(request);
        }
    }
}
```

FIGURE 8-16 The C# console application code to update the Azure ML model.

This method should be called using the results of the BES batch update. The parameters passed into this routine would come from parsing the results returned as "output1" from the C# console application as shown in figure 8-14. These results contain the BaseLocation, RelativeLocation, and SasBlobToken for the newly trained model. Additionally, update the apiKey and endpointUrl variables to reflect the URL and API key for the Update Resource endpoint, not the new web service used to create the new model.

This service, once called, will update the model referenced as a resource by the income prediction web service. After that time, all additional calls to that service will use this newly created model. Leveraging this update process in the life cycle of your Azure Machine Learning experiment allows you to truly deploy machine learning.

Summary

In this chapter, we explored the mechanisms in Azure Machine Learning to programmatically retrain a predictive model. This capability represents a powerful new approach to enable machines to learn on their own by providing a constant stream of new training data to adapt to changing conditions.

This new technological capability for machines to rapidly adapt through the use of constant learning also reflects the current inflection point in society that is both empowering and alarming to many individuals today. Like most technologies that have been discovered throughout history, the intent of the implementer often becomes the defining point for whether it is ultimately used as a tool or a weapon.

Resources

For more information about retraining Azure Machine Learning models, please see the following resources.

- Retrain Machine Learning models programmatically

  http://azure.microsoft.com/en-us/documentation/articles/machine-learning-retrain-models-programmatically/

- Creating endpoints

  http://azure.microsoft.com/en-us/documentation/articles/machine-learning-create-endpoint/

- Scaling API endpoints

  http://azure.microsoft.com/en-us/documentation/articles/machine-learning-scaling-endpoints/

- Microsoft Azure Machine Learning Retraining demo

  https://azuremlretrain.codeplex.com/

- Azure Machine Learning Frequently Asked Questions (FAQ)

  http://azure.microsoft.com/en-us/documentation/articles/machine-learning-faq/

About the author

Jeff A. Barnes is a Cloud Solution Architect (CSA) on the Microsoft Partner Enterprise Architecture Team, where he engages with leading cloud architects and developers to present Microsoft's cloud vision. A 17-year Microsoft veteran, Jeff brings over 30 years of deep technical experience to the CSA role. He typically works with key ISVs and global partners to demonstrate how Microsoft Azure technologies can be best leveraged to meet the current and future demands of an organization transitioning to the cloud. Jeff has deep practical experience in the retail, financial, and manufacturing industries and he is a frequent speaker at Microsoft and third-party events. Jeff resides with his family in Miami, Florida, where his definition of "offshore development" usually equates to "fishing offshore."

```
\              \
 \\            [\
  \\    \     [ \__
   -=<<((((<<>)}@>>
  //    /     {/
  /            /
```