



Time Series Forecasting with Azure Machine Learning and R



Contents

Introduction

Forecasting in Azure ML

First-time setup

Reviewing a sample experiment

Forecasting with Adventure Works

Conclusion

Forecasting 101

What is Azure Machine Learning?

This hands-on lab will introduce you to machine learning capabilities available in Microsoft Azure. Microsoft Azure Machine Learning (Azure ML) allows you to create basic designs and complete experimentation and evaluation tasks.

In this lab, we will cover Azure ML Studio, a fully managed machine learning platform that allows you to perform predictive analytics. Azure ML Studio is a user facing service that empowers both data scientists and domain specialists to build end-to-end solutions and significantly reduce the complexity of building predictive models. It provides an interactive and easy to use web-based interface with drag-and-drop authoring and a catalogue of modules that provide functionality for an end-to-end workflow.

In some scenarios, a data scientist may not understand the domain data as thoroughly as a domain expert does. Microsoft Azure Machine Learning has made it possible for the people who know the most about the data to produce predictive analytics for that data.

Forecasting in Azure ML

During this lab, we will explore how to use R in Azure Machine Learning as well as exploring the industry standard principles of forecasting. The lab will be based around a sample experiment that is readily available as an Azure ML template in [the Cortana Intelligence Gallery](#). The sample experiment already has all the code and sample data we need to work on this lab.

Using R (or Python) to extend Azure ML as well as time series forecasting are considered advanced topics in Azure ML. There are options created by [the AzureML Team](#) to get you started with a working example that you can replace the example data with your own and/or further customize the model. The noteworthy options are:

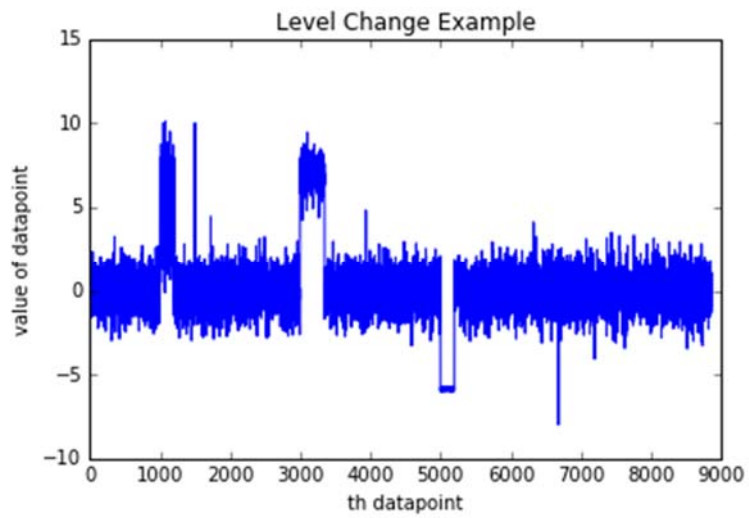
1. [Retail Forecasting](#). This is a 6-step full solution that separate each process in machine learning into its own ML experiment.
2. [Stephen Elston, PhD's Quick Start Guide to R in Azure Machine Learning](#). The California dairy production and price data is used in the time series prediction. There are two companion videos to explain how to use R in Azure ML.
3. [Neeraj Khanchandani's Time Series Forecasting using Custom Modules](#). The ML experiment contains pre-built Azure ML custom modules¹ that originally came from R Time Series code used in this lab.

In addition to using R (or Python) to extend Azure ML for forecasting, Azure ML provides a built-in Time Series model that is specialized in detecting anomaly in the input time series data. It supports two types of anomalous patterns in time series:

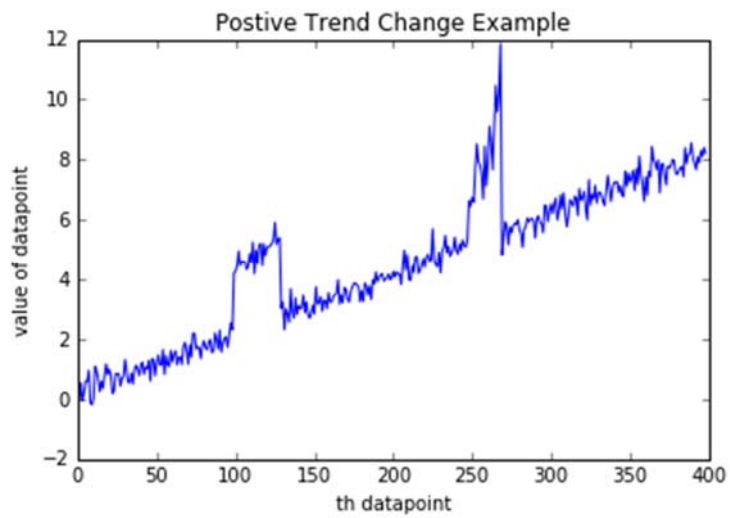
1. Upward and downward level changes

¹ By opening a solution with the same custom modules in the same Azure ML workspace multiple times, you'll have multiple copies of the custom modules that currently Azure ML doesn't provide a mechanism to remove them.

¹ Follow [this guide](#) to learn how to author custom R modules in Azure ML.



2. Positive or negative trends changes



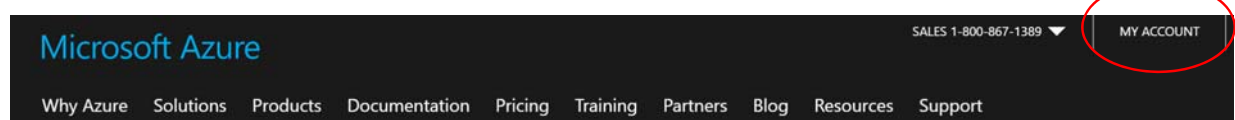
For details, please visit [the Time Series Anomaly Detection documentation](#).

First Time Setup

Microsoft Azure Machine Learning Studio (<https://studio.azureml.net>) is a user-friendly machine learning authoring and collaboration tool as part of many Azure Intelligent Cloud services offering. Azure ML Studio provides many world class ML algorithms created by Microsoft Research and Azure ML teams. There's no need to install any software. All you need to access ML Studio are your favorite web browser and internet connection, and so we proudly dub this tool as **Machine Learning as a Service**. The flavors of machine learning provided by Azure ML are supervised (binary and multi-class classification, regression, and anomaly detection), unsupervised, reinforcement learning² as detailed in the introduction section before this lab.

The main access for Azure services is Azure Management Portal (<https://portal.azure.com>), where you will need an Azure subscription to access the portal and to complete the lab. **We strongly recommend you obtain an approved Azure subscription from your IT department.** However, you can sign up for a free one month Azure trial³; using your Microsoft account (outlook.com, hotmail.com, live.com)⁴, or you may use your organization Microsoft Developer Network (MSDN) to access Azure.

Once you have an Azure subscription you can go to <http://www.azure.microsoft.com> and choose 'My Account' at the top of the page

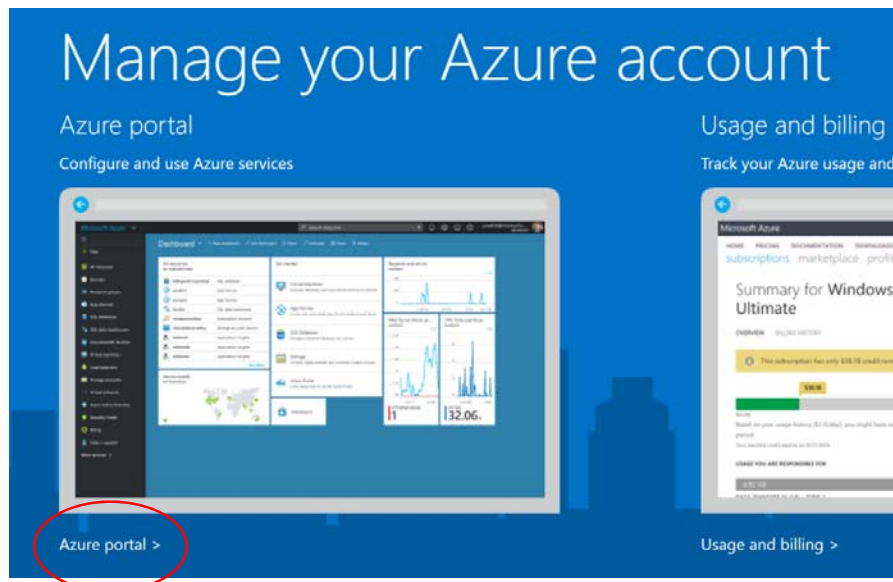


then choose the 'Management Portal' where it will ask you to sign in with your Microsoft account.

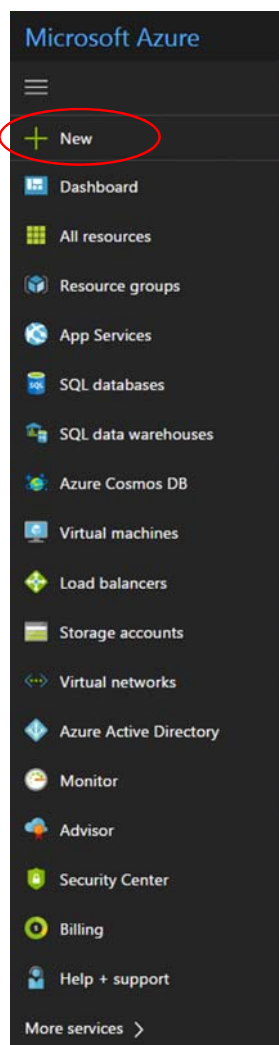
² More information about the algorithms available is at <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice>

³ Sign up to Azure for one month free trial here: <http://azure.microsoft.com/en-gb/pricing/free-trial/>. This option may require you to enter credit card information for the purpose to verify your identity only. Azure will not charge the credit card without your permission.

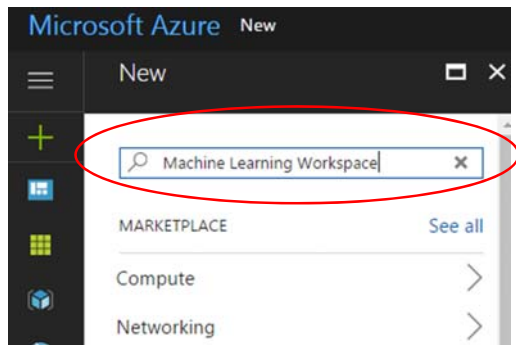
⁴ If you don't have a Microsoft account, create one here: <https://signup.live.com/signup.aspx>.



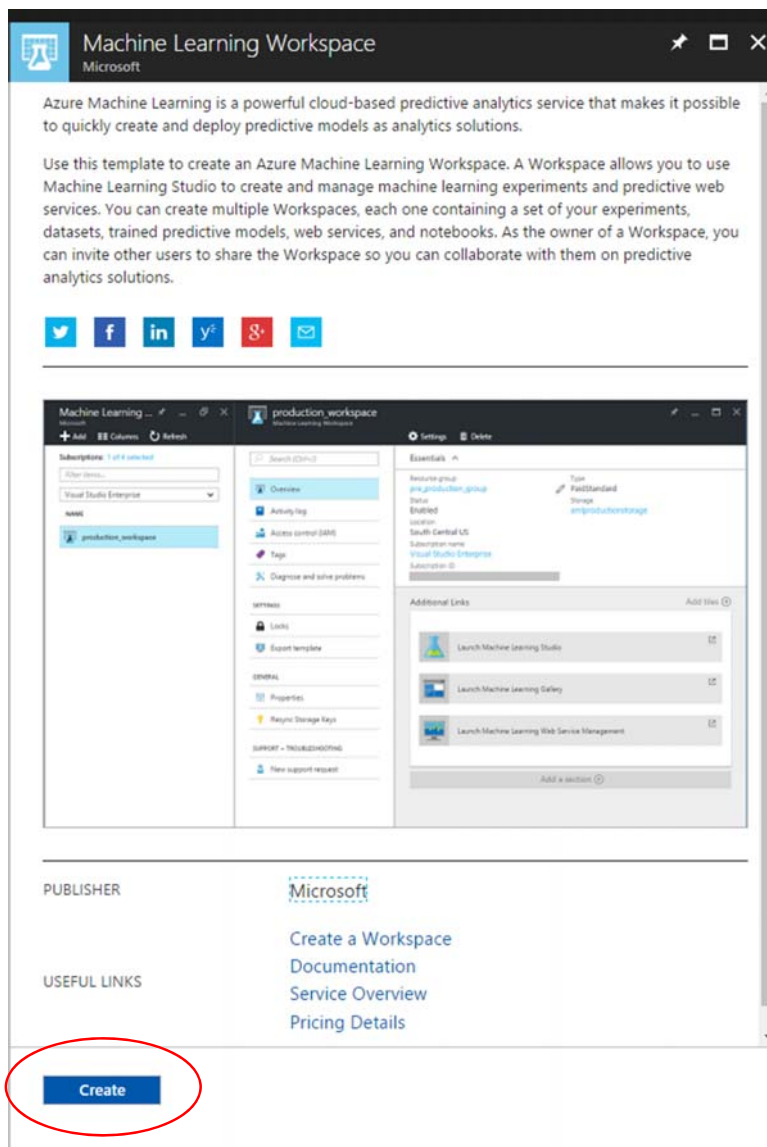
Once into Azure Portal, click + New to create a new Azure service, in this case, a new Machine Learning Workspace.



Type in the search box "Machine Learning Workspace", and a Machine Learning Workspace blade will open.



Click the Create button in the bottom of the blade.



Enter details for all the variables below:

- **Workspace Name:** Pick a name for your workspace.
- **Subscription:** Choose the Azure subscription.
- **Resource group:** Create new and enter a unique name (or pick the name of an existing resource group). A resource group is a grouping of Azure services for management purposes.
- **Location:** South Central (or a data center near you).
- **Storage account:** Create new storage account (or pick the name of a current storage account). A storage account is the space to store the Azure ML experiments, models, data, and other artifacts.

Choose one of the Web service plan pricing tier⁵ (Steps 1 and 2 in the figure). Check the Pin to dashboard checkbox (Step 3). Finally, click the Create button to create the workspace (Step 4).

The screenshot displays the 'Machine Learning workspace' configuration window. The left sidebar contains the following fields:

- Workspace name:** TimeSeriesWorkspace
- Subscription:** [Redacted]
- Resource group:** Create new (selected), TimeSeriesLab
- Location:** South Central US
- Storage account:** Create new (selected), timeseriesstore
- Workspace pricing tier:** Standard
- Web service plan:** Create new (selected), TimeSeriesWebService
- Web service plan pricing tier:** S1 Standard (highlighted with a red circle and '1')
- Pin to dashboard:** ☒ (highlighted with a red circle and '3')
- Create:** Button (highlighted with a red circle and '4')

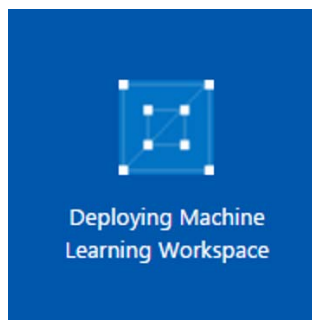
The right pane, titled 'Choose your pricing tier', shows the following options:

S1 Standard	S2 Standard	S3 Standard
25 Compute Hours	500 Compute Hours	12,500 Compute Hours
100,000 Transactions	2,000,000 Transactions	50,000,000 Transactions
Manual Scaling	Manual Scaling	Manual Scaling
3.23 USD/DAY (ESTIMATED)	32.26 USD/DAY (ESTIMATED)	322.58 USD/DAY (ESTIMATED)

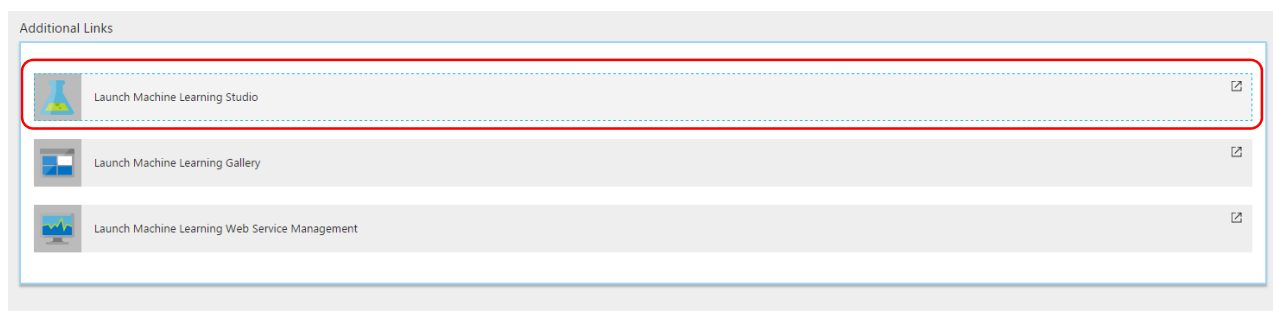
Below these is the 'DEVTEST Standard' option, which is 'Unavailable' and costs '0.00 USD/DAY (ESTIMATED)'. A red circle and '2' highlight the 'Select' button at the bottom of the pricing tier pane.

⁵ Azure allows one free DEVTEST Standard Web service plan per subscription. Learn more about different tiers and capabilities at <https://azure.microsoft.com/en-us/pricing/details/machine-learning/>.

While Azure is creating the machine learning workspace, you'll see Deploying Machine Learning Workspace on the dashboard (the front page of Azure Portal).



Once done, the portal will refresh to show you the summary⁶. Click Launch Machine Learning Studio.



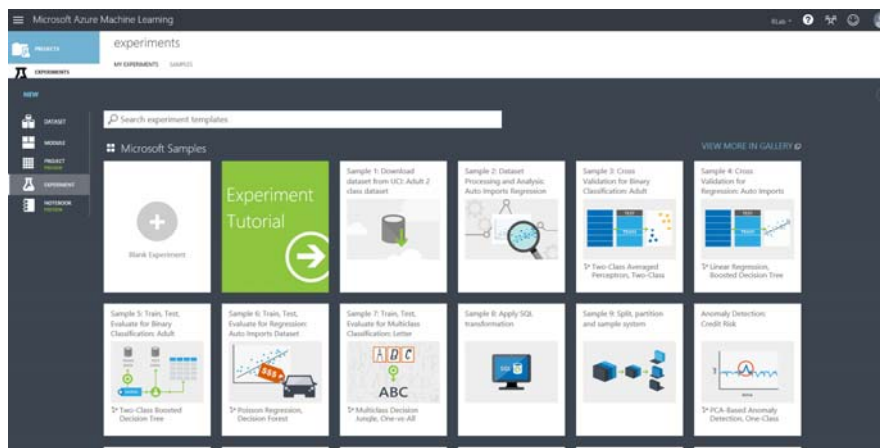
We are now ready to get into the world of Azure ML, R, and forecasting.

⁶ You'll also find an email from Microsoft Azure Machine Learning to confirm that you've successfully created an ML workspace.

Reviewing a sample experiment

We are now ready to begin to use R to extend Azure ML and have a brief look at forecasting. Forecasting, that is trying to predict patterns over time, is not currently one of the built-in modules in Azure ML; although, it is possible to use regression⁷. Fortunately, Azure ML by design is extensible via R and Python; thereby, users can write and implement their own algorithms. In addition, there are several examples of how to incorporate R and Python with Azure ML provided in the [Cortana Intelligence Gallery](#). In this lab, we'll be using R.

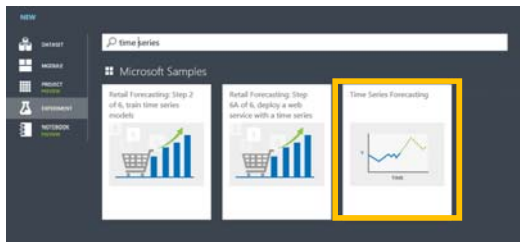
Once our Workspace has been created we can click on the link 'Sign-in to ML Studio' and we'll be taken to ML Studio⁸ and given the option to create a new experiment which can be blank or from one of the samples.



In the search box enter "time series".

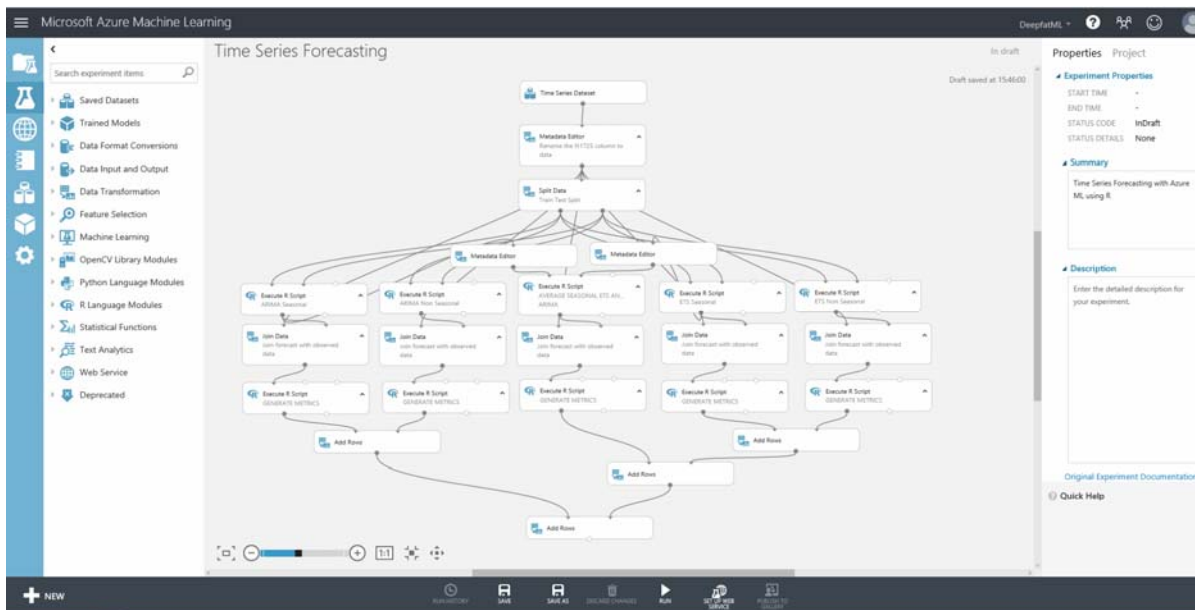
⁷ Regression is a technique in machine learning to fit data points to a line, e.g. linear regression fits points to a straight line.

⁸ A pop-up window with "Would you like a tour of Azure ML?" will be presented. We recommend users who are new to Azure ML to follow the short 5-step interactive tour. You will experience a binary classification to solve an income prediction problem, and it will familiarize users with the kind of flows from data to preprocessing, training, scoring, and evaluation of an ML model.




Select the [Time Series Forecasting](#) and then select view in gallery. This will bring us to the main web page of the experiment template, and **it is essential that you spend time with the contents of this page**. The dataset used in this ML experiment template is called N1725 that was part of an [M3-Competition](#) organized by International Institute of Forecasters (IIF).

When you have absorbed the contents, click on Open in Studio which allows us to save a local copy that we can edit and use, complete with the data. We'll be prompted to save it to our current ML workspace (which can be changed if we have more than one). Finally, the full experiment will open in ML Studio.



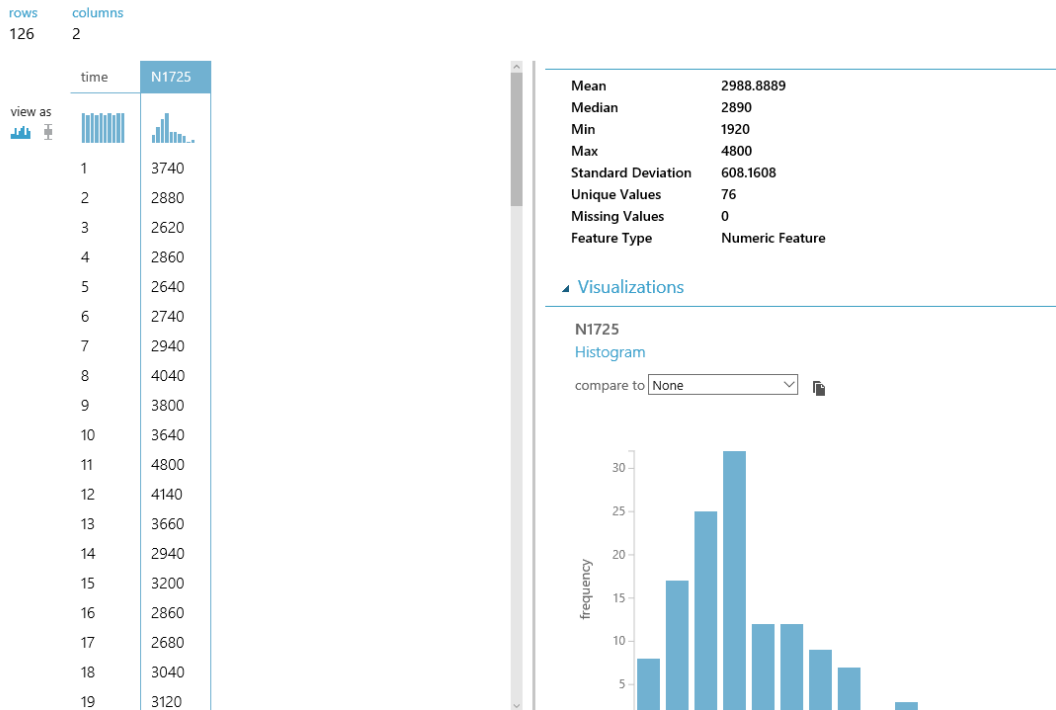
Here we can see how the experiment is wired up, where the grey lines show data flows through the various modules. New modules can be selected from the list on the left and then we can set the properties of the module we have focus on the right. Notice the experiment is branches into five parallel streams which are then recombined at the end. As described in the notes in the gallery each of these is a different type of forecasting, and there is a high-level guide introducing how these work at the end of this guide.

In order to investigate what is going on, all we need to do is run the experiment as is by clicking on Run  in the dark grey toolbar at the bottom. We'll see each module ticked in green as the run progresses. The runtime is about 3 minutes, and it's running in the Microsoft cloud, so you

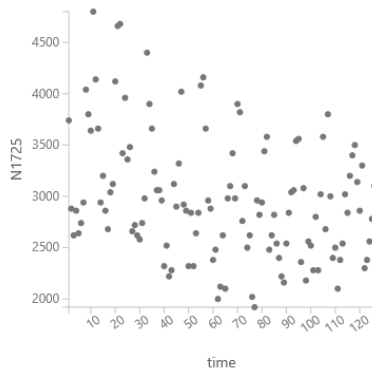
could turn off your laptop without harming the experiment in anyway. And when it's all finished, we can examine the results.

First let's look at the source data. Right click on the top module Time Series Dataset and select visualize. Click on the N1725 column to get some basic stats about that column.

Time Series Forecasting > Time Series Dataset > dataset



If we select time and then compare this to N1725 we get a scatterplot of the relationship between the variables.

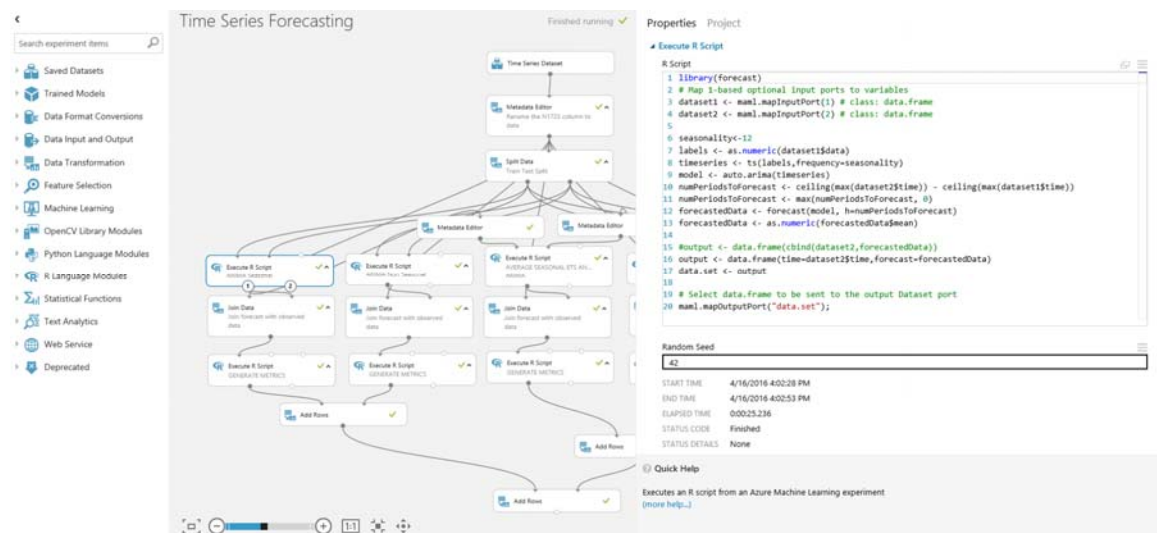


But we can and will do better with R, and that work has been done for us in the sample as we'll see later.

The next important module is the one for split, and this module outputs older data on the left port and the newer data on the right port by using a simple regular expression \"time\" <=108. This has been done so that we can train against the older data and then compare the forecast

against the actual and newer data as a test. The split module is essential in ML so that one part of the data is used to train the model(s), and another part is withheld to be used exclusively to score (aka. test) and evaluate the performance of the model(s).

If we look at each of the five parallel streams, we can see an R module works on a type of forecasting followed by a join followed by another R script to compute metrics to determine the accuracy of the forecast. If looking at the ARIMA seasonal module (on the left), we see that an R module has three input ports. Two of these are for data and the third allows us to pass in a zip file containing any custom R modules that are not included in Azure ML by default⁹. The R module also has two output ports. One for the flow of data and the other can be used for outputting other information from R, e.g. R messages and R plots. If we set focus on this module and rearrange the properties pane we can see the R code in this module.

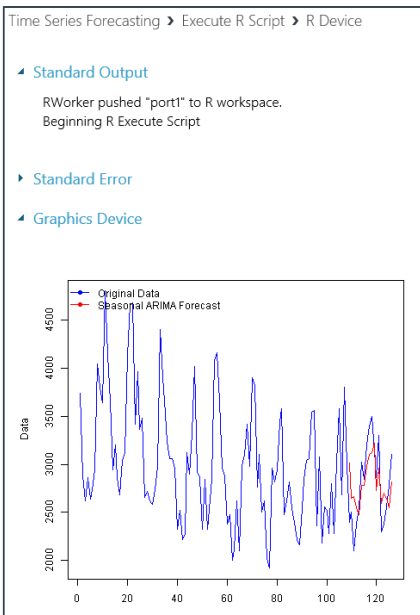


Notes

1. There are already a lot of standard R libraries pre-loaded in Azure ML, and so all we need to do is declare which we want to use. In this case, it's `library(forecast)`.
2. We can see how the input/output ports are exposed as R data frames – `maml.mapInputPort(1)`, `maml.mapInputPort(2)`, `maml.mapOutputPort("data.set")`.
3. The output from this R module is the forecast time and the forecast itself, but not the actual values from data set 2. Therefore, there is a join after this module. The join step could also be done in R if so choose.
4. The built-in R editor provides only basic functionality mostly to view the code. A recommendation is that users may use other R editors (IDEs) such as Visual Studio with R Tools enabled or R Studio to author and test R code before pasting the code in Azure ML.

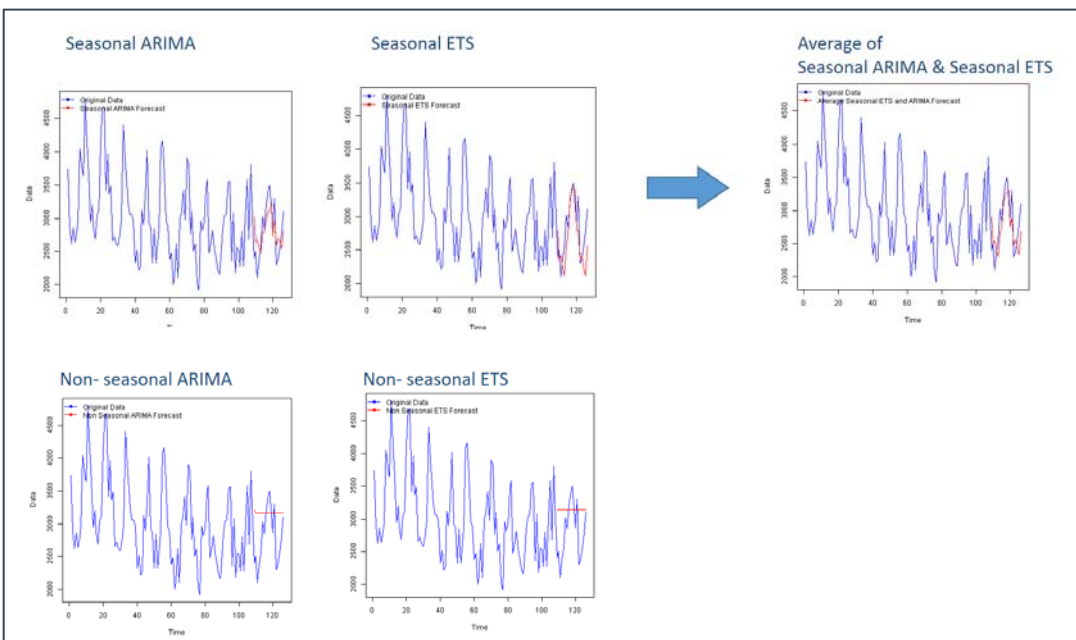
⁹ To find out what R modules already included in Azure ML, follow the guide at <https://msdn.microsoft.com/en-us/library/mt741980.aspx>.

If we now turn our attention to the Generate Metrics module underneath the ARIMA seasonal module, we can see that this much longer script not only computes metrics, but it also produces a time series plot. To visualize the plot (once we have run the module), we right click on the right output port of this module and choose Visualize.




The blue line is the actual data for the whole data set. The forecast line is in red superimposed for the periods greater than 108 (in other words the test set).

If we look at the outputs for the other modules, we can see that seasonality has a big bearing on the results.



The best fit is the Average of Seasonal ARIMA & Seasonal ETS, and it can become more obvious if we look at the output from the very final module (Add Rows), where each type algorithms is analyzed using a number of standard metrics in one grid.

Method	ME	RMSE	MAE	MPE	MAPE	MASE	sMAPE
							
seasonal arima	-1.676751	295.036934	252.157364	-1.460276	9.426425	0.833983	4.57093
non seasonal arima	-372.174512	553.416508	454.515075	-15.792259	18.203335	1.50326	7.990454
average seasonal arima & ets	65.010533	277.479422	230.363678	1.40434	8.445977	0.761903	4.231599
seasonal ets	131.697818	322.54912	255.748238	4.268956	9.323395	0.84586	4.872235
non seasonal ets	-344.702749	533.635548	438.509864	-14.776237	17.530166	1.450324	7.741309

The metrics represent different kinds of errors; therefore, the closer the number is to zero the better. To find out more about what these analyses mean click on the links.

- [Mean Error \(ME\)](#) - Average forecasting error (an *error* is the difference between the predicted value and the actual value) on the test dataset.
- [Root Mean Squared Error \(RMSE\)](#) - The square root of the average of squared errors of predictions made on the test dataset.
- [Mean Absolute Error \(MAE\)](#) - The average of absolute errors.
- [Mean Percentage Error \(MPE\)](#) - The average of percentage errors.
- [Mean Absolute Percentage Error \(MAPE\)](#) - The average of absolute percentage errors.
- [Mean Absolute Scaled Error \(MASE\)](#)
- [Symmetric Mean Absolute Percentage Error \(sMAPE\)](#)

Now that we have a basic understanding of how this experiment works, what if we wanted to adapt this to our own data? That's what we'll do in the next section.

Forecasting from Adventure Works

In this section, we will customize the previous experiment to use a new sample dataset from the standard Adventure Works Data Warehouse (AWDW) database included in SQL Server. This is to demonstrate a few use cases including how to connect live to a remote data source and what to look out for to modify if when you reuse an experiment template provided by Cortana Intelligence Gallery. Please be aware that this current exercise is not intended to perfect the time series analysis techniques used, so we'll leave it up to the reader to further improve the performance of the analysis.

First, we need to connect to the source database and to save time we already have an Azure SQL DW¹⁰ with the AWDW sample already setup. To connect to a data source, we need to use the Import Data module in Azure ML:

- In the search box above all the modules in ML Studio type import to locate the Import Data module.
- Drag the Import Data module onto the studio canvas and enter the following information to connect to the sample AWDW on Azure SQL Data Warehouse.

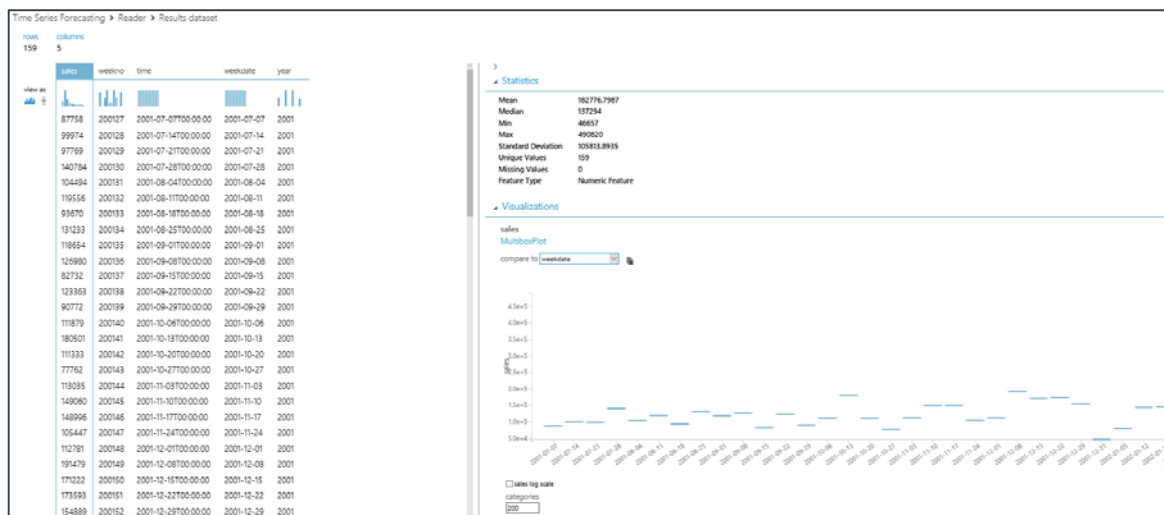
Data Source	Azure SQL Database
Database Server Name	[dwname].database.windows.net
Database Name	AdventureWorksDW
Server user account name	[username]
Server user account password	[password]

¹⁰ You can create one of these datasets for yourself in your Azure subscription. Just look for Azure SQL data warehouse and when you create it opt to include the sample database. However, be aware that Azure SQL DW is an enterprise scale solution, and so could be expensive to leave running. If you do decide to do this, pause it to stop the compute until you need it again. The ability to pause and start when needed demonstrates the economics of Cloud Computing.

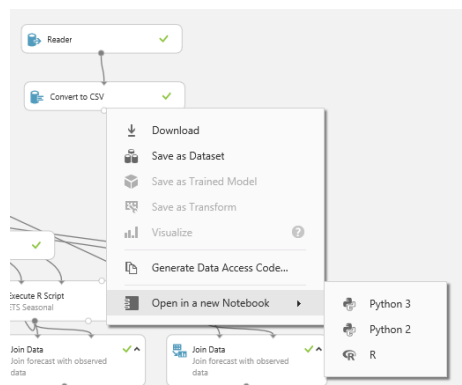
For the database query copy in the following:

```
SELECT
    CONVERT(Int,SUM(fis.SalesAmount)) AS sales,
    (dd.[CalendarYear]*100) + dd.[WeekNumberOfYear] AS weekno,
    MAX(dd.FullDateAlternateKey) AS [time],
    CONVERT(CHAR(10),MAX(dd.FullDateAlternateKey)) AS weekdate,
    dd.CalendarYear AS year
FROM
    [dbo].[FactInternetSales] fis
INNER JOIN
    [dbo].[DimDate] dd
ON
    dd.[DateKey] = fis.[OrderDateKey]
WHERE
    (dd.[CalendarYear]*100) + dd.[WeekNumberOfYear] < 200427 -- Later sales aren't useable for this.
GROUP BY
    (dd.[CalendarYear]*100) + dd.[WeekNumberOfYear],
    dd.CalendarYear
ORDER BY weekno
```

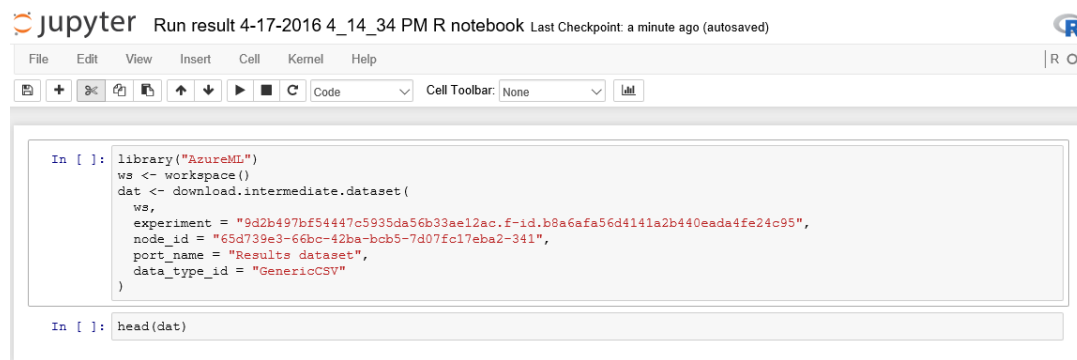
Don't connect the Import Data module to anything yet, but do run the experiment again to check the query is working OK and visualize the output.



Here the sales have been plotted against weekdate with 200 categories selected. However, what if we wanted to have a quick look at this data in R? All we need to do is drag on a convert to CSV module and connect the Import Data module to it and run the experiment again. We can then right click on the CSV Module and open the output in an R Jupyter Notebook.



And it looks like this.

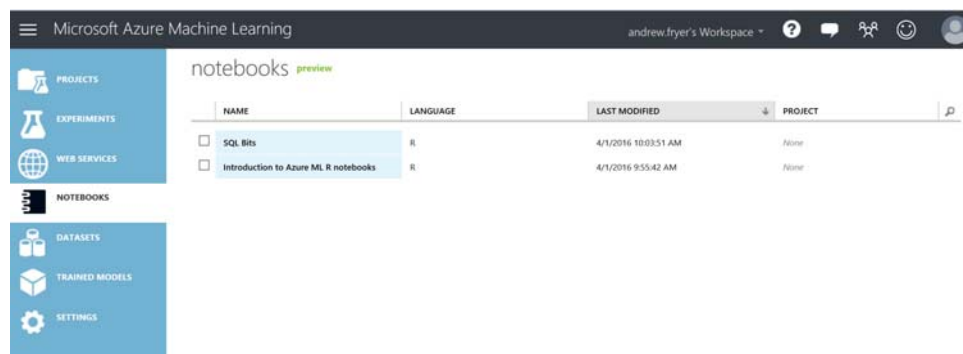


The screenshot shows a Jupyter Notebook window titled "Run result 4-17-2016 4_14_34 PM R notebook". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations and execution. The code editor contains the following R code:

```
In [ ]: library("AzureML")
ws <- workspace()
dat <- download.intermediate.dataset(
  ws,
  experiment = "9d2b497bf54447c5935da56b33ae12ac.f-id.b8a6afa56d4141a2b440eada4fe24c95",
  node_id = "65d739e3-66bc-42ba-bcb5-7d07fc17eba2-341",
  port_name = "Results dataset",
  data_type_id = "GenericCSV"
)

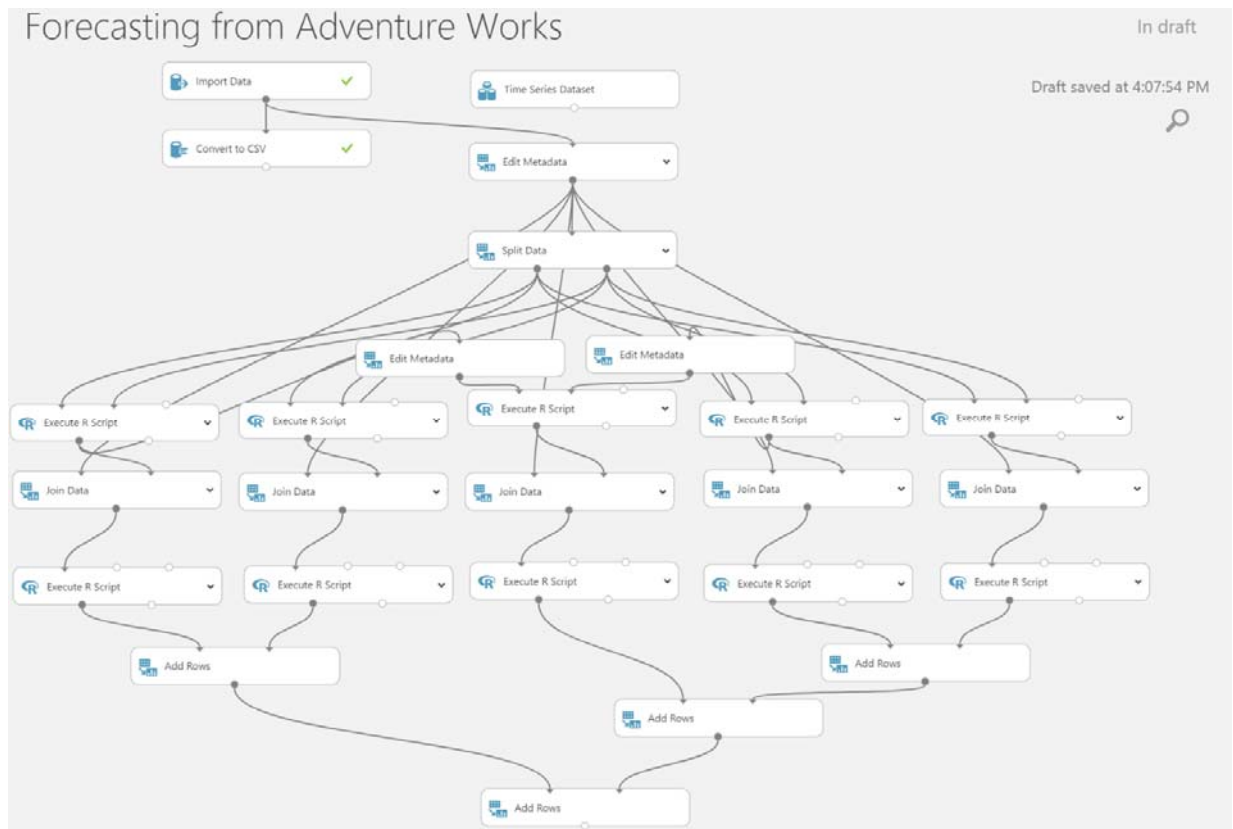
In [ ]: head(dat)
```

If we run all of that, we just get the first few rows back from the **head(dat)** function. However, the Jupyter Notebook can run anything we want in R including plots and any other R functions when we wish to get more information about the data we are working on. Also, this notebook will be automatically saved in our ML workspace where we can rename it and continue to work on it. For example, I have called mine SQL Bits:



In addition to viewing and analyzing the data in Jupyter Notebook, it's a good idea to save the data file locally or as an Azure ML dataset for future use. To do so, the options to Download and Save as Dataset are provided by right clicking the Convert to CSV module. We'll later use the data that are a CSV file (downloaded) and Save as Dataset in the next section. The dataset name can be anything of your choice, but we'll refer to it as AdventureWorksDW and AdventureWorksTimeSeries.csv.

Having understood this data, we can now adapt the experiment to see what the various forecasting algorithms can do with it. First, we need to replace the existing dataset. The easiest way to do that is to connect it to the metadata editor just under the Time Series data set. Let's also change the name of the experiment to such as Forecasting from Adventure Works.

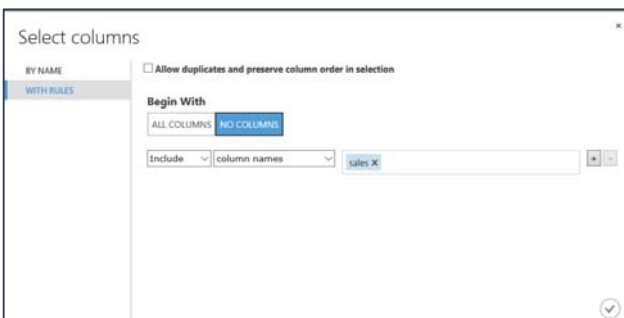


However, we can't just run the experiment yet as there are further modifications needed:

- Column names need to be changed to reflect the new data set. This new data set has proper dates in it where the N1725 dataset just had a unique number for each month.
- The seasonality of the old data was 12 as the data was monthly whereas this data is weekly.
- How will we split the data to train and test?
- There may be other breaking changes we need to fix.

Therefore, we need to go through the experiment top to bottom and test each step.

First, click on the Edit Metadata module we just connected the Import Data module to and launch the column selector. N1725 appears in red because we don't have that column anymore, so change it to sales and click the check icon.



While we are on this module change the comment **Rename the sales column to data.**

Now we need to change the way the Split Data module works. A good test might be to try and predict sales for 2004 which we can easily do by editing the properties of the split to `\ "year" < 2004`. Again, edit the comment for the module to reflect this change.

Now we can make some changes for each of the forecasting modules:

ARIMA seasonal and ETS seasonal

- Change the seasonality from 12 to 53 on line 6 - **seasonality <- 53**
- We want to forecast across the test data set (dataset2) so the number of periods to forecast is just the number of rows it has so we can comment out line 10 & 11 and on a new line enter - **numPeriodsToForecast <- nrow(dataset2)**

ARIMA non seasonal and ETS non seasonal

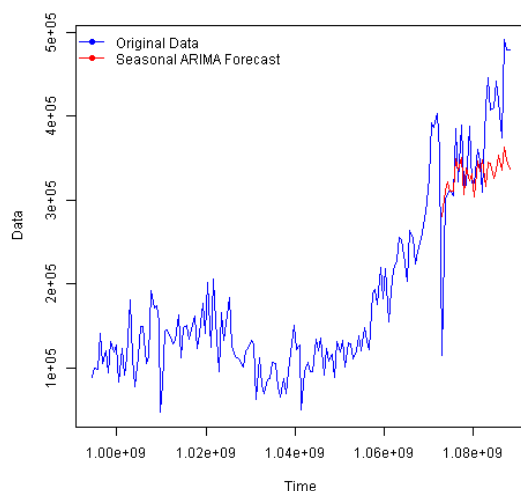
- comment out line 10 & 11 and on a new line enter - **numPeriodsToForecast <- nrow(dataset2)**

Average seasonal ETS and ARIMA

- Line 6 of this script is referencing columns by position and not by name, which is not a good practice, so comment that line out (to remind you not to do this!), and add this line in.

```
output_frame = data.frame(dataset1$time, output_forecast);
```

If we run this experiment now, we can see how the same algorithms work on this new data by visualizing the plots on each of the generate metrics. From this quick data swap and code modification, seasonal ARIMA produces more meaningful result than the other forecasting modules.



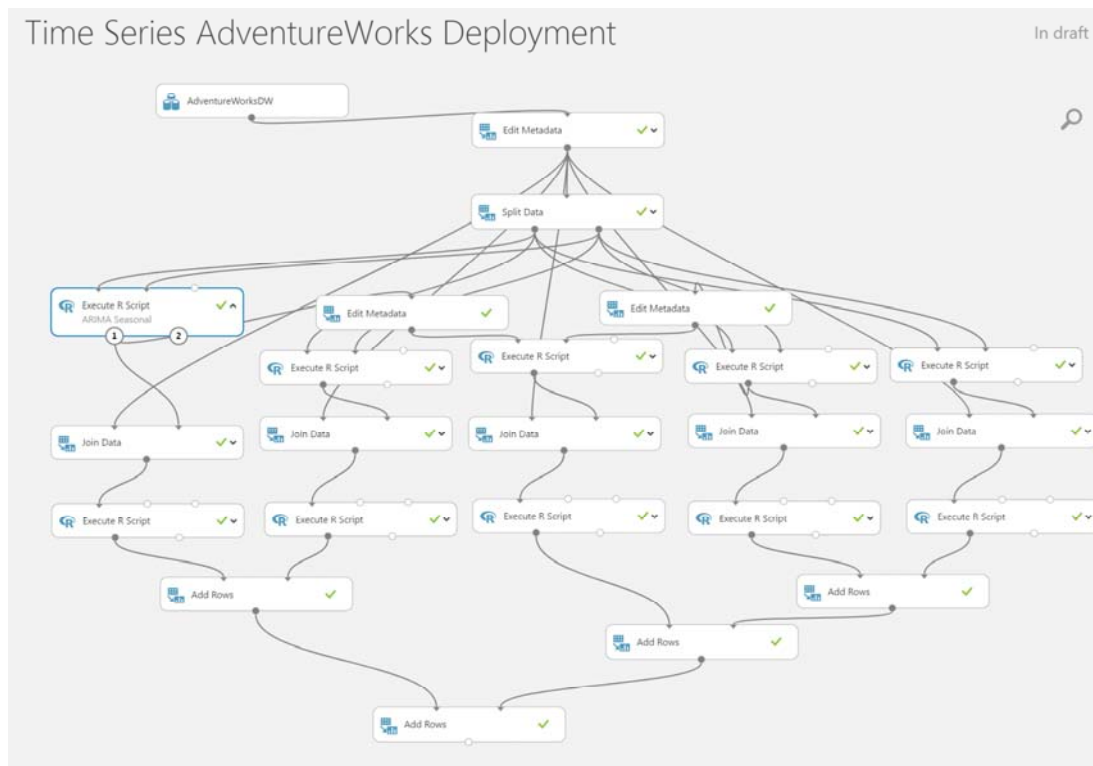
It may just be that this set of data is not easy to predict, but at least we can quickly establish this much considering the limited effort we spent for this experiment. In a real business scenario, we would further conduct analysis on this data to establish seasonality and trends in the data. A recommended resource for this is the [little book of Time Series in R](#).

Deploy Model and Predict in Excel

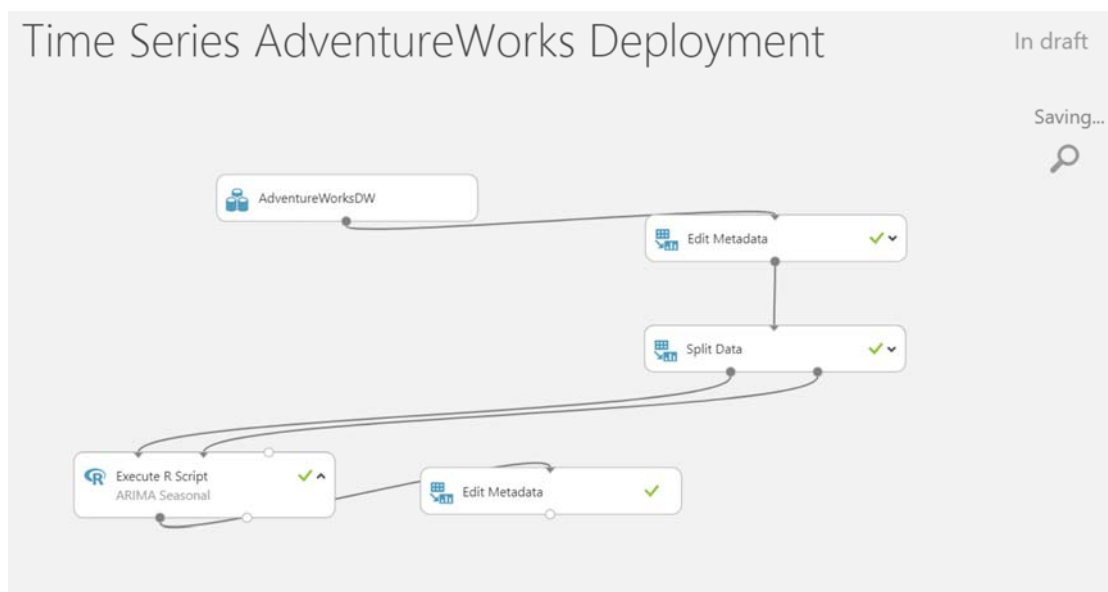
In this section, we'll productionize a selected forecasting model and consume the model via one of the most widely used data analysis tools, Microsoft Excel. Let's begin by a small data preprocessing task by opening AdventureWorksTimeSeries.csv and make sure that all the date or datetime columns are text. If they are not, they could cause error when we use them for Azure ML to forecast. If you find this preprocessing step too lengthy (and Excel skills are really not a learning objective of this lab), I have prepared a finished.xlsx file that can be downloaded from [here](#).

If you want to go through with this step, it's actually very easy. In order to make these date/datetime columns plain text, use the TEXT() function and adjust the parameters to fit with the original format of each column, such as TEXT(A1,"DD/MM/YYYY hh:mm:ss"). You'll need to copy and paste special "value" of the TEXT() columns. Lastly, you'll have to clean up the original as well as the TEXT() formulae to have the finished product.

Now, we're ready to continue with Azure ML. The next step is to rename the experiment to Time Series AdventureWorks Deployment. Bring in the saved dataset, AdventureWorksDW, and remove the Import Data and Convert to CSV modules. Your experiment then should look like this.

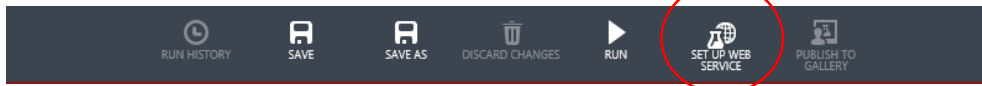


We found from the previous section that the ARIMA Seasonal model (the left most model) performed better comparing to others. Therefore, we'll deploy this model into production. The goal of the deployment is to get forecasted values as outputs. Since the original experiment contains many parallel experiments as well as R plots that we don't need in the deployment, we'll remove these modules out. As a result, the experiment will look like this.

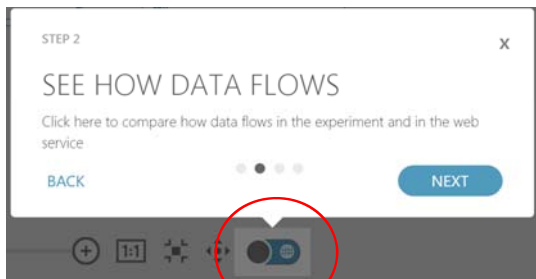


The Azure ML deployment is based on the Web Services API RESTful technology that is lightweight, scalable, and independent from predictive experiments. To learn more about this technology and how easy it is for software developers to integrate predictions in their conventional, web, and/or mobile applications, please visit <https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-consume-web-services>. For the purpose of this lab, we'll deploy the ARIMA forecasting model and consume the forecasted values using Excel. The basis of the model still stays the same that 1. the data prior to 2004 will be used to learn and anything in or after 2004 will be forecasted, and 2. the output columns are time and seasonal_arima.

We are now ready to setup Web Service by clicking at the SETUP WEB SERVICE button.



You should be seeing a quick animation that is actually adding two Web service modules to the experiment as well as a quick 4-step guide. Click Next to Step 2 that points out the toggle control to switch between the predictive experiment and web service views.



Please continue to click next on the guide. Once done, click RUN to run the new experiment. Wait a few moments for the run to finish then click Deploy Web Service [New] Preview.



This will bring you to a new tab, where you edit the name of the Web Service and choose the Price Plan. After that, click the green Deploy button.

Deploy "Time Series AdventureWorks Deployment" experiment as a web service

Web Service Name:

Storage Account:

The storage account shown is used by the workspace. The same storage account will be used for the new web service.

Price Plan:

Important: The plan tiers default to the plans in your default region and your web service will be deployed to that region.




By clicking on "Deploy", you agree to pay the plan charges in accordance with the [Pricing Page](#).

It'll take a few moments to process, and the page will refresh to confirm that it's done.

Quickstart | Dashboard | Batch Request Log | Configure | Consume | Test | Swagger API

← Web Services

Time Series AdventureWorks Deployment

BASICS	MANAGE & MONITOR	DEVELOP
		
Test Web Service Configure Web Service Use Web Service Launch in Excel	View usage statistics	Swagger Documentation Tutorial: How to build apps

Do spend time to browse all the tabs. If desire to include some sample rows of data with the Web Service, click the Configure tab, and toggle the Sample Data Enable? to Yes, and click Save. For our use case, this option is not useful.

Storage Account Name: ☐ Update Key for kayazureml

Sample Data Enabled?

Enable Logging: [Logging Help](#)

Because this R forecasting requires historical data that is older than 2004 to function, the Test tab that is a "vanilla" Web application will not work correctly in this setting (unless you further modify the R code by separating the learning R code from the forecasting R code). We'll proceed to the

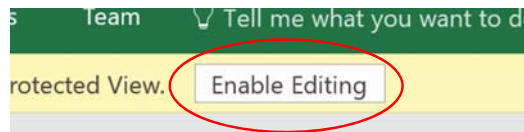
Consume tab and download one of two Excel files. In this lab, we'll show you the Excel 2013 or later option.

Time Series AdventureWorks Deployment

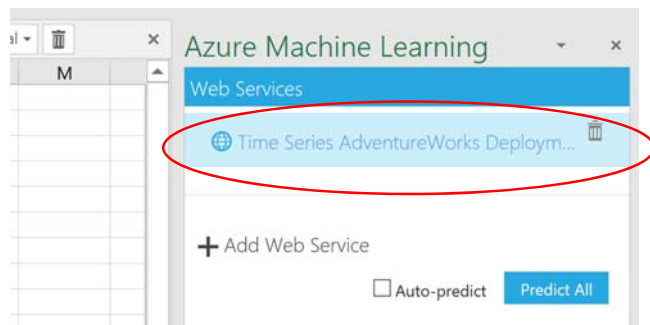
Web service consumption options



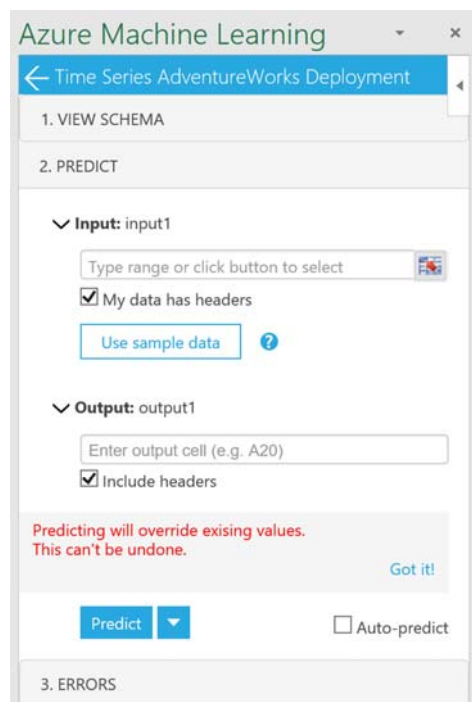
Save the file as AdventureWorksDeploy.xlsx and open it. Click Enable Editing and after a moment the Web Services will be loaded.



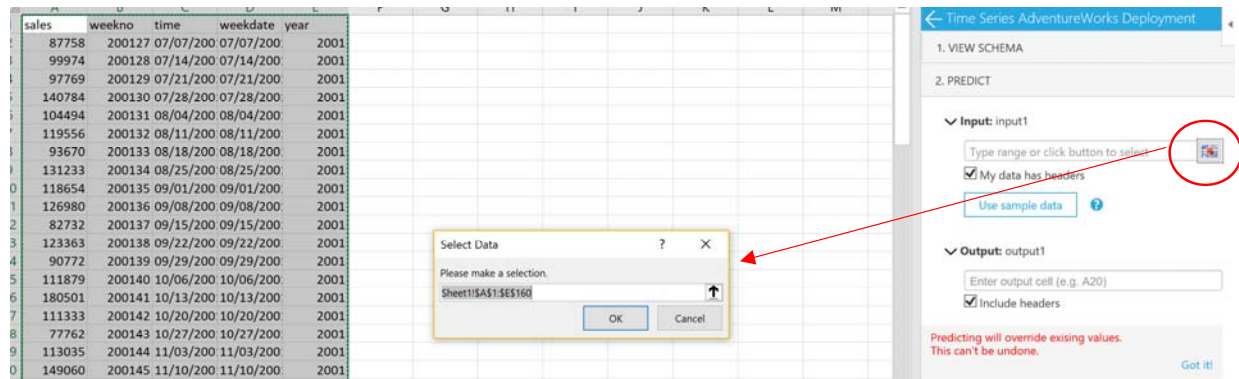
Click Time Series AdventureWorks Deployment to connect to the Web Service API.



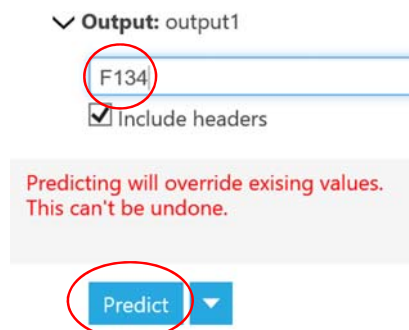
The right pane will refresh to look like the following.



Next is to obtain the data to paste into this Azure ML Excel file to predict. We'll open the preprocessed CSV data file, AdventureWorksTimeSeries.csv (or [AdventureWorksTimeSeries-ForPrediction.xlsx](#) from git). Copy the entire data (A1 to E160) and paste it on the Azure ML Excel file. While the pasted cells are still highlighted, click to select input data as shown. A window will pop up to confirm the selected area that is Sheet1!\$A\$1:\$E\$160. Click OK to close the popup window.



Type F134 in the Output cell where the header row of the forecasted values will be. Lastly click Predict and enjoy!



There you have it! The forecasted values from Azure ML right within Excel.

226989	200353	12/31/200	12/31/200	2003	time	seasonal_arima
114957	200401	01/03/200	01/03/200	2004	1/3/2004 0:00	279645.0004
300780	200402	01/10/200	01/10/200	2004	1/10/2004 0:00	311241.0004
307710	200403	01/17/200	01/17/200	2004	1/17/2004 0:00	321878.0004
311407	200404	01/24/200	01/24/200	2004	1/24/2004 0:00	310283.0004
305392	200405	01/31/200	01/31/200	2004	1/31/2004 0:00	310696.0004
385585	200406	02/07/200	02/07/200	2004	2/7/2004 0:00	349715.0004
322124	200407	02/14/200	02/14/200	2004	2/14/2004 0:00	335272.0004
389217	200408	02/21/200	02/21/200	2004	2/21/2004 0:00	351198.0004
317685	200409	02/28/200	02/28/200	2004	2/28/2004 0:00	306196.0004
317620	200410	03/06/200	03/06/200	2004	3/6/2004 0:00	338626.0004
387906	200411	03/13/200	03/13/200	2004	3/13/2004 0:00	322918.0004
319950	200412	03/20/200	03/20/200	2004	3/20/2004 0:00	330598.0004
320733	200413	03/27/200	03/27/200	2004	3/27/2004 0:00	303049.0004
360943	200414	04/03/200	04/03/200	2004	4/3/2004 0:00	346394.0004
346463	200415	04/10/200	04/10/200	2004	4/10/2004 0:00	334440.0004
308973	200416	04/17/200	04/17/200	2004	4/17/2004 0:00	348460.0004
411466	200417	04/24/200	04/24/200	2004	4/24/2004 0:00	315992.0004
445899	200418	05/01/200	05/01/200	2004	5/1/2004 0:00	344850.0004
407358	200419	05/08/200	05/08/200	2004	5/8/2004 0:00	343660.0004
409377	200420	05/15/200	05/15/200	2004	5/15/2004 0:00	325978.0004
441308	200421	05/22/200	05/22/200	2004	5/22/2004 0:00	335699.0004
418023	200422	05/29/200	05/29/200	2004	5/29/2004 0:00	353569.0004
374252	200423	06/05/200	06/05/200	2004	6/5/2004 0:00	335910.0004
490820	200424	06/12/200	06/12/200	2004	6/12/2004 0:00	362895.0004
479463	200425	06/19/200	06/19/200	2004	6/19/2004 0:00	347525.0004
478327	200426	06/26/200	06/26/200	2004	6/26/2004 0:00	337095.0004

Conclusion

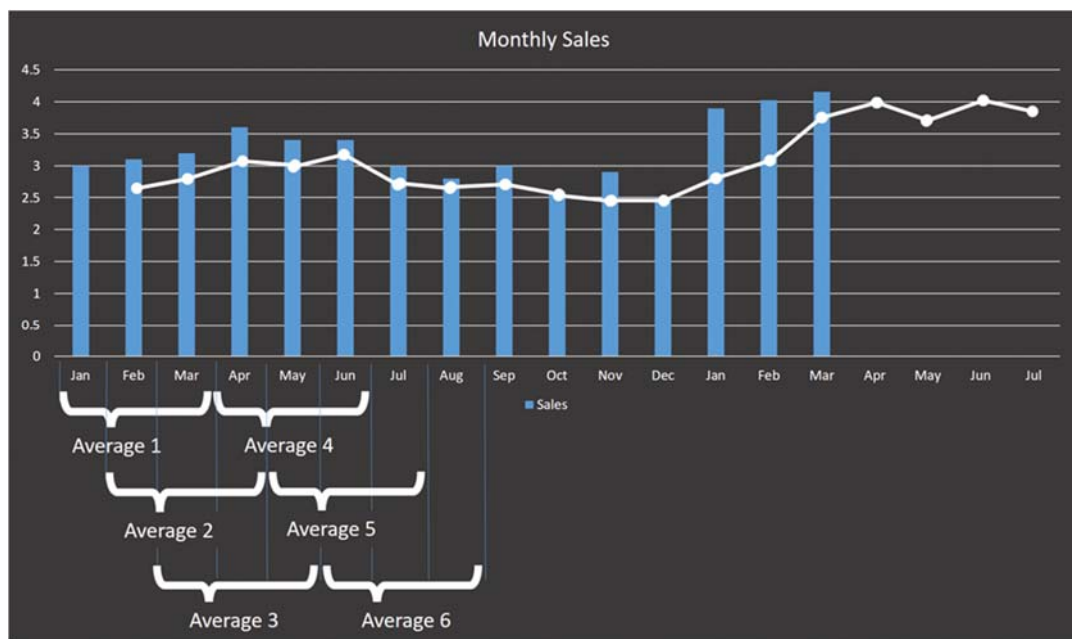
This hands-on lab was intended to introduce you to the basic concepts of Machine Learning and how to use R code and R notebooks in Azure ML.

Next Steps:

- R resources on a page <http://revolutionanalytics.com/r-language-resources>

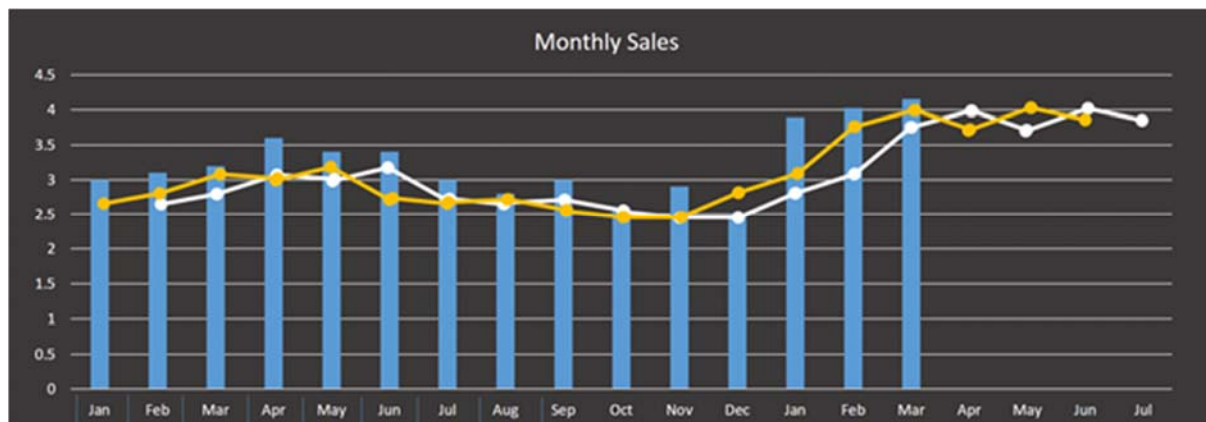
Forecasting 101

Let's begin by understanding the basics of how statistics based forecasting works, beginning with moving averages.



The graph above shows monthly sales in blue. From this, we can calculate averages 1 to 6 where each average is calculated over successive three-month periods. If we plot these averages, we get the line in white, where average 1 is shown against February and so on (the plot is for demonstration only). While the curve sort of matches the sales, there are a few of problems here:

- How many months should I average for each point on the curve given that the bigger this is the smoother the line will be.
- I have a built-in lag where the average is literally behind the curve. What I really want is to shift it left as shown the yellow line here.



Even when we could slide the curve, but then it's not that good a fit. So perhaps, we could put more of a weight on the most recent month in each average rather than treating each equally say 25% month1 + 25% month2 + 50% Month3 or some other value?

The whole process here is called smoothing (although the curves don't get smoother!). The most common technique is exponential smoothing where what we do is take a weighting (aka smoothing factor) W of say 45% and apply that backwards multiplying the factor by itself (hence exponential) right back to the beginning of our analysis

- last month (month-1) = $W = 45\%$
- month -2 = $(1 - W) * W = 55\% * 45\% = 24.75\%$
- month -3 = $(1 - W) * W * W = 55\% * 45\% * 45\% = 13.6\%$

that can look like a lot of work. But, we'd build this up as each new month appears. What happens is that we can use the result of the last forecast we did to get the next forecast.
 $(\text{last month's sales} * \text{Weighting}) + (\text{previous forecast} * 1 - \text{Weighting})$

Exponential Smoothing is available in Excel! but in a more sophisticated form called Exponential Triple Smoothing (ETS), which takes account of underlying trends in the data and regular rhythms or seasonality (we'll come back to seasonality later).

Another take on the moving average is ARIMA (Auto-Regressive Integrated Moving Average). The way to understand this is that what we want to do is to flatten the curve - there is no upward trend and there is no seasonality in the data (stationarity). In other words, the data wobbles predictably around an average value and so is said to auto correlate (where auto means with itself). The imperfections in this behavior are referred to as noise where what we are trying to get at is the signal (a lot of this idea came from electrical engineering).

To get data into this shape, we need to use math to transform our input data. For the technique to work, we need a good amount of historical data to work with. ARIMA uses either 3 or 6 arguments (6 for seasonal) and is written as ARIMA $p, d, q \times (P, D, Q)$ where x is the seasonality (say 12 months in our sales example):

- p is the number of autoregressive terms.

- d is the number of differences needed for stationarity so the wavelength. For example, if this is 1 then the next term is only based on the previous term.
- q is the number of lagged forecast errors in the prediction equation.

These terms can be considered more like switches in that they radically alter the behavior of the prediction. At this point, let's stop any attempt at giving a layman's terms explanation, and simply refer you to a good site at [Duke University](#) that goes into this in some detail. However, don't panic if you don't understand all of this as when it comes to Azure ML and R. These settings are not needed unless we need them to be and all we have to worry about is measuring how well the system we choose is forecasting against our data.

Terms of Use

© 2016 Microsoft Corporation. All rights reserved.

By using this Hands-on Lab, you agree to the following terms:

NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCTIONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER

This lab contains only a portion of the features and enhancements in Microsoft Azure. Some of the features might change in future releases of the product.