

R

Introduction

Contents

Forecasting in Azure ML

First-time setup

Reviewing a sample experiment

Forecasting with Adventure Works

Conclusion

Forecasting 101

What is Azure Machine Learning?

This 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. This will be based around a sample experiment (an Azure ML projects) in the Cortana Intelligence Gallery which already has all the code and sample data we need.

First Time Setup

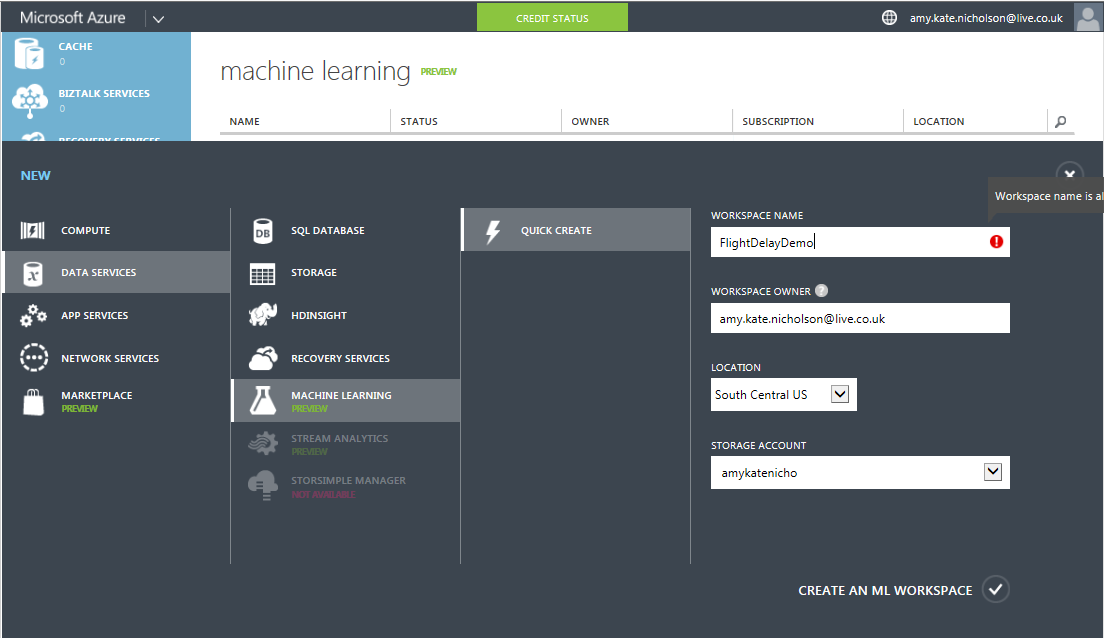
The Microsoft Azure ML Studio is in the Azure Management Portal, therefore you will need an Azure subscription to complete the lab. You can sign up for a free Azure one month trial; using your Microsoft account; where you get £125 to try out all the services in Azure.

Sign up to Azure for one month free trial here: <http://azure.microsoft.com/en-gb/pricing/free-trial/>

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

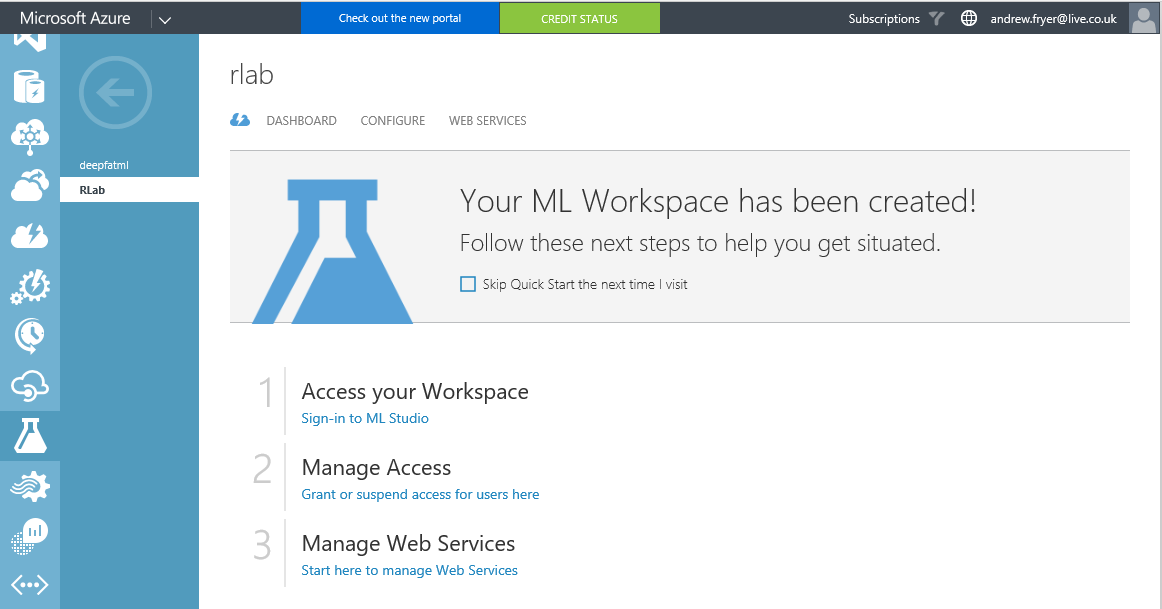
Once you have an Azure subscription you can go to http://[www.azure.microsoft.com](http://www.azure.microsoft.com) and choose ‘My Account’ at the top of the page then choose the ‘Management Portal’ where it will ask you to sign in with your Microsoft account.

Once into the management portal scroll down the toolbar on the left side of the screen and find the Machine Learning Icon. To create a new workspace click on the ‘New’, ‘Data Services’, ‘Machine Learning’ and ‘Quick Create’.



Enter details for all the variables below:

* **Workspace Name:** Pick a name for your workspace
* **Workspace Owner:** the Microsoft account to use to login to Azure ML.
* **Location:** W Europe (UK data centres may have this service when they come online)
* **Storage:** Create new storage account or pick the name of a current storage account.

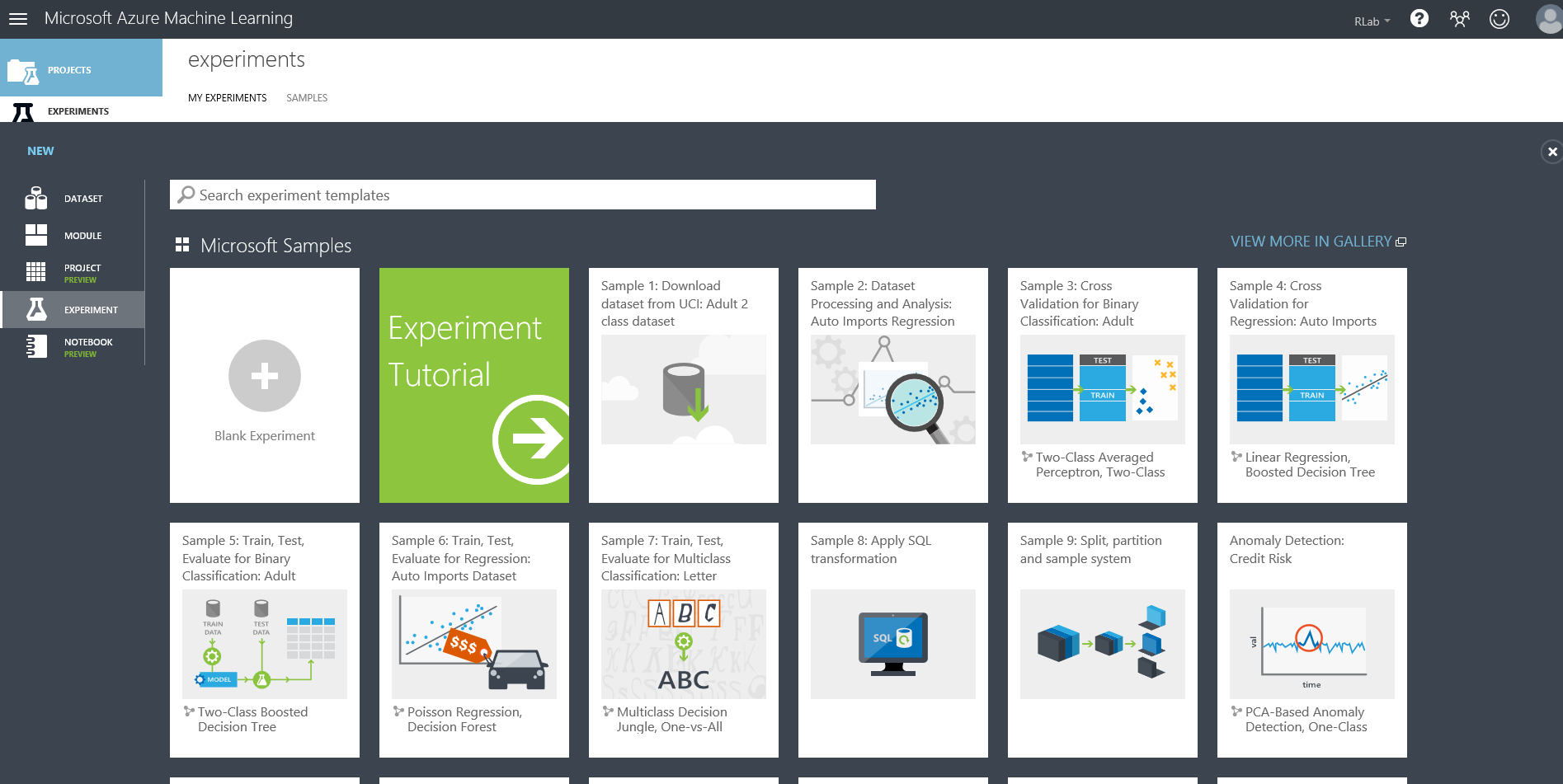


We are now ready to get into the world of R and forecasting.

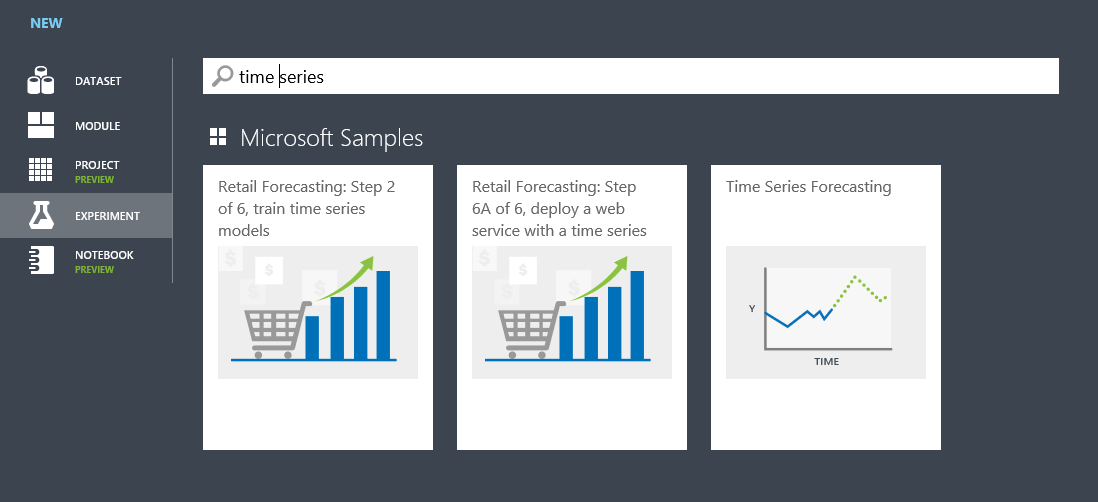
Reviewing a sample experiment

We are now ready to begin to use R in Azure ML and have a brief look at forecasting. Forecasting that is trying to predict patterns over time is not one currently one of the built in modules in Azure ML although it is possible to use regression [[1]](#footnote-2). For anything more sophisticated we have to write our own algorithm and we have two choices R & Python. For this lab we are using R. Fortunately, there are several examples of how to do this in the [Cortana Intelligence Gallery](http://gallery.cortanaintelligence.com/).

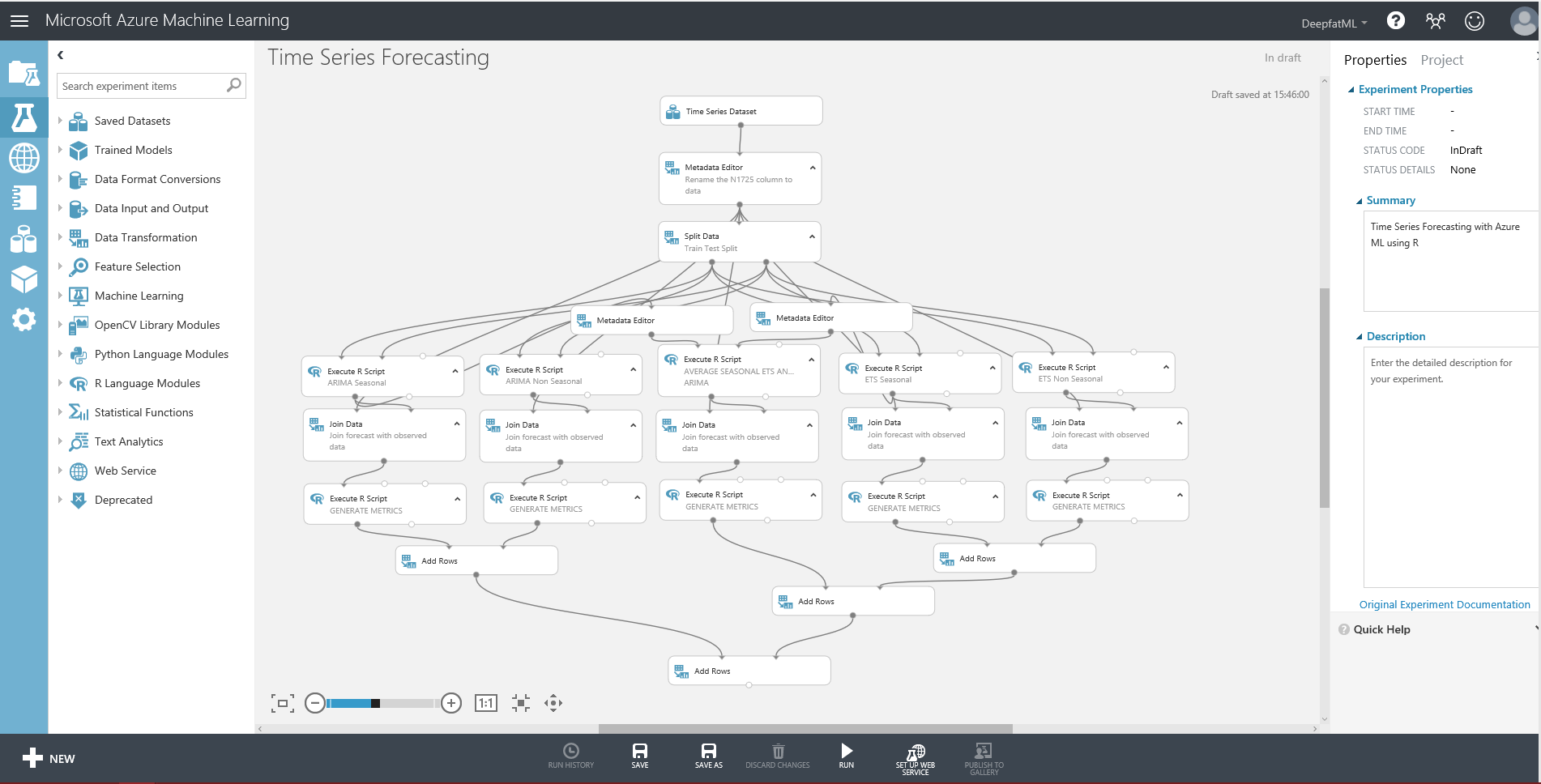
Once our Workspace has been created we can click on the link ‘Sign-in to ML Studio’ and we’ll be taken to ML Studio and given the option to create a new experiment which can be blank or from one of the samples..



In the search box enter “time series”



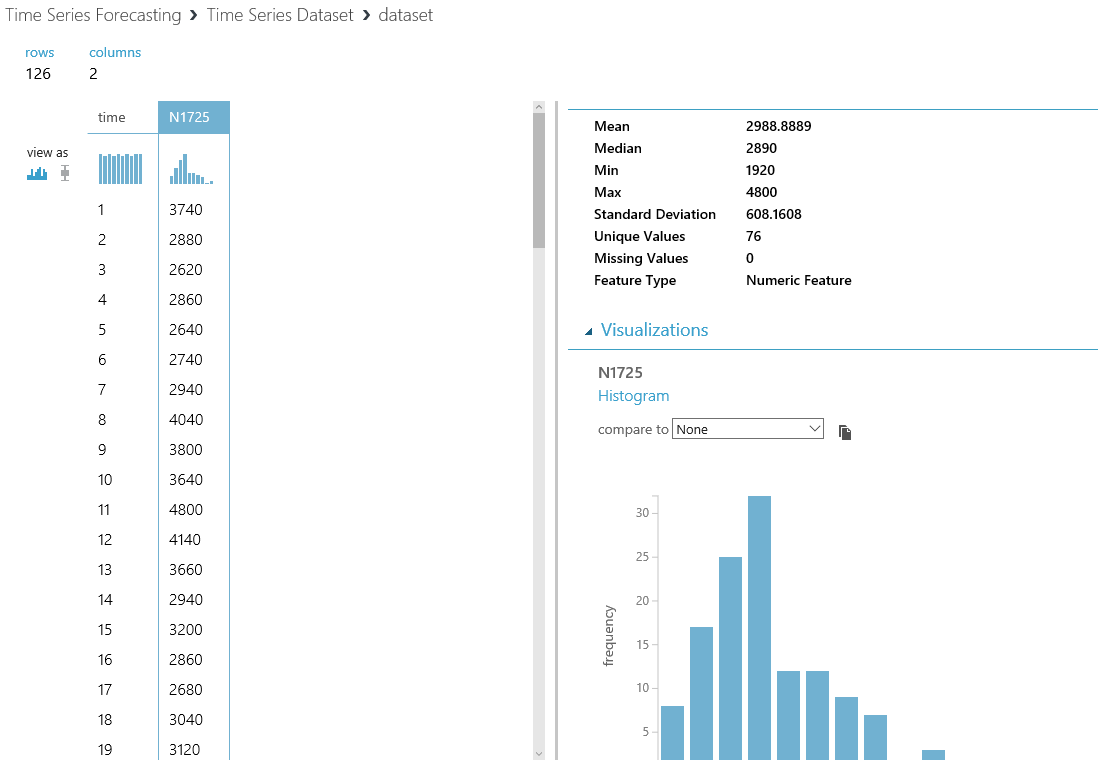
so we can find the sample we want. Select the Time Series Forecasting and then select view in gallery so we can find out more about it. We’ll get a detailed description of how this works and how to use it so do spend some time reading this. When you have absorbed that click on Open in Studio which allows us to save a local copy 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) and then it will open in ML Studio..



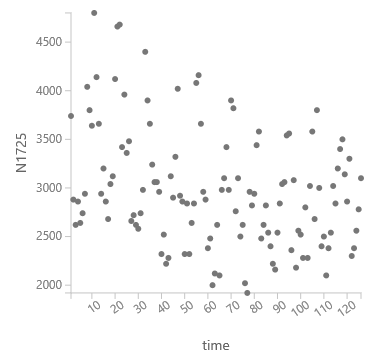
Here we can see how the experiment is wired up where the grey lines show data flows through the various modules. New modules can be selected from the list on the left and then we can set the properties of the module we have focus on the right. Notice the experiment is branches into five parallel streams 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 rough guide introducing these how these works at the end of this guide.

In order to see 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 of ML studio. We’ll see each module ticked in green as the run progresses (runtime is about 3 minutes) 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 visualise.. click on the N1725 column to get some basic stats about that column..



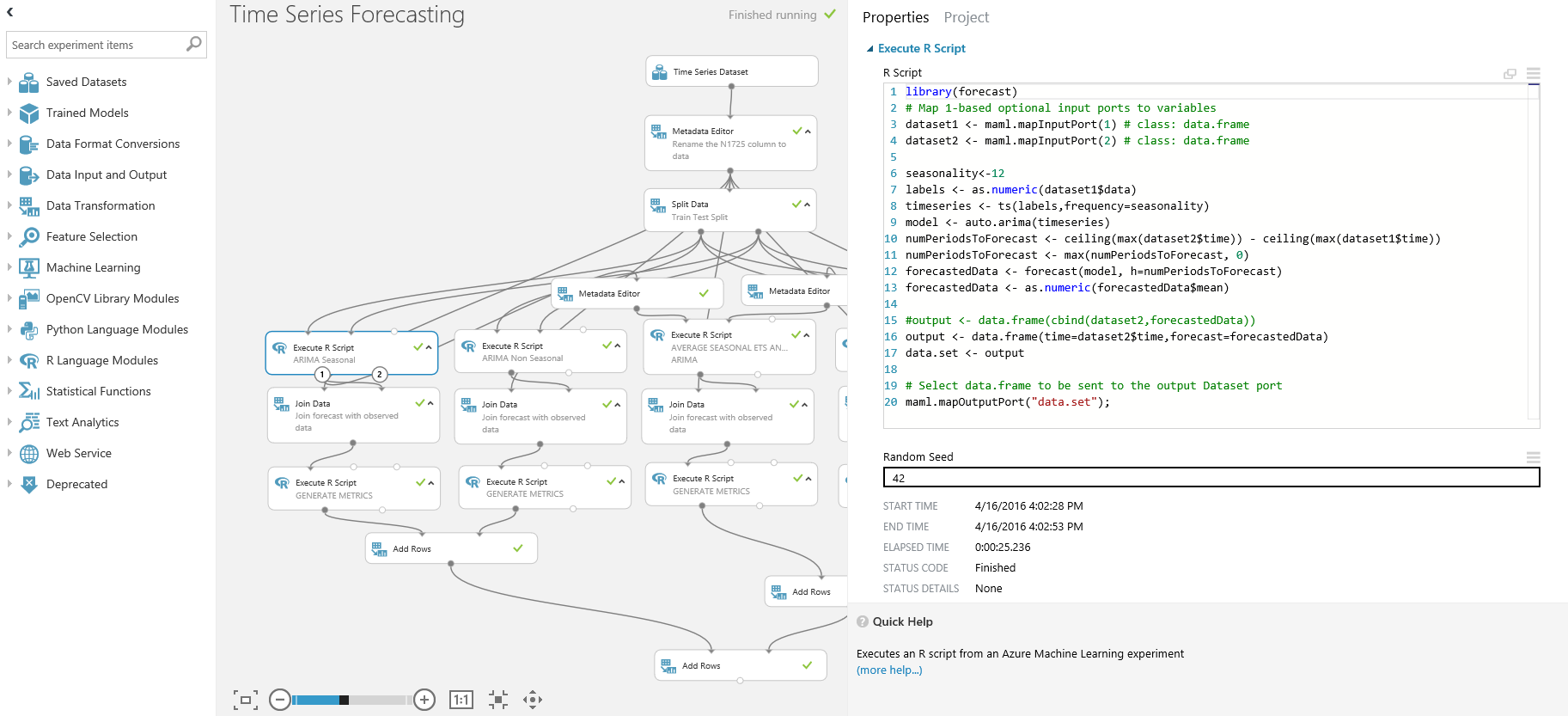
If we select time and then compare this to N1725 we get a scatterplot of the relationship between the variables..



But we can 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 just outputs older data on the left port and the newer data on the right port by using a simple expression \”time” <=108. This has been done so that we can train against the older data and then compare the forecast against the actual later data as a test.

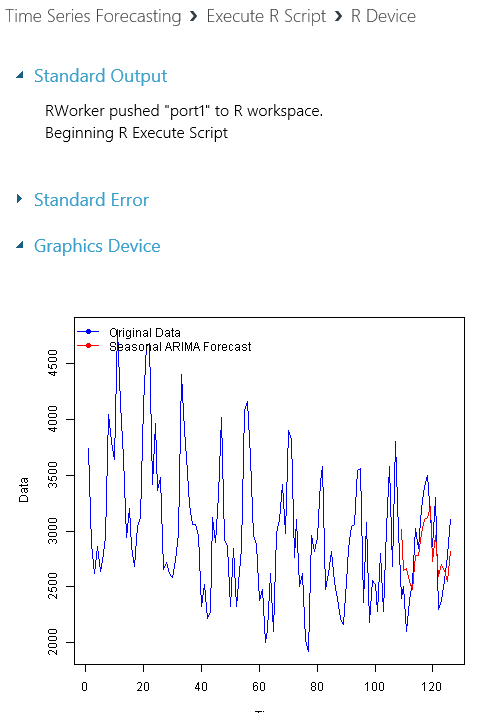
If we look at each of the five columns we can see an R module to do the particular type of forecast followed by a join followed by another R script to compute metrics to determine the accuracy of the forecast. If Looking at the ARIMA seasonal module (on the left) we that an R module has three import 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 default. The R module also has tow output ports one for the flow of data and the other can be used for outputting other information from R e.g. messages and plots. If we set focus on this module and rearrange the properties pane we can see the R in this module.



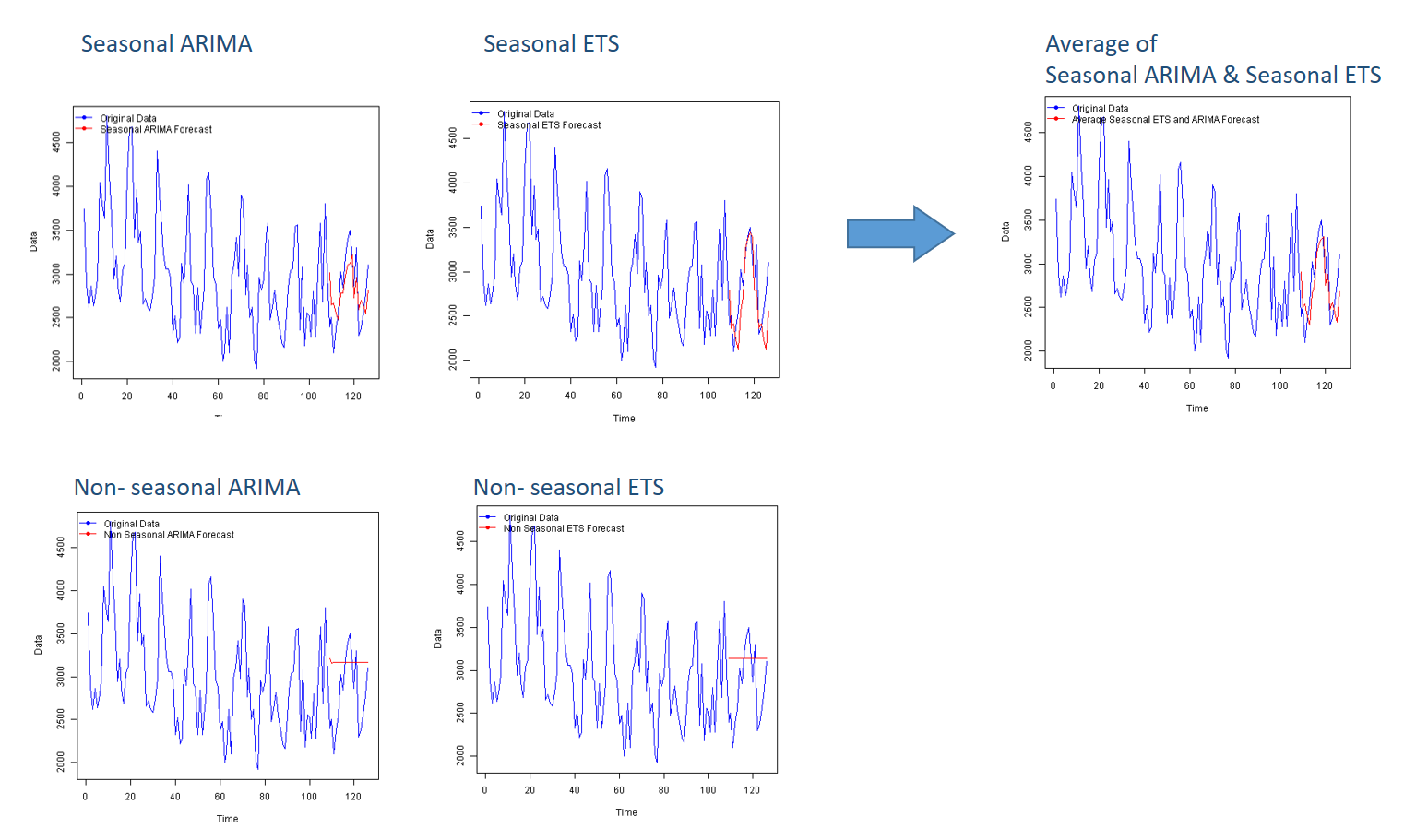
notes

1. There are already a lot of standard libraries in Azure ML and so all we need to is declare which we want to use in this case library(forecast)
2. We can see how the import/output ports are exposed as R dataframes – maml.mapInputPort(1), maml.mapInputPort(2) , maml.mapOutputPort(“data.set”).
3. The output from this is the forecast time and the forecast itself and not the actual values form data set 2 which is why there is a join after this module where this could have been done in R.
4. However the editor is pretty primitive so we’ll probably want to use R Studio or Visual Studio to write anything complex.

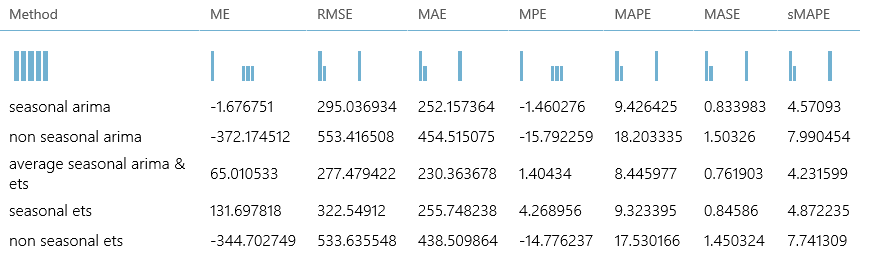
If we now turn our attention to the Generate Metrics module underneath the ARIMA seasonal module we can see that this much longer script not only computes metrics it also does a plot of the data and we can get to this (once we have run it by right clicking on the right output (2) of this module..

Where the blue line is the actual data for the whole data set with the forecast 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..



It’s not easy to see form the graphs but the best fit is the average and this is more obvious if we look at the output from the last module where each type algorithms is analysed using a number of standard metrics in one grid..



Where 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)](http://en.wikipedia.org/wiki/Mean_signed_difference) - 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)](http://en.wikipedia.org/wiki/Root-mean-square_deviation) - The square root of the average of squared errors of predictions made on the test dataset.
* [**Mean Absolute Error** (MAE)](http://en.wikipedia.org/wiki/Mean_absolute_error) - The average of absolute errors
* [**Mean Percentage Error** (MPE)](http://en.wikipedia.org/wiki/Mean_percentage_error) - The average of percentage errors
* [**Mean Absolute Percentage Error** (MAPE)](http://en.wikipedia.org/wiki/Mean_absolute_percentage_error) - The average of absolute percentage errors
* [**Mean Absolute Scaled Error** (MASE)](http://en.wikipedia.org/wiki/Mean_absolute_scaled_error)
* [**Symmetric Mean Absolute Percentage Error** (sMAPE)](http://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error)

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 apply our knowledge of R to see how to adapt this experiment to use a new sample dataset from the standard Adventure Works Data Warehouse (AWDW) database included in SQL Server.

First we need to connect to the source database and to save time we already have an Azure SQL DW [[2]](#footnote-3)with the AWDW sample already setup. To connect to a data source we need to use the Reader module in Azure ML:

* In the search box above all the modules in ML Studio type reader
* Drag the Reader module onto the design surface and enter the following information to connect to the sample AWDW on Azure SQL Data Warehouse..

|  |  |
| --- | --- |
| Data Source | Azure SQL Database |
| Database Server Name | INSUFFICIENTDATA.database.windows.net |
| Database Name | AWDW |
| Server user account name | SQRL |
| Server user account password | SQLbits16! |

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 ARE INCORRECT

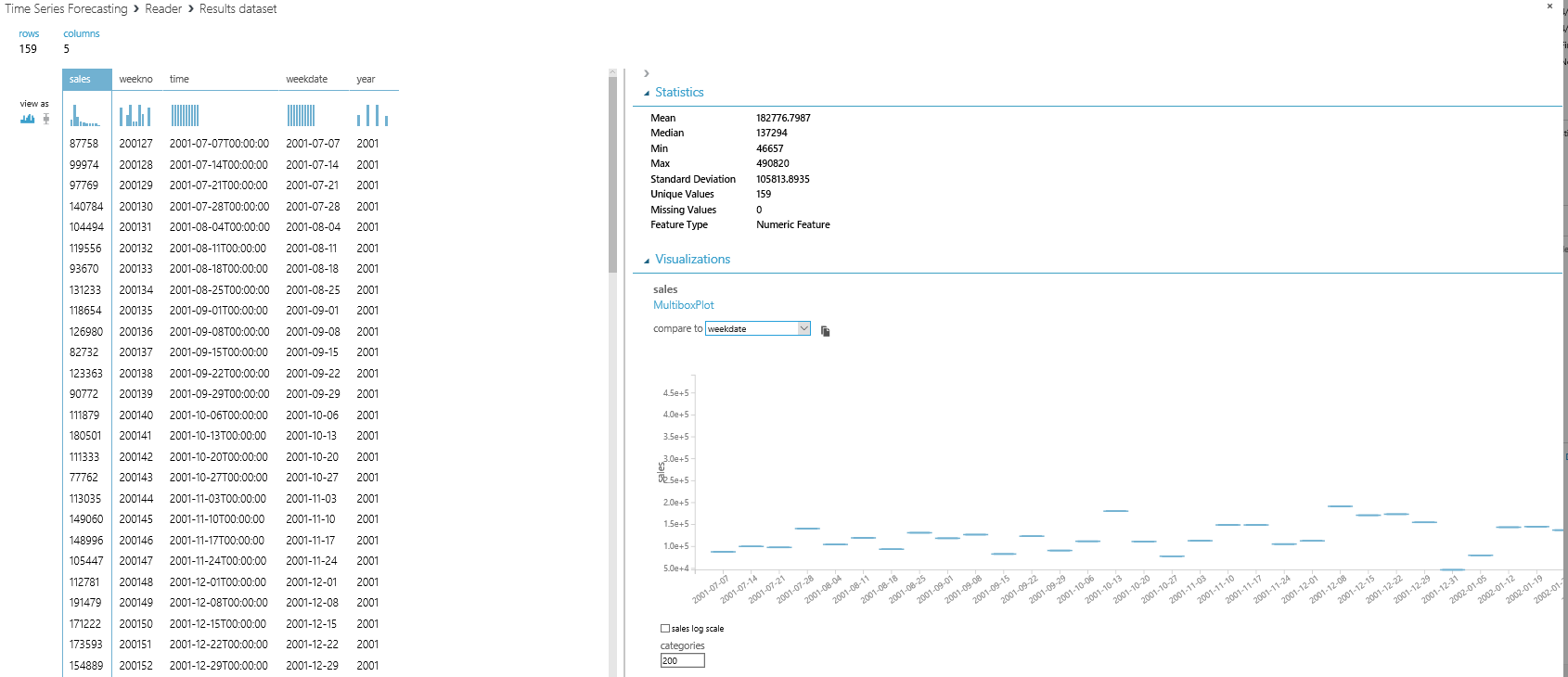
GROUP BY

(dd.[CalendarYear]\*100) + dd.[WeekNumberOfYear],

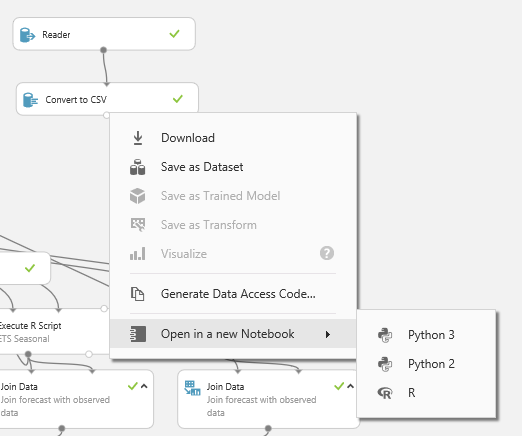
dd.CalendarYear

ORDER BY weekno

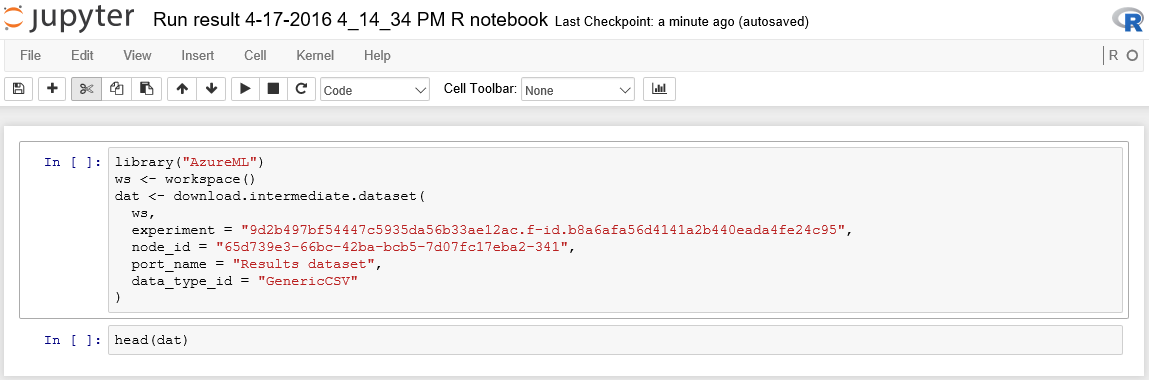
Don’t connect the Reader module to anything but do run the experiment again to check the query is working OK and visualise the output.



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 reader to it and run the experiment again and we can then right click on the CSV Module and open the output in an R Jupyter Notebook..

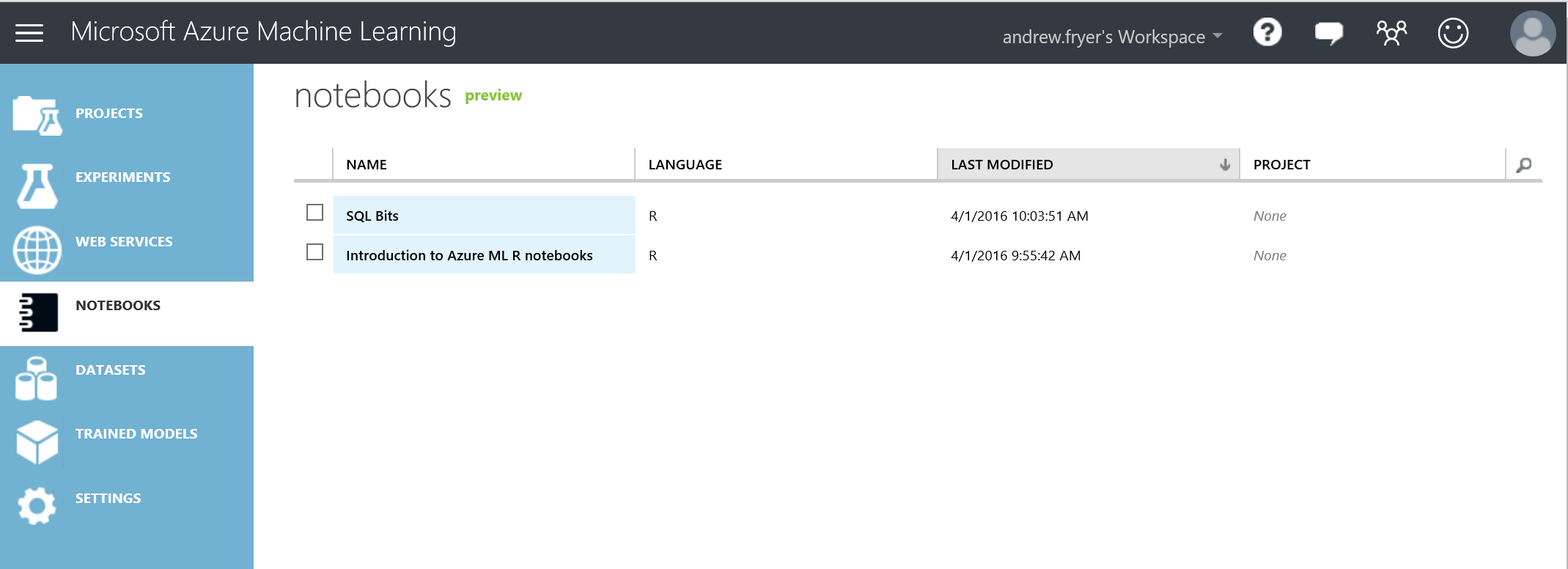


Which looks like this..



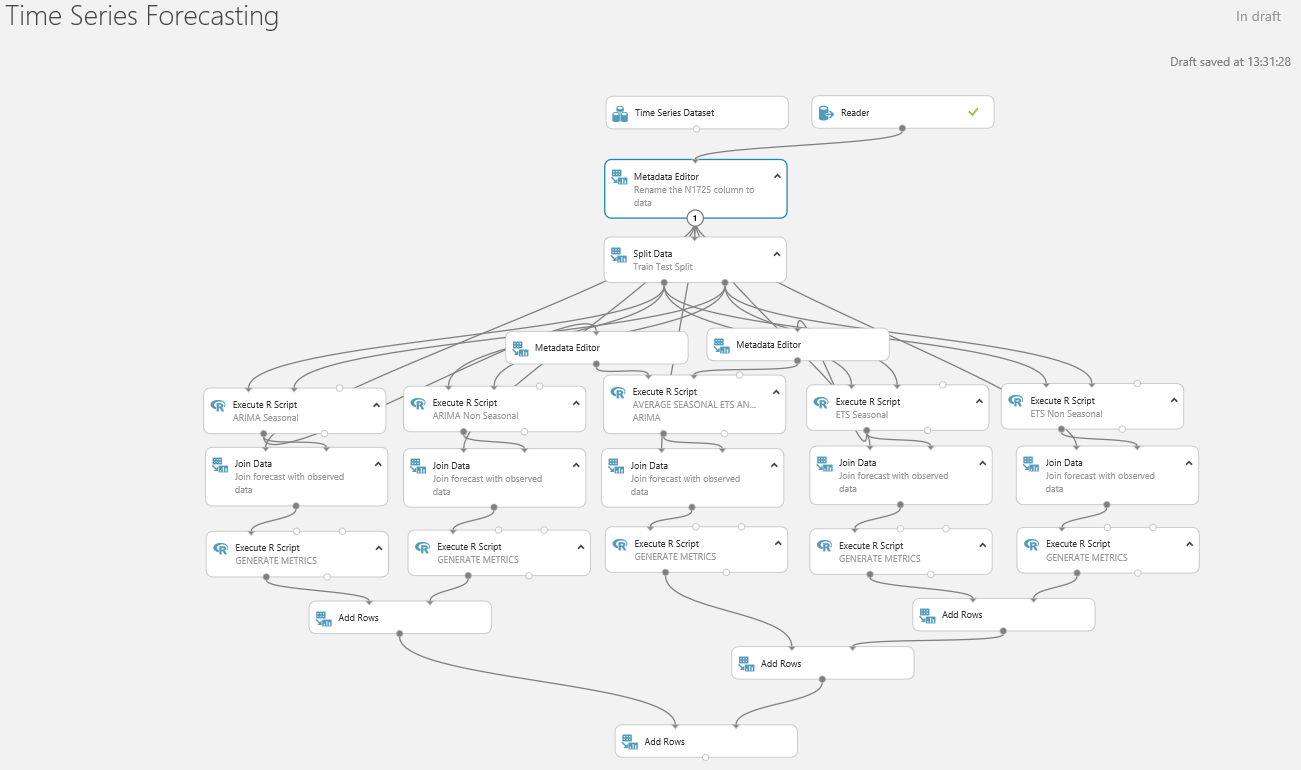
If we run all of that we just get the first few rows back from the head(dat) function but from here we can run whatever we want in R e.g.

Also this notebook will be automatically saved in our ML workspace where we can rename it and continue to work on it, here I have called min SQL Bits:



We can also add plots and any other functions we wish to get more information about the data we are working on.

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 and the easiest way to do that is to connect it to the metadata editor just under the Time Series data set...

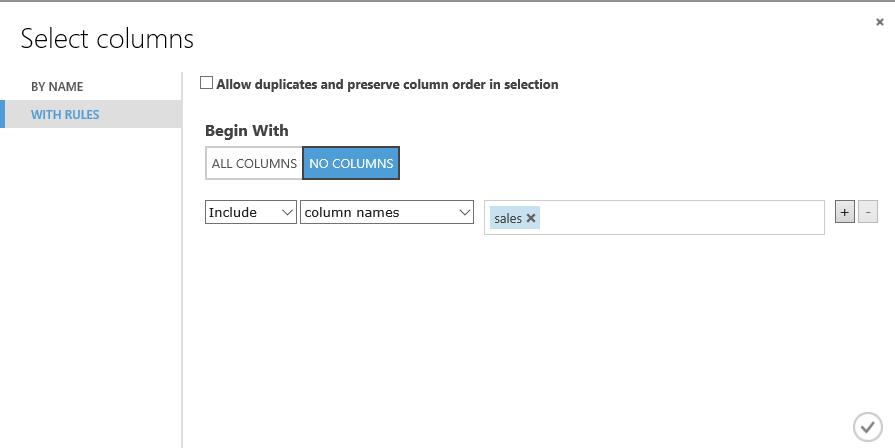


However, we can’t just run the experiment now as lots of things are broken:

* Column names need to be changed to reflect the new data set and 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 shall we split the data to train and test?
* There may be other breaking changes we need to fix

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

First let’s change the name of the experiment to something like AW Time Series Forecasting. Next click on the Metadata module we just connected the reader 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 tick box



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

Now we need to change the way the split 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 and 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

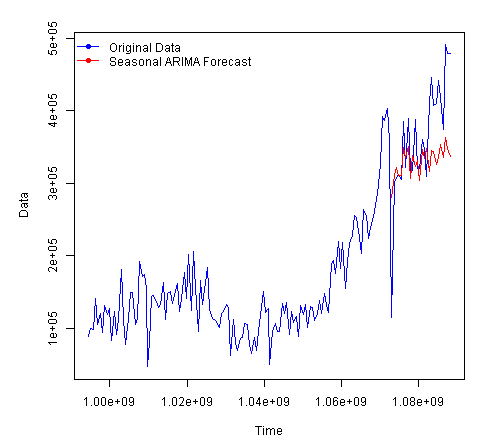
* comment out line 10 & 11 and on a new line enter - numPeriodsToForecast <- nrow(dataset2)

Average seasonal ETS and ARIMA

* Line 6 of this script is referencing columns by position and not by name which is bad practice so comment that line out (to remind you not to do this!) and 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 visualising the plots on each of the generate metrics. Frankly only the seasonal ARIMA produces any kind of meaningful result..



It may just be that this set of data is just not easy to predict but at least we can quickly establish this with this experiment. In real life we would conduct analysis on this data to establish seasonality and trends in the data and a great resource for this is the [little book of Time Series in R.](https://media.readthedocs.org/pdf/a-little-book-of-r-for-time-series/latest/a-little-book-of-r-for-time-series.pdf)

Conclusion

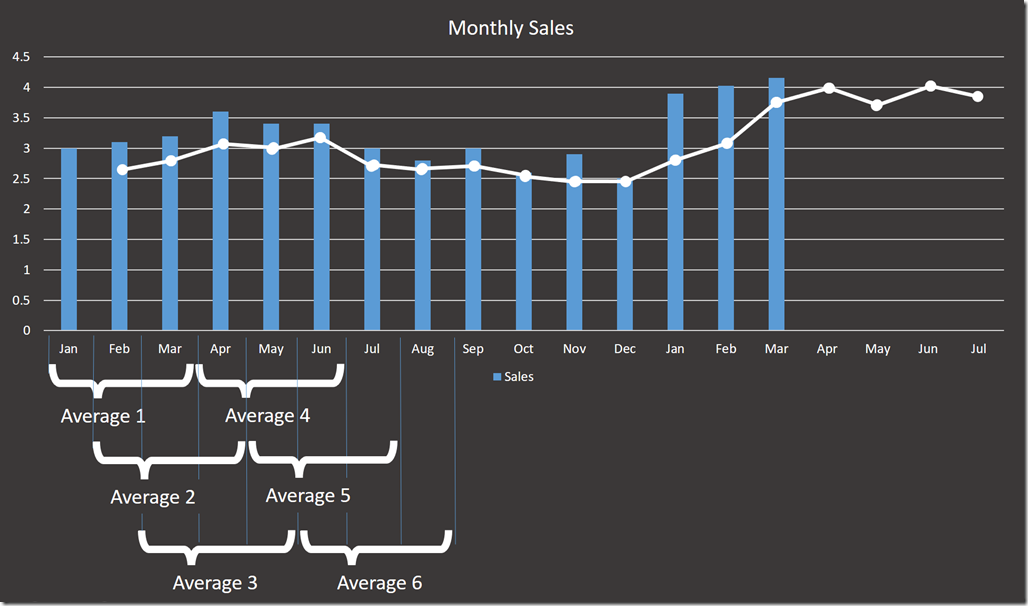
This lab was intended to introduce you to the basic concepts of Machine Learning and how to use R code and R notebooks in Azure ML

**Next Steps:**

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

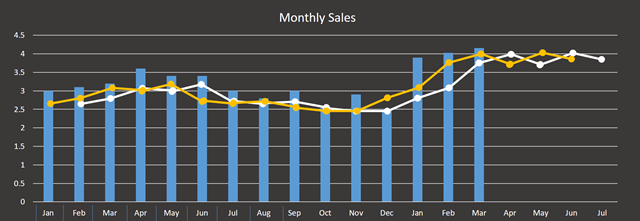
Forecasting 101

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

[](file:///C:\Users\afryer\AppData\Local\Temp\OpenLiveWriter-899803736\supfiles27185D6B\image42.png)

The graph above shows monthly sales in blue and from tis 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 not exact- just for show). 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 shunt it left like the yellow line here..

[](file:///C:\Users\afryer\AppData\Local\Temp\OpenLiveWriter-899803736\supfiles27185D6B\image47.png)

So we could slide the curve but even then it’s not that good a fit so perhaps I 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!)and 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 in reality we’d build this up as each new month appears and what actually happens is that we can use the result of the last forecast we did to get the next forecast..

(last month’s sales \* Weighting) + (previous forecast \* 1-Weighting)

Exponential Smoothing is even 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 stuff came form electronics).

To get data into this shape we need to use maths to transform our input data and 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 x(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 behaviour of the prediction and at this point I’ll stop any attempt at plagiarising and simply refer you to a good site at [Duke University](http://people.duke.edu/~rnau/411arim.htm#arima100) 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 setting 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.

1. Regression is the business of fitting points to a line so linear regression fits points to a straight line. [↑](#footnote-ref-2)
2. You can create one of these 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 this is an enterprise scale solution and so is expensive to leave running so if you do decide to do this pause it to stop being charged for it until you need it again. [↑](#footnote-ref-3)