HOW TO BUILD AND VISUALIZE MACHINE LEARNING PREDICTIONS WITH MICROSOFT AZURE AND MICROSTRATEGY

Presented by

MicroStrategy

Introduction

Haven't you always wondered how machine learning could actually be used by your business? In this one hour workshop, you will learn the how to use Azure Machine Learning to gain insight from data that is easily consumed by MicroStrategy Desktop.

You will use publicly available data to make predictions and learn about the thought process behind this kind of effort. You will also learn how to build your first model with Azure Machine Learning ML, and visualize these predictions with MicroStrategy Desktop. Once finished, you will have a deeper understanding of Machine Learning and gain insight into additional possibilities about how you can leverage these techniques to add sophistication to your own applications.

In this workshop, we will use flight data to demonstrate how you can make a prediction using Azure ML, and use MicroStrategy to visualize this information and prediction.

Overview

The main sections of this workshop are:

- Prerequisites

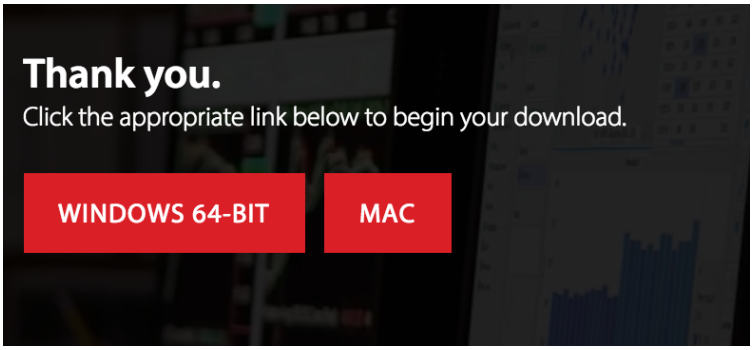
    a. Azure Machine Learning only requires network connectivity to the internet, no sign-up, no account required
    b. Download and install MicroStrategy Desktop
    c. Download and install R
    d. Download and install R Integration Pack
    e. Download "Microsoft Azure Workshop.mstr" dashboard file, the "FlightML.R" script file, and the **R Metrics for Microsoft ML workshop.pdf** that are available in the supporting files folder.

- Azure Machine Learning - How to use Machine Learning ML, for predicting if a flight will be late
- MicroStrategy Desktop – How to visualize predictions with MicroStrategy dashboards

Prerequisites and Setup:

Download and Install MicroStrategy Desktop

1. If you do not have the latest version of MicroStrategy Desktop installed on your laptop, you can download it from the below link for free: https://www.microstrategy.com/us/desktop
2. Please select the version corresponding to your laptop Operating System (Windows or Mac).



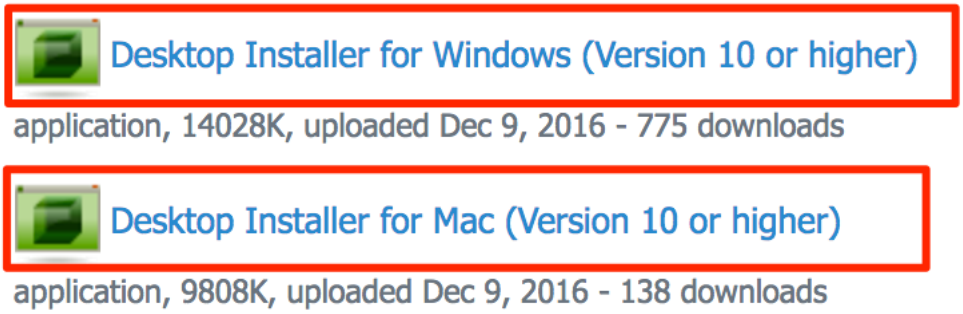
3. To install Desktop, go to your downloads folder to find the zip file you just downloaded, unzip it and run the installation: “MicroStrategyDesktop-64bit.exe” for Windows and “MicroStrategy Desktop.pkg” for Mac.
4. Follow the installation wizard and accept all default settings to finish the installation.

Download and install R

5. If you are using a Windows laptop, please download the Microsoft R Client from: http://aka.ms/rclient/download. Alternatively, Windows and Mac can download CRAN R from https://cloud.r-project.org/bin/macosx/
6. Install R on your laptop and accept all default settings.

Download and install the R Integration Pack

7. Go to: http://rintegrationpack.codeplex.com/releases/view/630028
8. On the web page, locate and download the appropriate version for your laptop operating system.



9. Install the R Integration Pack on your laptop and accept all default settings.

Download Workshop Supporting files

10. Download the "Microsoft Azure Workshop.mstr" file, the FlightML.R script file, and the **R Metrics for Microsoft ML workshop.pdf** that are available in the **supporting files folder.**

Azure Machine Learning for Flight Prediction

We will use a simple example in Azure Machine Learning to demonstrate concepts that can be leveraged for other use cases. The example leverages publicly available data to use machine learning to predict when a flight will be late. A web service is created from the model, and MicroStrategy Desktop will use the web service to pass in flight data to predict whether a flight will be late.

Access Azure ML

11. Set up access to Azure ML

This Azure Machine Learning Workshop can be completed with any of the following:

* Guest Access - http://studio.azureml.net/home/anonymous

This does not require an Azure subscription or a credit card. It is anonymous access, and after 8 hours the workspace resets.

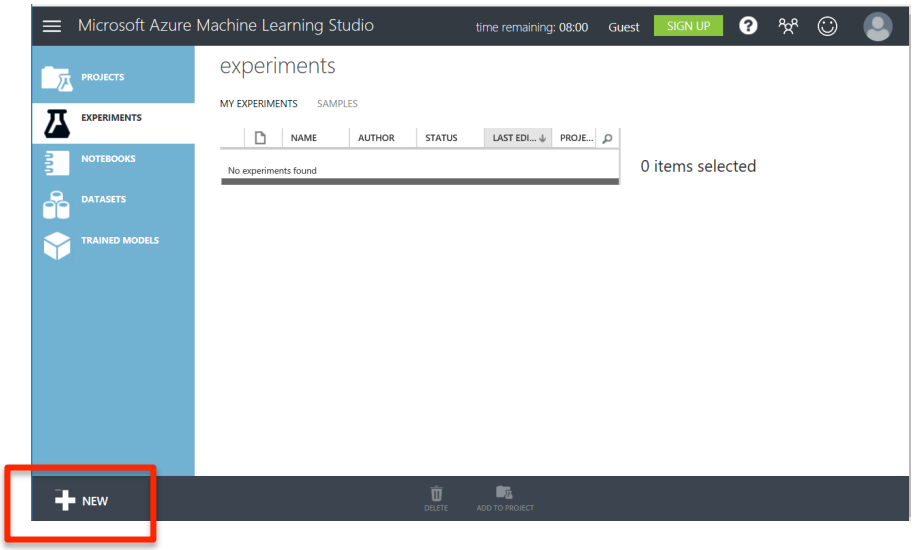
* Your Own Account - https://studio.azureml.net/Home

Sign in and use a work, school, or Microsoft account.

Create an Experiment

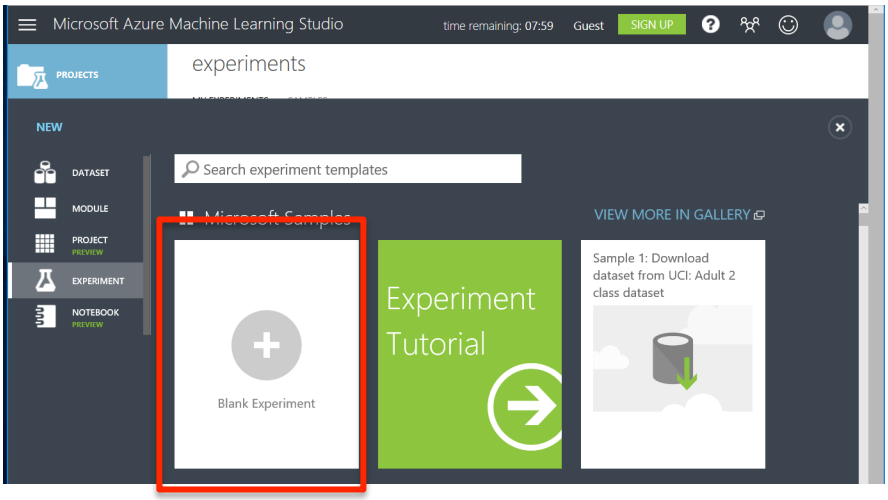
Let's get started by making a new experiment.

12. Select +New in the lower left corner.



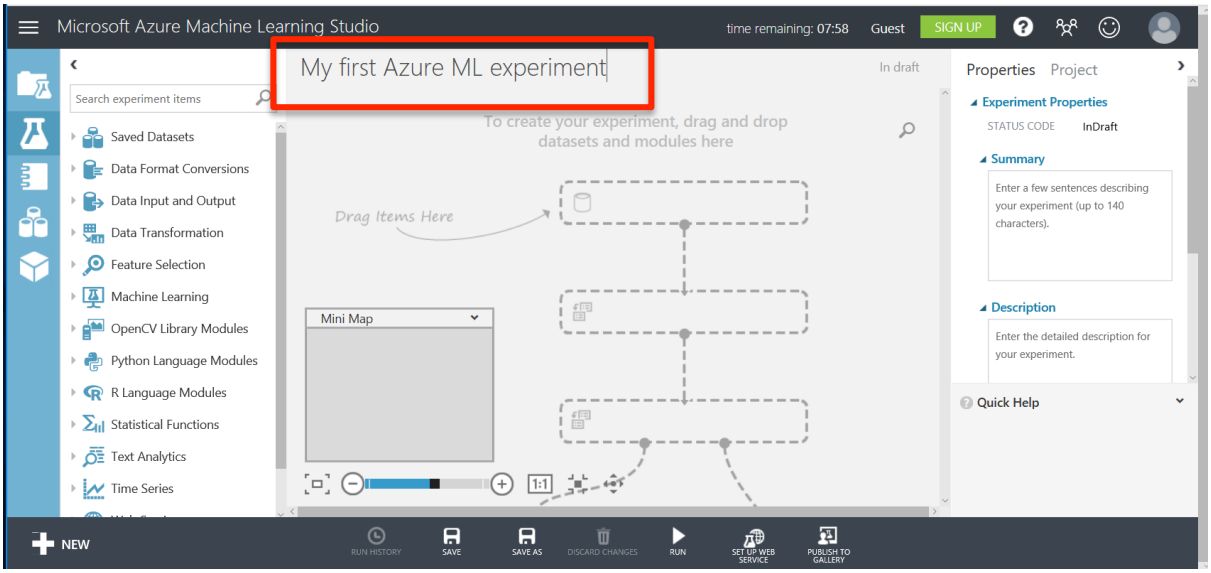
To the right of **Experiment**, you will see a tile with a plus sign and the words “Blank Experiment.”

13. Select + **Blank Experiment.**



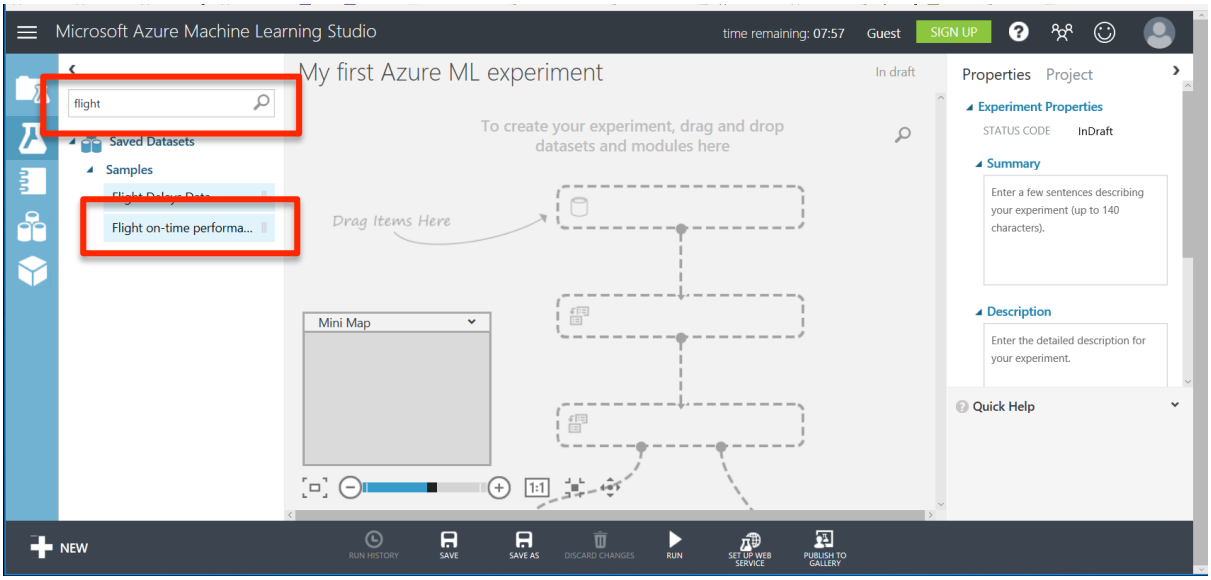
By default, a title is created with a name like "***Experiment created on 1/19/2017.***"

14. Give the experiment a title. Change the title to **"My first Azure ML experiment"**

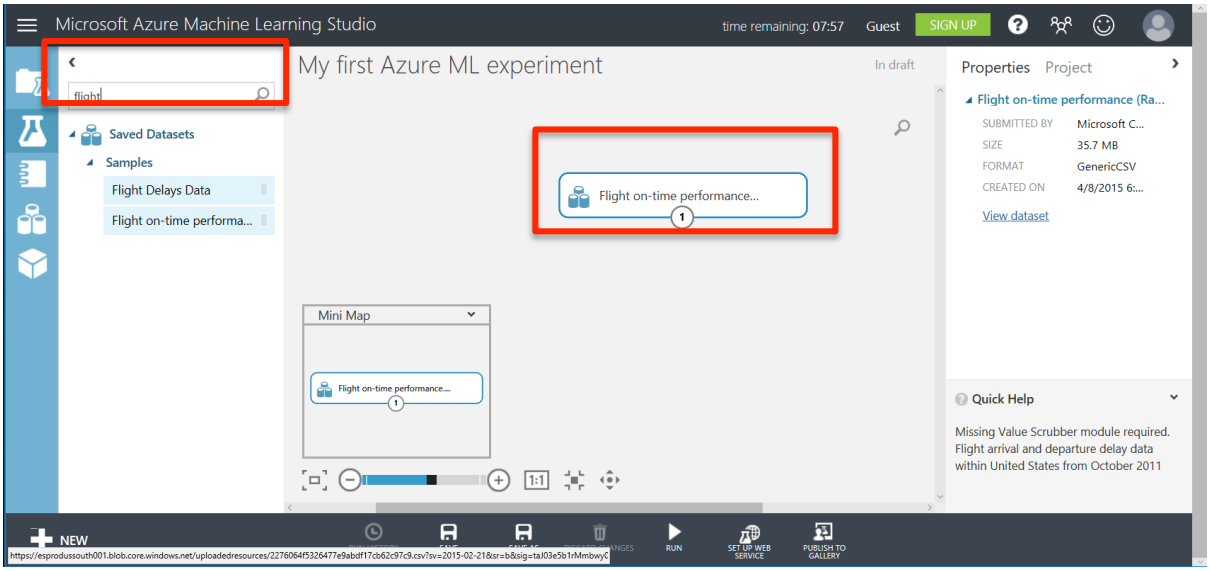


Import, Review, and Clean Data

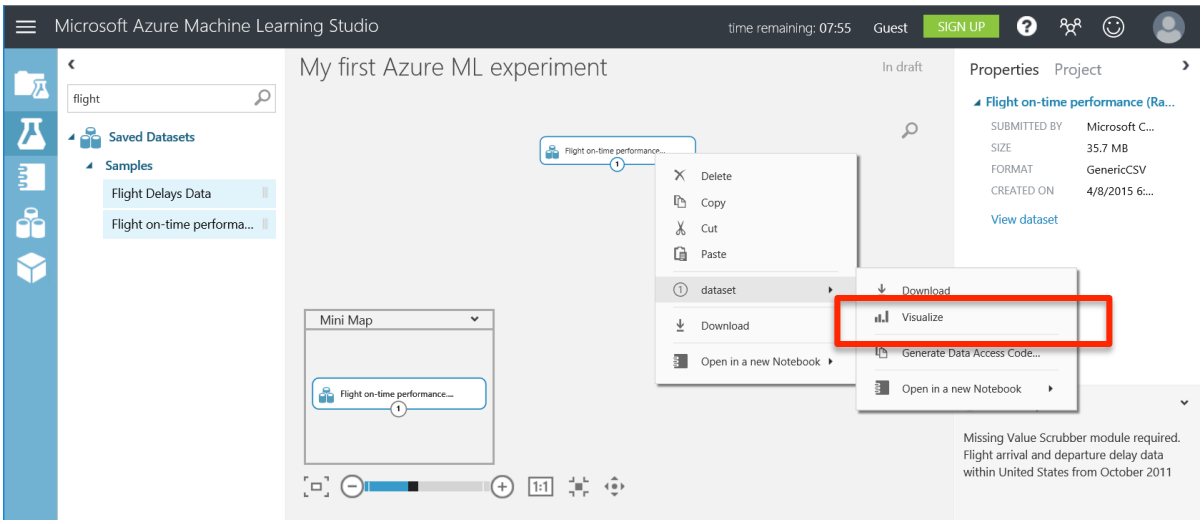
15. To search for flight data, type “flight” into the search bar.



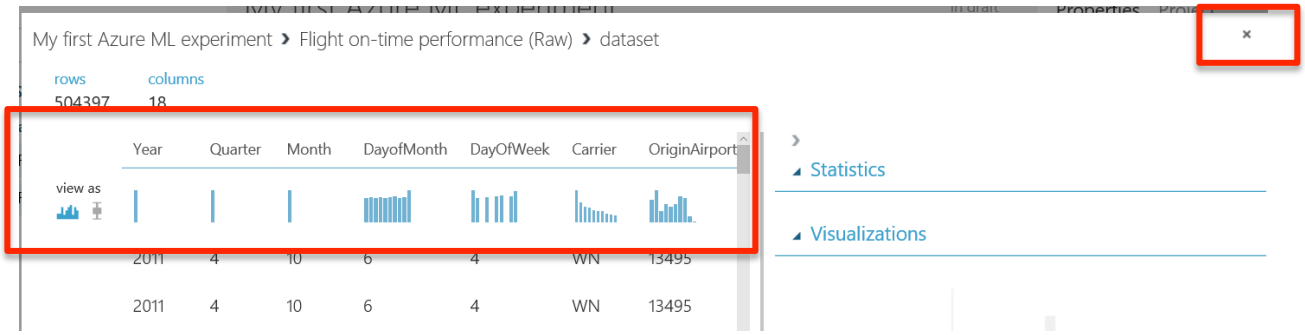
16. To import data, drag the **Flight on-time performance** dataset to the workspace.



17. To review data, right-click on the dataset in your workspace and select **dataset > visualize** from the pop-up menu.



Note: Notice the graphs or charts at the top of each data column. Explore the dataset by clicking on different columns. It's essential in Machine Learning to be familiar with your dataset and visualizing your dataset is a great first step. This dataset provides information about flights and whether they arrived on time. We are going to use Machine Learning to use this data to create a model that predicts whether a given flight will be late.



Note: In a real-world data science experiment, it is likely going to be necessary to data wrangle or clean dirty data. For this example, the data set is clean enough for our use.

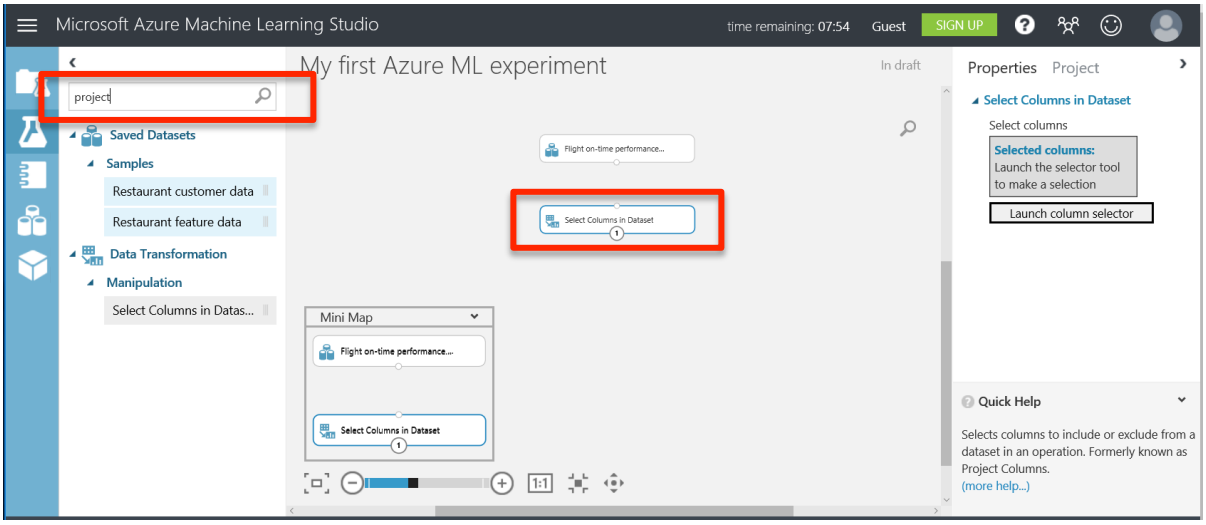
You can find the column definitions for this data on the US Department of Transportation site, http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time

18. Close the data visualization window, by clicking on the "**X**" in the top right corner of the window.

Specify Columns to Use

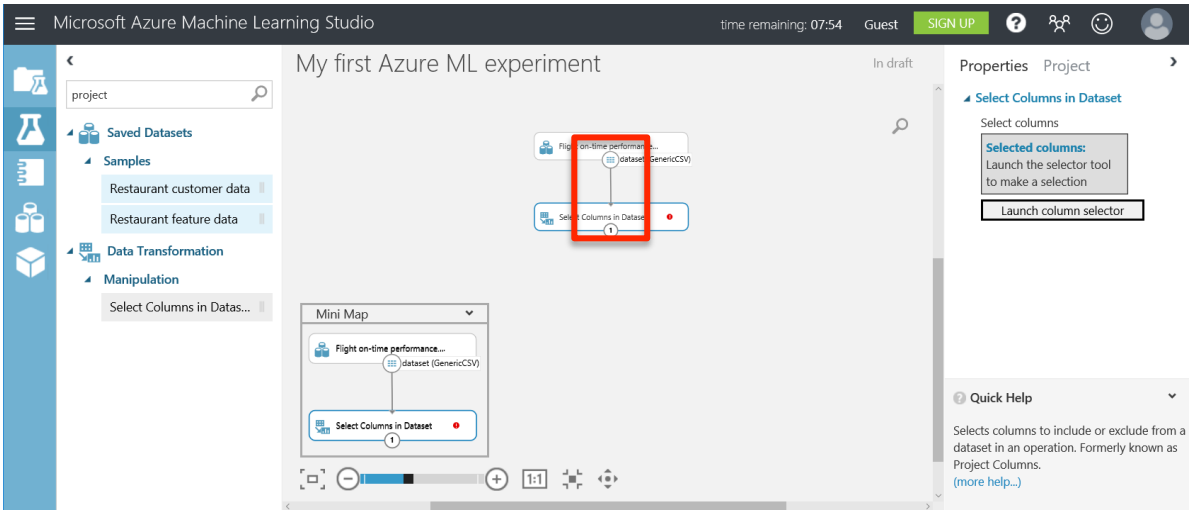
You need to review the data in the dataset and decide which columns represent data that you think will affect whether or not a flight is delayed. You also need to select the column that you want to predict. In this case, we are going to predict the value of ArrDel15. This is a binary state, 0/1 that indicates whether a flight arrival was delayed by more than 15 minutes.

19. To add the “Select Columns in Dataset” **manipulation**, type "**project**" into the search bar and drag the **Select Columns in Dataset** manipulation to the workspace.

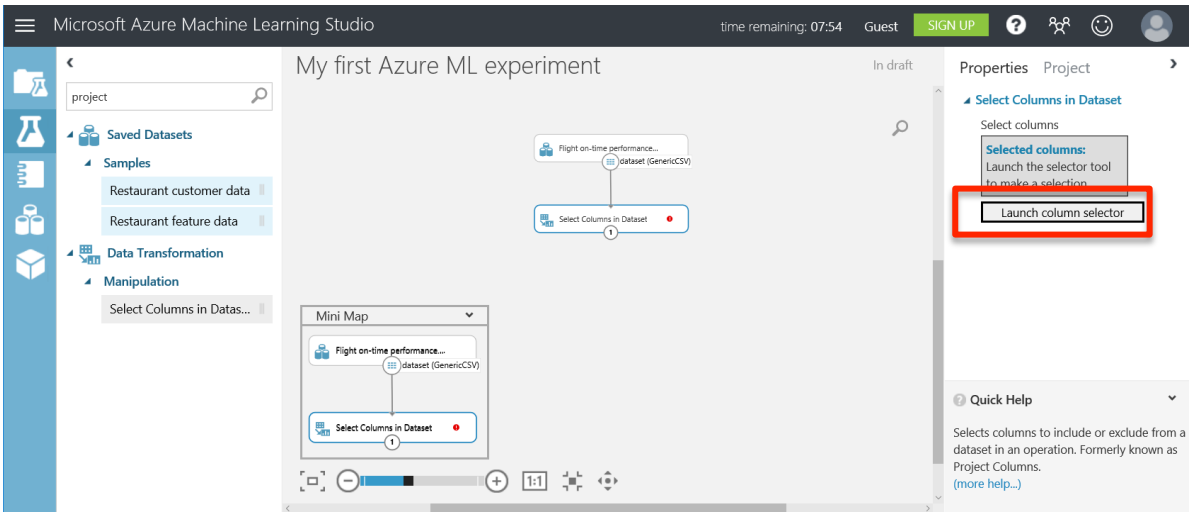


This manipulation enables you to specify which columns in the data set you think are significant to the prediction.

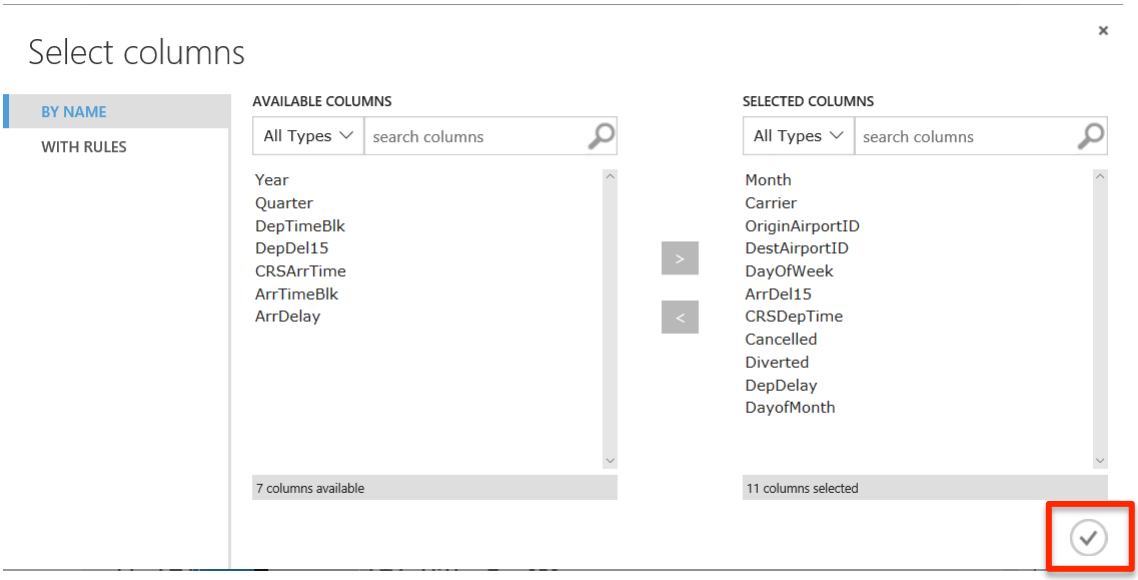
20. Connect the tasks - **Flight on-time performance** task and the **Select Columns in Dataset** task by clicking on the lower center dot and dragging it down to connect with the top center dot of the **Select Columns in Dataset** task.



21. Click on the Select Columns in Dataset module on the far right of your screen and select **Launch column selector**.



22. Select the columns you think will affect whether or not a flight is delayed as well as the column we want to predict. Select **Month, Carrier, OriginAirportID, DestAirportID, DayofWeek, ArrDel15, CRSDepTime, Cancelled, Diverted, DepDelay, DayofMonth.**



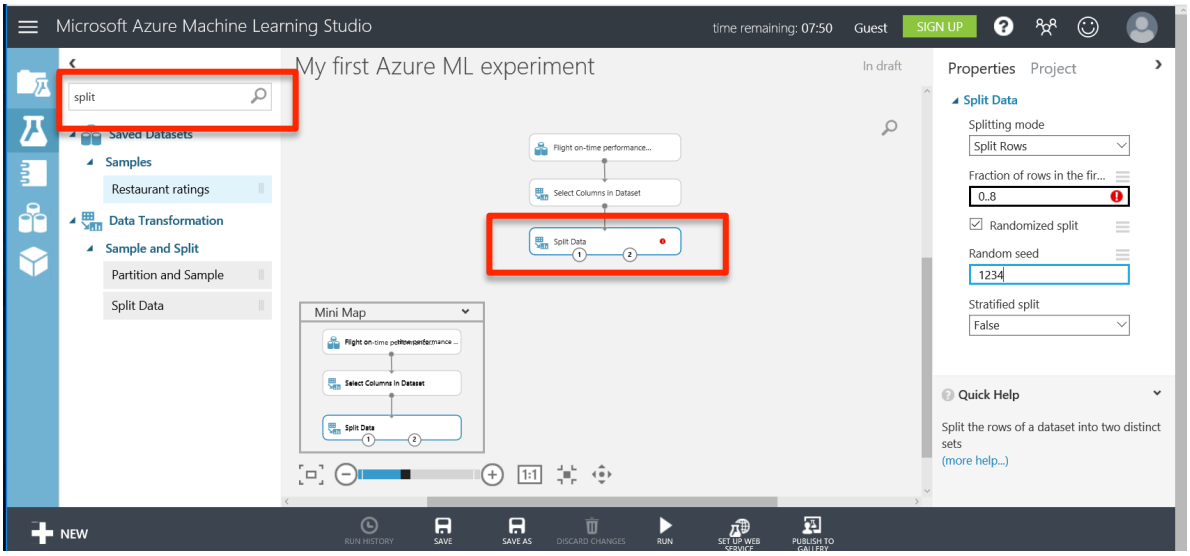
23. Complete the **Column Selection** process by selecting the checkbox in the lower right of the **Select columns** window.

Split the data into a training and test set

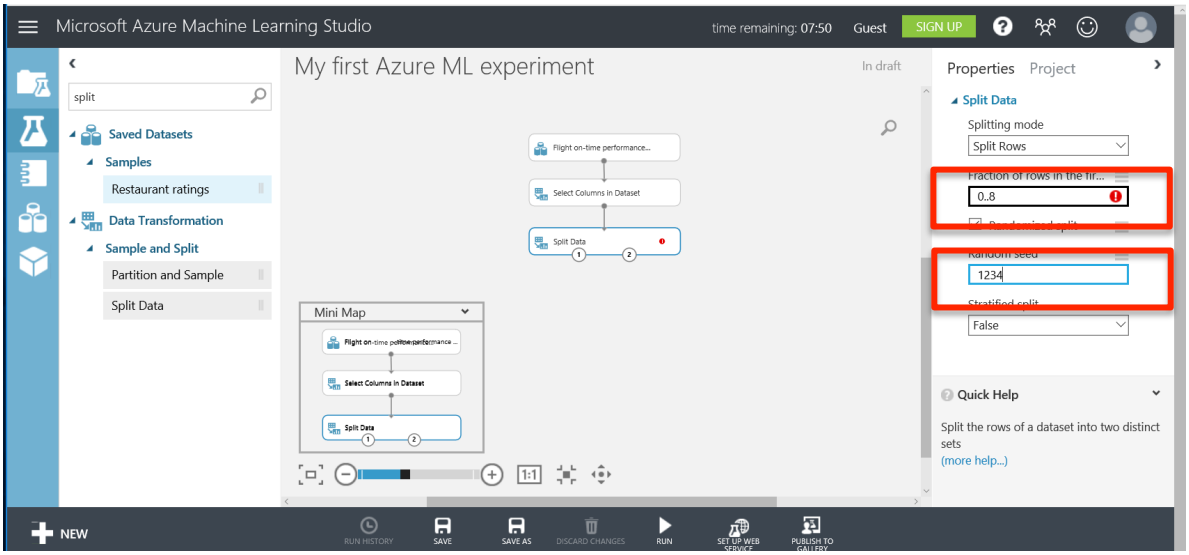
The Split Data task allows us to divide up our data. We need some data to try and find patterns to make predictions, and other data to test if the model created successfully works.

Traditionally, you will split the data 80/20 or 70/30. For today's workflow, everyone will use an 80/20 split. That means 80% of the data will be used to train the model and 20% will be used to test the accuracy of the model we develop.

24. Add the **Split Data Task** by typing "**split**" into the search bar and drag **Split Data** task to the workspace. Connect the output of the **Select Columns in Dataset** task to the input of the **Split Data** task (using the same process by which we connected the Flight Data to the Select Columns modules).



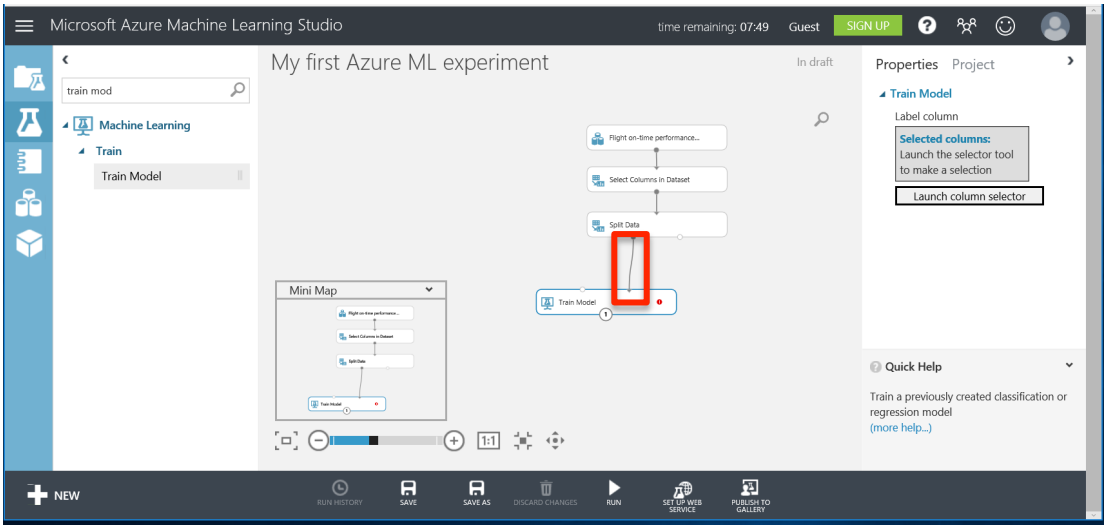
25. Next, we'll split our input data, by clicking on the **Split Data** task to bring up the **Properties Pane**. Specify 8 as the fraction of rows. Change random seed to 1234.



Train the model

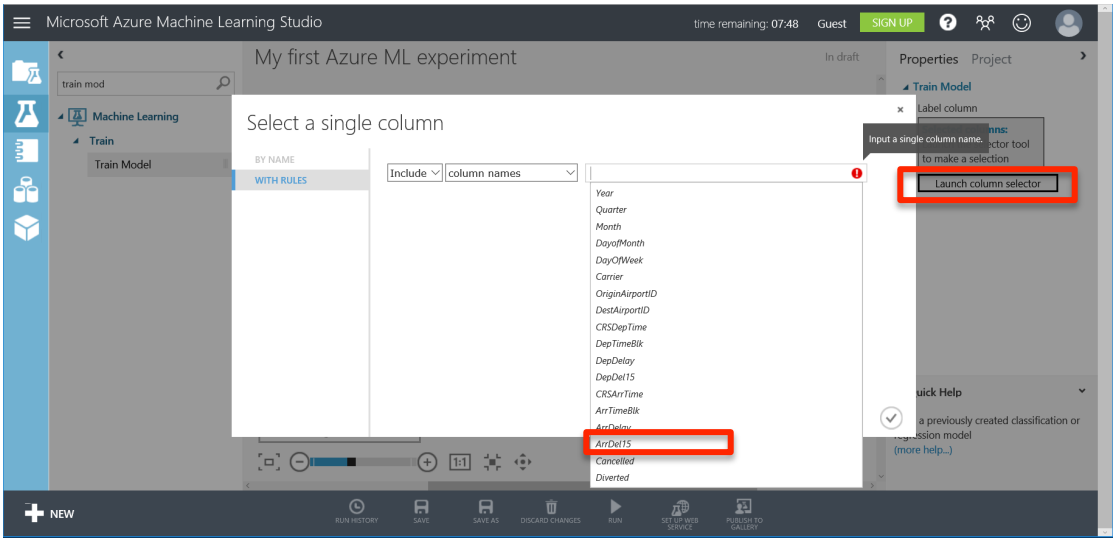
Next, we need to identify which data is to be predicted. In our case, we are predicting the value of the column **ArrDel15** that indicates if a flight arrival time was delayed by more than 15 minutes.

26. Type "**train model**" into the search bar. Drag the **Train Model** task to the workspace.



27. Hovering over the input and output dots will reveal what each input/output represents. Connect the first output, Results Dataset1, (the circle on the left) of the **Split Data** task to the right input of the **Train Model** task. This will take 80% of our data and use it to train/teach our model to make predictions.

28. Identify Predicted Value, by clicking on the Train Model task.
29. In the **Properties** window, select **Launch Column Selector**.
30. Select the column ArrDel15 by typing "**ArrDel15**" in to the text box (a smart filter of columns will appear). Click the checkbox in the lower right corner of the widow to complete the operation.



Select an algorithm

If you are a data scientist who creates your own algorithms, you could now import your own R code to analyze the patterns. But, Azure ML provides a number of standard algorithms, which are available for your use.

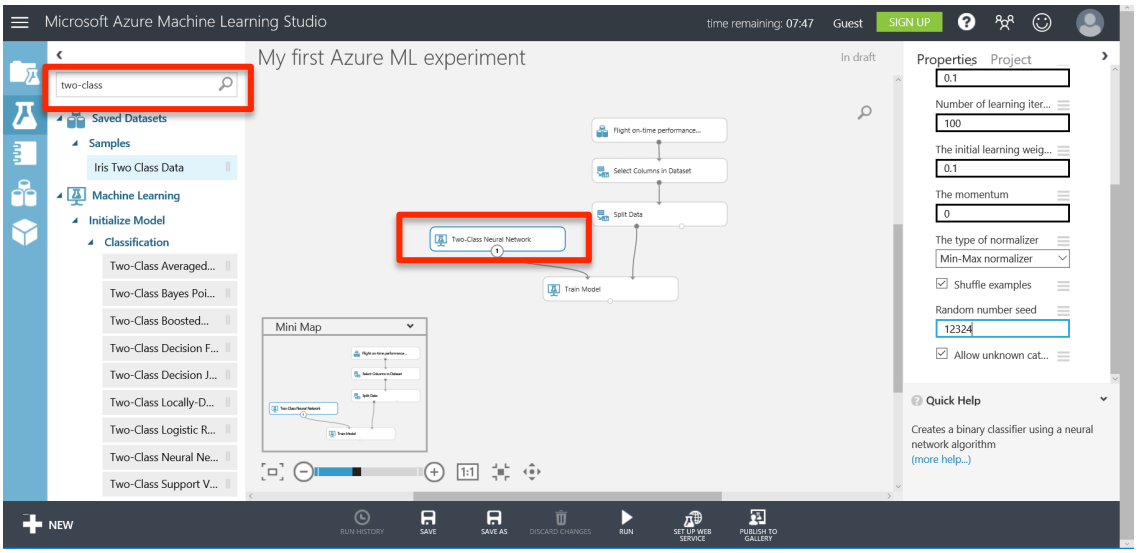
Selecting an algorithm is easier when using the Azure ML Cheat Sheet, https://aka.ms/azuremlcheatsheet, has been created. By narrowing the type of problem you are trying to solve, you can find the algorithms that will most likely generate a good model.

Today we are doing binary classification (also known as two-class classification). Using the cheat sheet, we can narrow our selection to a standard algorithm called Two-Class Neural Network.

As you can see, there are multiple two-class algorithms that we can choose from so we may want to try different ones out as we refine our model. Swapping out or even comparing two algorithms is made easy with Azure Machine Learning.

31. Type “**two-class**” into the search bar.
32. Select the **Two-Class Neural Network** and drag it to the workspace.

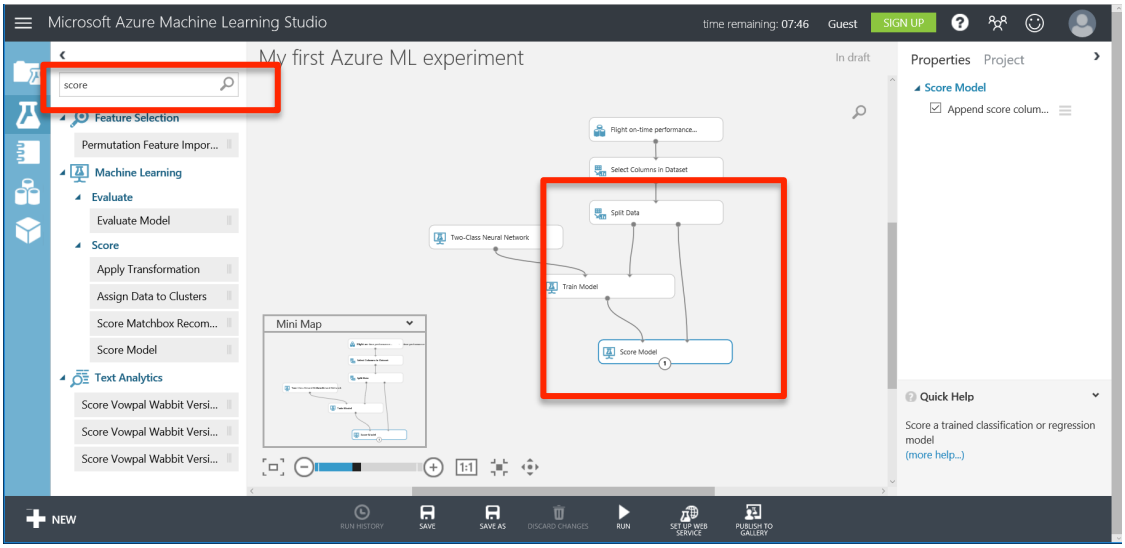
33. Connect the output of the **Two-Class Neural Network** task to the left input of the **Train Model** task.



Score the Model

After the model is trained, it is evaluated to determine how well it predicts delayed flights, so the model is scored by testing it against the Test Data (the remaining 20% of the data).

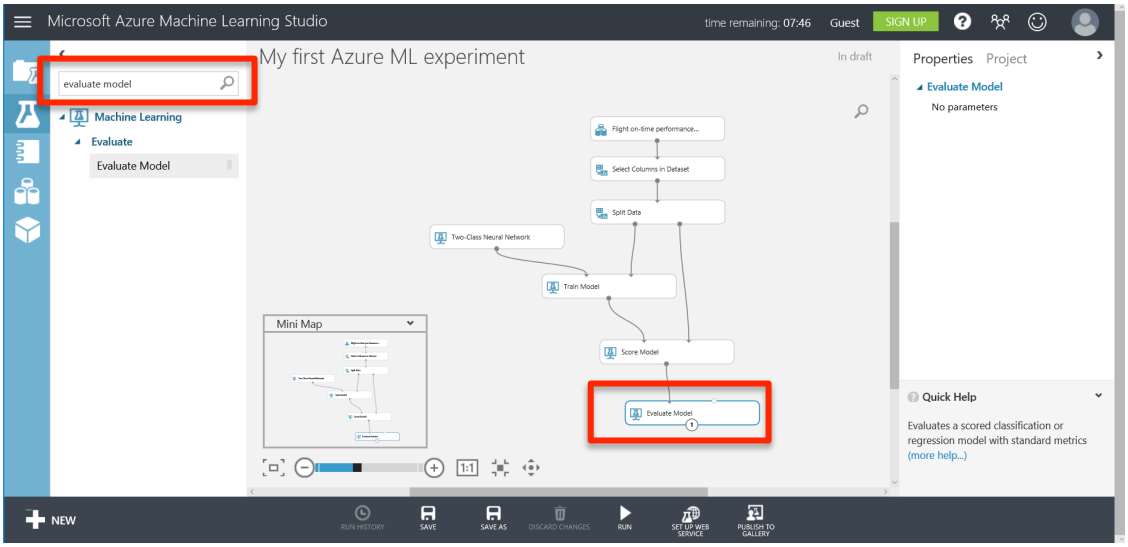
34. Type "**score**" into the search bar and drag the **Score Model** task to the workspace.
35. Connect the output of **Train Model** to the **left input** of the **Score Model task**.
36. Connect the Test Data, the **right output** of the **Split Data** task to the **right input** of the **Score Model** task as shown in the following screenshot. The output of this task is a scored dataset.



Evaluate Model

Next, the model is evaluated to determine its accuracy. This is done by evaluating the trained model using the test data.

37. Determine accuracy of model: Type "**evaluate**" into the search bar and drag the **Evaluate Model** task to the bottom of the workspace.
38. Connect the output of the **Score Model** task to the **left input** of the **Evaluate Model** task. The other input and output of the Evaluate Model task are not connected at this time.

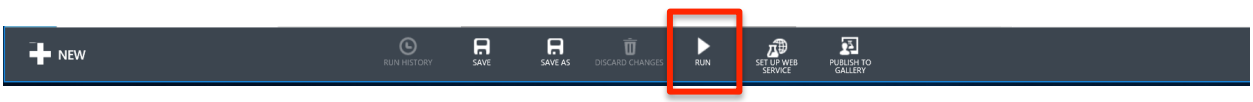


You are now ready to run your experiment!

Run Experiment

39. Next, select **Run** on the bottom toolbar. You will see green check marks appear on each task as it completes. The data is flowing through your Machine Learning Workflow, starting with data selection, being trained against the model, and finally being evaluated.

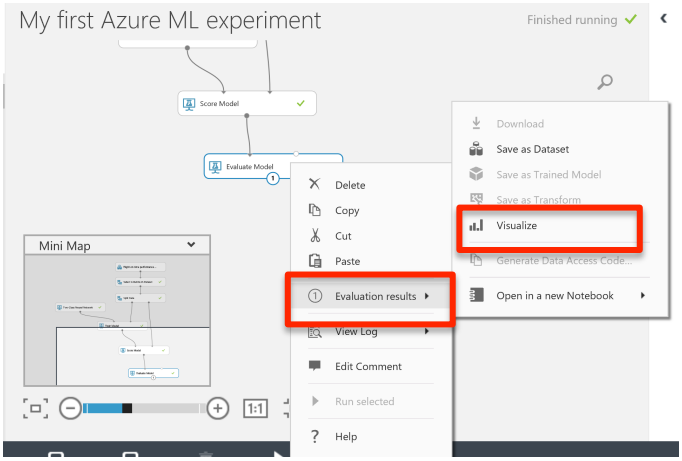
This process can take several minutes. When there is a green check mark on the **Evaluate Model** task the process is complete.



Post Run: Evaluate Model

To refine your model, it is usually necessary to evaluate the model, improve it, re-run it, and repeat.

40. Evaluate the model: When the entire experiment is completed, right-click on the **Evaluate Model** task and select "**Evaluation results > Visualize**" to see how well the model predicted delayed flights.

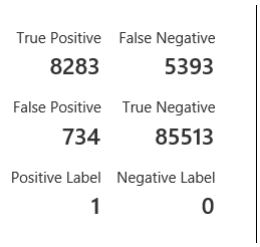


Interpreting Results

The first run of a model is a baseline and is considered a first step.

One useful piece of the evaluation results is the first graph the **True Positive Rate versus False Positive Rate** graph. This graph is a representation of the Area Under the Curve. A 45-degree flat line indicates guessing randomly. A more accurate model than random guessing looks like the image below, our current model.

Model Accuracy



Scroll down to see the accuracy. You can also see the number of false and true positive and negative predictions.

- **true positives** are how often your model correctly predicted a flight would be late

- **false positives** are how often your model predicted a flight would be late, when the flight was on time (your model predicted incorrectly)

- **true negatives** indicate how often your model correctly predicted a flight would be on time (arrDel15 is false)

- **false negatives** indicate how often your model predicted a flight would be on time, when in fact it was delayed (your model predicted incorrectly)

You want higher values for true positives and true negatives, you want low values for false positives and false negatives.

From this run of the model, there were only a few false positives, which is good. There are a few false negatives which can be considered for future work. There were also a good number of true positives and true negatives, which indicates the model predicted those correctly and that this is a very solid attempt at prediction. If planned for production, we would iterate by changing the algorithm, or choosing additional data inputs, or cleaning up more of the data.

Congratulations! You have created a successful training experiment!

Next, we are going to continue to **Step 44**. You will use an existing web service with MicroStrategy Desktop.

Note: This section of the workshop, up to Step 44 is for informational purposes alone.

We include instructions below on how to publish your own web service for informational purposes. These steps will demonstrate how to publish your own training experiment as a web service so MicroStrategy Desktop can use the model to make predictions. MicroStrategy Desktop will then be able to provide inputs to the model, which result in predictions and related data.

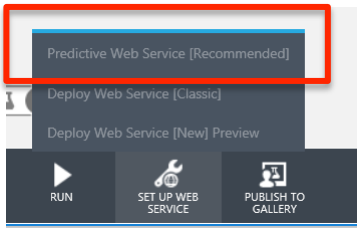
Publish Model as a Web Service

We have 2 additional steps to publish the training experiment on the web:

- **Convert training experiment to a predictive experiment:** This step defines the inputs/outputs and removes unneeded training info
- **Deploy it as a Web Service:** This step creates an API key and a URL for the model

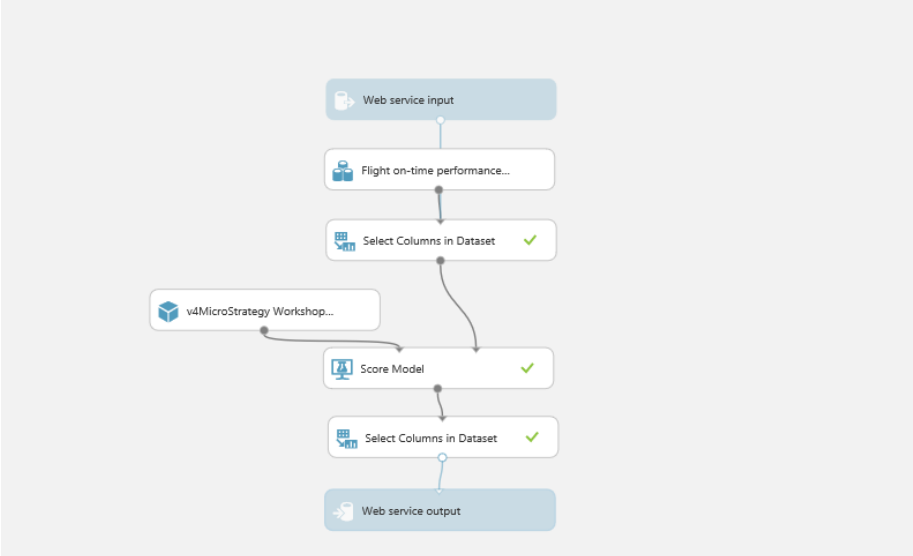
Create a predictive experiment

Creating a predictive experiment from the training experiment is the first step to create a web service from a training experiment. In this experiment, this is done automatically by the system which will handle the outputs as well as include the trained model into the predictive model. In more complicated experiments, extra items needed to train the model will be removed.

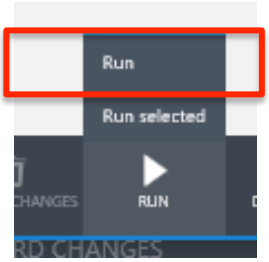


41. Click on **Set Up Web Service**
42. Click **Predictive Web Service**

Your model will be automatically modified to add web service inputs/outputs and the trained model as shown below. You will see a tab with the words Predictive experiment at the top.

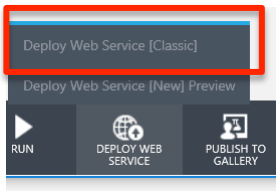


43. Click **Run** to run the model.



Publish a Web Service

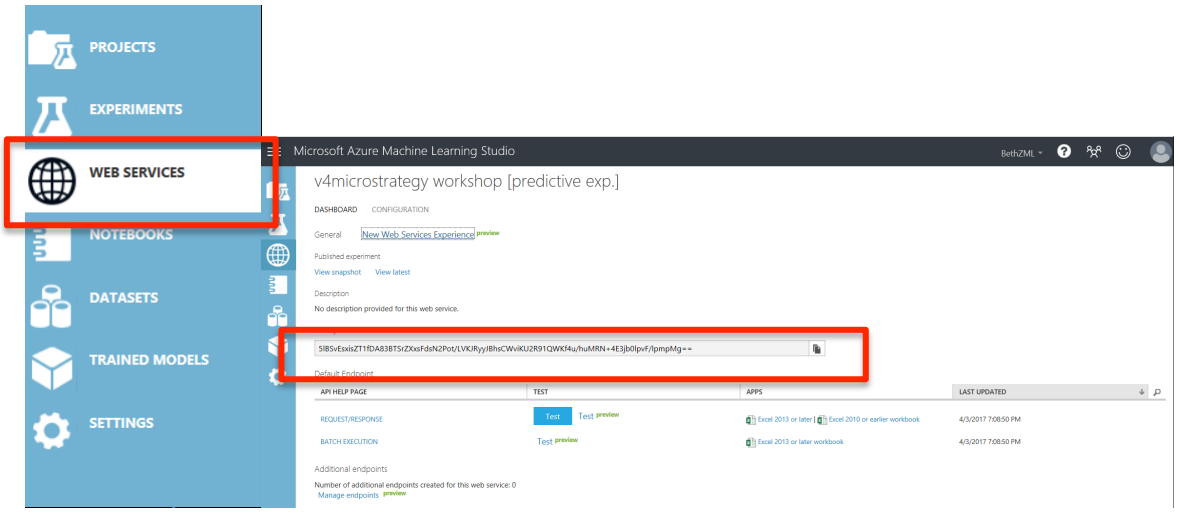
44. Next, click **Deploy Web Service [Classic]**. You now have an active Web Service ready for consumption.



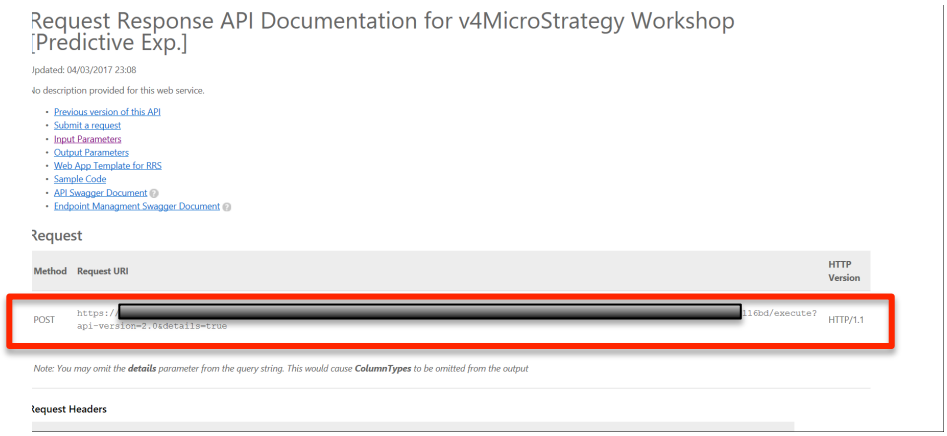
To consume the model in MicroStrategy, you need both the API Key and URL.

Note: In this workshop, we will provide these inputs so you do not need to capture them yourself.

API Key



URL

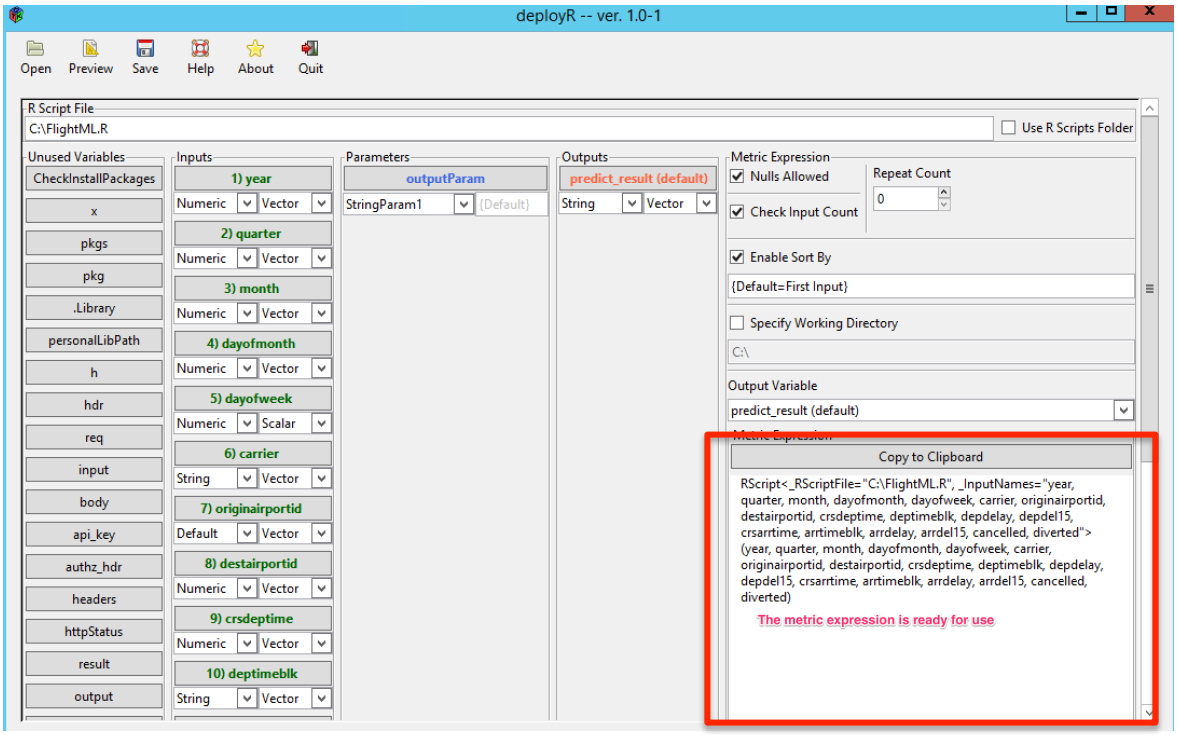


Azure ML automatically creates R code for use in MicroStrategy Desktop. All that's required are minor modifications.

Note: In this workshop, you are not required to do this step. We will provide you with the appropriate R script.

As you can see, it generates C#, Python, and R.

Once you have the R script, you can use the deploy() function in the R package "MicroStrategyR" to obtain the metric expression – as seen in the bottom right of this editor.



Note: In this workshop, you are not required to do this step and we will provide you with the metric expression.

Next, we'll use MicroStrategy Desktop to easily provide inputs to the model and create dashboards. The API key and URL are the way that the Desktop and Azure ML model communicate.

Visualizing Data with MicroStrategy Desktop

Downloading the R script and data file

45. If you have not done so already, please download the R script file "**FlightML.R**" and place it in the C:/ for Windows machines and anywhere you can access it from for Macs. The location of this file will be relevant on Step 50. Also download the dashboard file "**Microsoft Azure Workshop.mstr**" from the supporting files folder.

TIP: *Microsoft R Open, formerly known as Revolution R Open (RRO), is the enhanced distribution of R from Microsoft Corporation. It is a complete open source platform for statistical analysis and data science.*

The current version, Microsoft R Open 3.3.2, is based on (and 100% compatible with) R-3.3.2, the most widely used statistics software in the world, and is therefore fully compatible with all packages, scripts, and applications that work with that version of R. It includes additional capabilities for improved performance, reproducibility, as well as support for Windows and Linux-based platforms.

Like R, Microsoft R Open is open source and free to download, use, and share. It is available from https://mran.microsoft.com/open/

Launch MicroStrategy Desktop

46. Launch **MicroStrategy Desktop.**

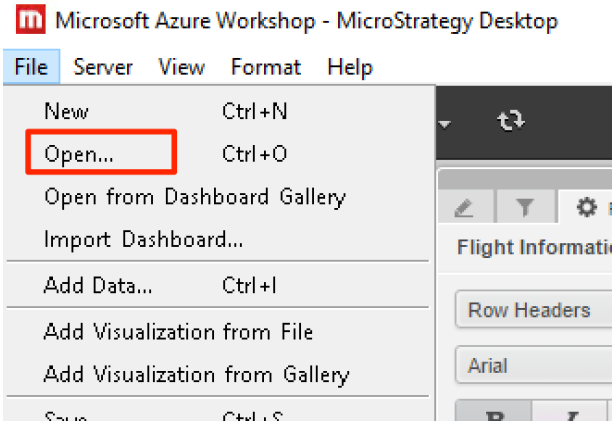
47. If this is the first time you are accessing MicroStrategy Desktop, the **Quick Tips** guided product tour will be displayed. Click **Hide Quick Tips** to remove the notes.

Open Existing Dashboard

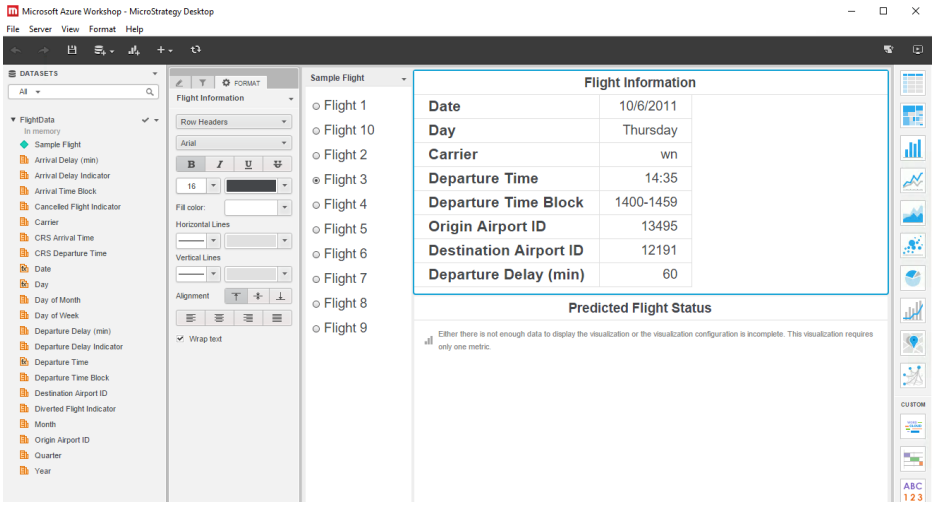
To focus on the integration with Azure Machine Learning, we have prepared a dashboard for you that is available in the supporting files folder. You will start this section by opening this pre-built dashboard by clicking on the “Microsoft Azure Workshop.mstr” file. This dashboard contains data for several flights and will let you make a prediction using MicroStrategy Desktop to call an Azure Machine Learning Model.

TIP: *To learn how to build a MicroStrategy dashboard by importing your own data, attend the Introduction to Data Discovery Workshop*

48. From the "**File**" menu, select **Open** to find and select the file "**Microsoft Azure Workshop.mstr**"



49. The dashboard will now be displayed. It contains a filter and a table with flight information. Switch flights by using the filter on the left. Selecting a different flight will display additional information for that flight.

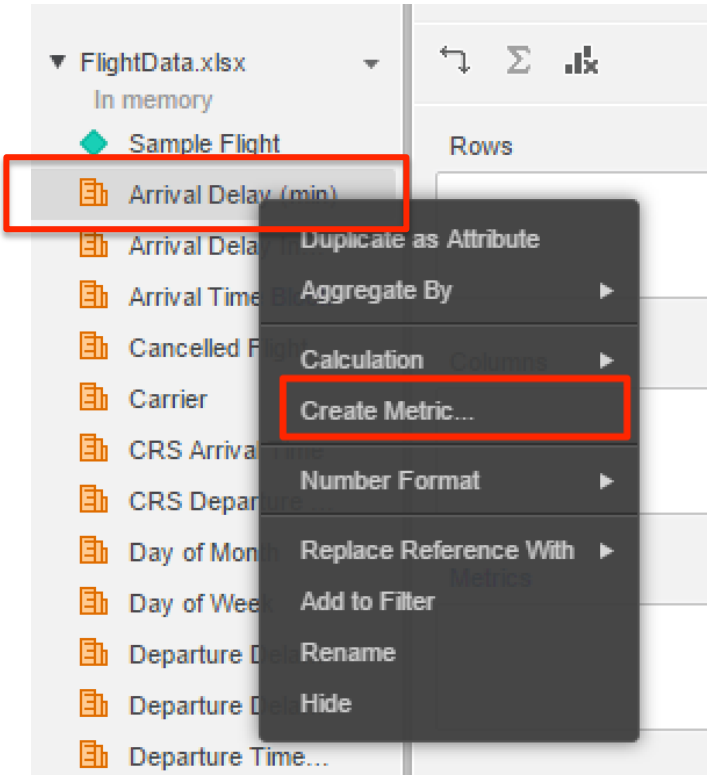


Note: The predicted flight status is currently empty and we will call an Azure Machine Learning Model to generate that next.

Make Predictions on New Data

MicroStrategy can visualize R metrics. In order to do that, we will need to create a derived metric that uses the downloaded R script to call the Azure Machine Learning Model which will then return the predicted flight status.

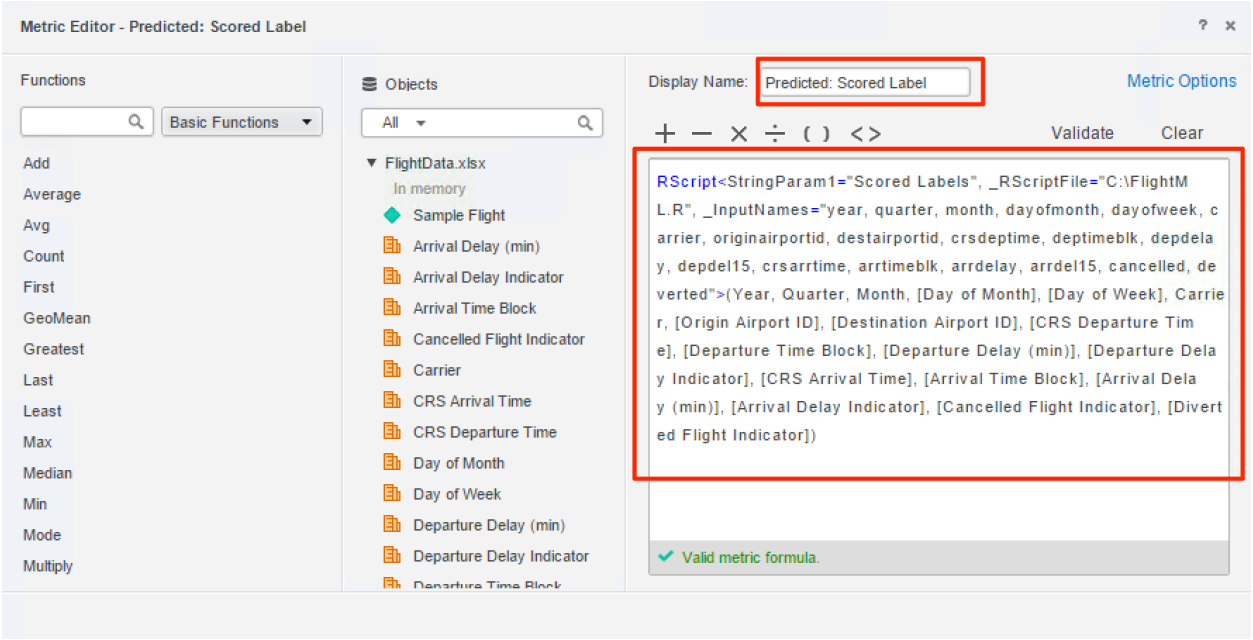
50. To create a metric for displaying the predicted flight status right-click on any metric (e.g. "**Arrival Delay (min)"),** and select "**Create Metric**" from the menu.



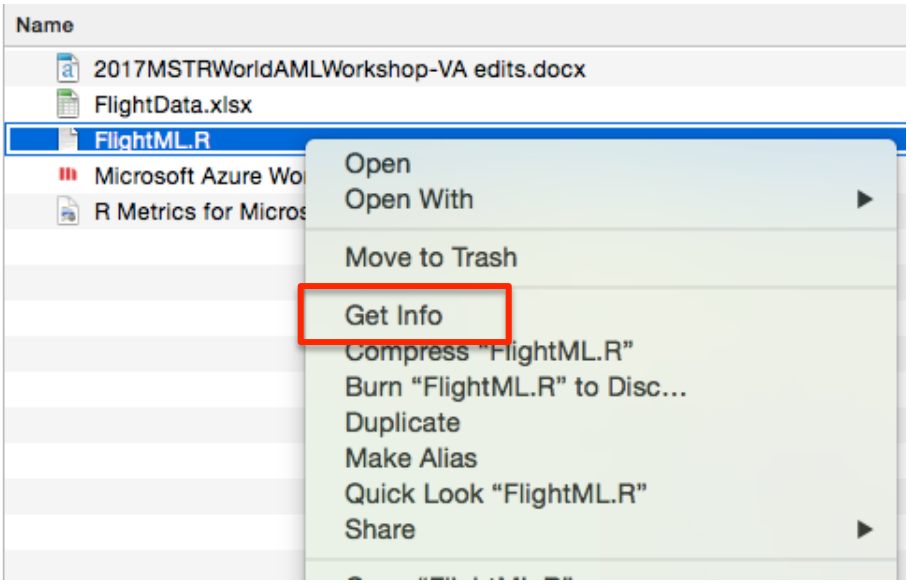
51. In the **Metric Editor,** copy and paste the metric expression below. This expression includes a script to invoke an Azure Machine Learning Model and defines the data Input Names that will be provided to the model. (If you store the downloaded R script file in a different location or you are using a Mac, please modify the file path so that it can find the R script where you saved it.) For Mac users, see tip on next page.

```
RScript<StringParam1="Scored Labels", _RScriptFile="C:\FlightML.R", _InputNames="year, quarter, month, dayofmonth, dayofweek, carrier, originairportid, destairportid, crsdeptime, deptimeblk, depdelay, depdel15, crsarrtime, arrtimeblk, arrdelay, arrdel15, cancelled, diverted">(Year, Quarter, Month, [Day of Month], [Day of Week], Carrier, [Origin Airport ID], [Destination Airport ID], [CRS Departure Time], [Departure Time Block], [Departure Delay (min)], [Departure Delay Indicator], [CRS Arrival Time], [Arrival Time Block], [Arrival Delay (min)], [Arrival Delay Indicator], [Cancelled Flight Indicator], [Diverted Flight Indicator])
```

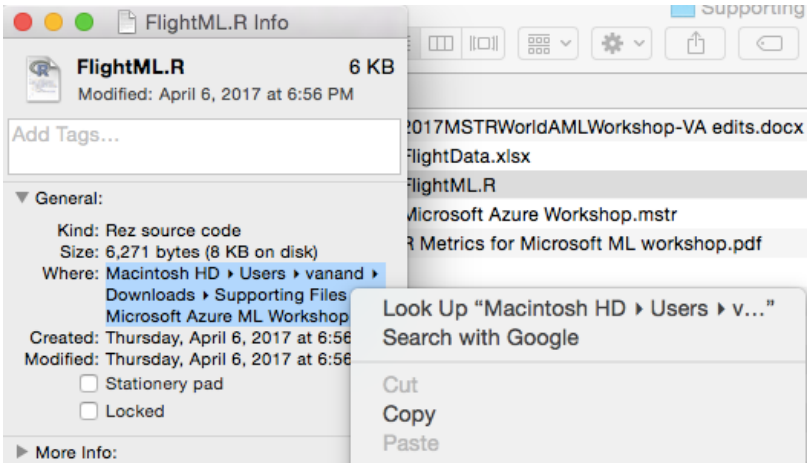
Note: This script can be copied from **R Metrics for Microsoft ML workshop.pdf** that is available in the supporting files folder.



55. Don't forget to change the lication of the R script. Windows users can use the location of the file as shown in the screenshot.
56. Mac users can get the location of the file, by right-clicking on the R script and clicking on "**Get Info**"



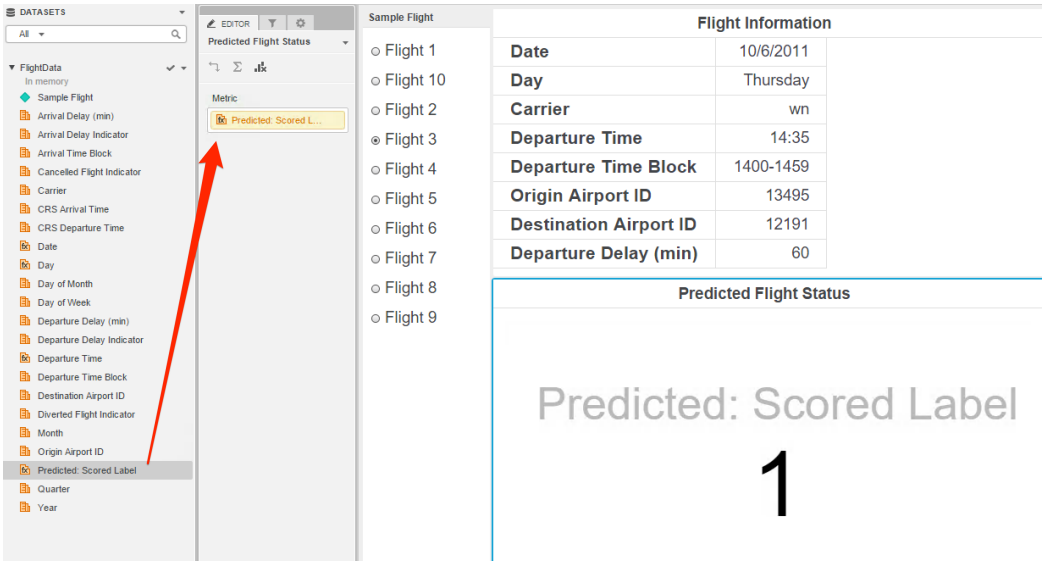
Tip: Mac users, can copy he location by selecting the full directory location as below:



57. Replace the location of the R file to point to the right location.
58. Rename the metric by changing the **Display Name or Metric Name** to "Predicted: Scored Labels".
59. Validate the metric formula by clicking on **Validate**.

    A checkmark under the script box with the phrase "Valid metric formula" indicates success.
60. Click **Save** button in the Metric Editor to save the metric.

    Now, let's take a looked at the predicted result.
61. Click on the empty visualization called "Predicted Flight Status" that is beneath the grid (you will see a blue border around it).
62. Drag and drop the newly created metric "**Predicted: Scored Label**" to the **Metric drop** zone of the **Editor Panel**.



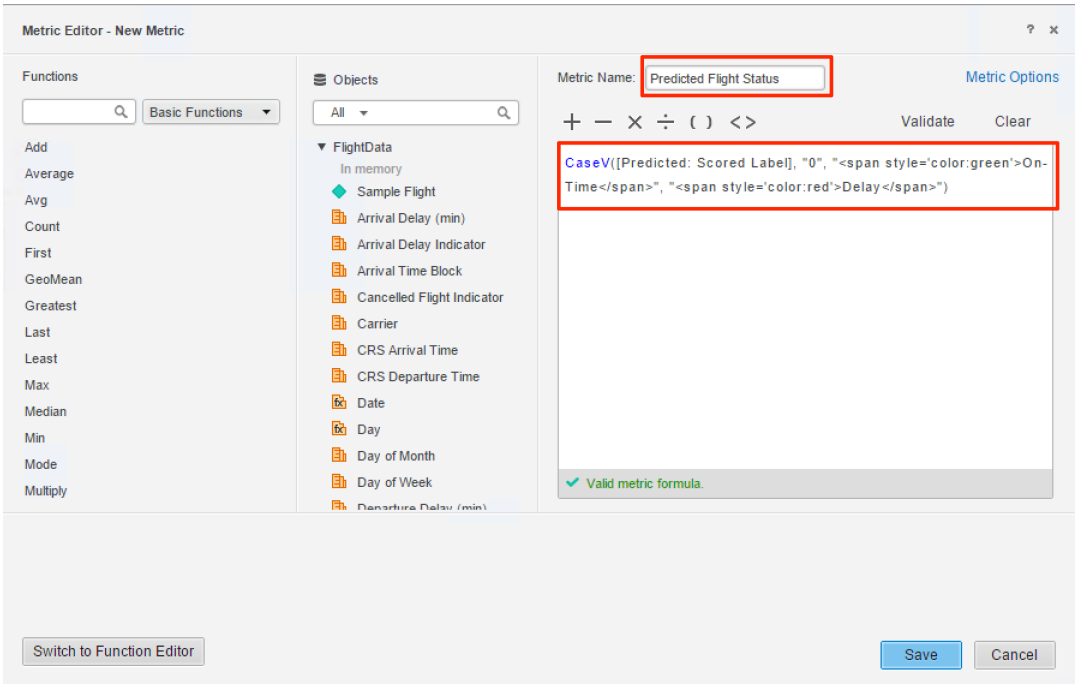
63. Switch between flights by using the filter on the left side of the dashboard. When **Predicted Flight Status** is 1, it means the flight is predicted to be delayed. When it is 0, it means the flight is predicted to be on-time.
    Congratulations! You've supplied flight data to the Azure Machine Learning model using MicroStrategy Desktop and have successfully created your own flight predictions!

Next, we will change the dashboard from displaying 1 or 0 for the flight prediction to displaying our prediction in text form.

64. In the **Datasets panel** on the MicroStrategy Desktop interface, right-click on any metric (e.g. "Arrival Delay (min)"), and select "**Create Metric**" from the menu.
65. In the **Metric Editor**, copy and paste the metric expression below, and rename the metric "Predicted Flight Status".

```
CaseV([Predicted: Scored Label], "0", "<span style='color:green'>On-Time</span>", "<span style='color:red'>Delay</span>")
```

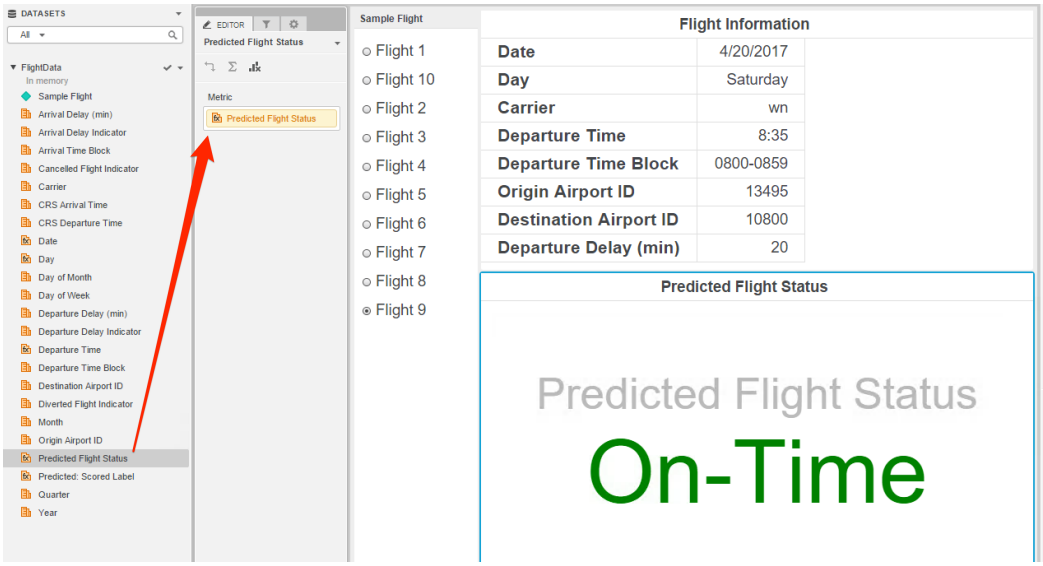
Note: This script can be copied from **R Metrics for Microsoft ML workshop.pdf** that is available in the supporting files folder.



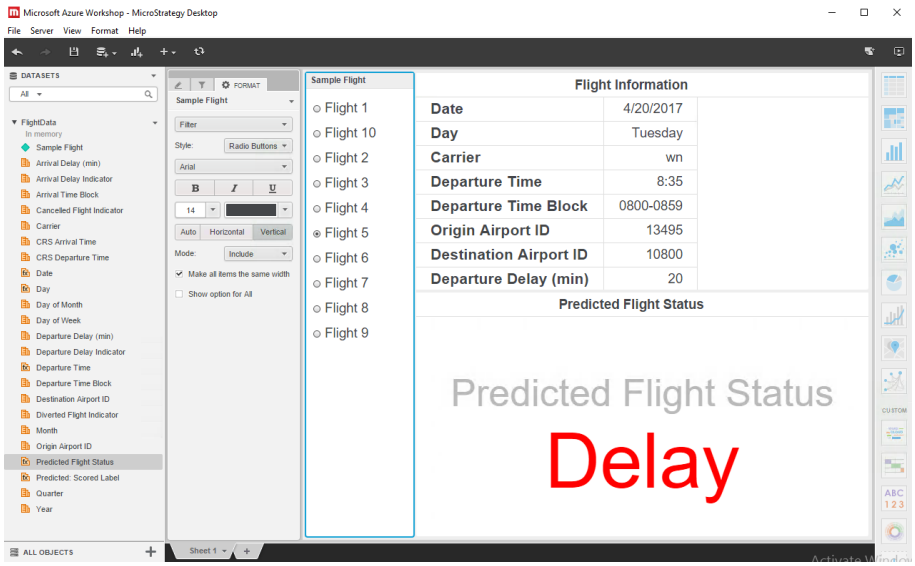
66. Validate the metric formula by clicking **Validate**. A checkmark under the script box with the phrase “Valid metric formula” indicates success.
67. Click the **Save** button.

Now we will use the “**Predicted Flight Status**” to replace the previous metric on the dashboard.

68. Select the visualization “**Predicted Flight Status**” beneath the grid (once selected, you will see blue border around it).
69. Drag “Predicted Flight Status” and place it on top of the old metric “Predicted: Scored Label” in the metric zone of the **Editor** panel.



70. Select different sample flights on the filter and you will see both on-time and delay predictions for flights. Here is what the dashboard looks like for flight 5:



Congratulations! You've learned how to use Machine Learning in Azure to make a flight prediction and visualized it with a dashboard on MicroStrategy Desktop.