

How to Build and Visualize your own Machine Learning prediction with Microsoft Azure and MicroStrategy

Introduction

Haven’t you always wondered about where machine learning could actually be used by you? In under one hour, you will learn the magic of Azure Machine Learning to gain insight from data that is easily consumed by MicroStrategy Desktop to quickly create useful dashboards.

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

MicroStrategy leveraged Azure Machine Learning, ML, to make effective predictions in their applications. In this workshop, we will use flight data to demonstrate how you can make a prediction using Azure ML, and to highlight areas where MicroStrategy leverages Azure ML to help their business and yours.

Overview

The main sections of this workshop are:

1. Prerequisites
   1. Azure Machine Learning only requires network connectivity to the internet, no sign-up, no account required
   2. MicroStrategy Desktop – Download and Install MicroStrategy Desktop
   3. R – Download and install R
   4. R Integration Pack – Download and install R Integration Pack
   5. Download data excel file and R script file to your C: drive or any other convenient place
2. Step-by-step Azure Machine Learning for Flight Prediction - how to use Machine Learning, ML, for predicting if a flight will be late
3. Step-by-step MicroStrategy Desktop – Integrate MicroStrategy with Azure ML to make predictions and easily create effective dashboards

**Pre-requisites:**

**Download data and R script files from Github**

* 1. Download the data excel “FlightData.xls” and R script file “FlightML.R” to your C: drive if you have Windows laptop or any other convenient place for Mac from:

<https://github.com/bethz/AzureML-FlightPrediction/tree/master/MicroStrategy>

Download and Install MicroStrategy Desktop

* 1. If you do not have MicroStrategy Desktop 10.7 installed on your laptop yet, you can download it from the below link for free: <https://www.microstrategy.com/us/desktop>
  2. Please select the version for corresponding to your laptop system (Window or Mac)



* 1. Go to your Downloads folder to find the zip file you just downloaded, unzip it and run installation, “MicroStrategyDesktop-64bit.exe” file for Windows and “MicroStrategy Desktop.pkg” for Mac.
  2. Follow the installation wizard and accept all default settings to finish the installation.

Download and install R on Windows.

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

Download and install the R Integration Pack

* 1. Go to the link: <http://rintegrationpack.codeplex.com/releases/view/630028>
  2. On the web page, locate and download the version for your laptop.

For Windows: 

For Mac: 

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

**Step-by-step Azure Machine Learning for Flight Prediction**

Here is an on-line version of the Workshop Manual:

<https://github.com/bethz/AzureML-FlightPrediction>/MicroStrategy/2017MSTRWorldAMLWorkshop.docx

A paper manual will be provided at the conference.

A simple example in Azure Machine Learning will be used to demonstrate concepts that can be leveraged for others uses. The example uses some 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 for the model to predict whether a flight will be late. We begin by accessing Azure ML to make a model.

# Azure Machine Learning for Flight Prediction

This workshop uses publicly available Flight Data to predict when a flight will be late. We begin by accessing Azure ML. Next steps:

[Access Azure ML](https://github.com/bethz/AzureML-FlightPrediction#1-access-azure-ml)  
[Create an Experiment](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#10-run-experiment)  
[Import, Review and Clean Data](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#3-import-review-and-clean-data)  
[Specify Columns to Use](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#4-specify-columns-to-use)  
[Split The Data Into A Training And Test Set](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#5-split-the-data-into-a-training-and-test-set)  
[Train the model](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#6-train-the-model)  
[Select Algorithm](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#7-select-algorithm)  
[Score the Model](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#8-score-the-model)  
[Evaluate Model](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#9-evaluate-model)  
[Run Experiment!](https://github.com/bethz/AzureML-FlightPrediction/blob/master/README.md#10-run-experiment)

## **Access Azure ML**

* 1. Set up access to Azure ML

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

\* [Guest Access](http://studio.azureml.net/home/anonymous) - <http://studio.azureml.net/home/anonymous>

This does not require an Azure subscription or a credit card. It is anonymous access. After 8 hours the workspace gets reset. This is a great option for evaluation and this workshop and is recommended for easy setup.

\* [Your Own Account](https://studio.azureml.net/Home) - <https://studio.azureml.net/Home>

Sign in and use a work, school or Microsoft account. Please do not take a tour now, explore the tour later.

There are two types of accounts:

\* Free Account ([Microsoft Account](https://signup.live.com/signup) required) - <https://signup.live.com/signup>

10 GB of Storage, R and Python Scripts and Web Service access

\* Enterprise Grade ([Azure Subscription](https://azure.microsoft.com/en-us/free/) required) - <https://azure.microsoft.com/en-us/free/>

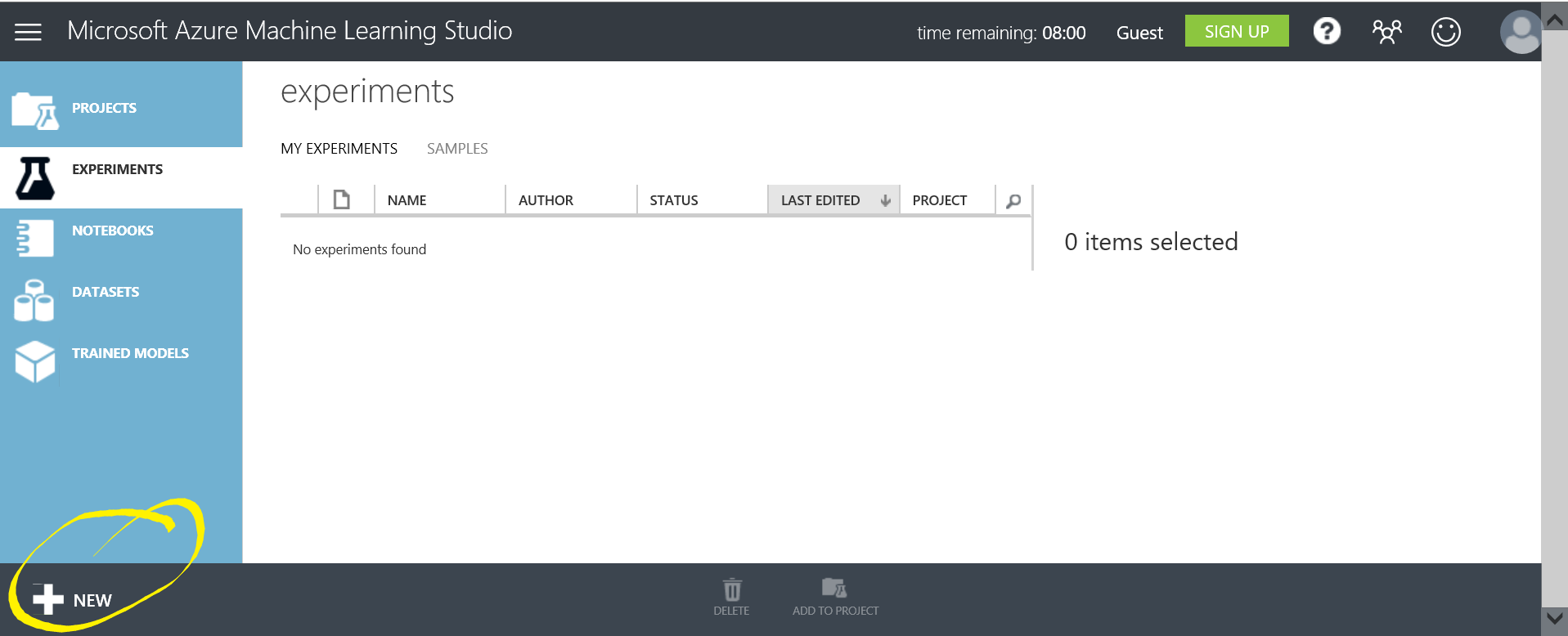
Full SLA, Bring your own Azure Storage, parallel graph execution, Elastic Web Service Endpoints

## **Create an Experiment**

Let's get started by making a new experiment.

* 1. Make a new experiment

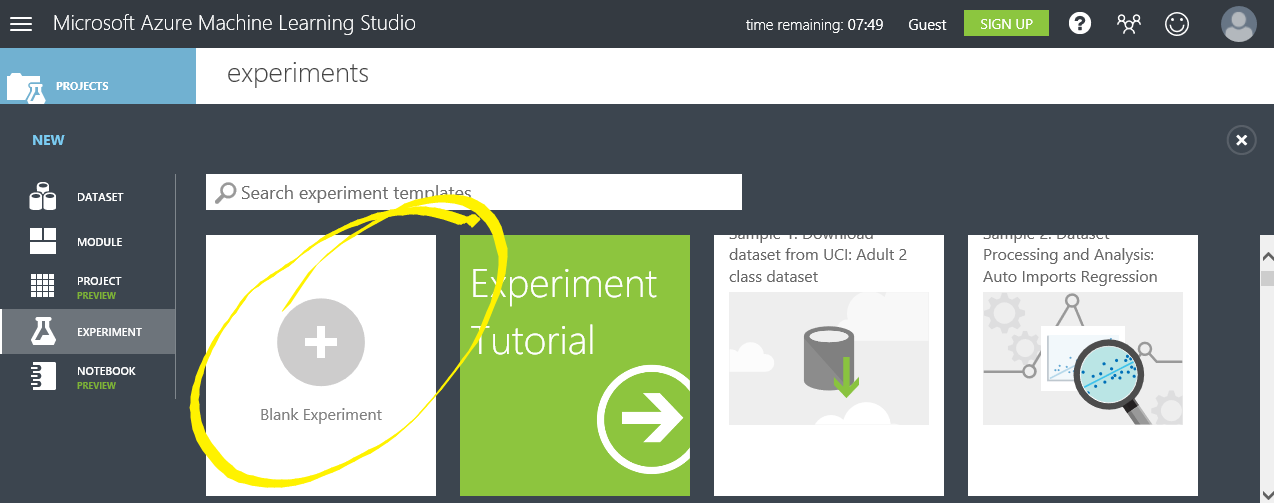
Select +New in the lower left corner.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/2ANewExperiment.png)



* 1. Select Blank Experiment

To the right of Experiment, you will see a tile with a plus sign and the words Blank Experiment. Select + Blank Experiment.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/2bBlankExperiment.png)



* 1. Give the experiment a title

By default, a title is created with a name like "***Experiment created on 9/24/2016***". Change the title to "My first Azure ML experiment" by editing the provided title.

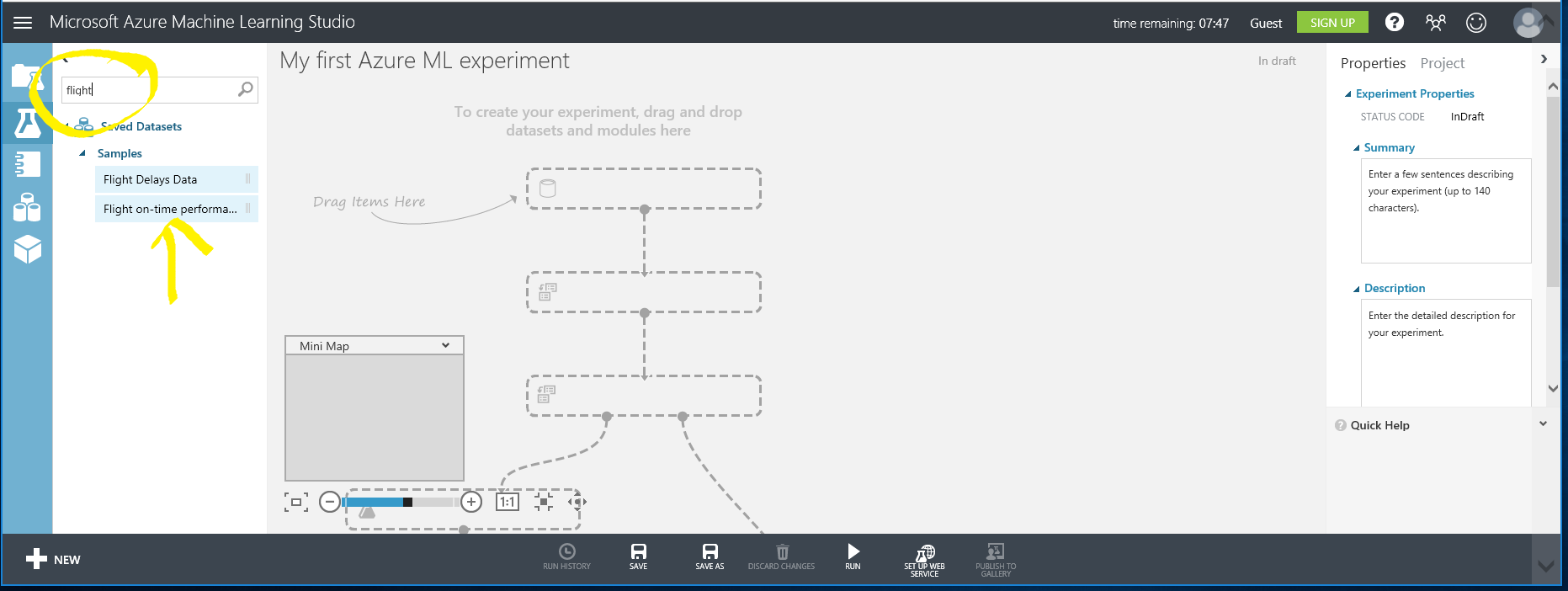
## <https://cloud.githubusercontent.com/assets/6098674/18649061/7e817444-7e8b-11e6-815d-2a7dd7b154ac.png>



## **Import, Review and Clean Data**

* 1. Search for flight data

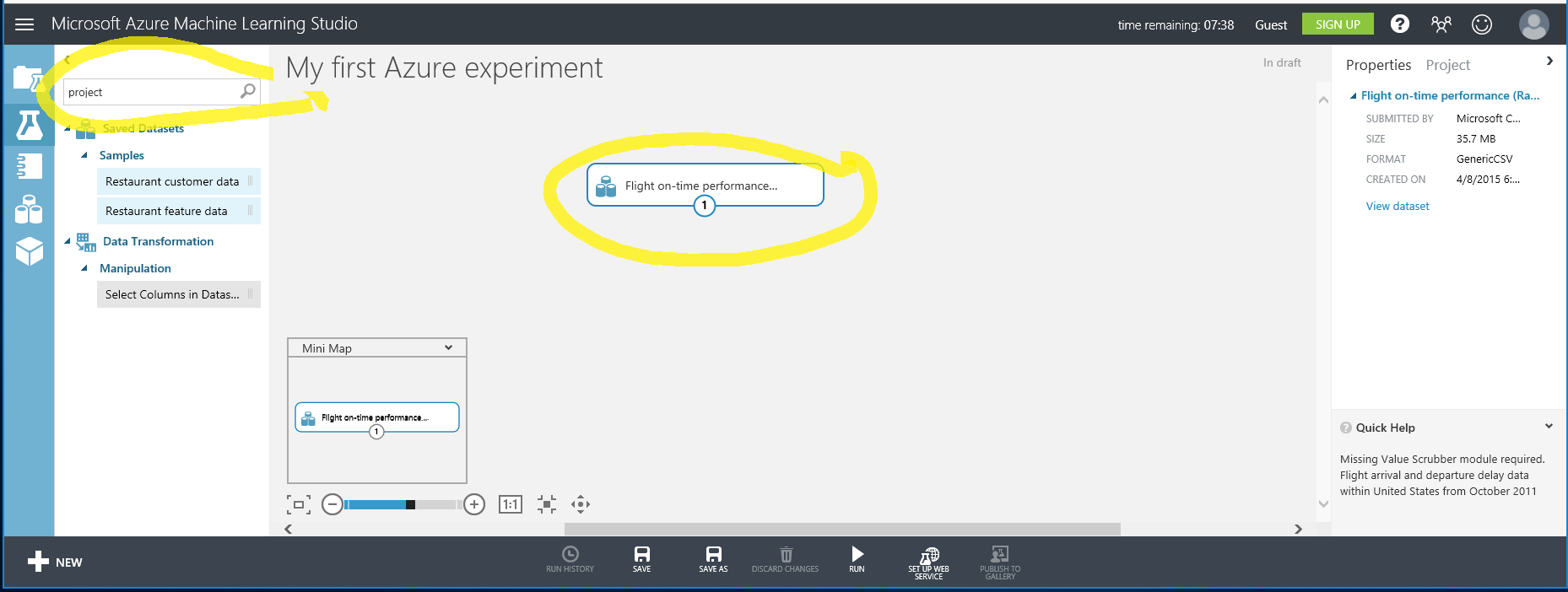
Type “flight” into the search bar.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/3A.png)



* 1. Import data

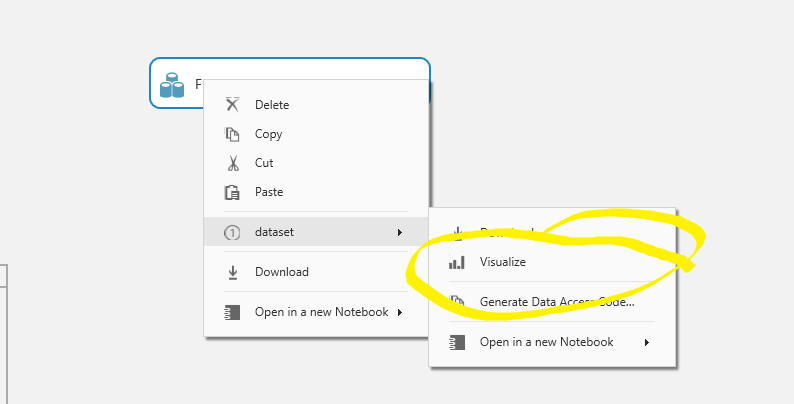
Drag the Flight on-time performance Dataset to the workspace as show in the image. This is one of many sample datasets built into Azure Machine Learning Studio designed to help you learn and explore the tool.

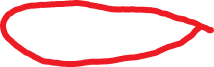
[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/3b.png)



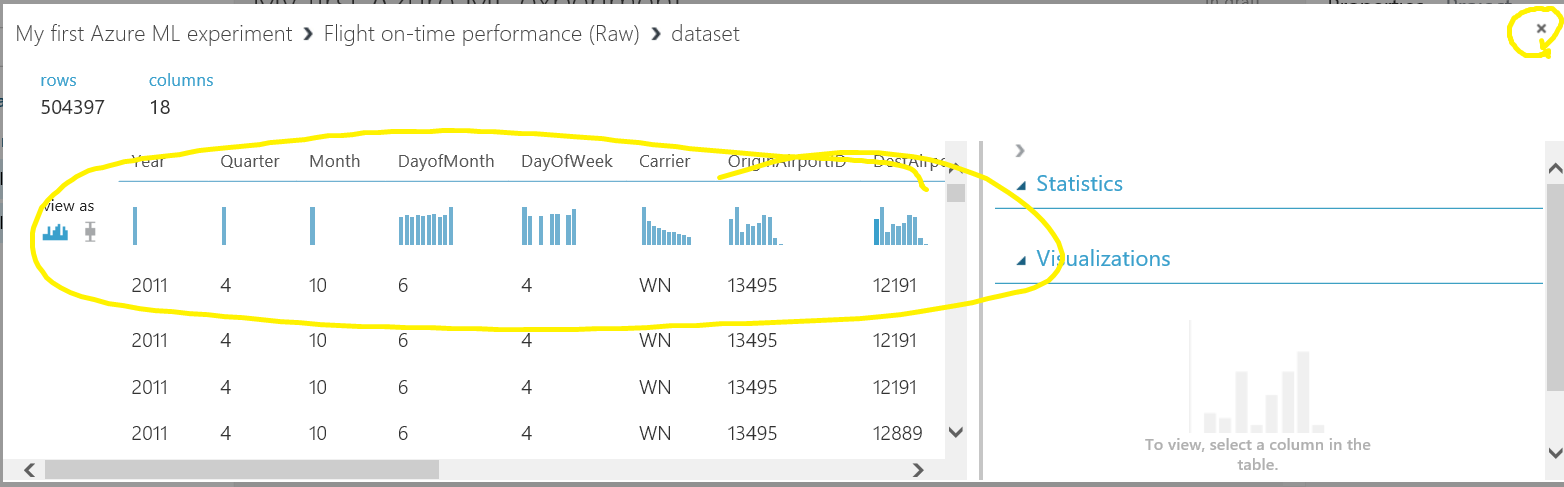
* 1. Review Data

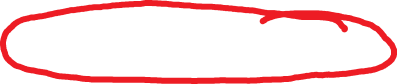
Right click on the dataset on your worksheet and select **dataset | visualize** from the pop-up menu.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/3cvis.png)



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 data and visualizing your dataset is a great first step. This dataset provides a great deal of information about flights and whether or not 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.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/3cdataset.png)



Note: In an actual 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), http://www.transtats.bts.gov/DL\_SelectFields.asp?Table\_ID=236&DB\_Short\_Name=On-Time.

* 1. Close the data visualization window

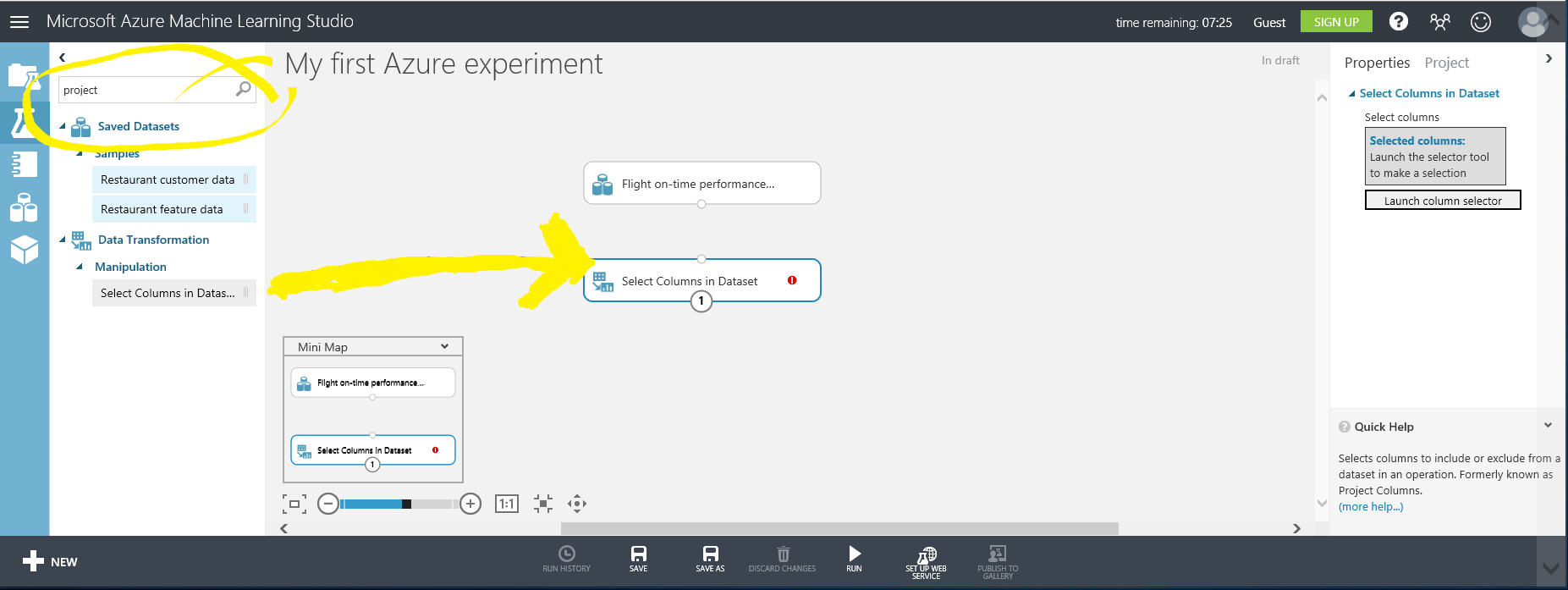
Click on the X in the top right corner of the window to close the data visualization window..

## **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.

* 1. Add Manipulation to Select Columns in Dataset

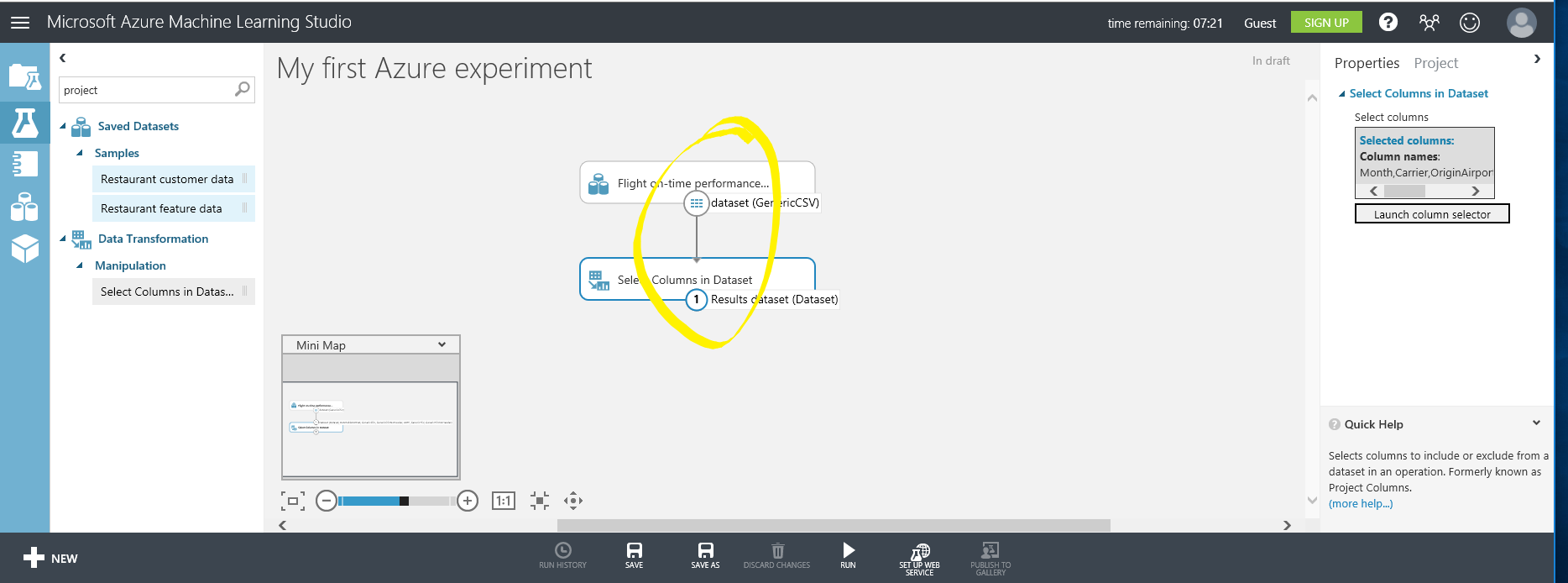
First, type "**project**" into the search bar and drag the Select Columns in Dataset manipulation to the workspace. This manipulation enables you to specify which columns in the data set you think are significant to the prediction.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/4a.png)



* 1. Connect Flight on-time performance task to Select Columns in Dataset task

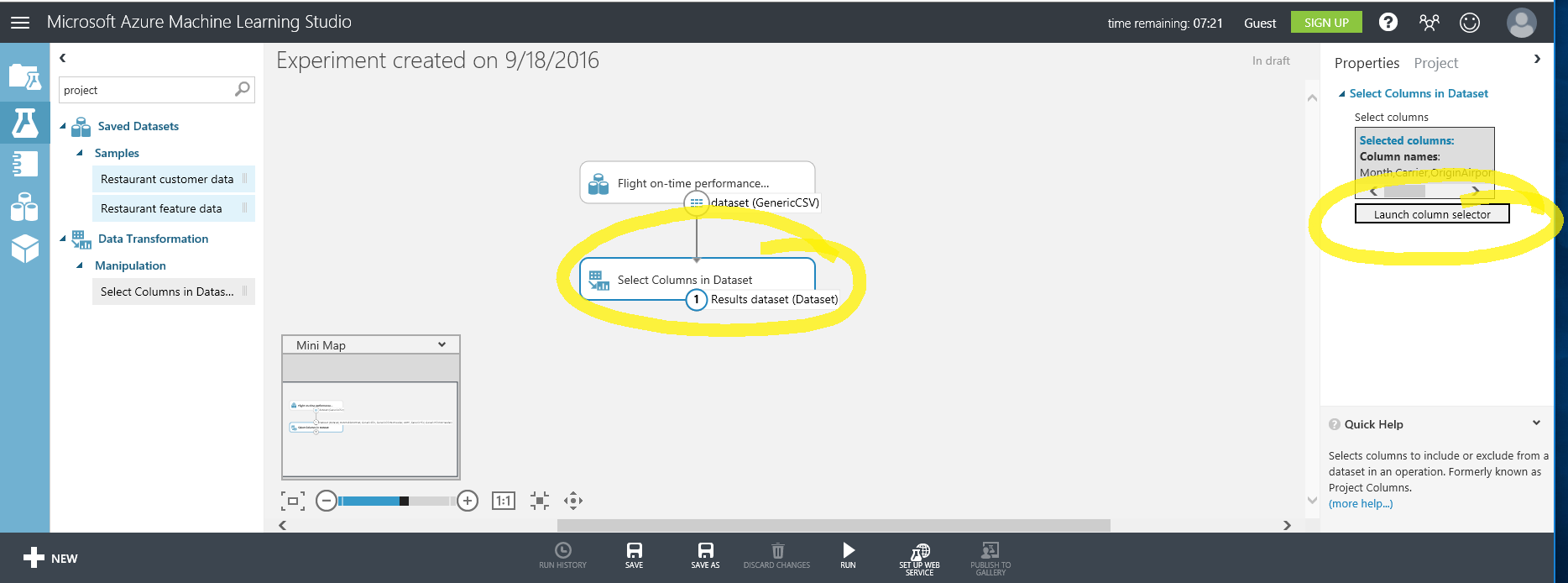
Connect the output of Flight on-time performance dataset to the input of the Select Columns in Dataset by clicking on the lower center dot and dragging to the input, top center dot, of the Select Columns in Dataset task.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/4b.png)



* 1. Launch Column Selector

Click on the Select Columns in Dataset module, then on the far right, select **Launch column selector**.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/4c.png)



* 1. Select Columns

Select the columns you think affect whether or not a flight is delayed as well as the column we want to predict ArrDel15. In the following screenshot, I selected Month, Carrier (airline), OriginAirportID, DestAirportID, DayofWeek, ArrDel15, CRSDepTime, Cancelled, Diverted, DepDelay, DayofMonth. You might choose to select more or less columns in subsequent runs to determine how they affect the outcome. The columns used will affect the prediction.

### 



* 1. Complete Column Selection

Select the checkbox in the lower right of the **Select columns** window.

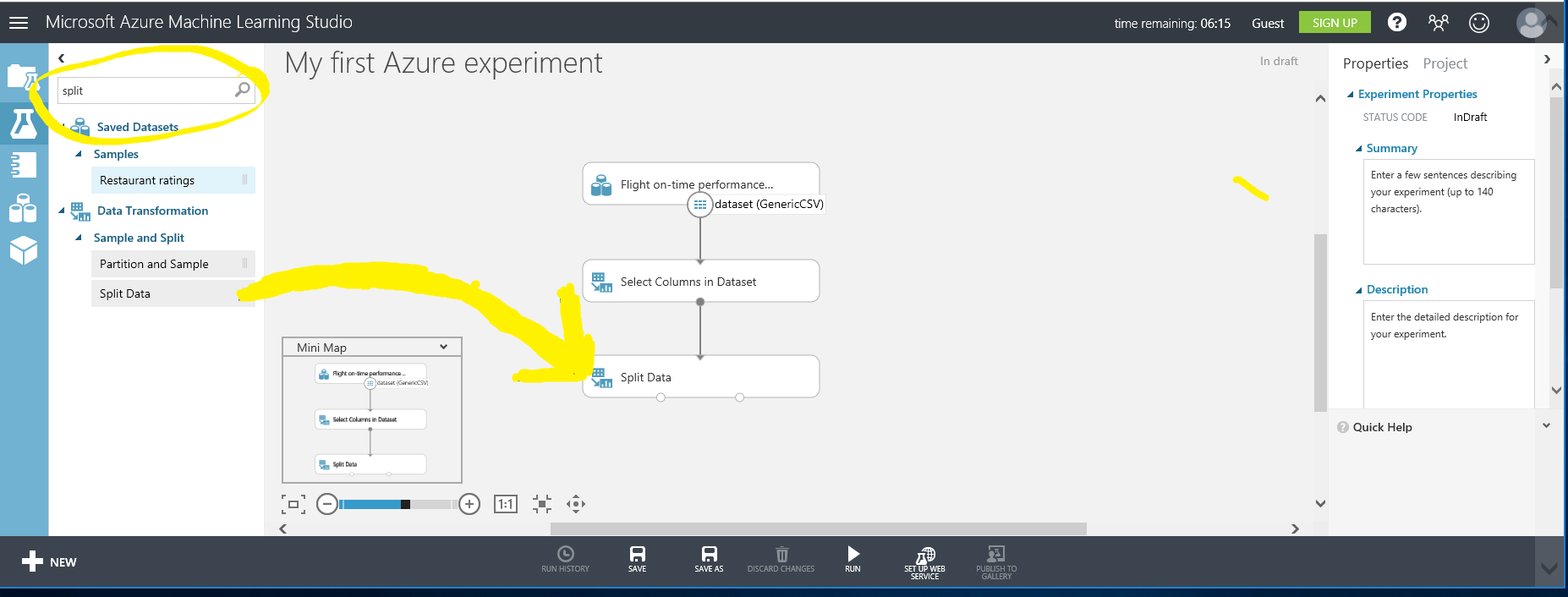
## **Split The Data Into A Training And Test Set**

The Split Data task allows us to divide up our data, we need some of the data to try and find patterns so we can make predictions. We need to save some of the data to test if the model we create successfully makes predictions.

Traditionally, you will split the data 80/20 or 70/30. For today’s challenge 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.

* 1. Split Data Task

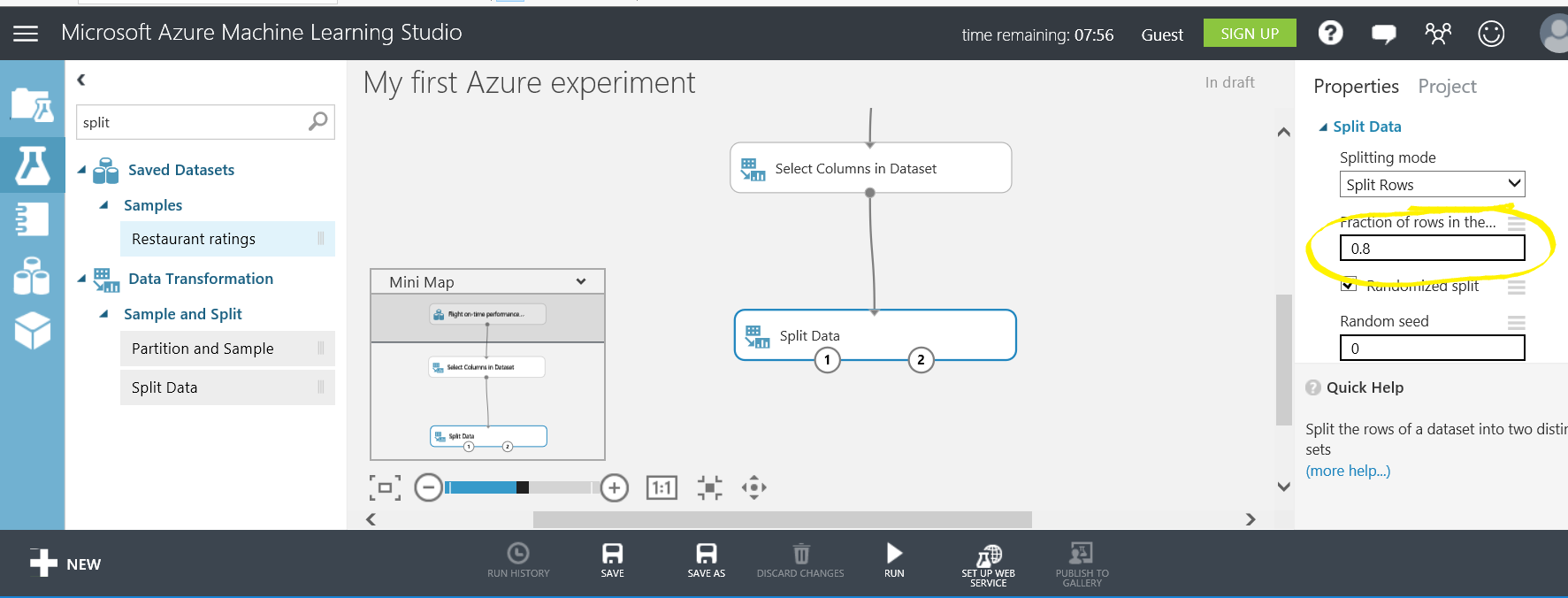
Type “**split**” into the search bar and drag the Split Data task to the workspace. Connect the output of Select Columns in Dataset task to the input of the Split Data task (same way we connected the Flight Data to the Select Columns modules).

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/5a.png)



* 1. Split our input data

Click on the Split Data task to bring up the Properties Pane and specify .8 as the Fraction of rows. Change random seed to 1234.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/5b.png)



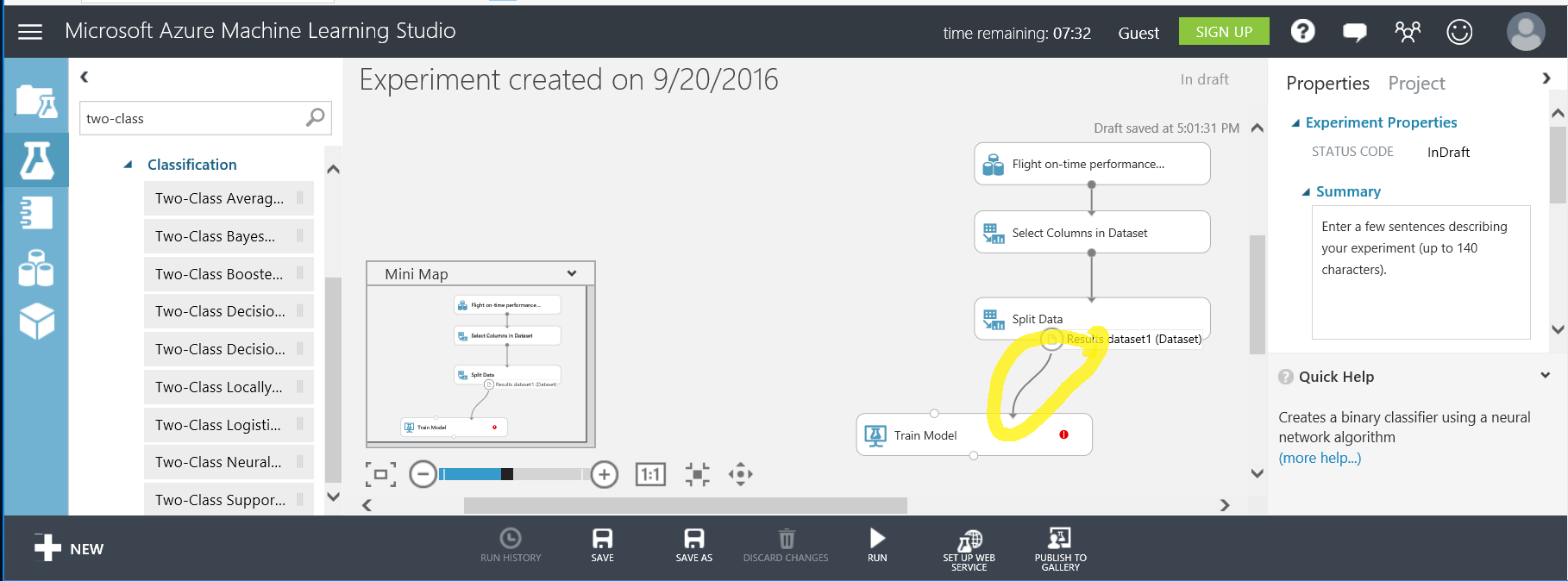
## **Train The Model**

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

* 1. Connect Data

Type “**train model**” into the search bar. Drag the Train Model task to the workspace.

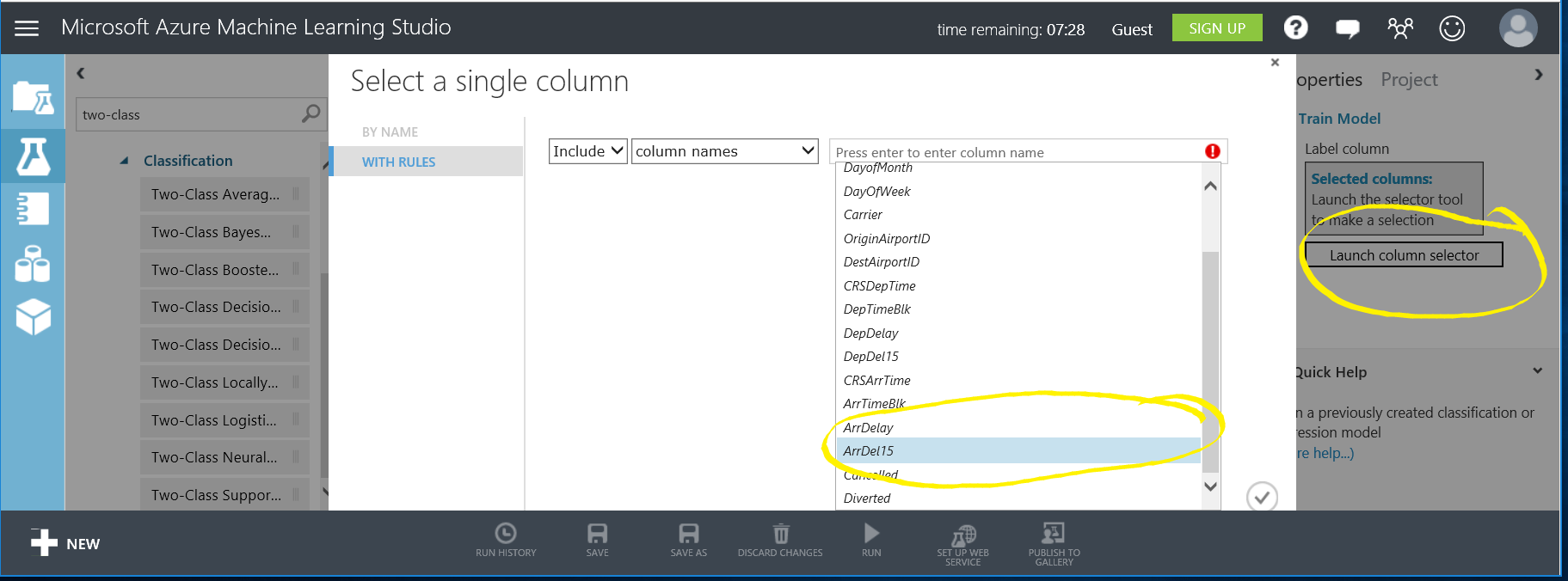
Hovering over the input and output dots will reveal what each input/output represents. Connect the first output, Results Dataset1, (the circle on the left) of the Split Data task to the **rightmost** input of the Train Model task. This will take 80 % of our data and use it to train/teach our model to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

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/6a.png)



* 1. Identify Predicted Value

Click on the Train Model task. In the **Properties** window, select Launch Column Selector. 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 to complete the operation.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/6b.png)



## **Select Algorithm**

If you are a data scientist who creates their 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 use.

Selecting an algorithm can be overwhelming, to help narrow the process a [Azure ML Cheat Sheet](https://aka.ms/azuremlcheatsheet), <https://aka.ms/azuremlcheatsheet>, has been created. By narrowing the type of problem you are solving can find the algorithms that will be most likely to generate a good model.

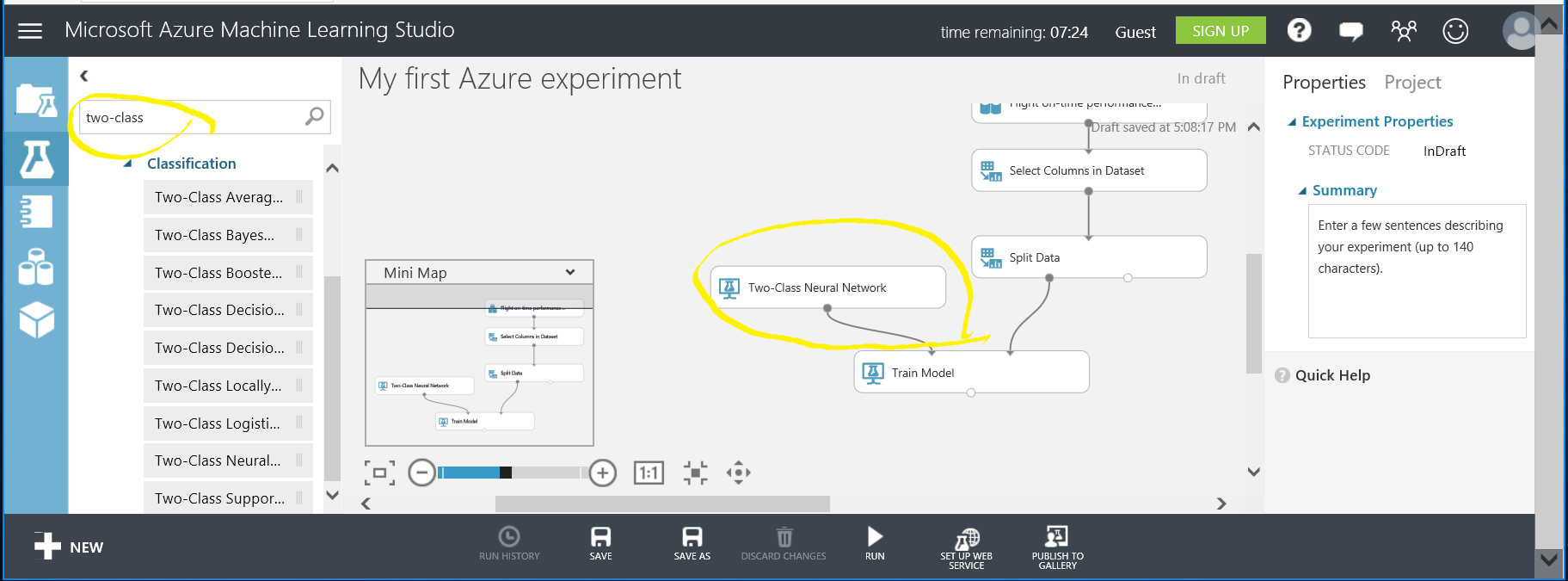
Today we are doing binary classification also known as Two-Class Classification. Using the cheat sheet we can narrow our selection to a standard algorithm called Two-Class Neural Network.

As you can see there are many 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 as you will see.

* 1. Connect algorithm

Type “**two-class**” into the search bar. You will see a number of different classification algorithms listed and each has its own advantages and disadvantages. Each of the two-class algorithms is designed to predict a binary outcome.

Select Two-Class Neural Network and drag it to the workspace. Connect the output of the Two-Class Neural Network task to the **leftmost** input of the Train Model task.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/7a.png)



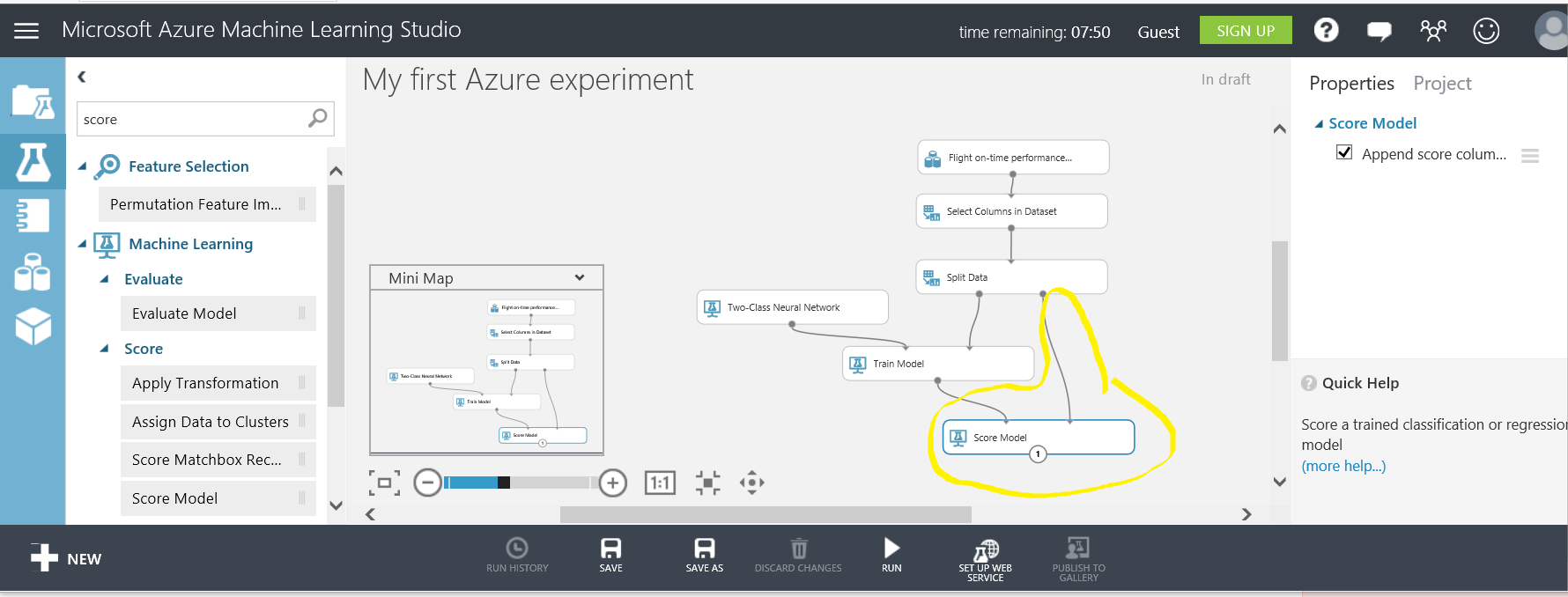
## **Score the Model**

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 which is the remaining 20% of the data we split to the second output of the Split Data task.

* 1. Connect test data

Type “**score**” into the search bar and drag the Score Model task to the workspace.

Connect the output of Train Model to the **left input** of the Score Model task. 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.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/8a.png)



## **Evaluate Model**

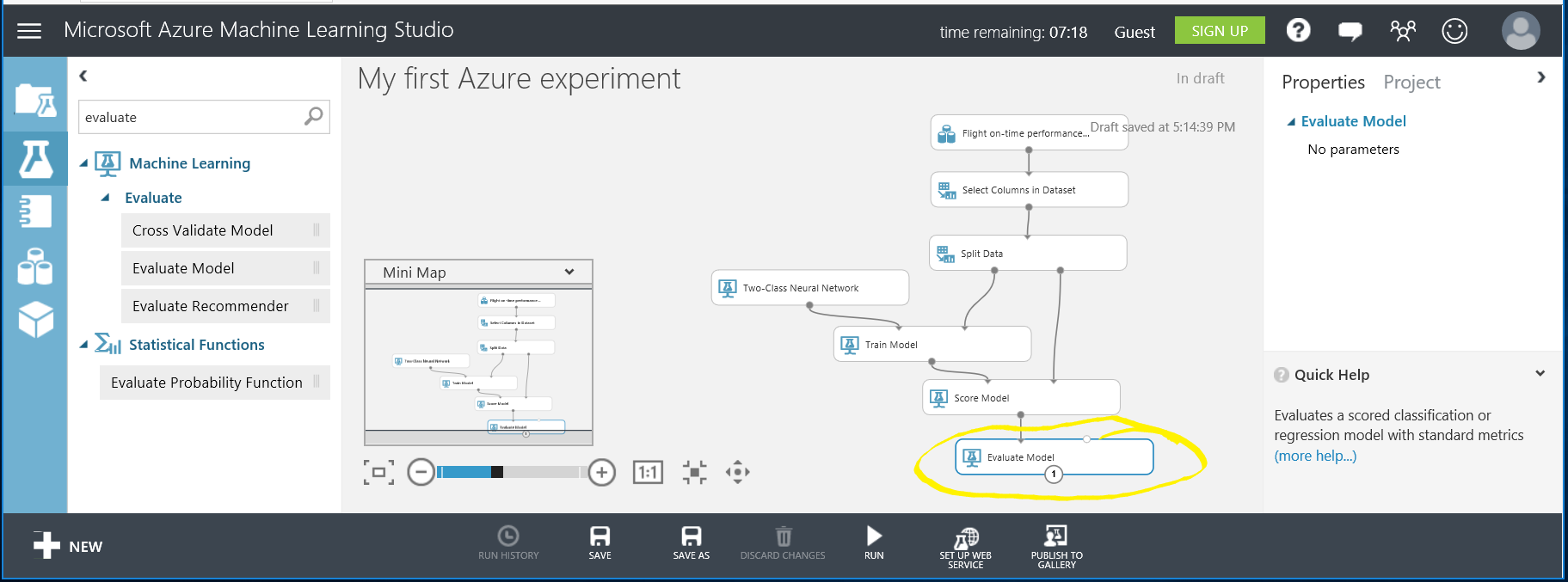
Next, the model is evaluated to determine its accuracy. This is done by evaluating the trained model by using the test data.

* 1. Determine accuracy of model

Type “**evaluate**” into the search bar and drag the Evaluate Model task to the bottom of the workspace.

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!

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/9a.png)



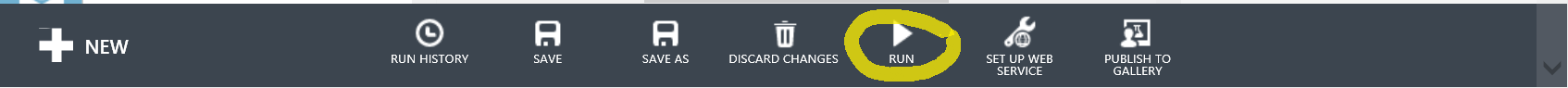
## **Run Experiment**

### 

* 1. Select Run

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.

[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/10a.png)

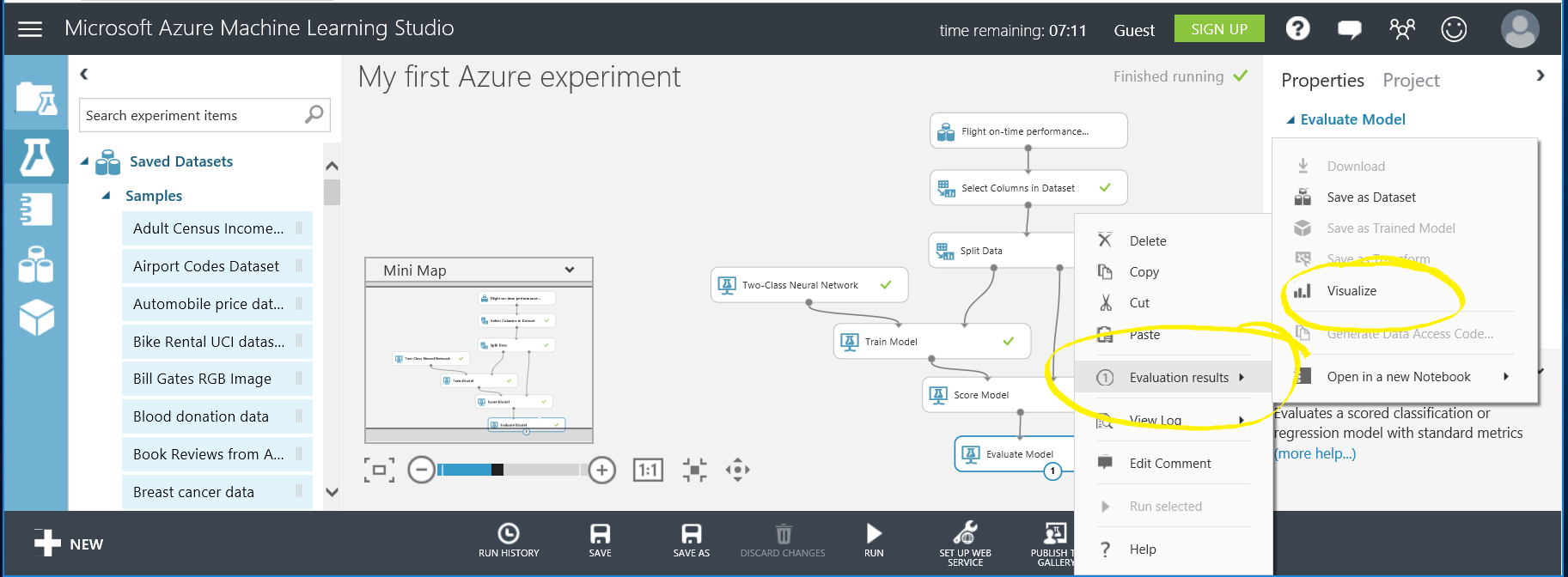


## **Post Run: Evaluate Model**

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

* 1. 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.

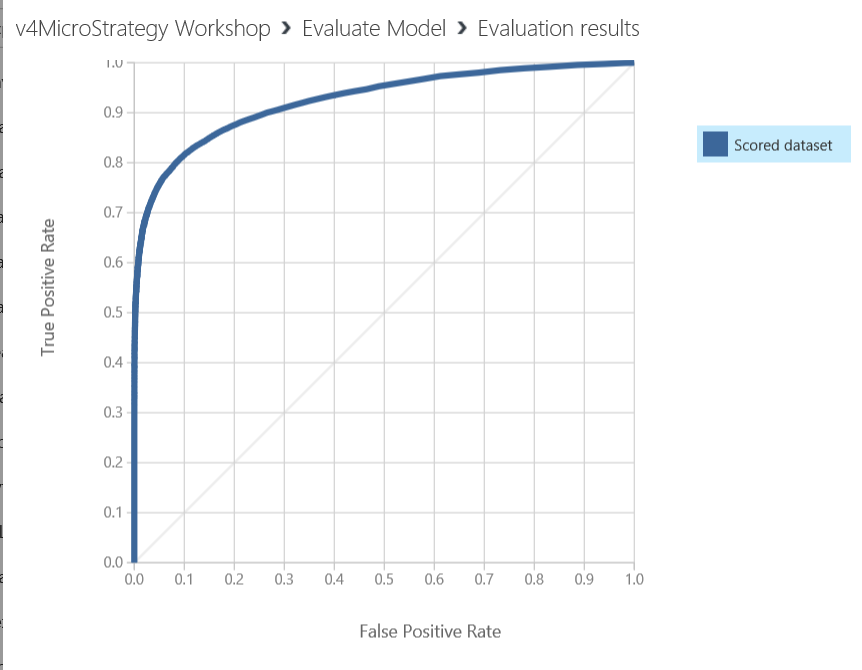
[](https://github.com/bethz/AzureML-FlightPrediction/blob/master/images/11a.png)



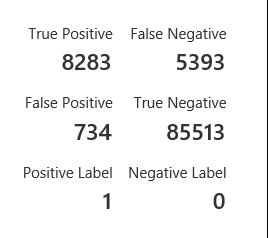
## **Interpreting Results**

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**. This graph is a representation of the Area Under the Curve. A 45 degree flat line on this chart indicates guessing randomly. A more accurate model than random guessing looks like the image below, our current model.



Model Accuracy



If you scroll down you can see the accuracy – Higher accuracy is good! 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 actually 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 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 a good number of True Positive and True Negative which indicates the model predicted those correctly and 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 publish the training experiment as a web service so MicroStrategy Desktop can use the model to make predictions. The 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:

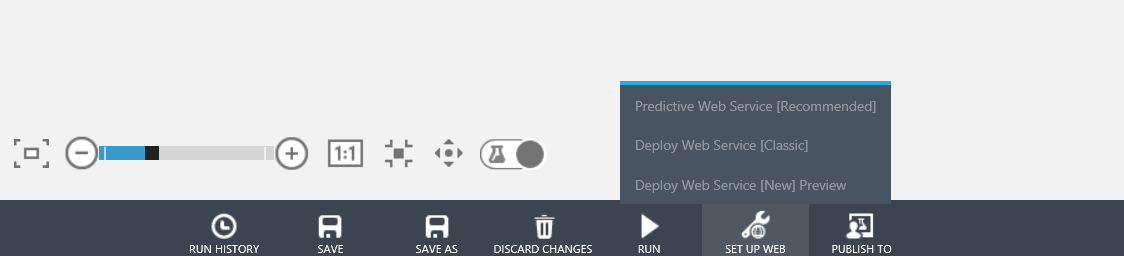
Convert training experiment to a predictive experiment

* 1. This step defines the inputs/outputs and removes unneeded training info

Deploy it as a Web Service

* 1. This step creates an API key and a URI for the model

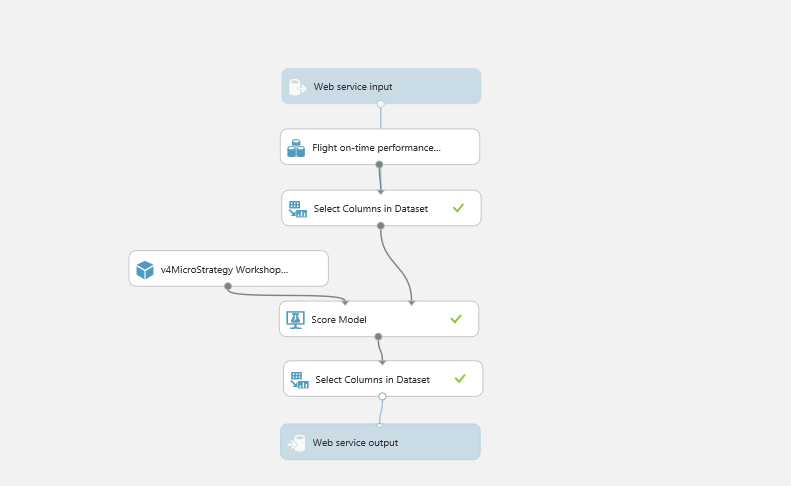
Create a predictive experiment

Creating a predictive experiment from the training experiment is the first step to make 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. 



* 1. 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.

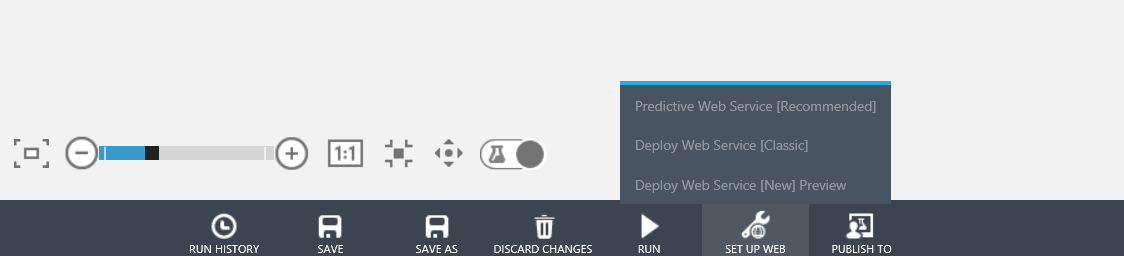


* 1. Click Run

It is necessary to run the model by clicking run.

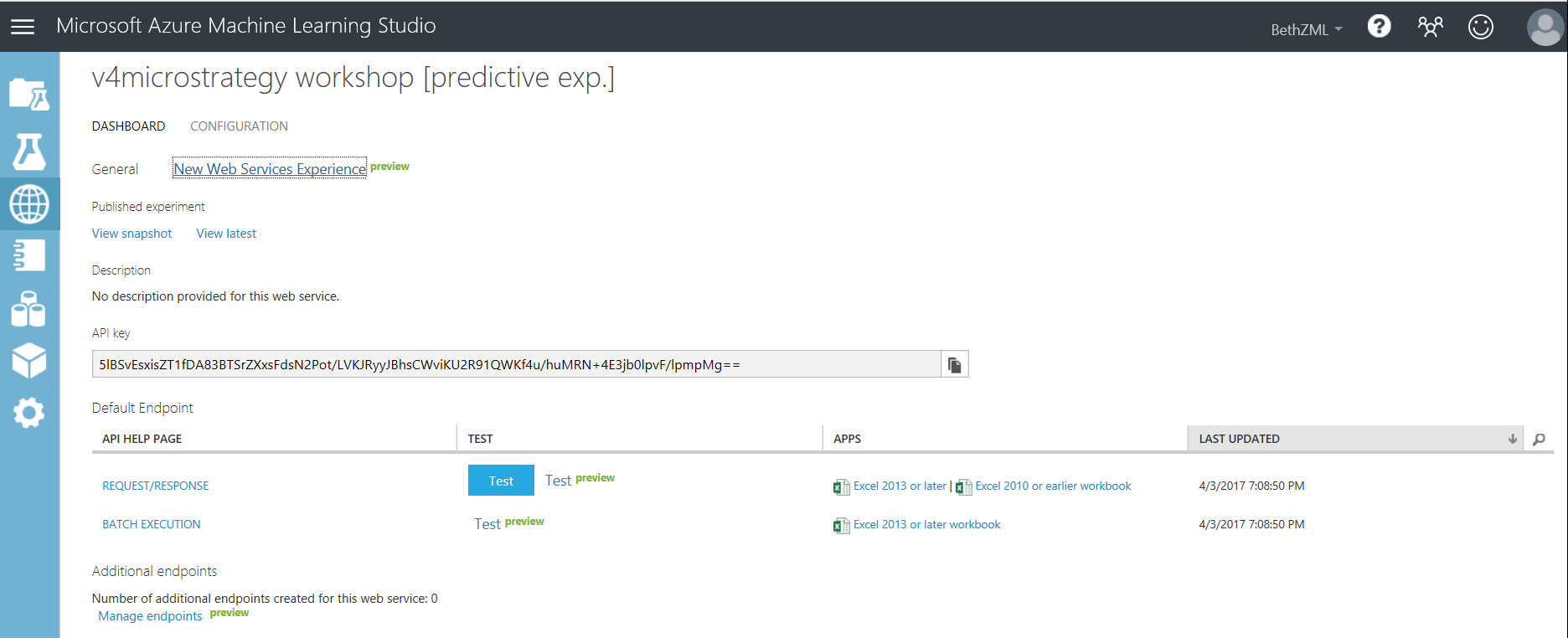
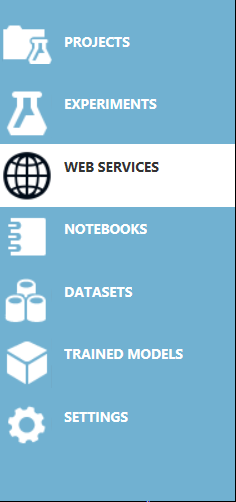
Publish a Web Service

* 1. Next, click Deploy Web Service [Classic] and you have an active Web Service ready for consumption.

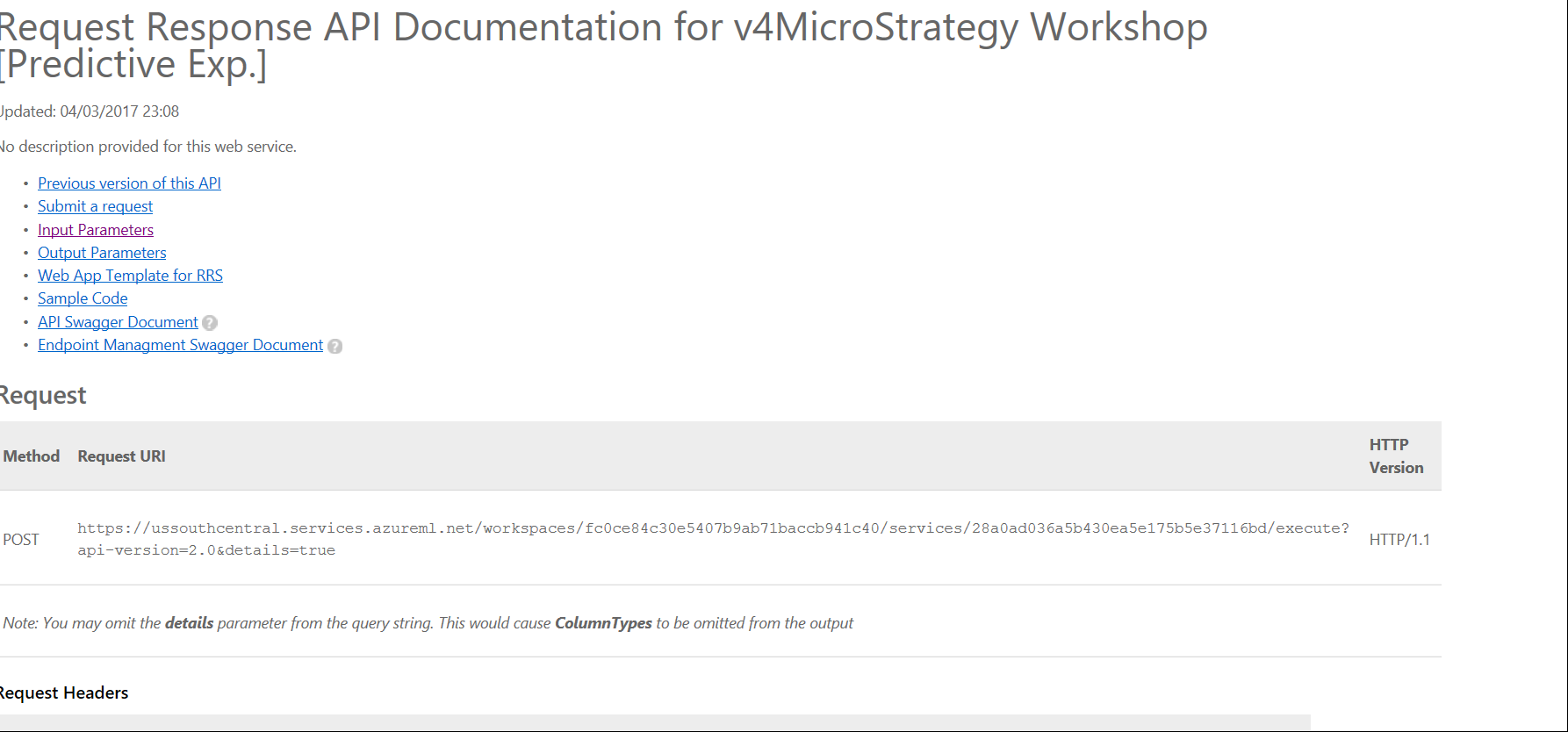




The API Key and URI to the model are needed to consume the model. Here is how to obtain those inputs for MicroStrategy Desktop. In this workshop, we will provide these so you will not need to capture them.





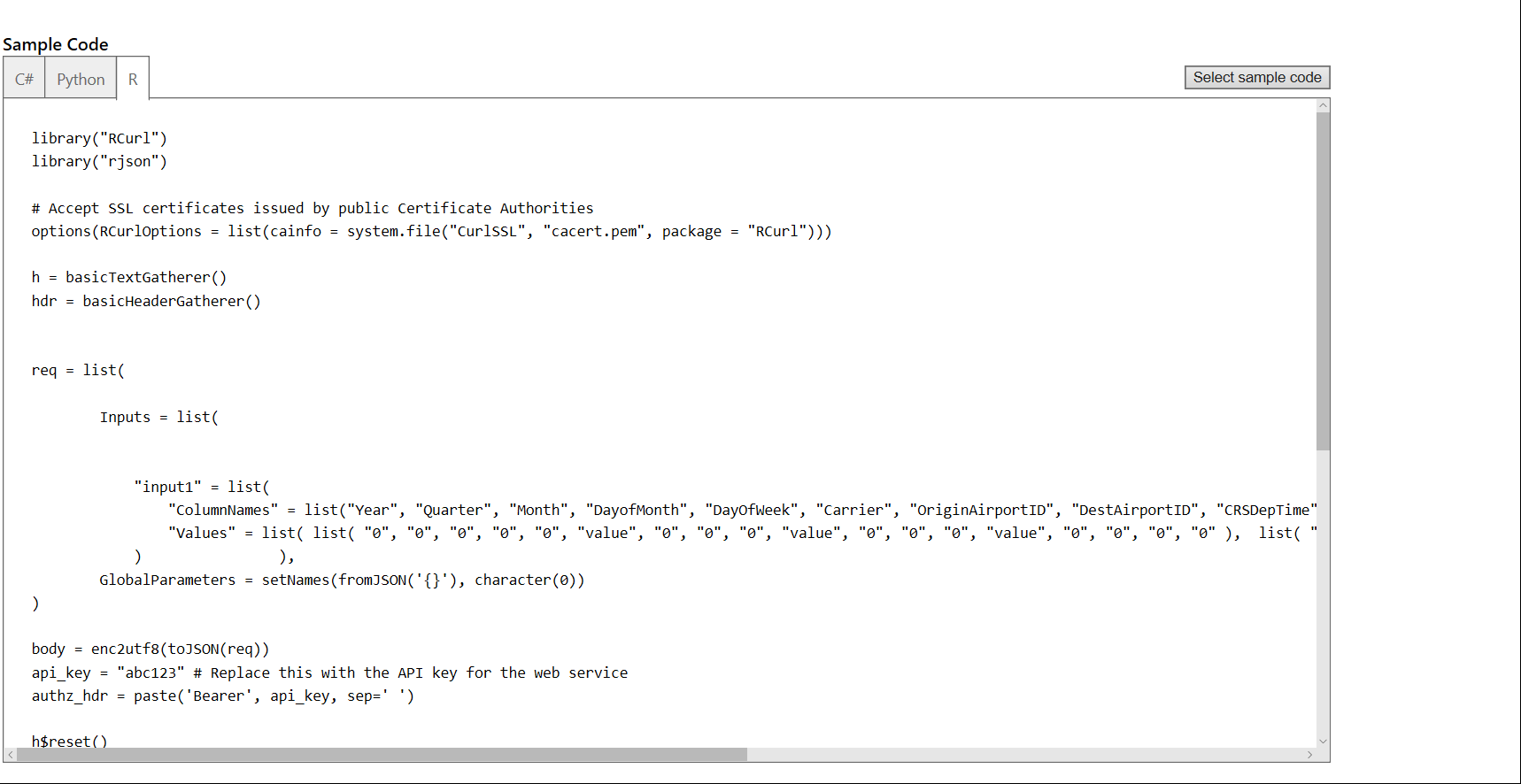




Azure ML automatically creates R code for use in MicroStrategy Desktop with very minor modifications, including provide the correct API key and web service URI, and convert the output to be a vector.

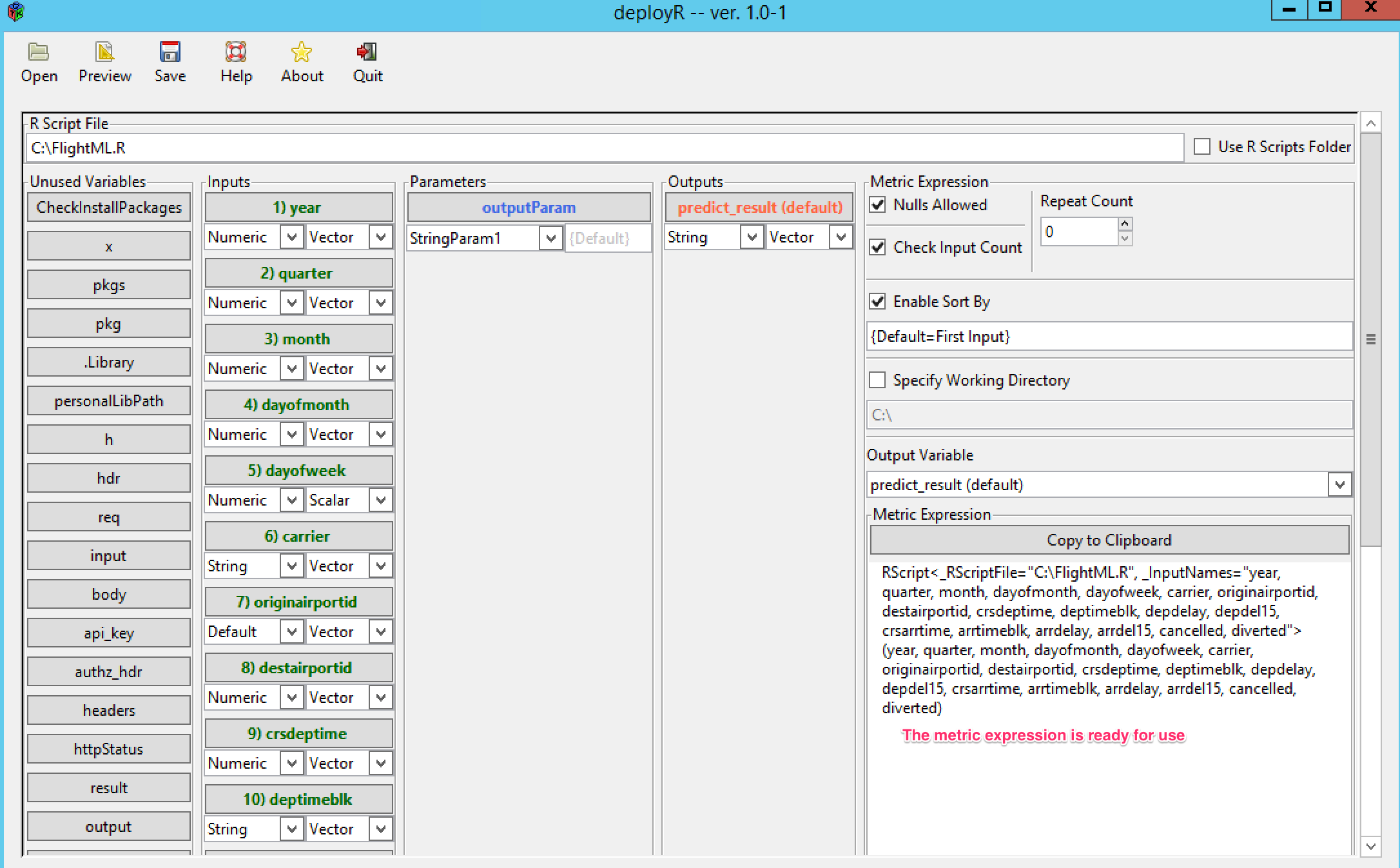
**In this workshop, you are not required to do this step and we will provide you with the R script code.**

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





After you have the R script, you can use the deploy() function in R package “MicroStrategyR” to obtain the metric expression.



**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 URI are the way that the Desktop and Azure ML model communicate.

1. Step-by-step MicroStrategy Desktop

Downloading R script file and data file

1. If you have not done so, please download the R script file “FlightML.R” and the data file “FlightData.xlsx”, and put them on your C: drive if you have Windows laptop or any convenient place if you have a Mac. Download the files from here: <https://github.com/bethz/AzureML-FlightPrediction/tree/master/MicroStrategy>

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 compatibility with all packages, scripts and applications that work with that version of R. It includes additional capabilities for* [*improved performance*](https://mran.microsoft.com/rro/#intelmkl1)*,* [*reproducibility*](https://mran.microsoft.com/rro/#reproducibility)*, as well as support for* [*Windows and Linux-based platforms*](https://mran.microsoft.com/documents/rro/installation/#sysreq)*.*

*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

1. Launch MicroStrategy Desktop.

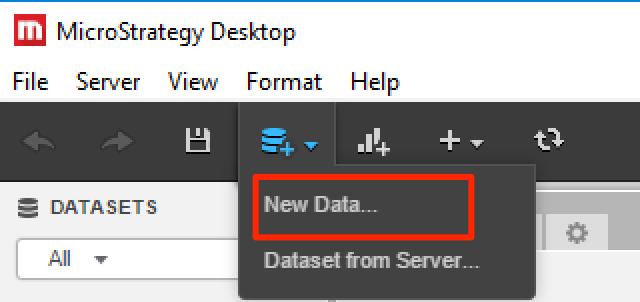


1. If this is the first time you are accessing MicroStrategy Desktop, the Quick Tips guided product tour will be displayed. Click Hide Quick Tips to remove the notes.

Connect to your data

1. Next, we will import an xlsx file with some sample flight data into MicroStrategy Desktop so we can use its interface to invoke the Azure ML web service created in section 2 and use it to make some of our own flight predictions and generate dashboards.

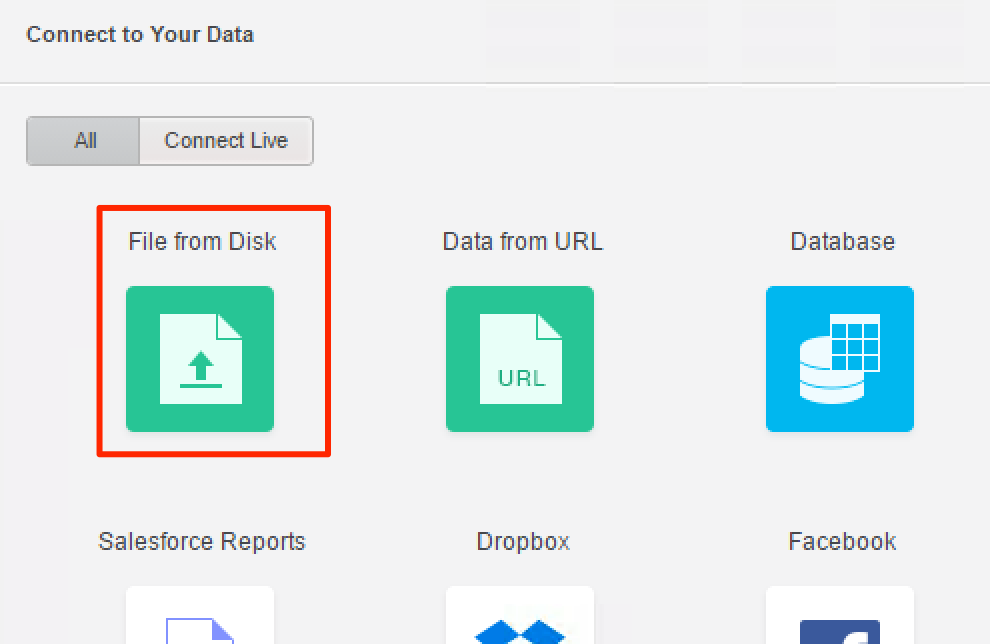
From the Dashboard Datasets panel, click on the Add Data button and it will show a New Data… or Dataset from Server… choice.



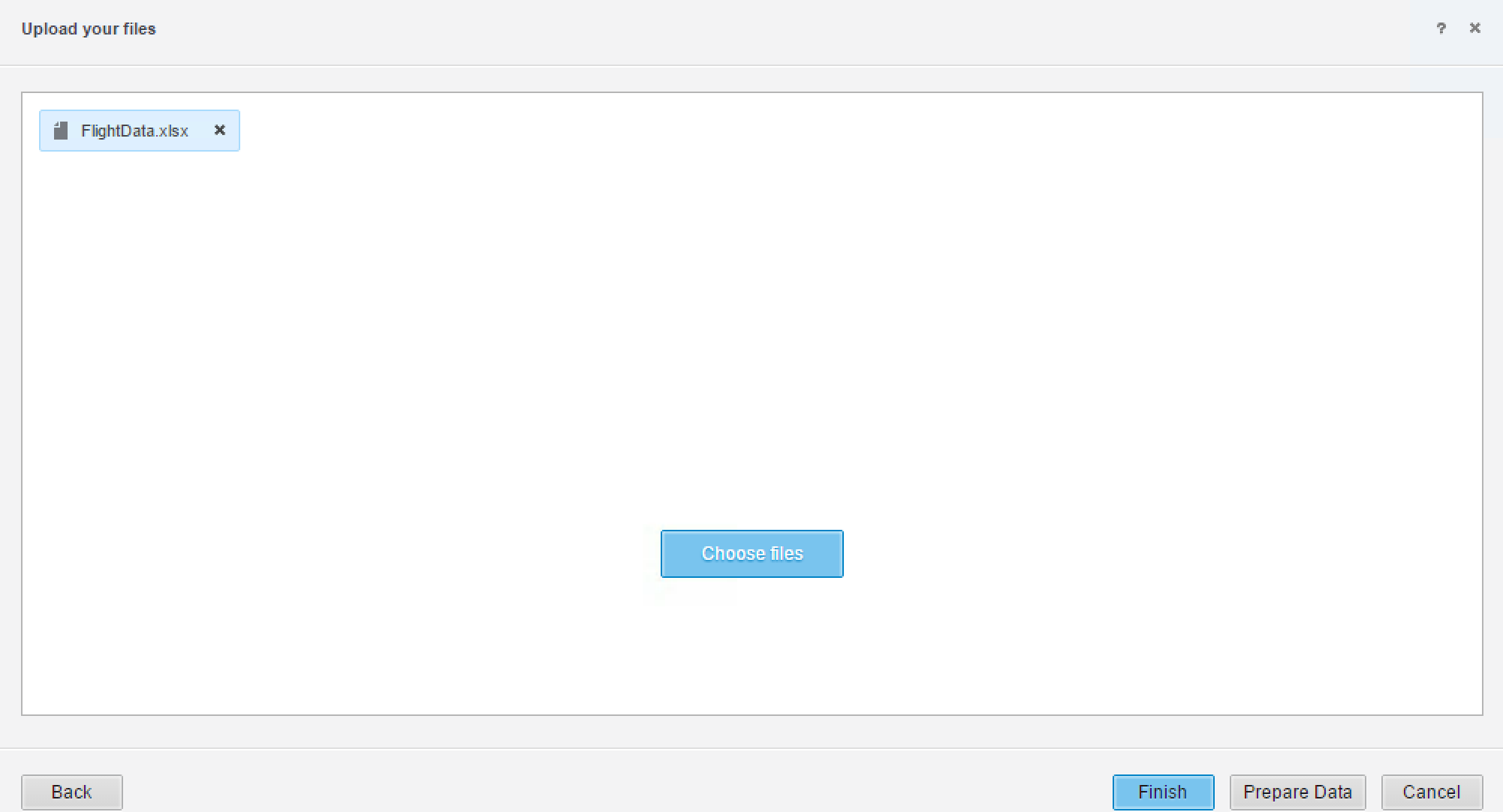
1. Click New Data and the Connect to Your Data wizard will be shown.

TIP: MicroStrategy Desktop allows you to quickly and intuitively connect to almost any data source, from a tabular data file like Excel spreadsheets, to Hadoop big data sources, cloud- based applications, and relational databases.

1. Next, we will import the Excel data file. Choose the File from Disk option.

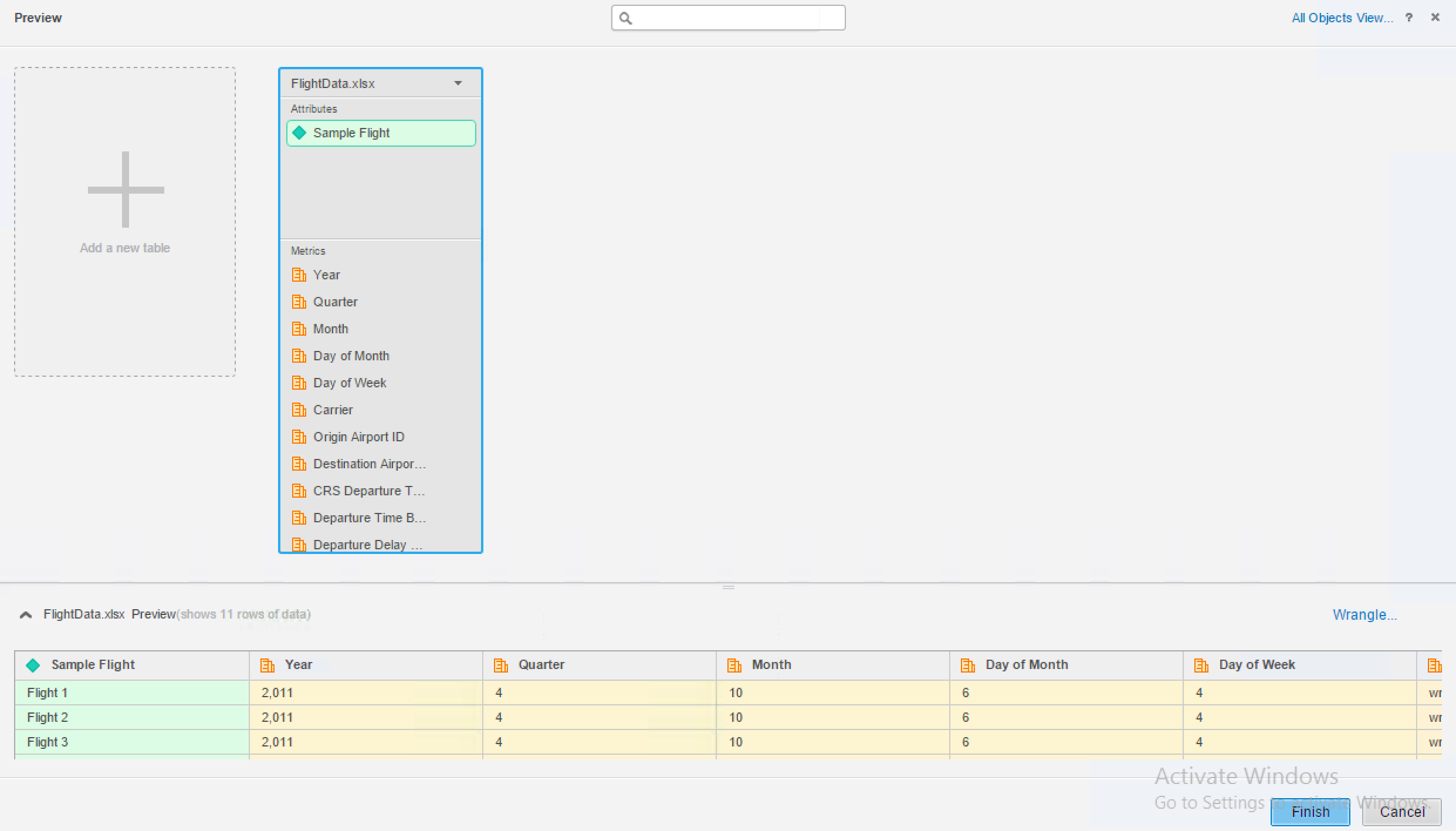


1. An Upload your files window will be shown, on that screen, click the blue Choose files button.
2. In the dialog that appears, find and select the file “FlightData.xlsx”, which was downloaded previously and click Open.
3. Click the Prepare Data button in the dialog box to preview the data you are importing. This will also provide the opportunity to refine the data.





1. In the Preview page, we will move some attributes to the Metrics section. Do not move the “Sample Flight” attribute. Move a group of attributes by clicking at the top of the Attributes list and then shift-clicking an attribute at the end of the list. Then, drag and drop them to the Metrics section. The page should look like the image below.



On this screen, you are presented with a preview of the data. Here you can review a sample of the data in the dataset, the mapping of the columns as attributes or metrics, and the data type of each column.

1. After reviewing your data click Finish button to complete data import.

TIP: MicroStrategy automatically maps the columns as attributes (your business dimensions) and metrics (your performance indicators or KPIs) that are available for analysis. This mapping is based on the column data types, and content. If any adjustments are needed, they can be performed here as in the above step.

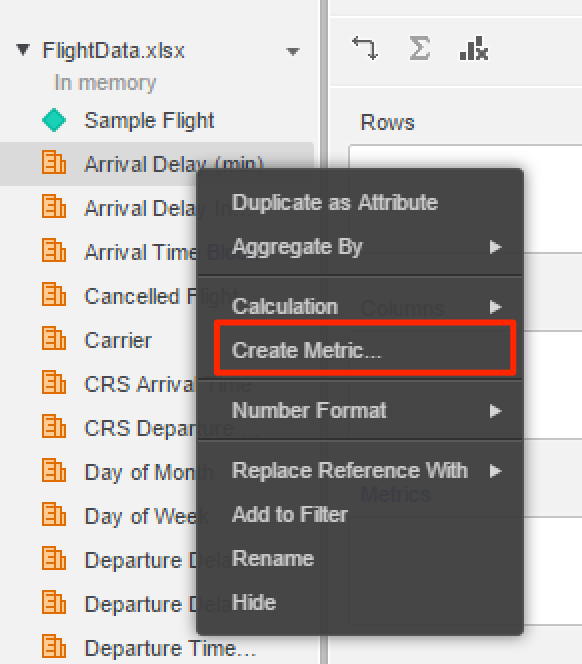
Make Predictions on new data

Now that you have your new data imported and prepared, you are ready to make predictions about them! Use the Azure Machine Learning model you prepared in step 2. Before you start visualizing your data, take a moment to save the dashboard.

1. Click the Save button.



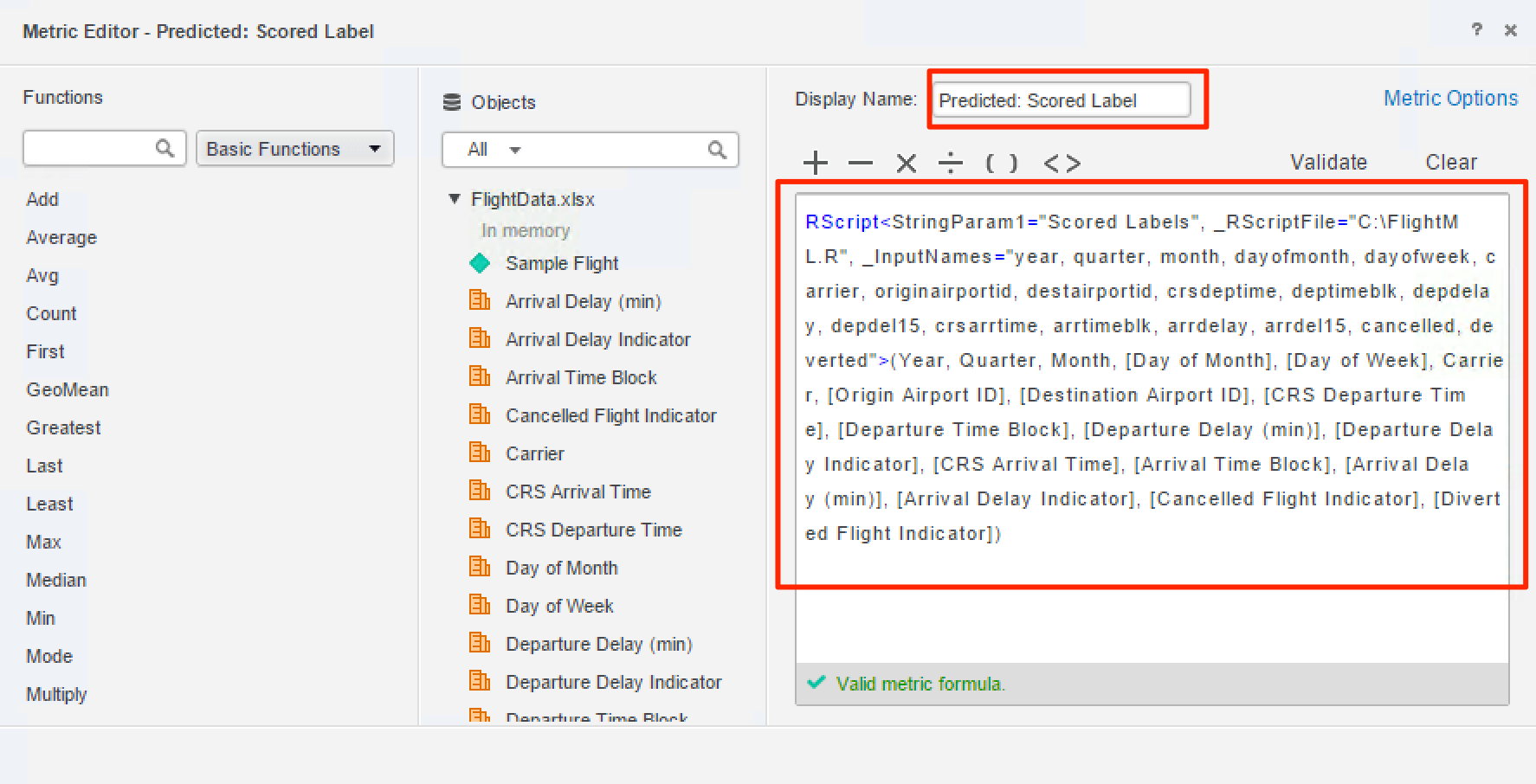
1. In the dialog box, navigate to the Desktop and save your dashboard as Azure ML Workshop. It is now saved as a shareable MicroStrategy file.
2. Next, we are going to create a metric for displaying the predicted result. In the DATASETS panel, right click on any metric (e.g. “Arrival Delay (min)”), and select “Create Metric” from the menu.



1. In the Metric Editor, copy and paste the metric expression below as shown in the next image. 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.

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, deverted">(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])

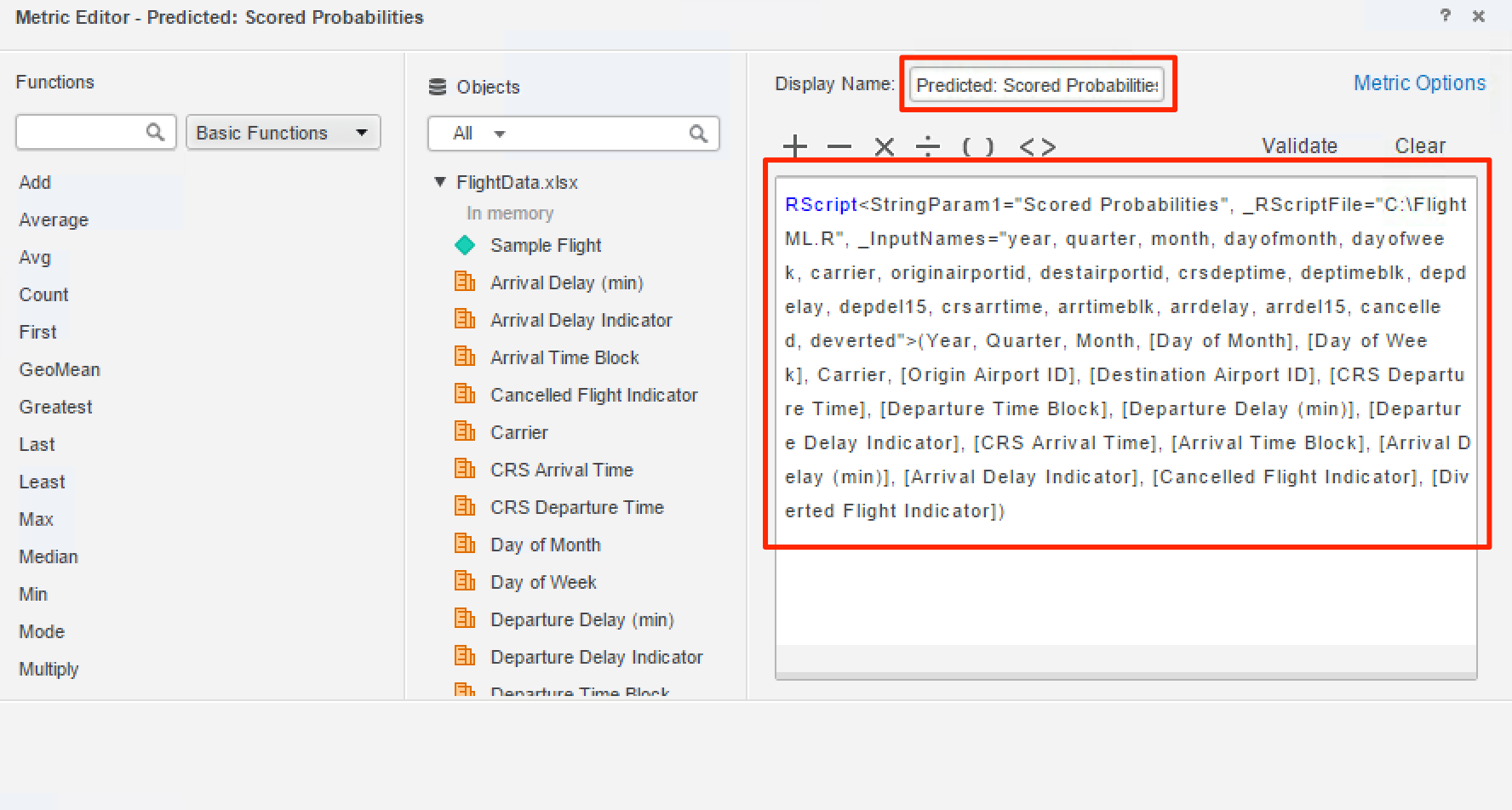
*If you are using a Mac, please modify the above yellow filename with the full path of your downloaded R script file*.



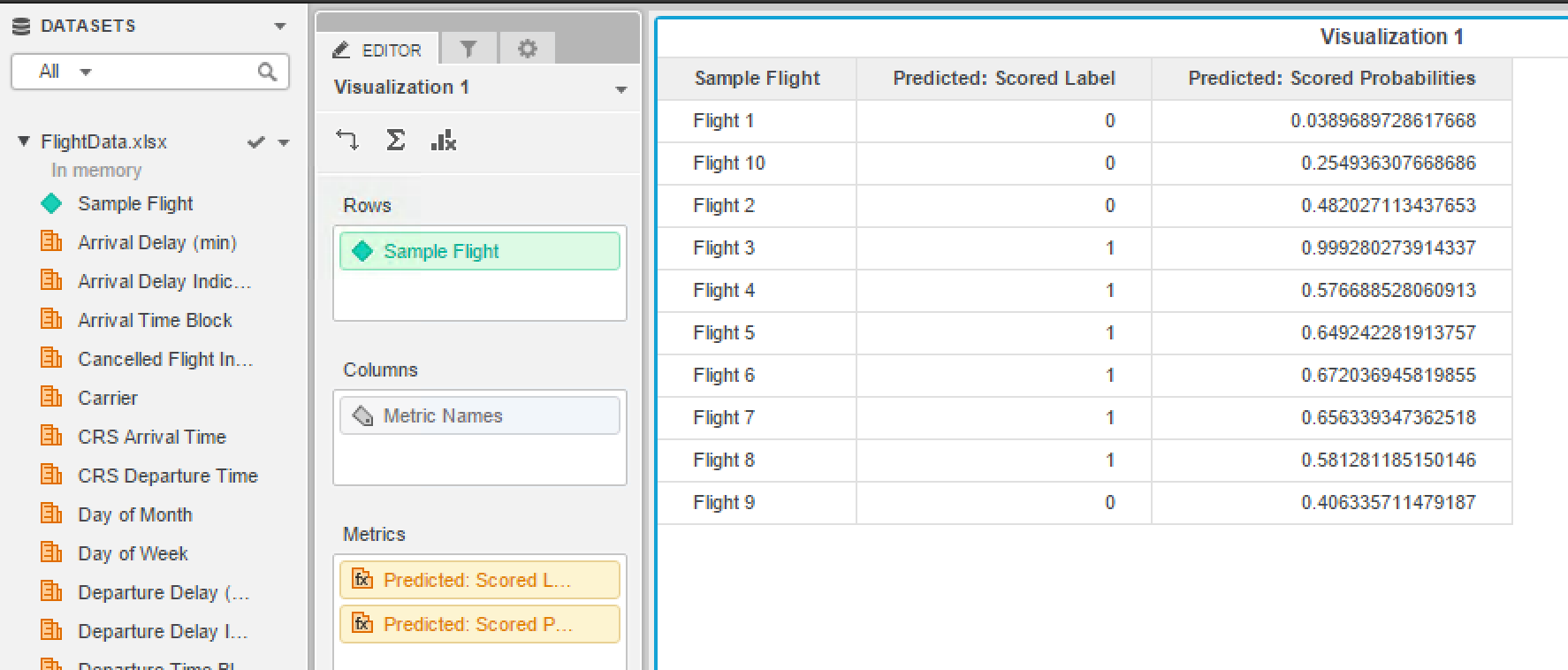
1. Rename the metric by changing the Display Name to “Predicted: Scored Labels”.
2. Validate the metric formula by clicking Validate. A checkmark under the script box with the phrase “Valid metric formula” indicates success.
3. Click Save to save this work.
4. Next, create a metric which includes the on-time or delayed flight prediction along with the data used to generate that prediction. To create the metric, in the DATASET panel, right click on any metric (e.g. “Arrival Delay (min)”), and select “Create Metric” from the menu.
5. In the Metric Editor, copy and paste the metric expression below, and rename the metric as “Predicted: Scored Probabilities”.

RScript<StringParam1="Scored Probabilities", \_RScriptFile="C:\FlightML.R", \_InputNames="year, quarter, month, dayofmonth, dayofweek, carrier, originairportid, destairportid, crsdeptime, deptimeblk, depdelay, depdel15, crsarrtime, arrtimeblk, arrdelay, arrdel15, cancelled, deverted">(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])

*If you are using Mac, please modify above yellow filename with the full path of your downloaded R script file*.



1. Validate the metric formula by clicking Validate.
2. Click Save to save this work.
3. Now, let’s take a looked at the predicted result. Drag and drop the attribute “Sample Flight” to Rows, and the 2 metrics we created to Columns, and you will get following dashboard.



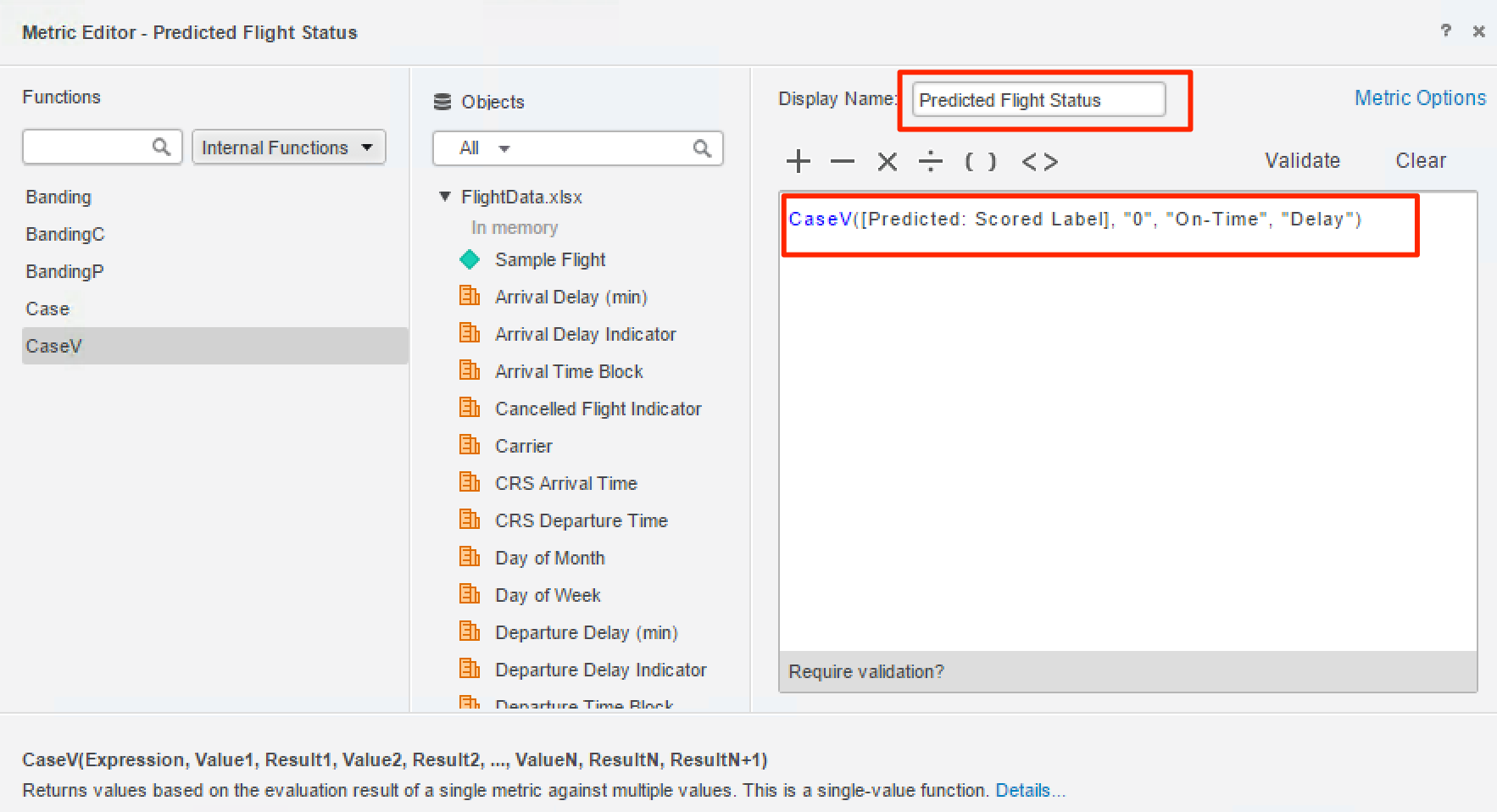
Congratulations! You’ve supplied flight data to the Azure Machine Learning model and now have flight predictions!

However, the dashboard looks a little boring for now, let’s continue work on it.

The prediction results we get by metric “Predicted: Scored Label” has a value 0 or 1, where 0 means “On-Time” and 1 means Delay, so let’s change it to be more descriptive by creating another metric.

1. In the DATASET panel, right click on any metric (e.g. “Arrival Delay (min)”), and select “Create Metric” from the menu to create another metric.
2. In the Metric Editor, copy and paste the metric expression below, and rename the metric as “Predicted Flight Status”.

CaseV([Predicted: Scored Label], "0", "On-Time", "Delay")

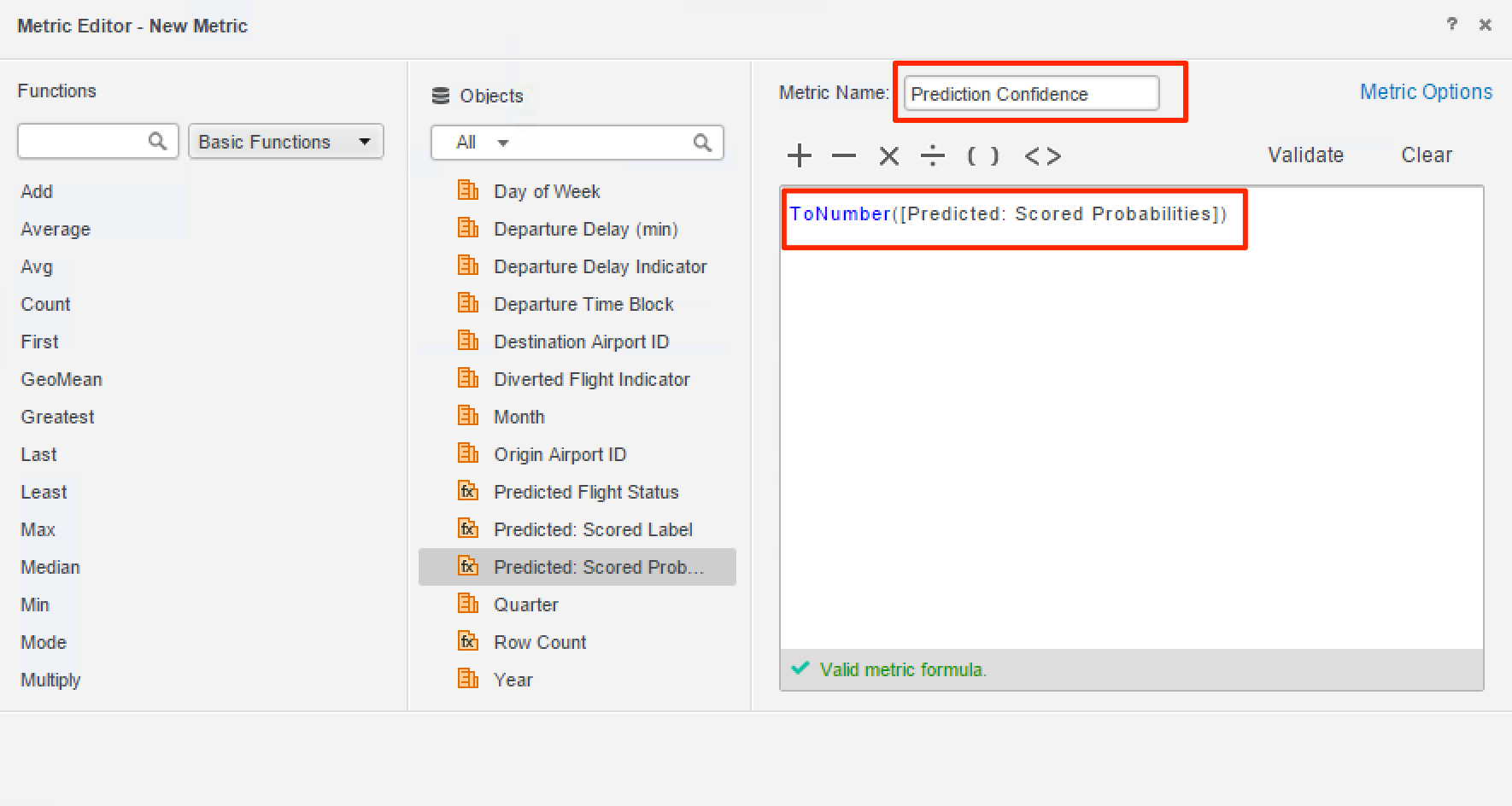


1. Validate the metric formula, and click Save.

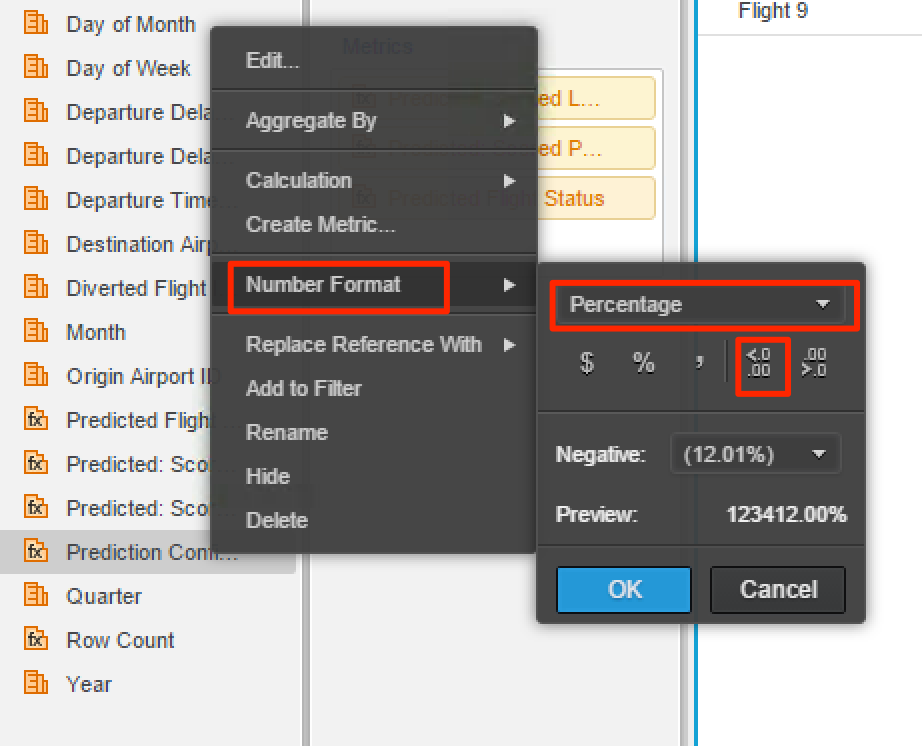
The prediction confidence is represented by the metric “Predicted: Scored Probabilities” is currently with string data type, to format it into as a percentage value, we’d like to convert its data type by creating another metric.

1. In the DATASET panel, right click on any metric (e.g. “Arrival Delay (min)”), and select “Create Metric” from the menu to create another metric.
2. In the Metric Editor, copy and paste the metric expression below, and rename the metric as “Prediction Confidence”.

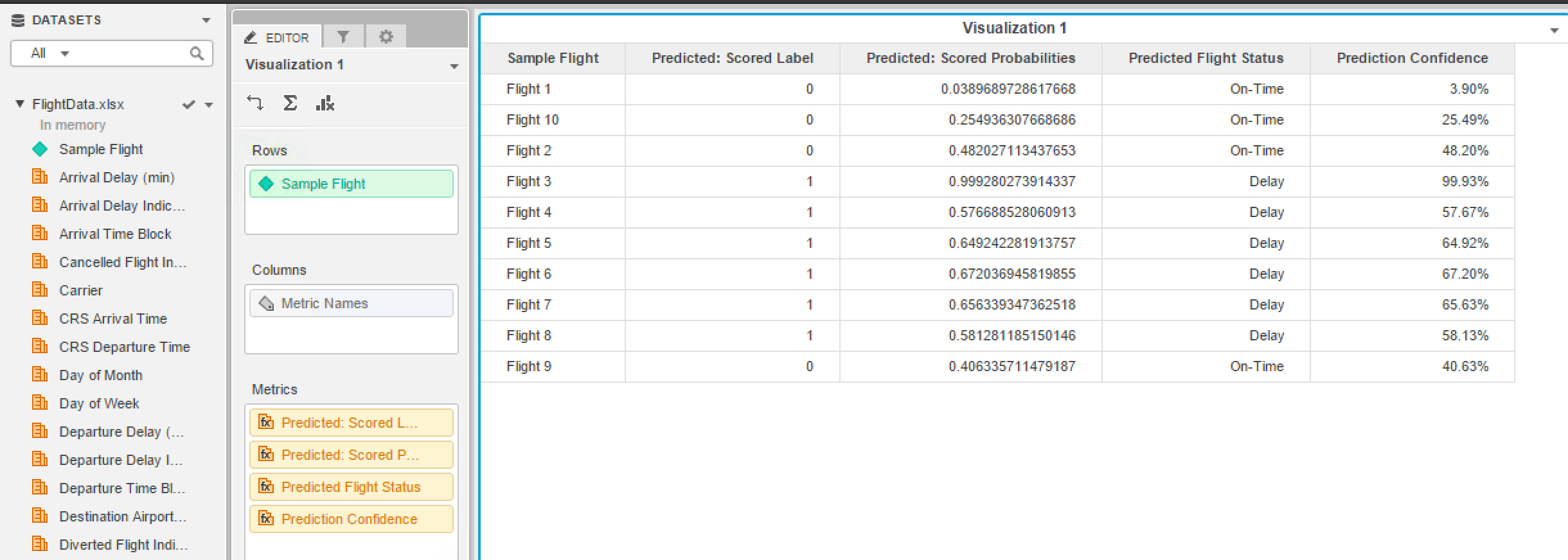
ToNumber([Predicted: Scored Probabilities])



1. Validate the metric formula, and click Save.
2. Next is to format the “Prediction Confidence” as percentage. In the DATASET panel, right click on metric “Prediction Confidence”, and select “Number Format”; set the format as Percentage, and have 2 decimals by clicking on the increase decimal button twice.



1. Click OK.
2. Add the 2 newly created metrics to dashboard, to see the result. Much better, right?

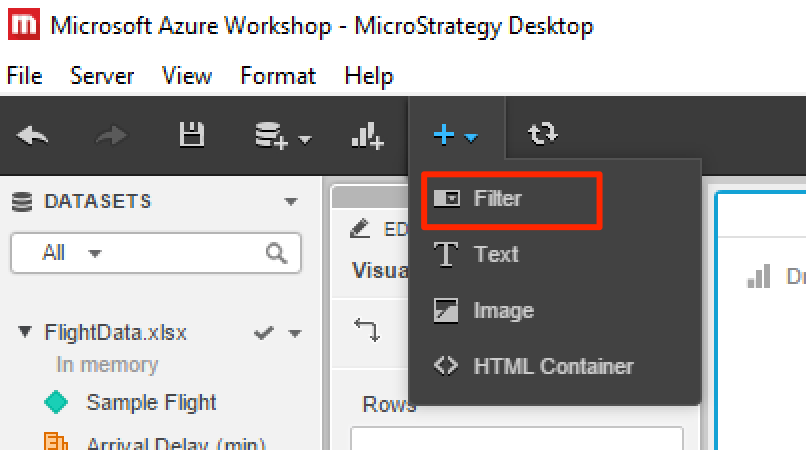


As now we have all the data ready, let’s continue to finalize our dashboard to make it more visual appealing.

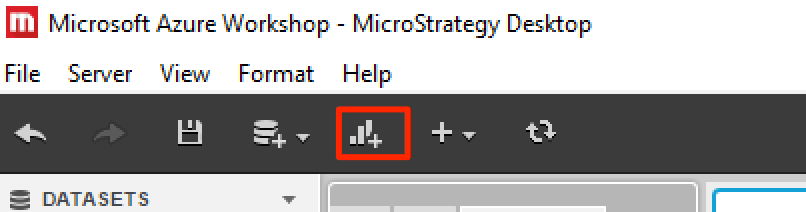
1. Clearing the currently grid, by clicking on the “Remove Data” button.



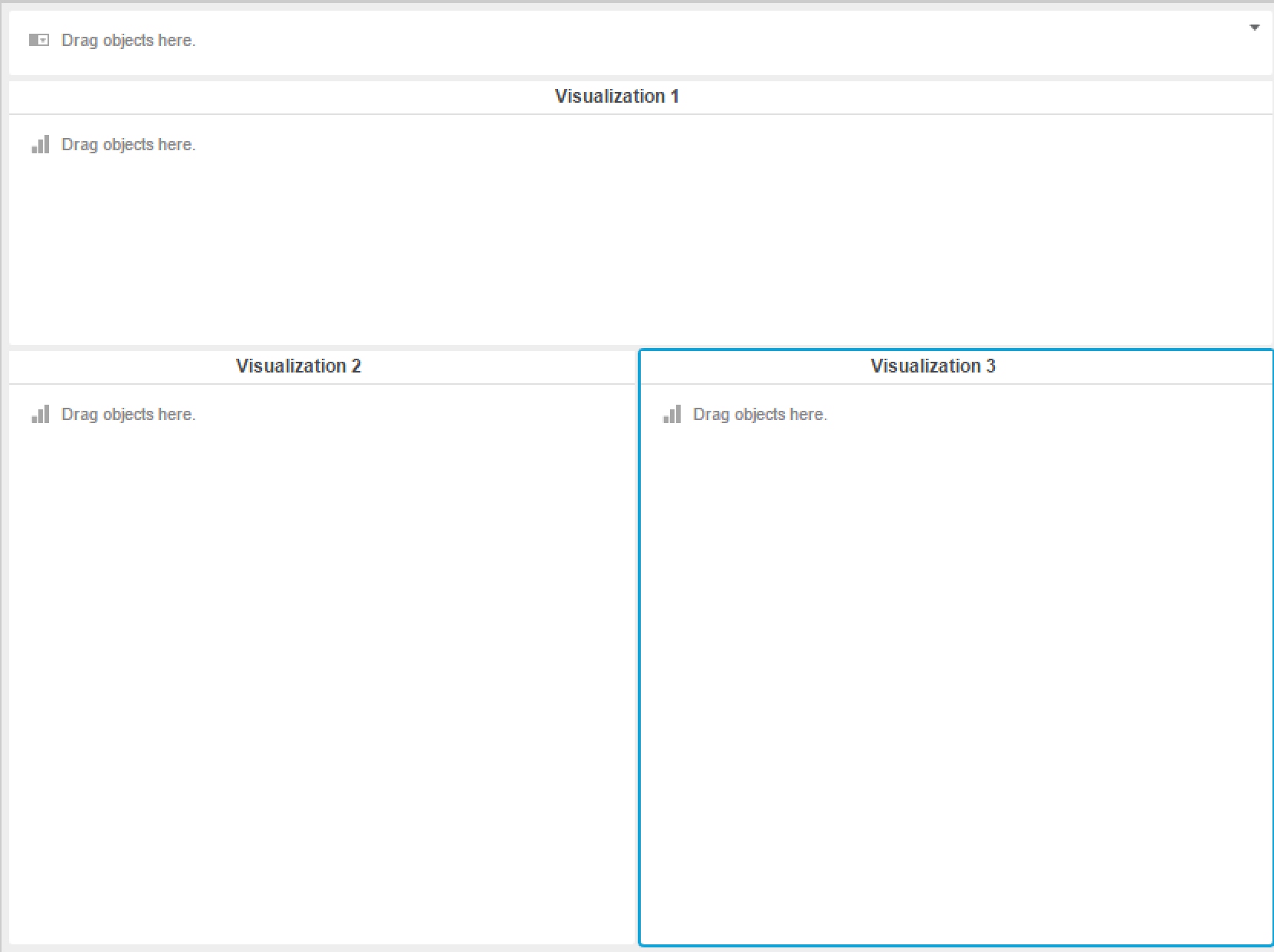
1. From the tool bar, add Filter to the dashboard.



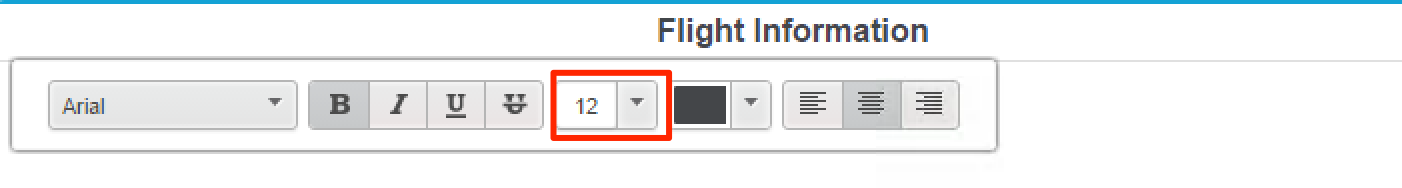
1. Add 2 more visualizations to the dashboard, by click on the “Insert Visualization” button twice.



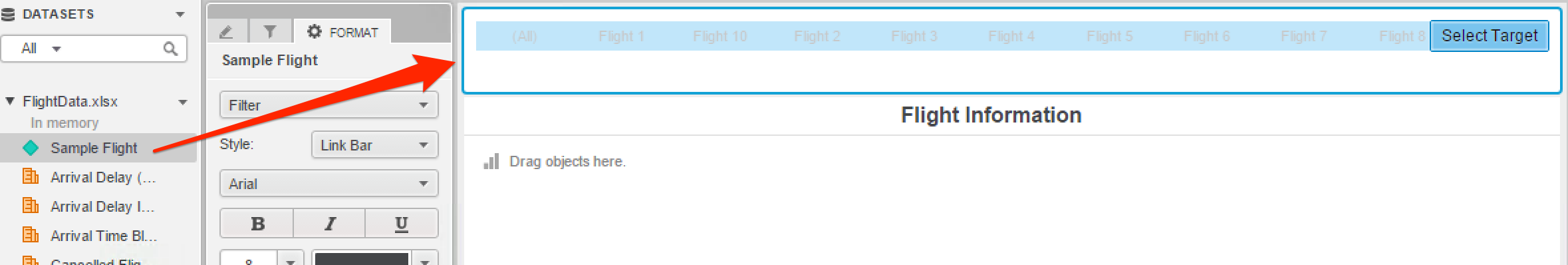
1. Drag and drop to layout the visualizations as following:



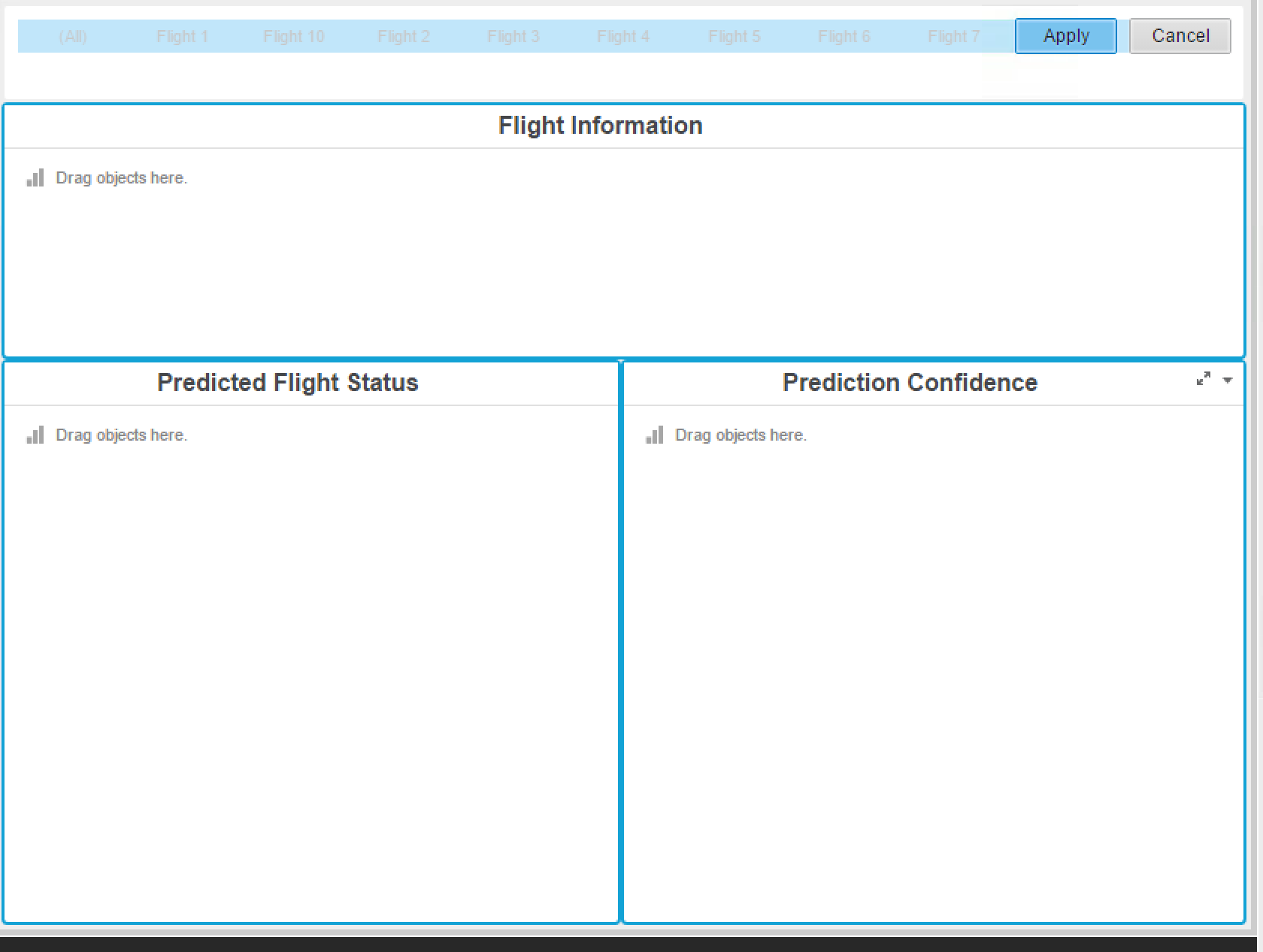
1. Double click on the visualization headers, and rename them to be “Flight Information”, “Predicted Flight Status” and “Prediction Confidence”.
2. Right click on the visualization headers, and select “Format”, and change the font size to be 12. Do the same for all three visualizations.



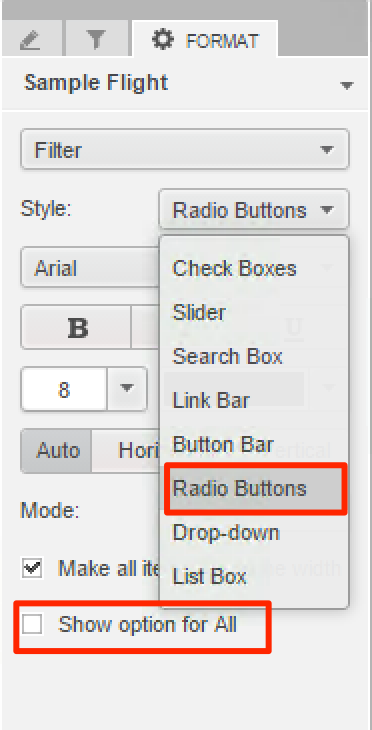
1. From the DATASETS panel, drag and drop attribute “Sample Flight” to the filter we added to the dashboard.



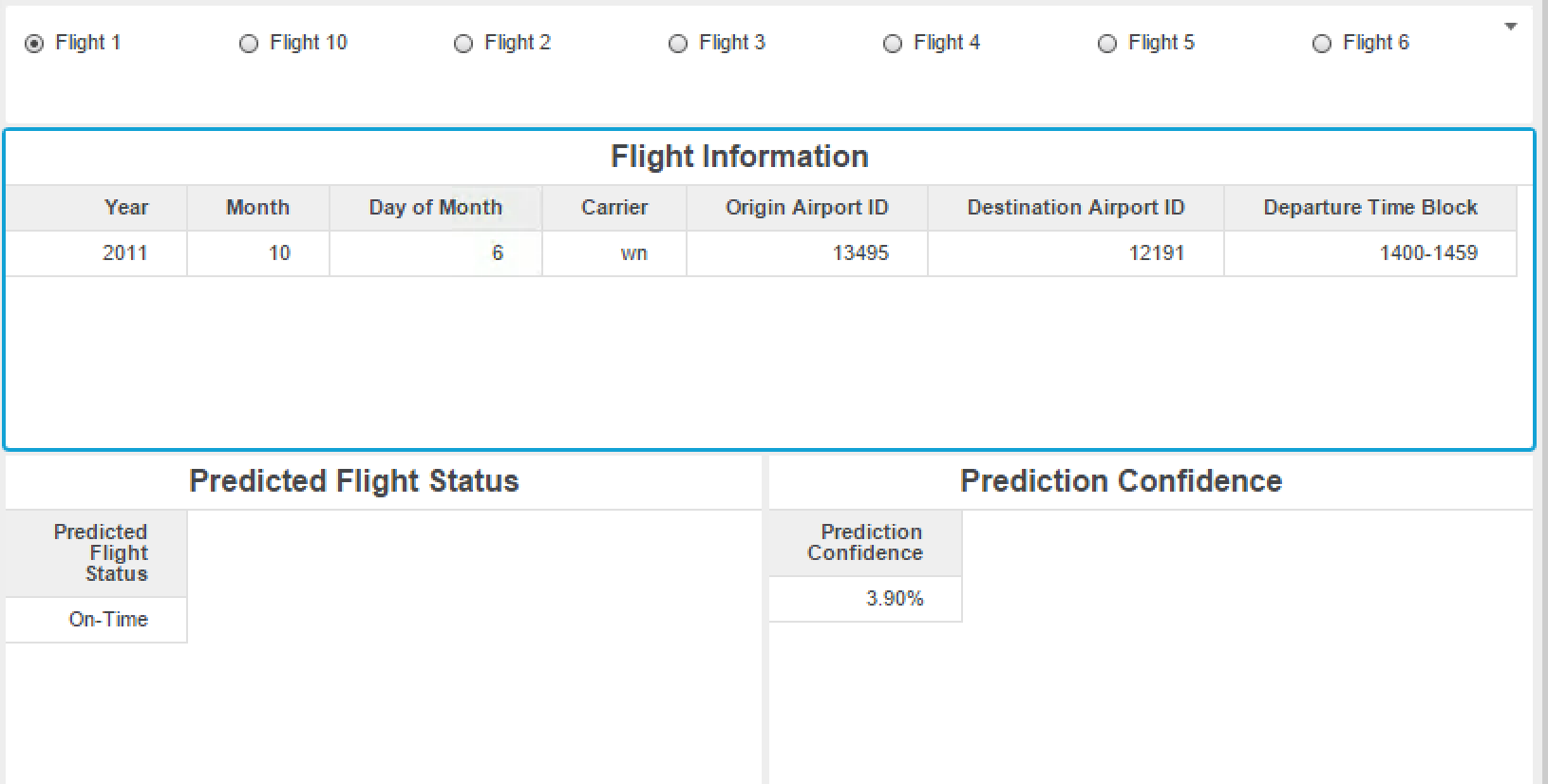
1. Click “Select Target” button, and click on all the 3 visualizations to select all of them (the visualization will have blue border when correctly selected), and then click “Apply”.



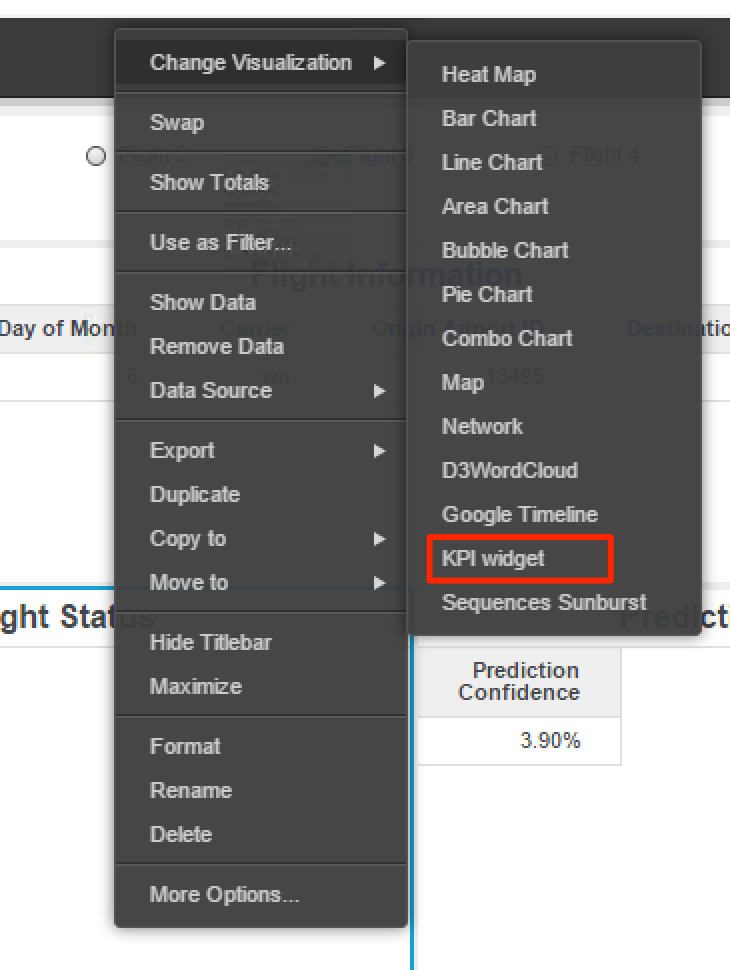
1. Keep the Filter selected (it has a blue border), and switch to the FORMAT panel in the middle of the dashboard. Set the style to be “Radio Buttons” and uncheck option “Show option for All”.



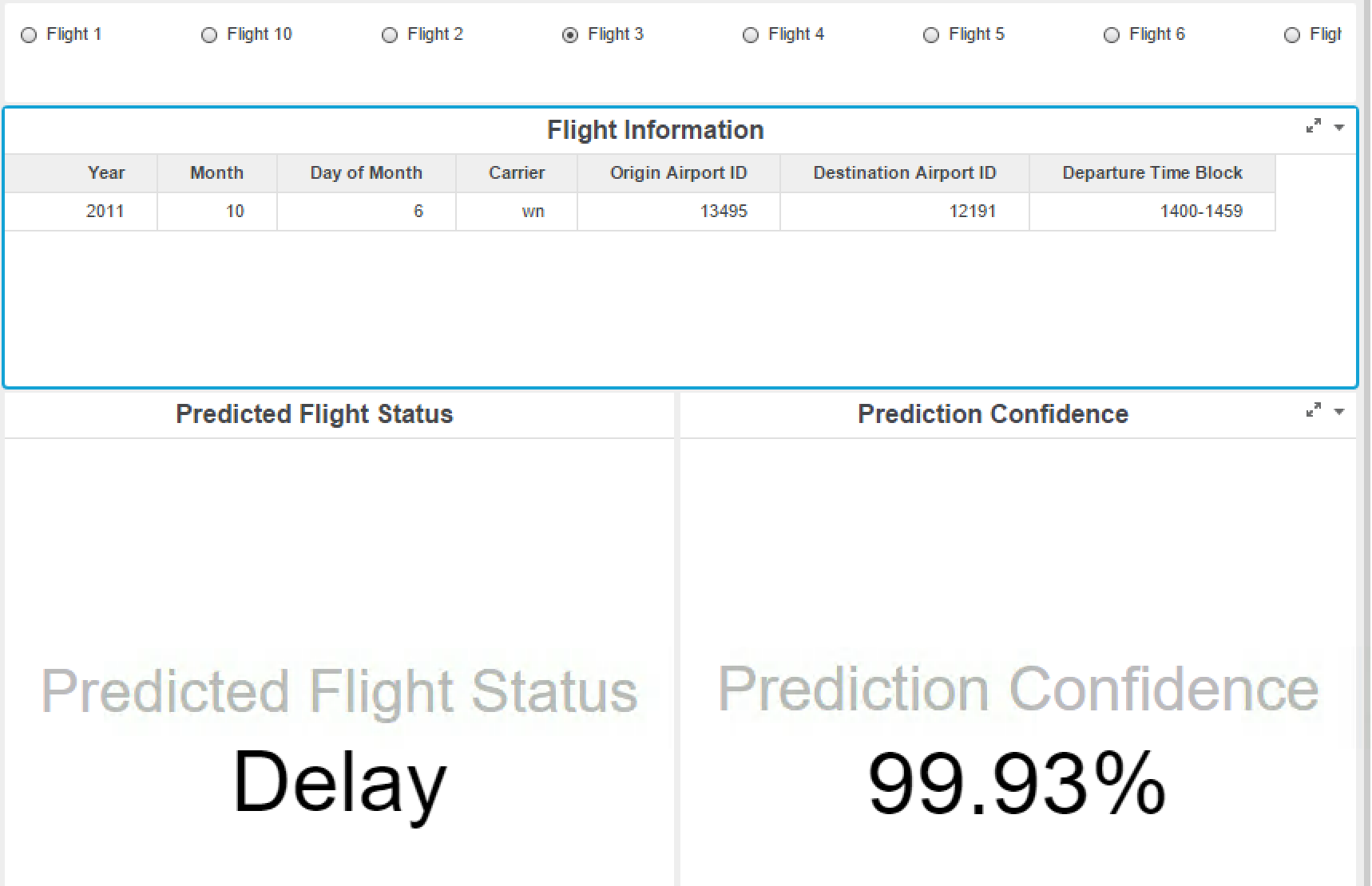
1. Select “Flight 1” on the Filter.
2. Select the visualization “Flight Information”, drag and drop metrics “Year”, “Month”, “Day of Month”, “Carrier”, “Origin Airport ID”, “Destination Airport ID” and “Departure Time Block” to the Metrics drop zone.
3. Select the visualization “Predicted Flight Status”, drag and drop metric “Predicted Flight Status” to the Metrics drop zone.
4. Select the visualization “Prediction Confidence”, drag and drop metric “Prediction Confidence” to the Metrics drop zone. Now you dashboard looks like following:



1. Click the little triangle on the top right of visualization “Predicted Flight Status”, under menu “Change Visualization” and select “KPI Widget”.

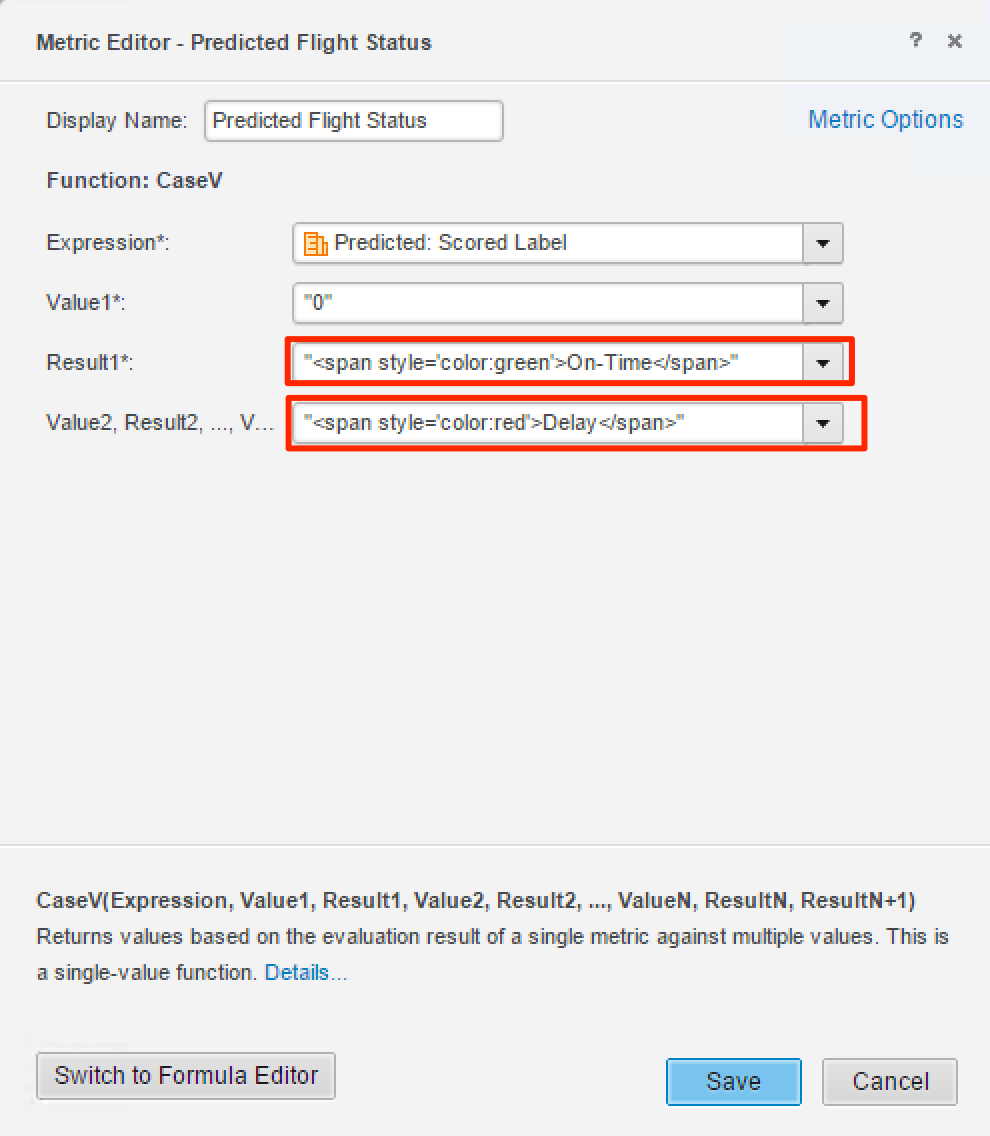


1. Do the same to visualization “Prediction Confidence”.
2. Now you get following dashboard, and if you switch selection of different flight, you will get its basic information and the prediction result.



The last change we want to do with the dashboard is to show Delay prediction result in red while On-Time result in green. To achieve this, we need to modify the metric “Predicted Flight Status”.

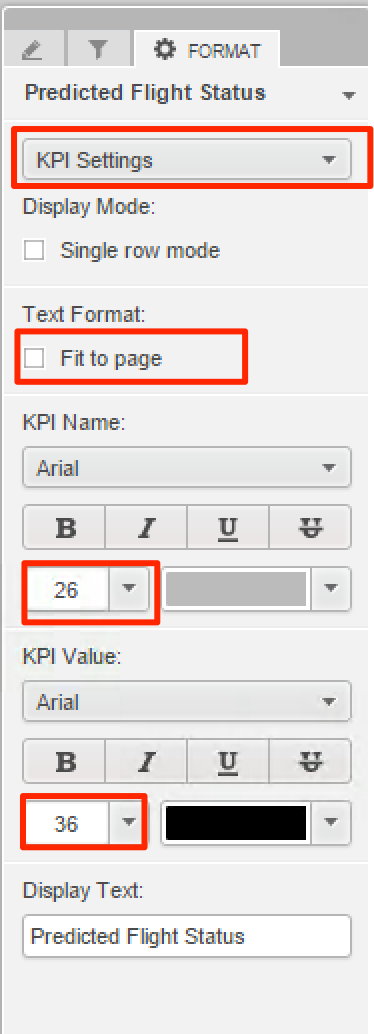
1. In the DATASETS panel, right click on metric “Predicted Flight Status”, and select Edit.
2. In the Metric Editor, set the “Result1” as: "<span style='color:green'>On-Time</span>", and “Result2” as: "<span style='color:red'>Delay</span>"



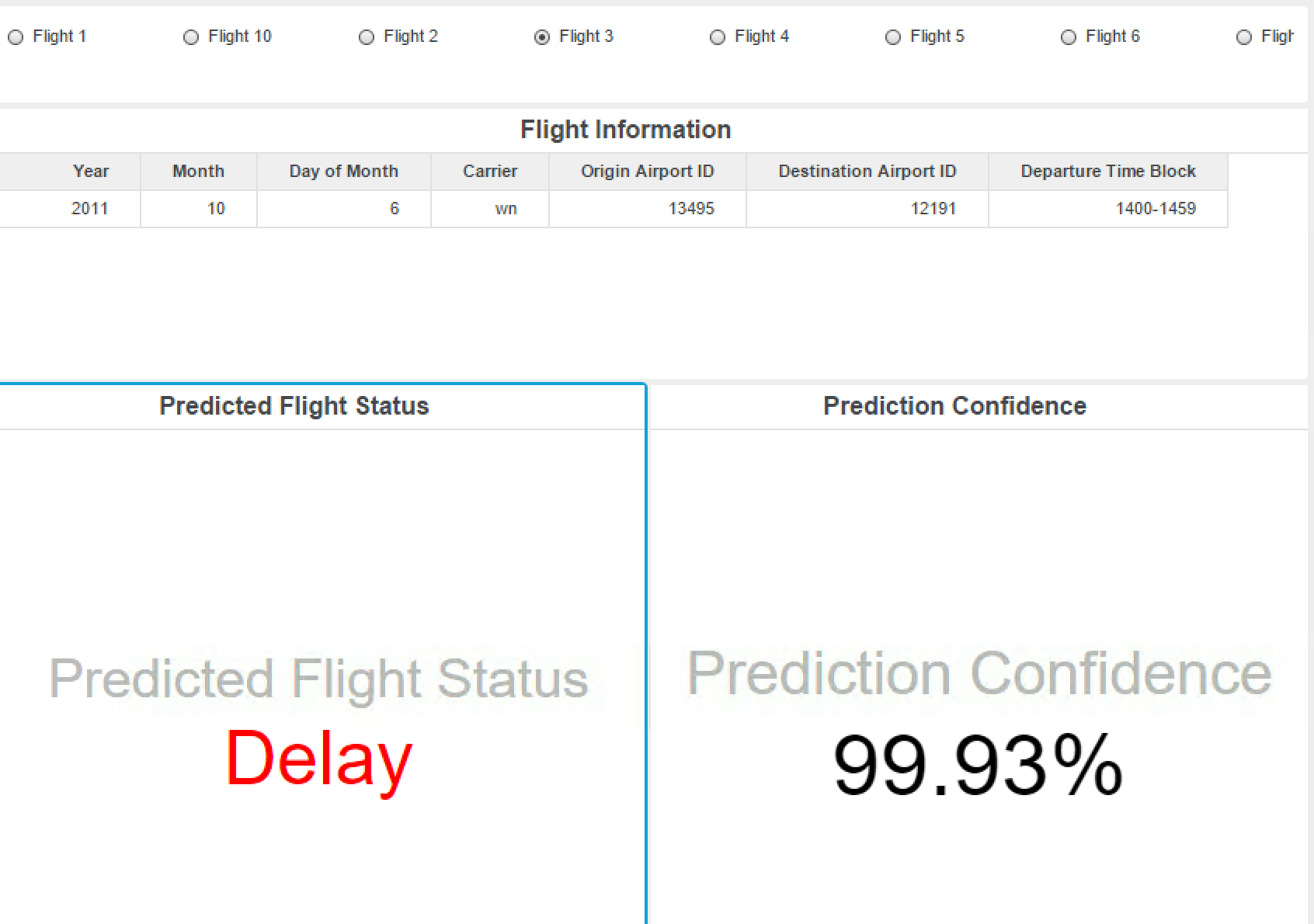
1. Click Save and switch selection of different flights, and you will find the Delay status is in red and On-Time status is in green.

However, the font for the flight status looks too small, to make it larger:

1. Select the visualization “Predicted Flight Status”, and switch to “FORMAT” panel in the middle of the dashboard.
2. Select “KPI Settings” in the drop-down list, uncheck “Fit to page”, set font size to be 26 for KPI Name and 36 for KPI Value.



1. Then we get own final dashboard.



Congratulations! You’ve now learned how to use Machine Learning in Azure to make a flight prediction by incorporating it with MicroStrategy Desktop to provide inputs and create dashboards of the result.