

Microsoft Azure

Machine Learning

Lab

**Lab Creator**: Fidan Boylu Uz

**Contributors**: Danielle Dean, Muxi Li

Table of Contents

[Requirements 3](#_Toc436253189)

[Executive Overview 3](#_Toc436253190)

[Download materials (including this documentation) 4](#_Toc436253191)

[First time set-up instructions 4](#_Toc436253192)

[Uploading a Dataset from Local Machine to Azure ML 5](#_Toc436253193)

[Data exploration 6](#_Toc436253194)

[Through Experiments 6](#_Toc436253195)

[Through IPython Notebook 8](#_Toc436253196)

[Training and testing binary classification model 9](#_Toc436253197)

[Parameter optimization 14](#_Toc436253198)

[Performance Evaluation 16](#_Toc436253199)

[Executing R scripts 18](#_Toc436253200)

[Creating a web service for binary classification model 22](#_Toc436253201)

[Azure ML Web service call from Excel 24](#_Toc436253202)

[Visualizations for classification using Power BI 26](#_Toc436253203)

[Clustering 29](#_Toc436253204)

[Through Experiment Canvas 29](#_Toc436253205)

[Through IPython Notebook 30](#_Toc436253206)

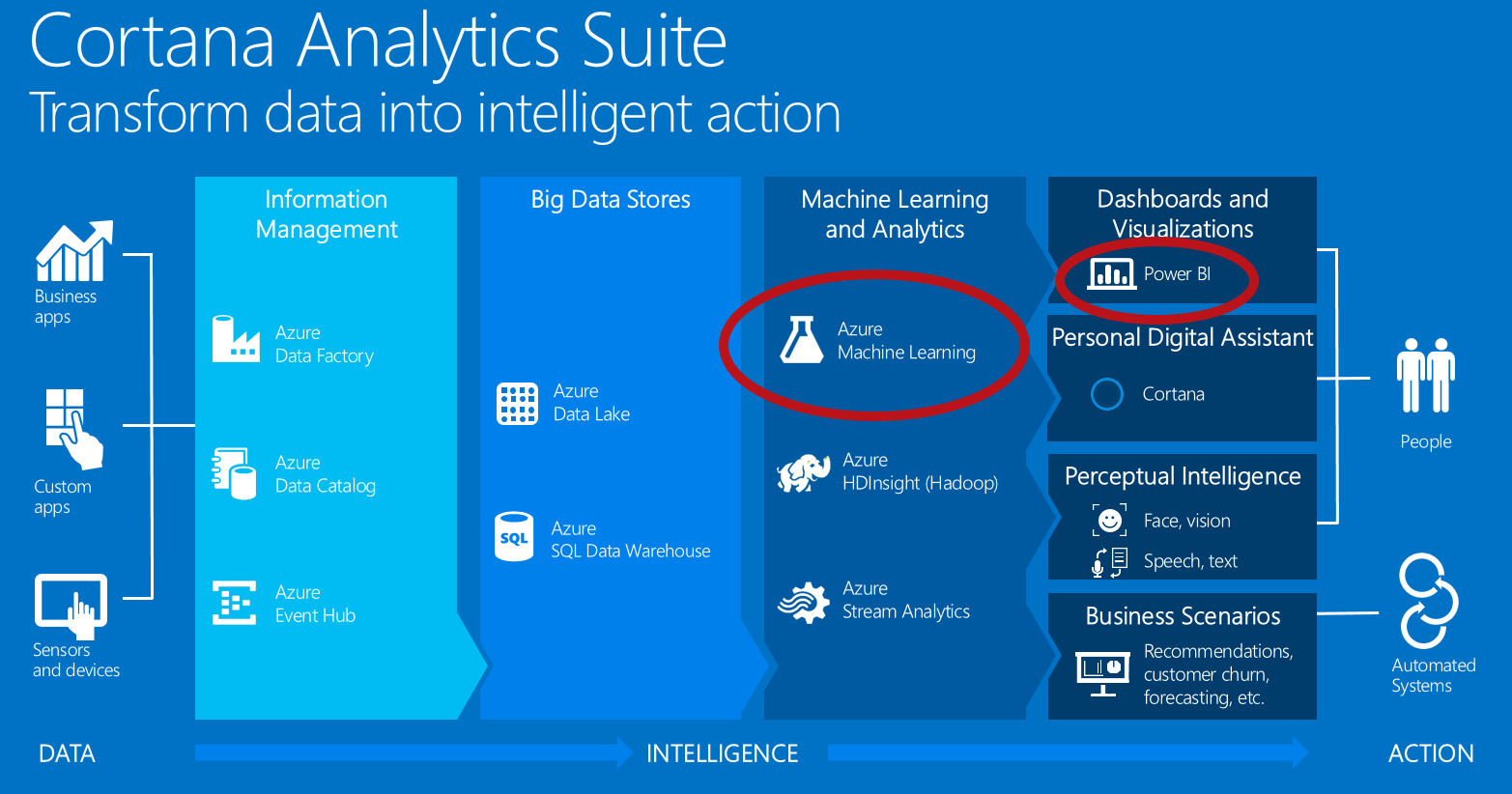
[Summary 30](#_Toc436253207)

# Requirements

1. Laptop
2. Access to internet
3. Internet Browser such as Internet Explorer or Chrome

# Executive Overview

The purpose of this lab is to introduce you to hosting machine learning models in Microsoft Azure cloud, including basic design, experimentation and development tasks. In the 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 data scientists and domain specialists to offer customers end-to-end solutions by significantly reducing the complexity to build predictive models. Azure ML provides an interactive and easy to use web-based interface with a drag-and-drop authoring model and a catalogue of modules that encapsulate functionality for the end-to-end model construction workflow. Azure Machine Learning is part of the Cortana Analytics Suite, a fully managed big data and advanced analytics suite that enables you to transform your data into intelligent action. This lab will also cover the basics of Power BI Desktop, the report authoring tool that can be downloaded for free, which connects to Power BI – a collection of online services and features that enables you to find and visualize data, share discoveries, and collaborate in intuitive new ways.



As a part of the lab, you will be creating two different models based on the dataset called **Churn**. The dataset consists of records belonging to 4667 customers of a fictitious telecom service provider. The columns of the dataset hold information such as the length of customer account, total day, and night, evening and international minutes used.

The first model you will create is called churn analysis known as customer attrition which is the problem of identifying the customers who are likely to leave a service or a business. The goal of the analysis is to contact these high risk individuals and take necessary actions such as providing special offers and discounts to prevent them from leaving the business. You will model the problem using the binary classification technique. Additionally, sections are provided to create a web service for the model and visualizing the classification results using Power BI desktop.

The second model you will create is a segmentation model where the objective is to find natural clusters of customers within the data sets who have similar characteristics. This is also extremely beneficial to understand the customer base for targeted marketing applications where the goal is to target the right individuals in order to grow the business.

# Download materials (including this documentation)

To make following along easier, you can download this documentation at <http://aka.ms/AzureMLChurnTutorial>. You also need to go to that URL to download the data that is used in this tutorial.

**Step-by-step**

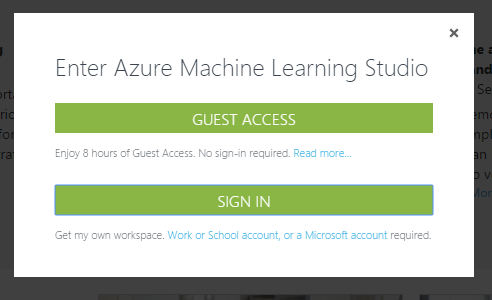
1. Go to: <http://aka.ms/AzureMLChurnTutorial> and download all the materials by clicking the **Download ZIP** button.
2. Extract (un-zip) these files to have a local copy.

# First time set-up instructions

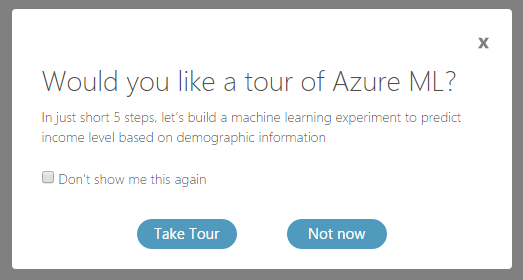
You will need an Azure Machine Learning workspace for this lab. There is a free and a standard tier of Azure Machine Learning – we will use the free tier in this lab. For more details, see pricing page here: <https://azure.microsoft.com/en-us/pricing/details/machine-learning/>. This step takes about 10 minutes to complete.

**Step-by-step**

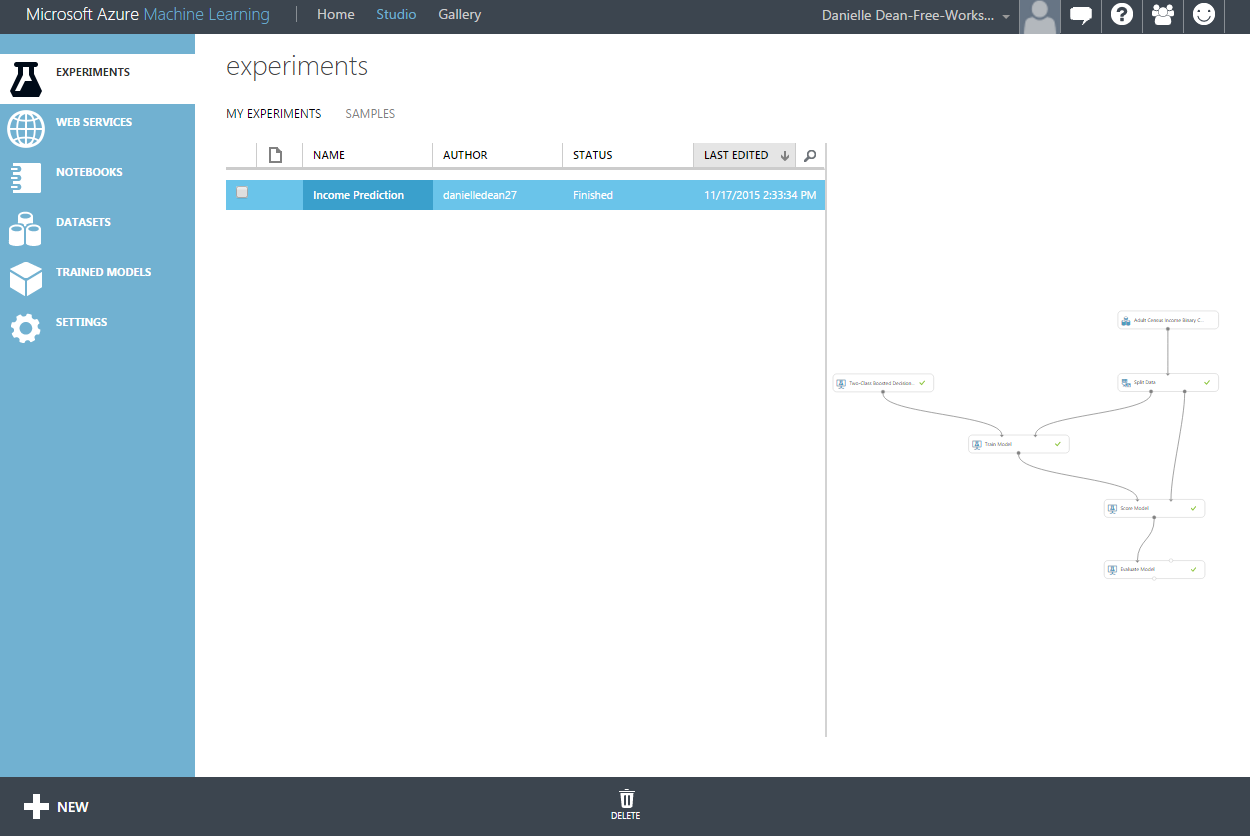
1. Go to <https://azure.com/ml>. Click on **Get Started Now** link on top-middle part of the page.
2. We recommend doing ‘**Sign In’** option to create a free tier workspace, so that the work you do as part of the tutorial today can be later accessed. If this doesn’t work for you, **Guest Access** will also work for the tutorial (but you will not be able to access your work again another day). If your email address is already a “Microsoft account” then simply sign in, otherwise you can click to ‘**Don't have a Microsoft account? Sign up now**’ and associate your email with a Microsoft account to log in.



1. Once you have signed in, Azure ML Studio will ask you if you’d like to take a tour. Click on “**Take Tour**” and walk through the steps to understand the basics of the Studio.



1. When you are finished, click on “finished” to get back to the Studio. You should then have a workspace that looks something like this:



# Uploading a Dataset from Local Machine to Azure ML

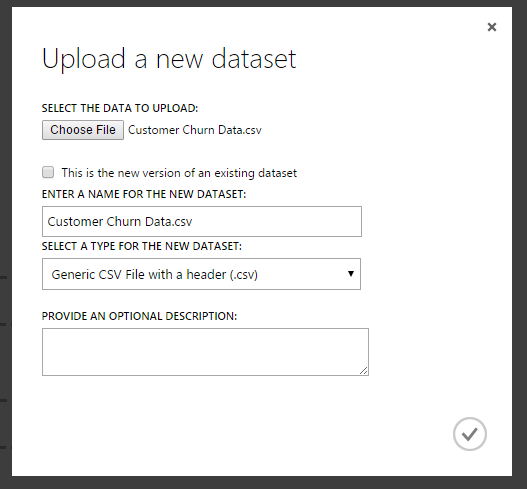
In this exercise, you will upload a new data set stored on your local machine in the form of a .csv file. This step takes about 5-10 minutes to complete.

**Key Points**

1. The table includes information about 4500 fictitious individuals that are subscribed to a mobile telecommunications service provider.
2. The information contained in the table includes the length of the account, plan subscriptions, international minutes, etc.

**Step-by-step**

1. The data file is part of the ZIP file that was downloaded in the first part of the tutorial. Once the folder is extracted, the CSV is called **CustomerChurnData.csv**.
2. Click on **+New** link at the bottom left corner of Azure ML Studio. Click **DATASET** on the left bar and then **FROM LOCAL FILE**. Browse to the location where you downloaded the file and select the file **CustomerChurnData.csv**. Notice that the type of the data set is set to **Generic CSV File with a header (.csv)** and click the check mark at the bottom of the window.



1. You now have a new dataset called CustomerChurnData.csv which you will find under My Datasets category on the modules list (you will see this in Section 4 that follows).

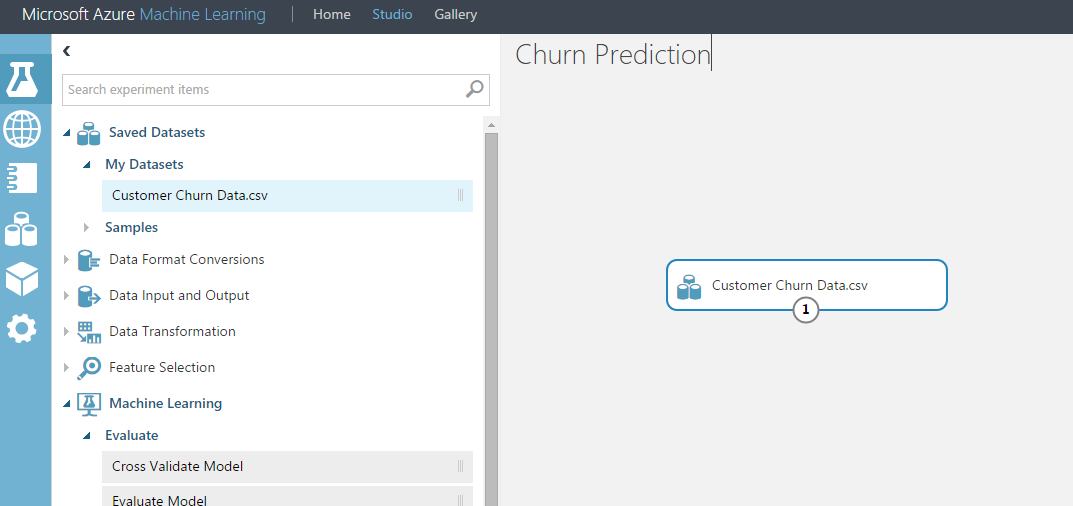
# Data exploration

First step in any data science project is to explore and understand the data. This requires some statistical analysis and visualizations to get a better view about its properties which are important criteria when deciding which machine learning model fits well for the data. To do this, we first need to create an Azure ML experiment – experiments allows us to create machine learning workflows with the drag-drop environment, or we can create a IPython Notebook which allows us to use Python code interactively with the data. Then we can start exploring the data in more detail. In the steps that follow, we will do both options, so you can see what works best for you.

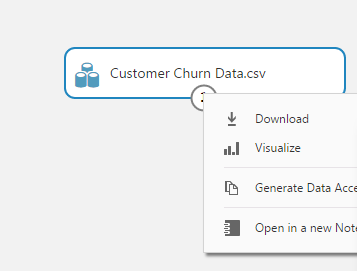
**Step-by-step**

## Through Experiments

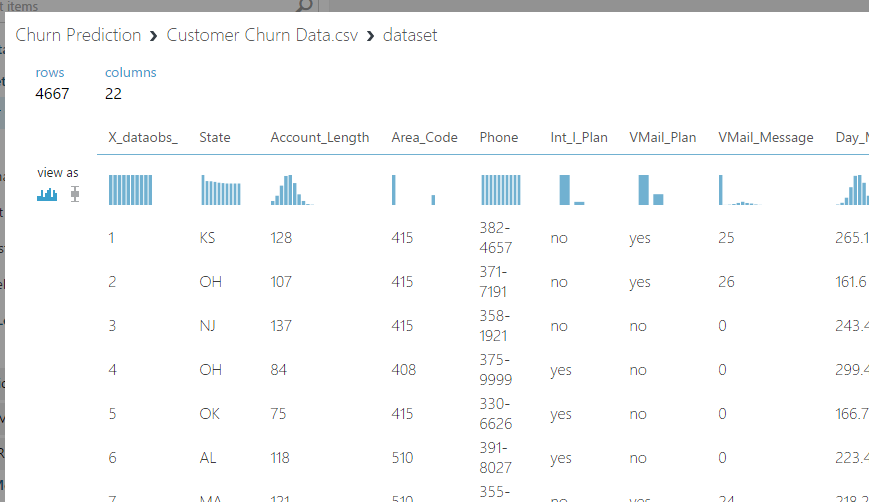
1. Click on **+New** link at the bottom left corner of Azure ML Studio and click on **EXPERIMENT** and then click on **Blank Experiment** on the top left corner. This will open a blank experiment.
2. You should see the Customer Churn Data within the **My Datasets** section of **Saved Datasets**. Drag the module onto the experiment canvas so we can start the experiment with that data. Rename the experiment at the top of the screen to **Churn Prediction**.



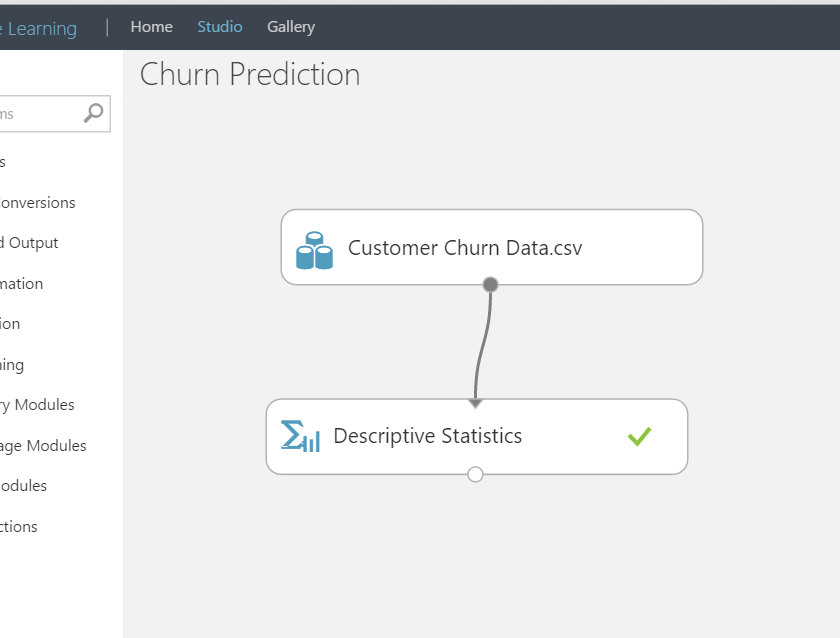
1. Select the dataset module that you’ve pulled onto the canvas, and right click the (1) that is at the bottom of the module. This is the output port. Select **Visualize** to see the data and make sure it was read in correctly.



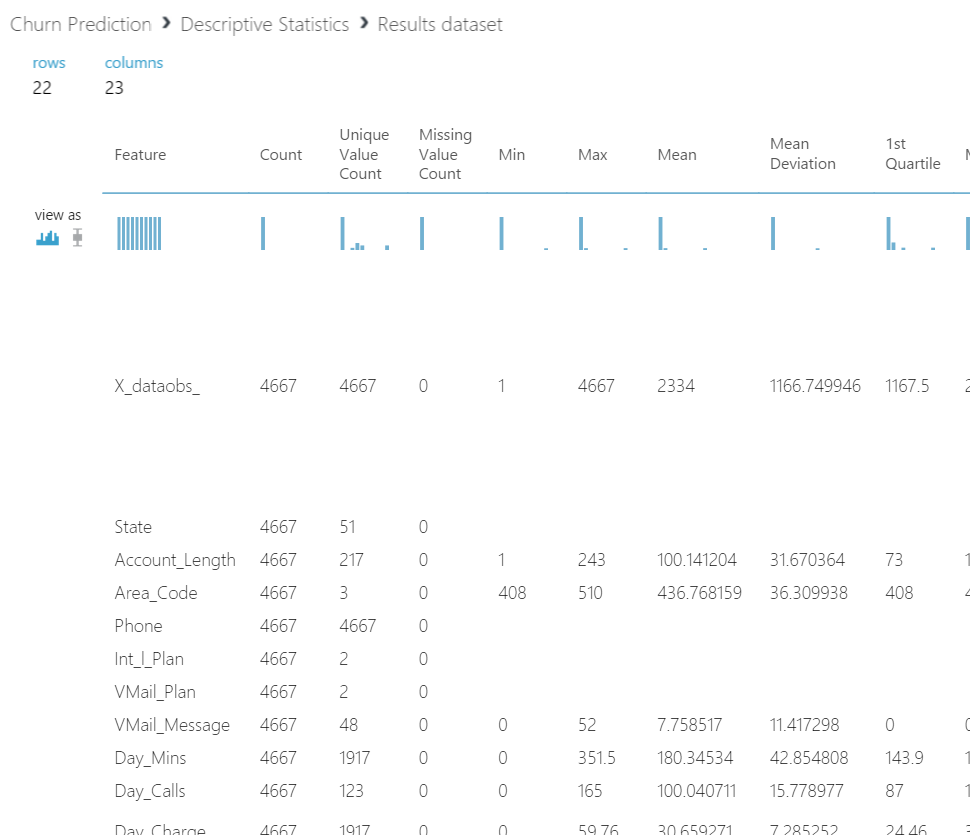
1. You should have 4,667 rows with 22 columns, and the data should have headers.



1. To understand the data in more detail, one option is to use the **Descriptive Statistics** module. You can do this by dragging it onto the canvas and connecting the data to the module.



1. After you’ve done this, run the experiment, then right click the output port of **Descriptive Statistics** and **Visualize** the results.



This gives descriptive statistics about the columns of the dataset.

## Through IPython Notebook

Microsoft has developed "Azure Machine Learning Python client library", which lets you access your Azure ML datasets from your local Python environment. With it, you can:

* Download datasets that are available in your ML workspace;
* Download intermediate datasets that were created within experiments;
* Upload new datasets and update existing datasets;
* Publish your defined functions as web services hosted on Azure;
* Consume published web services on Azure in your Ipython notebook.

Here, we are going to use this customer churn clustering problem as an example to demonstrate how you can utilize this library to integrate your Ipython Notebook with your Azure Machine Learning (Azure ML) workspace.

Note: This is a technology preview. The APIs exposed by the library and the REST endpoints it connects to are subject to change.

For more information, please see a complete documentation [here](https://github.com/DinoV/Azure-MachineLearning-ClientLibrary-Python/blob/master/README.md).

**Step-by-step**

1. Here, we have a pre-built notebook for you, in case you’re not familiar with the Python language or to just accelerate your work if you are familiar with Python. Click on **+NEW** at the bottom left of the screen, then select **Notebook** section, and click on **Python 2** Notebook – the code we have written only works with Python 2. Name your notebook (such as **Churn.ipynb**) and select ok to open it.
2. Now you have a blank IPython Notebook to begin exploring the data. Rather than starting from scratch, let’s load the notebook that is pre-built so you can get an idea of how this works.
3. Click on **File** then **Open** then **Upload** and browse to the location of the downloaded ZIP file which contains the IPYNB that you want to upload which is named **Customer Churn Clustering Analytics.ipynb** and select **Open**. Then select the blue **Upload** button to upload that notebook to the cloud. Refresh your page after a few seconds and the notebook should turn green with the word “running” written on the right. If it’s still uploading due to slow connection, you might have to wait a little longer and refresh the page again.
4. Click on **Customer Churn Clustering Analytics.ipynb** to open it within Azure Machine Learning Studio.
5. You will need to replace parts of the code, such as those surrounding workspace id and authorization token. Go to **Datasets** section of Machine Learning Studio, and then select the **Customer Churn Data.csv** dataset, then select the **Generate Data Access Code** at the bottom. Copy/paste the workspace\_id and the authorization\_token into the relevant section of the notebook.
6. Run code snippets by hitting **Shift+Enter** and feel free to modify or write new code to explore the data if you want. Stop when you get to the **Data Pre-Processing** part – we will return to this later.

# Training and testing binary classification model

There are five main machine learning techniques you can perform using Azure ML Studio. These are Anomaly Detection, Classification, Clustering, Regression, and Recommender methods. In this section, we will focus on creating a Binary Classification Model.

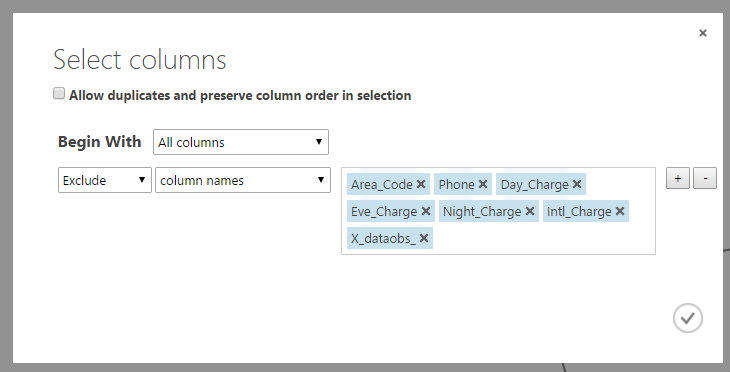
**Key Points**

1. There is an extensive library of binary classification algorithms included in Azure ML Studio. Most popular ones are Boosted Decision Trees, Logistic Regression and Support Vector Machines. You are strongly encouraged to research about these algorithms to get a better understanding of the techniques.
2. Before applying any machine learning model, an important step is the data cleaning and preparation. As an example, in this exercise, you will reduce down the dataset by eliminating highly correlated variables which we have preselected for you to eliminate.
3. In order to assess the quality of models, a common method in machine learning applications is to separate the data into **training** and **test** sets to measure the accuracy of predictions. In this part of the lab, you will first train the model with 70% of the total data points and test it with the remaining 30% of the data in order to assess the performance of the model. There are more sophisticated ways of validating such as cross-validation which aren’t covered in this section.

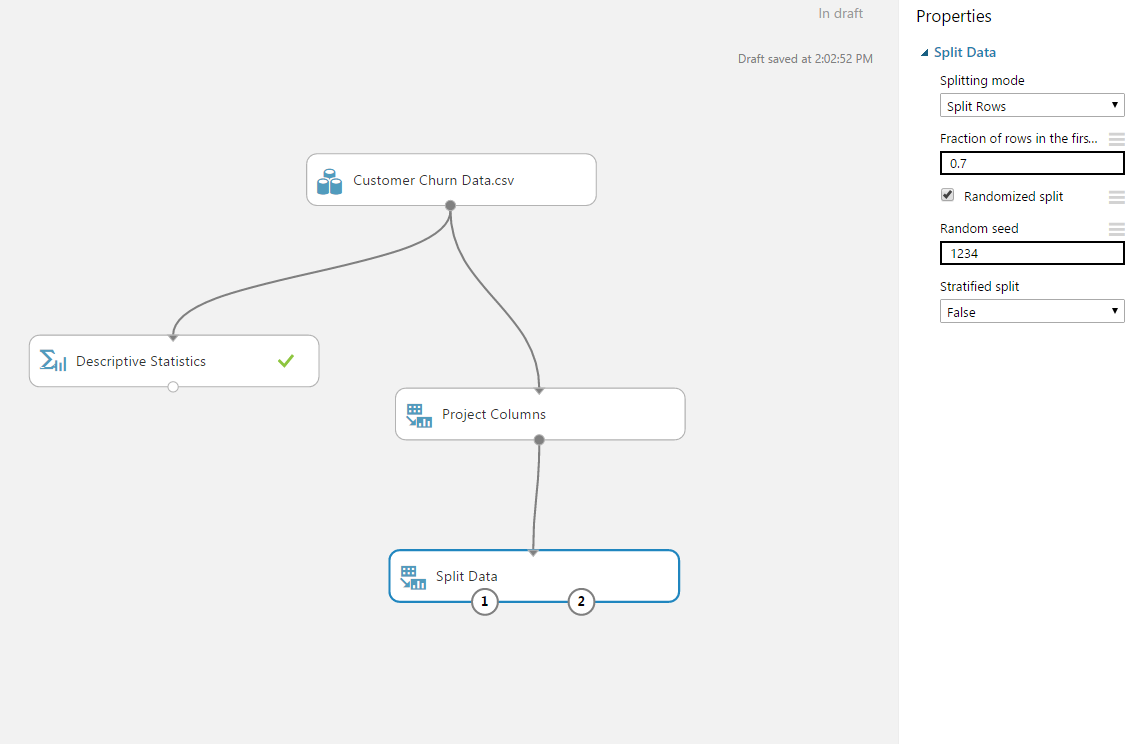
**Step-by-step**

In this part, you will extend the experiment for training a binary classification to identify customers who are more likely to churn. This step takes about 20 minutes to complete.

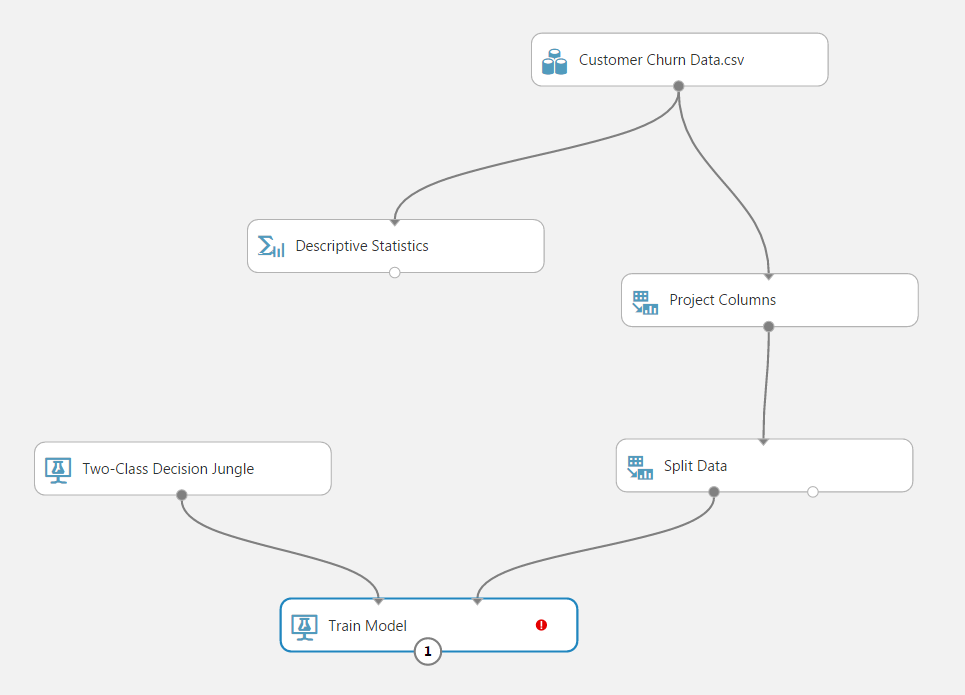
1. First, you will use the **Project Columns** module to exclude **Day\_Charge**, **Eve\_Charge, Night\_Charge** an**d Intl\_Charge** columns since these are highly correlated with **Day\_Mins**, **Eve\_Mins**, **Night\_Mins** and **Intl\_Mins**. You will also exclude **Area\_Code**, **Phone**, and **\_X\_dataobs\_**. To do this, drag the **Project Columns** module onto the canvas.
2. Click on the Project Columns module and click **Launch column selector** in the properties menu bar on the right side. Select *Begin Columns:* ***All columns*** and **Exclude**. In the text box, enter the following columns: **X\_dataobs\_, Area\_Code**, **Phone**, **Day\_Charge**, **Eve\_Charge**, **Night\_Charge**, and **Intl\_Charge,** click the check mark.



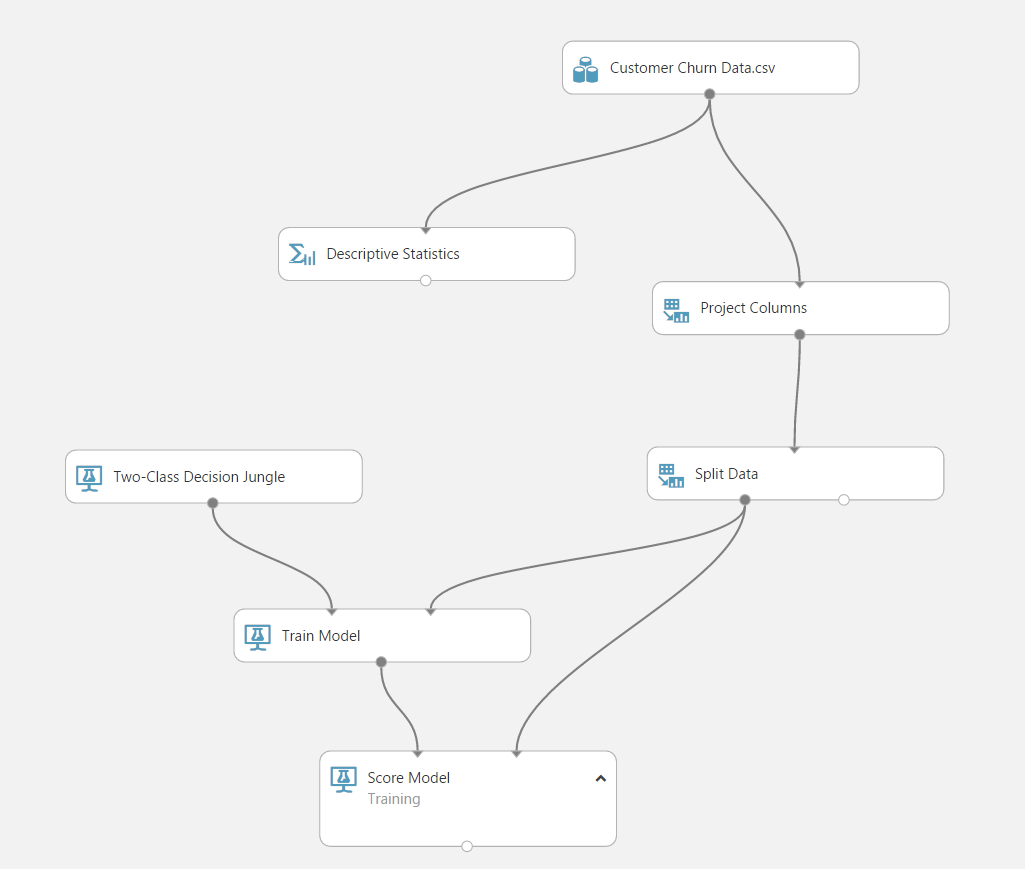
1. Next, on the left side navigation bar, type **Split**, drag/drop the module to the experiment and connect the output port of **Project Columns** module to the **Dataset** input of **Split**. Rearrange modules as in the screenshot below. Next, click **Split** and on the **Properties** navigation bar on the right side, type **0.7** in the textbox for **Fraction of rows in the first output dataset** and **1234** in **Random seed** textbox. Random seed is used to start the random process from the same seed every time the experiment is re-run.



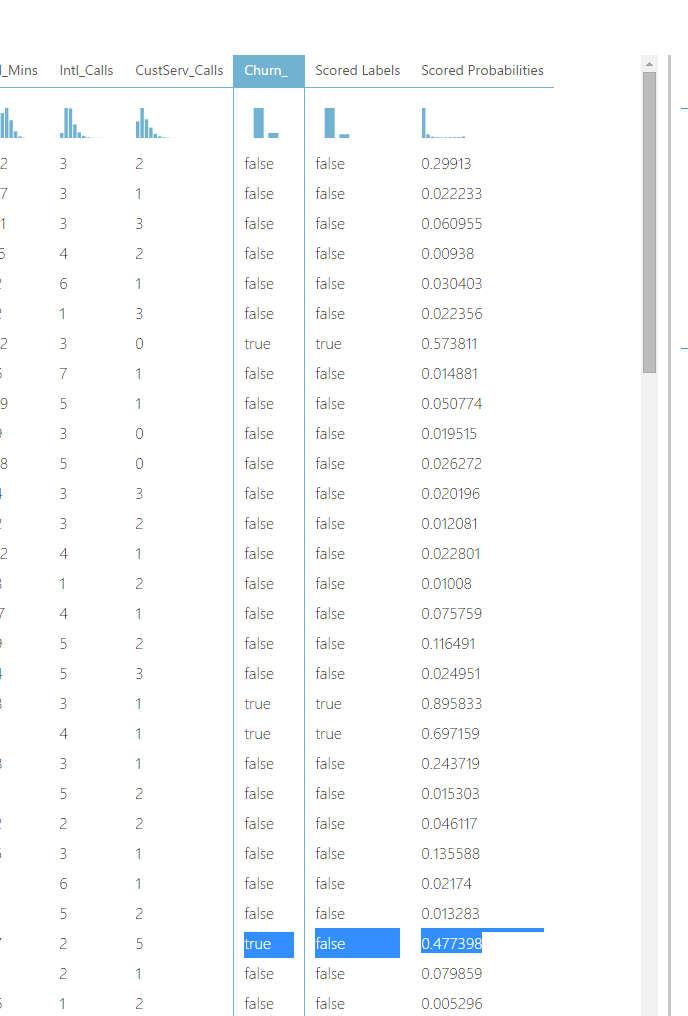
1. Now, on the left side navigation bar, type **Machine Learning** and under **Initialize Model, Classification** category**,** drag/drop **Two-Class Decision Jungle** moduleon the experiment canvas. [Decision jungles](http://go.microsoft.com/fwlink/?LinkId=403675) are a recent extension to [decision forests](http://go.microsoft.com/fwlink/?LinkId=403677). A decision jungle consists of an ensemble of decision directed acyclic graphs (DAGs). We will not go into the details of how the algorithm works here.
2. Next, under **Train** category, drag/drop **Train Model.** Connect **Results dataset1** output port of **Split** module to the **Dataset** input of **Train Model.** Also, connect **Untrained model** output port of **Two-Class Decision Jungle** to the **Untrained model** input port of **Train Model.**



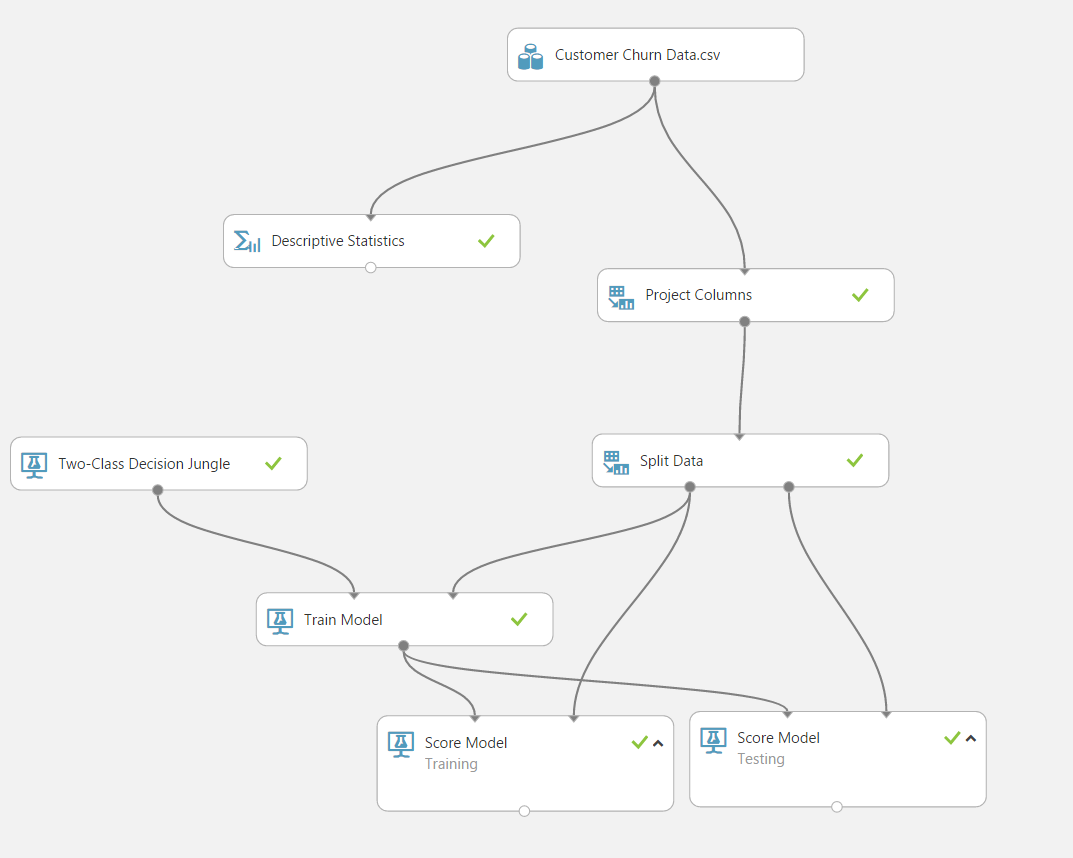
1. Click **Train Model** and on the Properties navigation bar on the right side, click **Launch column selector**, select **Column names** in the first drop down menu and type or select **Churn\_** in the textbox and click the check mark at the lower right corner.
2. Type **Score Model** in the search box on the left navigation pane and drag/drop the module to the experiment canvas. Double click that module and type **Training** in the Label box (use the arrow to unhide the label). Connect **Trained model** output of **Train Model** module to the **Trained model** input of **Score Model** module. Next, connect **Results dataset1** output port from the **Split** module to **Dataset** input port of **Score Model** module and click **Run** at the bottom menu bar.



1. Make sure all modules have green check marks and experiment is finished. Right click **Scored dataset** output port of **Score Model / Training** and select **Visualize**. In the output page, scroll right to the end and observe the **Scored Labels** and **Scored Probabilities** columns. These are the predictions that the trained model has made on the top 100 training dataset rows. Scroll down and compare **Churn\_** column with **Scored Labels** column to identify any false positives and false negatives. You will mostly find correct classifications where Churn\_ and Scored Labels columns have the same value. We will look into the classification error (false positives and false negatives) in the Performance Evaluation section. Close the page using the close button on the upper right corner.



1. For testing, search for **Score Model** module again and drag/drop on the experiment. Double click the module and type **Testing** in the Label box. Connect the **Results dataset2** output of the **Split** moduleto **Dataset** input of **Score Model - Testing**. Now, connect the **Trained model** output of **Train Model** to **Trained model** input of **Score Model / Testing** and **Run** the experiment.



1. Right click **Scored dataset** output port of **Score Model – Testing** and select **View Results**. In the output page, scroll right to the end of the page to observe the **Scored Labels** and **Scored Probabilities** columns. These are the predictions that the trained model has made on the test dataset. Scroll down and compare **Churn\_** column with **Scored Labels** column to identify any false positives and false negatives. Close the page using the close button at the upper right corner.

# Parameter optimization

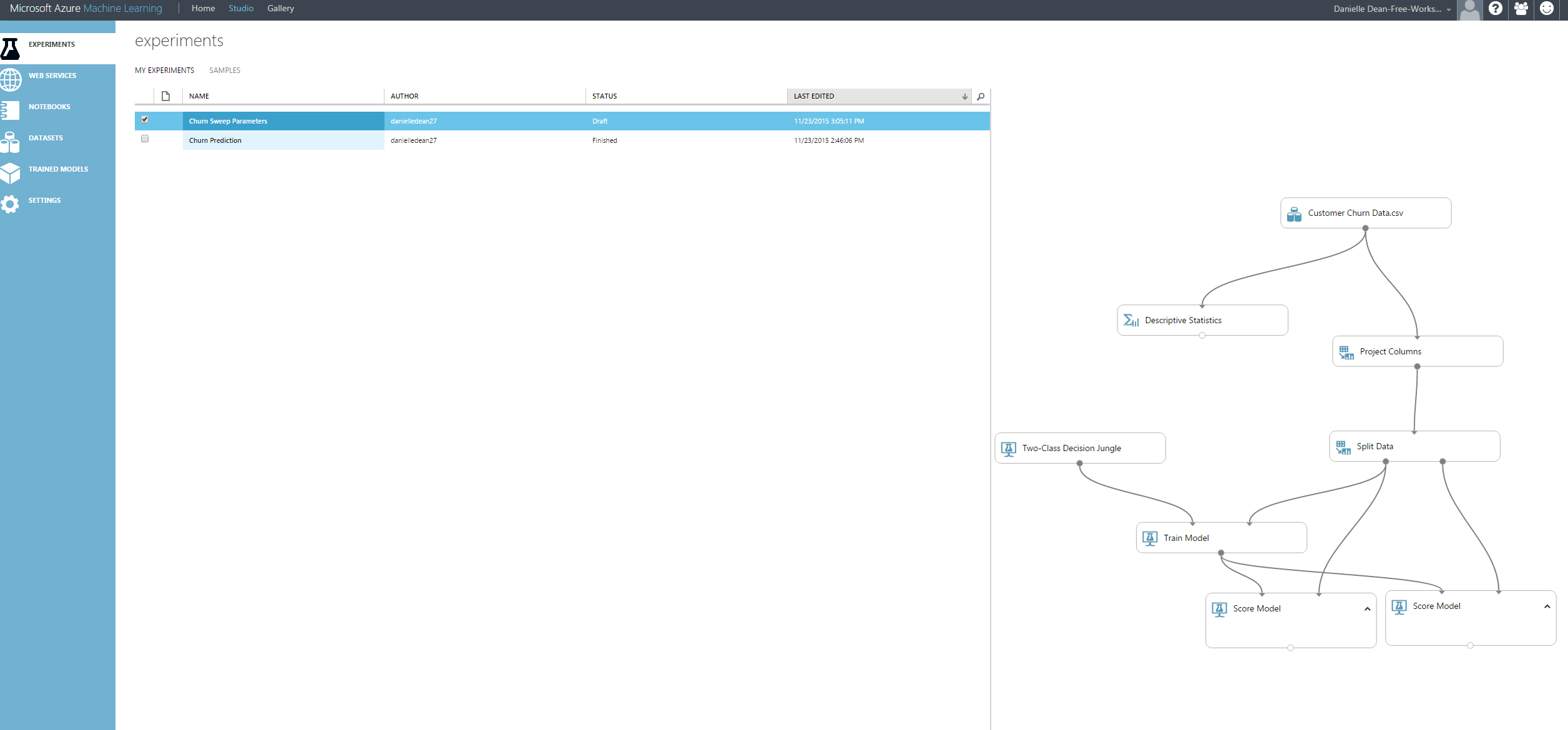
As a next step, you will use **Sweep Parameters** module to fine-tune your models.

**Key Points**

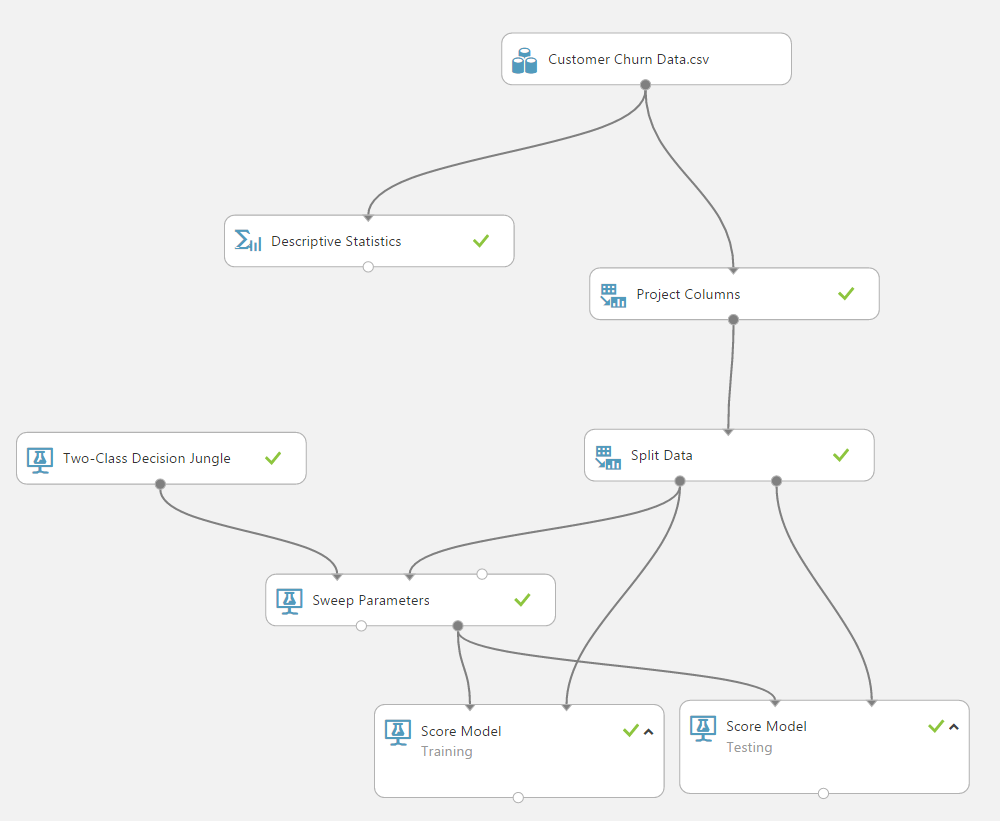
1. Machine Learning algorithms depend on parameters that can change the model performance significantly. The best values for these parameters are not computed while training the models so it is important to try different parameters to find the best performing parameter set.
2. As an example, some of the parameters of Decision Jungle algorithm are the number of decision DAGs (the total number of graphs that can be created in the ensemble) and the maximum depth of the decision DAG.

**Step-by-Step**

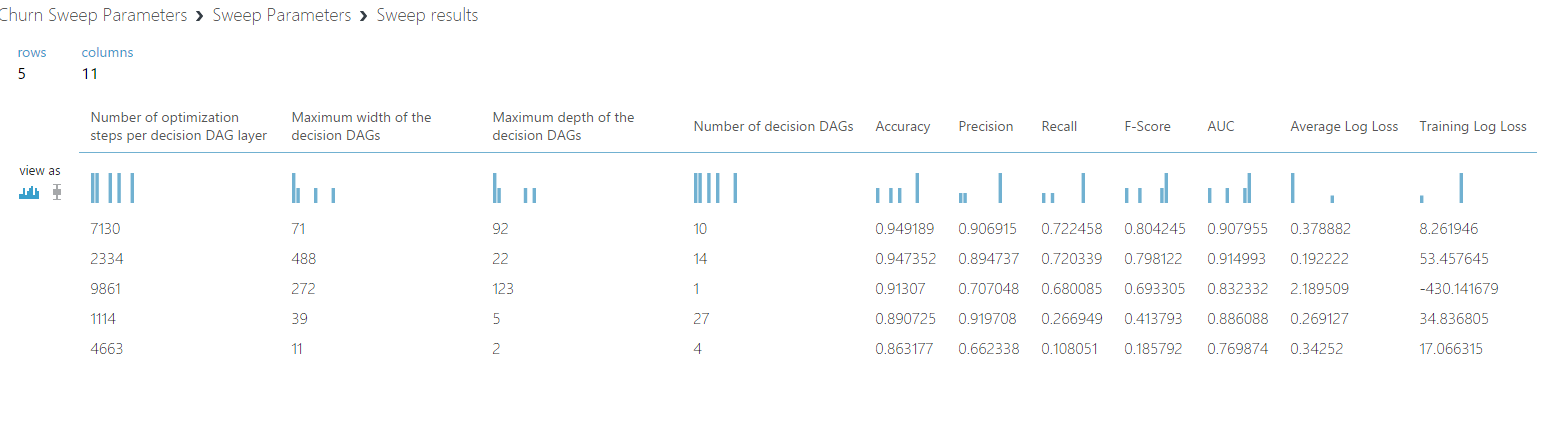
1. First, we will create a new experiment called **Churn Sweep Parameters**. Click the **SAVE** then **SAVE AS** button at the bottom bar and type Churn Sweep Parameters for the experiment name and click the check mark. You will see that your experiment is renamed. You now have two experiments in your workspace (three if you did the “tour” of Azure ML). You can verify that by clicking the experiments tab on the leftmost navigation bar. Notice that a preview of the experiment is displayed on the right side.



1. Go back to Churn Sweep Parameters experiment by clicking on the name of that experiment in the experiment list and in the search experiment items box, search for Sweep Parameters module and drop it on your experiment to replace Train Model module (the third input port will be blank and the second output port is used to connect to both Score Module modules). Observe the Properties of Sweep Parameters. It gives you the option to randomly sweep or sweep over a grid of the parameters of the training algorithm. Refer to the quick help page for more information on the Sweep Parameters module. In the properties pane, leave all options as default. However, you need to use column selector to select **Churn\_** column as the label column.



1. Run the experiment. When finished, click on the left output port of Sweep Parameters and observe the random set of parameters that were used to run the experiment 5 times and their performance metrics. These will be discussed in the next sections.



1. Click on the experiments tab and return back to the Churn experiment.

# Performance Evaluation

In this exercise, you will assess the predictive power of the model you have built so far. You will observe the performance of the model on the training set and compare that to the test set to understand the trade-off between over fitting and generalization.

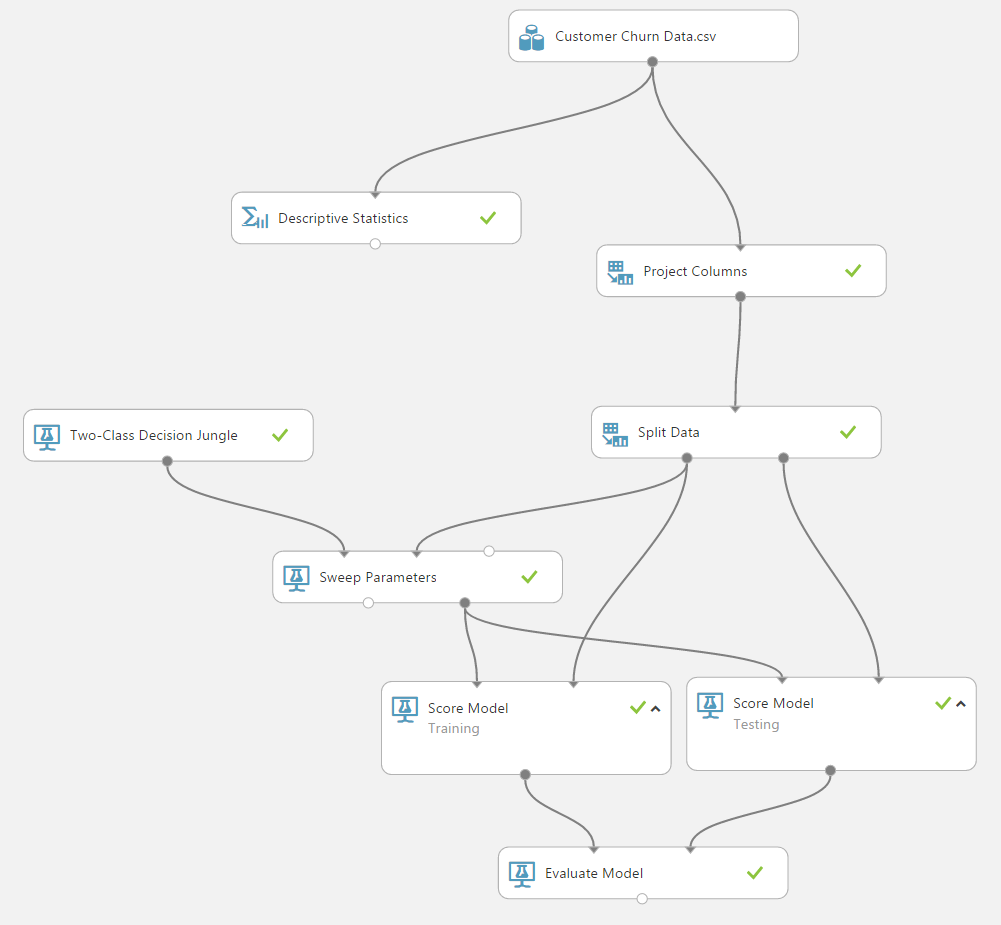
**Key Points**

1. To understand performance of a model, a few basic concepts must be introduced. For binary classification models, there are four possible outcomes:
   1. True Positive (TP) - a positive instance that is correctly classified as positive;
   2. False Positive (FP) - a negative instance that is incorrectly classified as positive;
   3. True Negative (TN) - a negative instance that is correctly classified as negative;
   4. False Negative (FN) - a positive instance that is incorrectly classified as negative).
2. There are three related metrics that are used to measure performance.
   1. Precision is calculated by TP/ (TP+FP)
   2. Recall is calculated by TP/ (TP+FN)
   3. Accuracy is calculated by TP+TN/ (TP+FP+TN+FN).
3. ROC curve is an important performance measure which will be explained in this section.

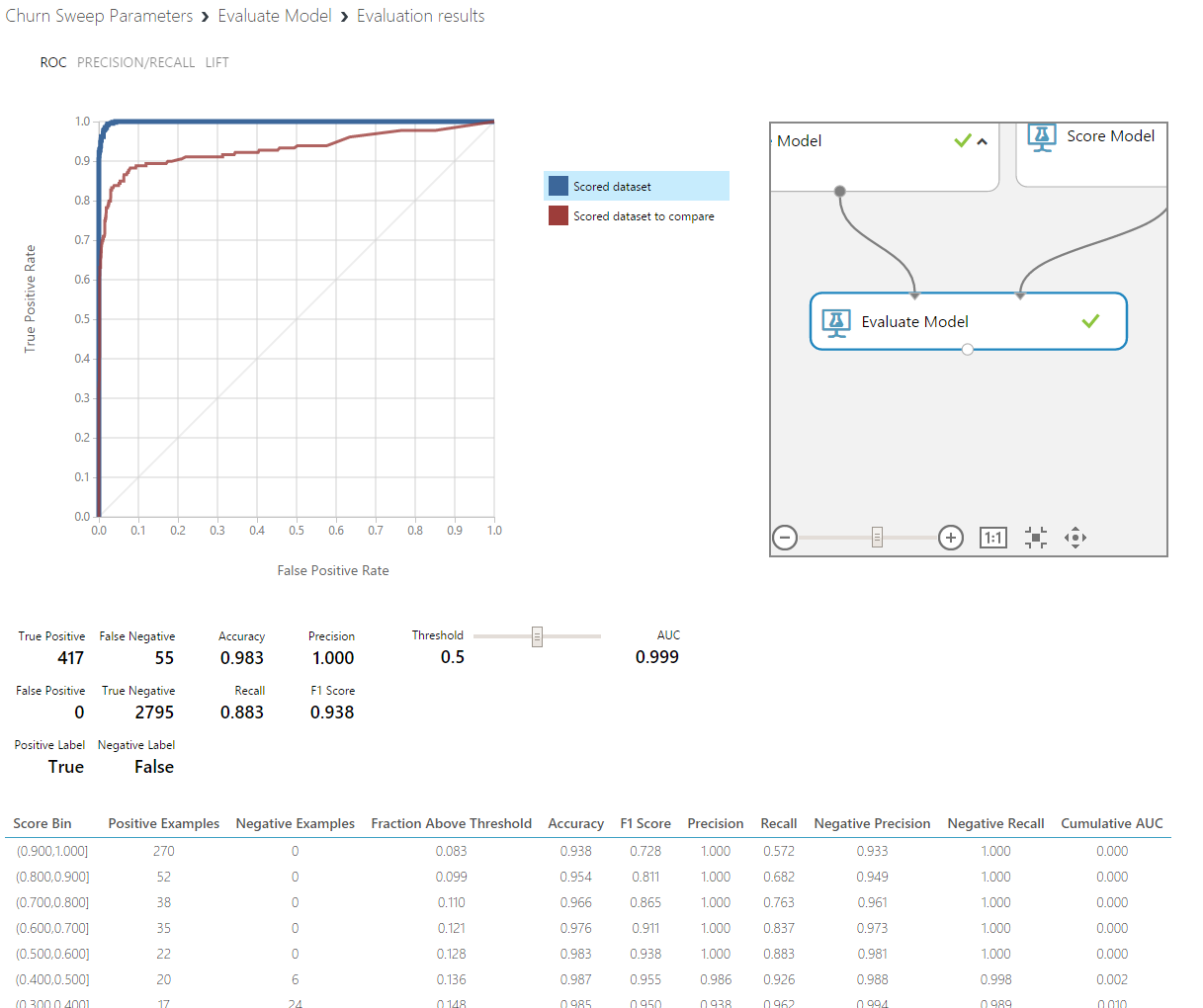
**Step-by-Step**

This exercise takes about 10 minutes to complete.

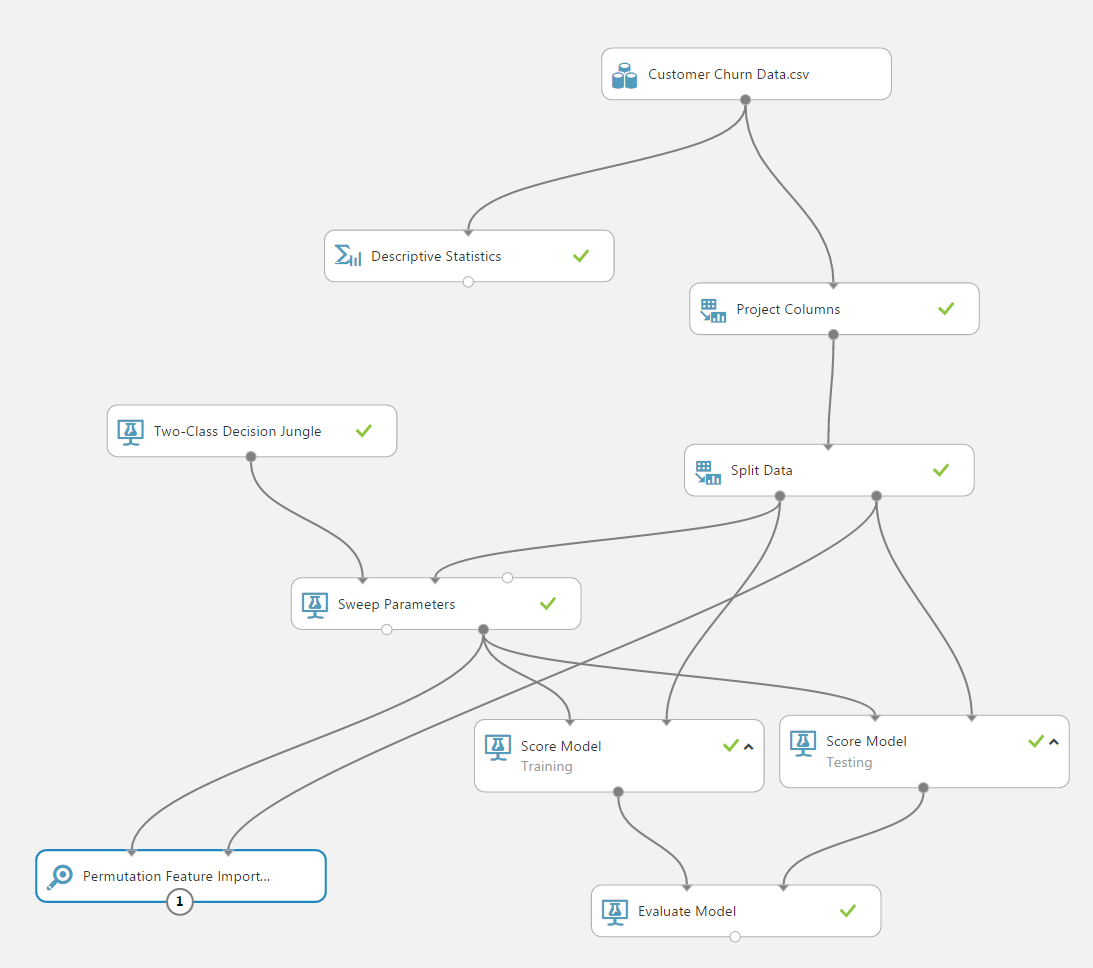
1. Go to your Churn experiment. Type **Evaluate Model** to the search box on the left and drag/drop it on the experiment. Connect the **Scored dataset** output port of **Score Model / Training** module to the left side **Scored dataset** input port of **Evaluate Model**.
2. Connect the **Scored dataset** output port of **Score Model / Testing** module to the right side **Scored dataset** input port of **Evaluate Model** and click **Run.**



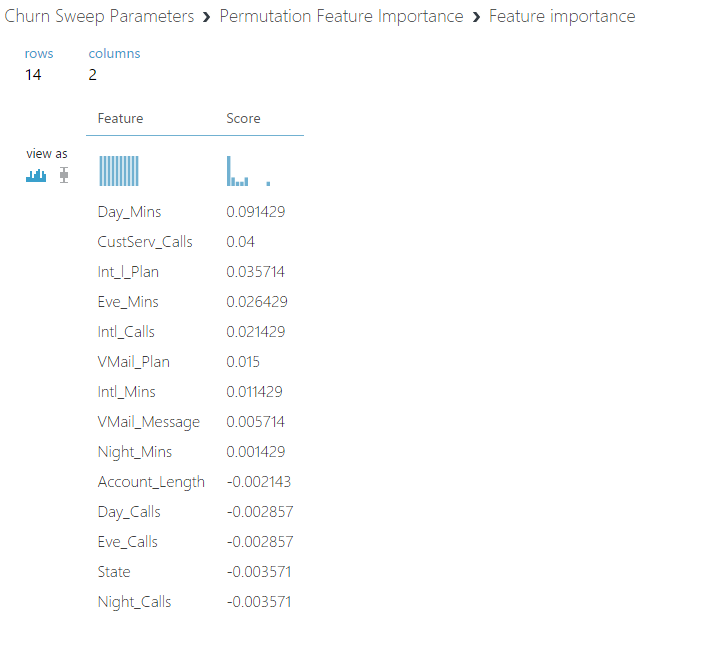
1. Right click the **Evaluation results** output port of **Evaluate Model** and select **View Results**



1. Notice that you are looking at **ROC** curves of both the training and test sets interlayered on top of each other for easy comparison. Also, when you scroll down the window, you will find the misclassification matrix. The blue curve (highlighted on the legend) belongs to the training set and shows that the classifier has near perfect performance since the training dataset has hardly any misclassifications (**False Positive**s or **False Negatives). AUC** is calculated as the area under the **ROC** curve which maps performance into a single scalar. The ROC curve is a two-dimensional representation of a model's performance. A perfect model will score an AUC of 1, while random guessing (diagonal line stretching from (0,0) to (1,1)) will score an AUC of 0.5 if the dataset is balanced between TRUE/FALSE’s on the outcome you are modeling.
2. Now, highlight the red curve by clicking the legend to examine the test set performance. You can see that the test set AUC is lower than the training set meaning that the classifier made errors on the test data. Now observe the confusion matrix for TP, FP, TN and FN counts. The model can be said to be over-fitting in this case, as the model does much better to the training data than the test data. However, the performance is still good on the testing data. Close the window to return to the experiment.
3. To see what features are important to the model, type in **Permutation Feature Importance** to the search box and drag the **Permutation Feature Importance** module onto the canvas. Connect the right output port of **Sweep Parameters** to the left input port of **Permutation Feature Importance**. Connect the right output port of **Split Data** to the right input port of **Permutation Feature Importance** and run the experiment.



1. Click the output port of **Permutation Feature Importance** and **Visualize** the results. This gives a rank ordering from the most important features to the least important features.



# Executing R scripts

In this exercise, you will use the **Execute R Script** module to fit a decision tree algorithm to the churn data and visualize it by using the pre-installed R package called **Party** within Azure ML. The purpose of this section is to get you introduced to the functionality in general and is not intended to dive into R scripting or to provide a detailed description of the package used.

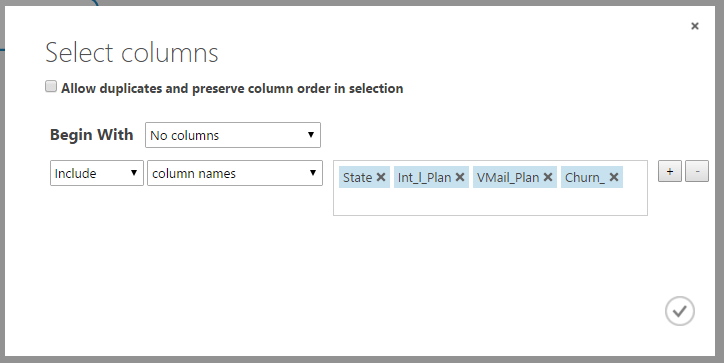
**Key Points**

1. Azure ML can easily be extended with open source R packages and scripts to complement the existing modules. It can also be extended with Python but that is not covered in detail here (besides IPYNB as shown before).

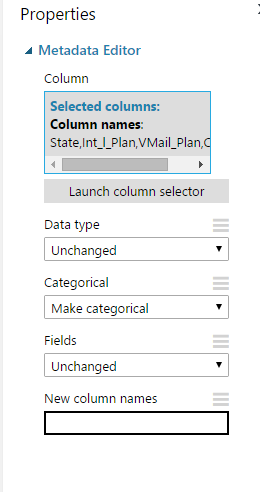
**Step-by-Step**

This exercise takes about 10 minutes to complete.

1. First, we will create a new experiment called Churn Decision Tree. Click the **SAVE AS** button at the bottom bar and type Churn Decision Tree for the experiment name and click the check mark. You will see that your experiment is renamed. You now have multiple experiments in your workspace. You can verify that by clicking the experiments tab on the leftmost navigation bar. Notice that a preview of the experiment is displayed on the right side.
2. Click on the Churn Decision Tree experiment, delete all the modules except **Customer Churn Data.csv** and **Project Columns**. Next, type **Metadata Editor** in the search box and drag and drop the module on the experiment canvas and connect it to the Project Columns. Click **on Launch column selector** on the properties section and on the **Select columns** window, make sure that the first drop down menu next to **Begin With** has **No columns**, next one is **Include** and the last one is **column names.** Click the text box next to it and select **State, Int\_l\_Plan, VMail\_Plan** and **Churn\_** and click the check mark to confirm your settings.



1. Under the properties of the module, Change the Categorical setting to **Make Categorical.** Your settings should look like the screen shot below.



1. Next, type **Execute R Script** in the search box and drag the module and connect its first input port to the output port of **Metadata Editor**. Replace the contents of the **R script text** box on the Properties panel with the following (you can also copy/paste if you have downloaded this documentation):

*# Map 1-based optional input ports to variables*

ChurnData <- maml.mapInputPort(1) # class: data.frame

library(party)

model<- ctree(Churn\_~ . ,data= ChurnData,control = ctree\_control(maxdepth=3))

plot(model, main = 'Churn Decision Tree')

print(model)

ChurnResponse<-as.data.frame(treeresponse(model))

ChurnProb<-as.data.frame(t(ChurnResponse))

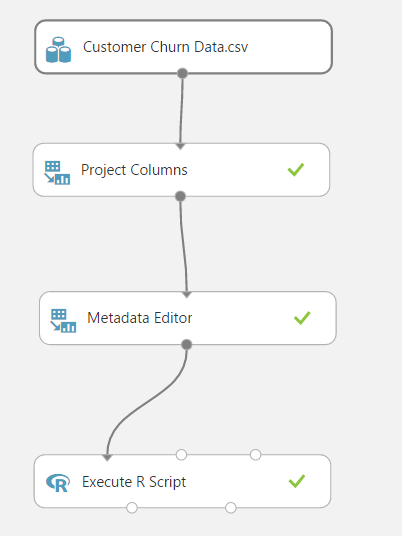
ChurnPred<- data.frame(ChurnData$Churn\_, predict(model),ChurnProb[,2])

colnames(ChurnPred)<-c("Churn\_","Scored Label","Scored Probability")

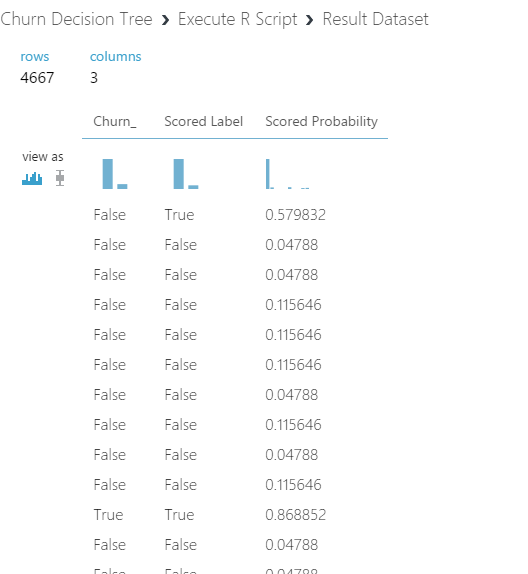
*# Select data.frame to be sent to the output Dataset port*

maml.mapOutputPort("ChurnPred");

1. Run the experiment.



1. The above script calls the R package **Party (**[**http://cran.r-project.org/web/packages/party/index.html**](http://cran.r-project.org/web/packages/party/index.html)**)** and computes a small decision tree with 3 levels and also outputs the predicted labels and probabilities. Right click the left output port of Execute R module and **Visualize** to observe the predictions of the decision tree that was trained on the whole dataset.



1. Close the window and this time right click the second output port of the Execute R script module and **Visualize**. You will fist see the split rules generated by the package and as you scroll down you will find the plot of the decision tree that was just computed. Please refer to the package documentation provided at the link above for more details of the usage of the functions included in the package and their outputs.
2. Next, place another **Execute R Script** on the experiment canvas and connect its first input port to the output port of **Metadata Editor**. Replace the contents of the **R script text** box on the Properties panel with the following (you can also copy/paste if you have downloaded this documentation):

# Map 1-based optional input ports to variables

ChurnData <- maml.mapInputPort(1) # class: data.frame

library(ggplot2)

# Scatter plot of Account\_Lenght vs CustServ\_Calls by churn

ggplot() +

facet\_wrap(~Churn\_) +

layer(

data=ChurnData,

mapping=aes(x=Account\_Length, y=CustServ\_Calls, color=Int\_l\_Plan),

geom="point",

position=position\_jitter()

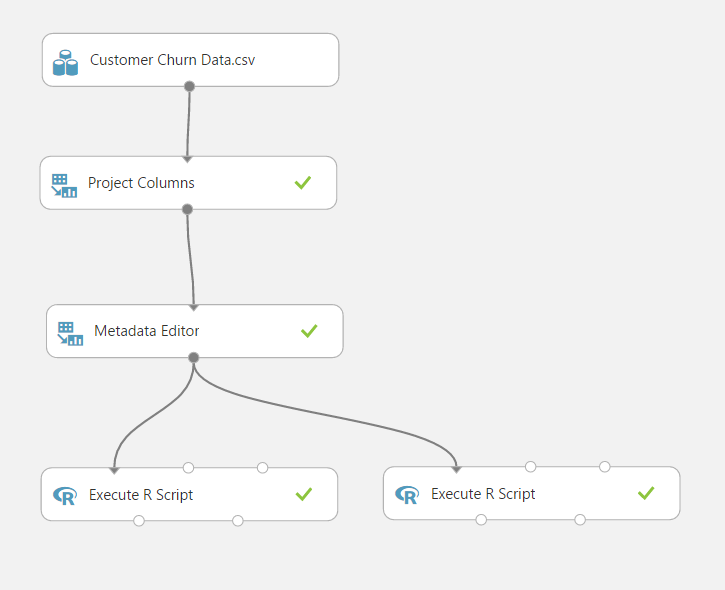
)

# Select data.frame to be sent to the output Dataset port

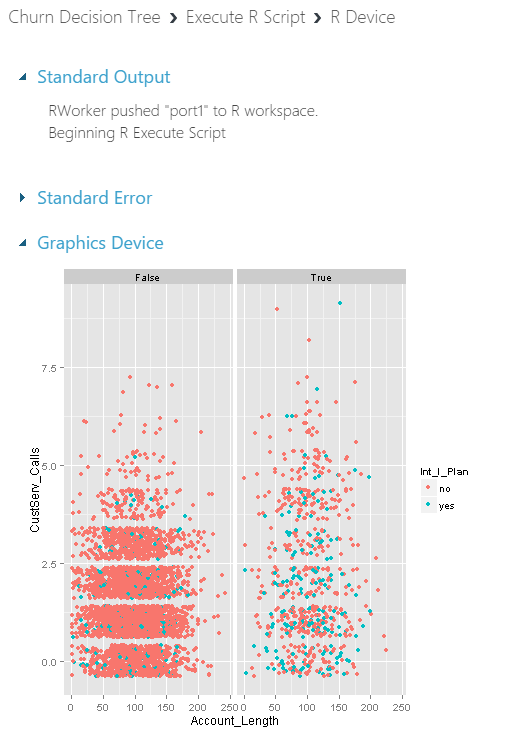
maml.mapOutputPort("ChurnData");

1. The above script calls the R package ggplot2 **(**[**https://cran.r-project.org/web/packages/ggplot2/index.html**](https://cran.r-project.org/web/packages/ggplot2/index.html)**)**

and creates scatter plots of Account\_Lenght and CustServ\_Calls by faceting according to Churn\_ column. When finished your experiment should like the below:



1. Right click the right output port of Execute R module and **Visualize** to observe the plots generated by the script.



# Creating a web service for binary classification model

In this part, you will create a Web service for the Churn experiment that you created before, operationalize and deploy your model to production.

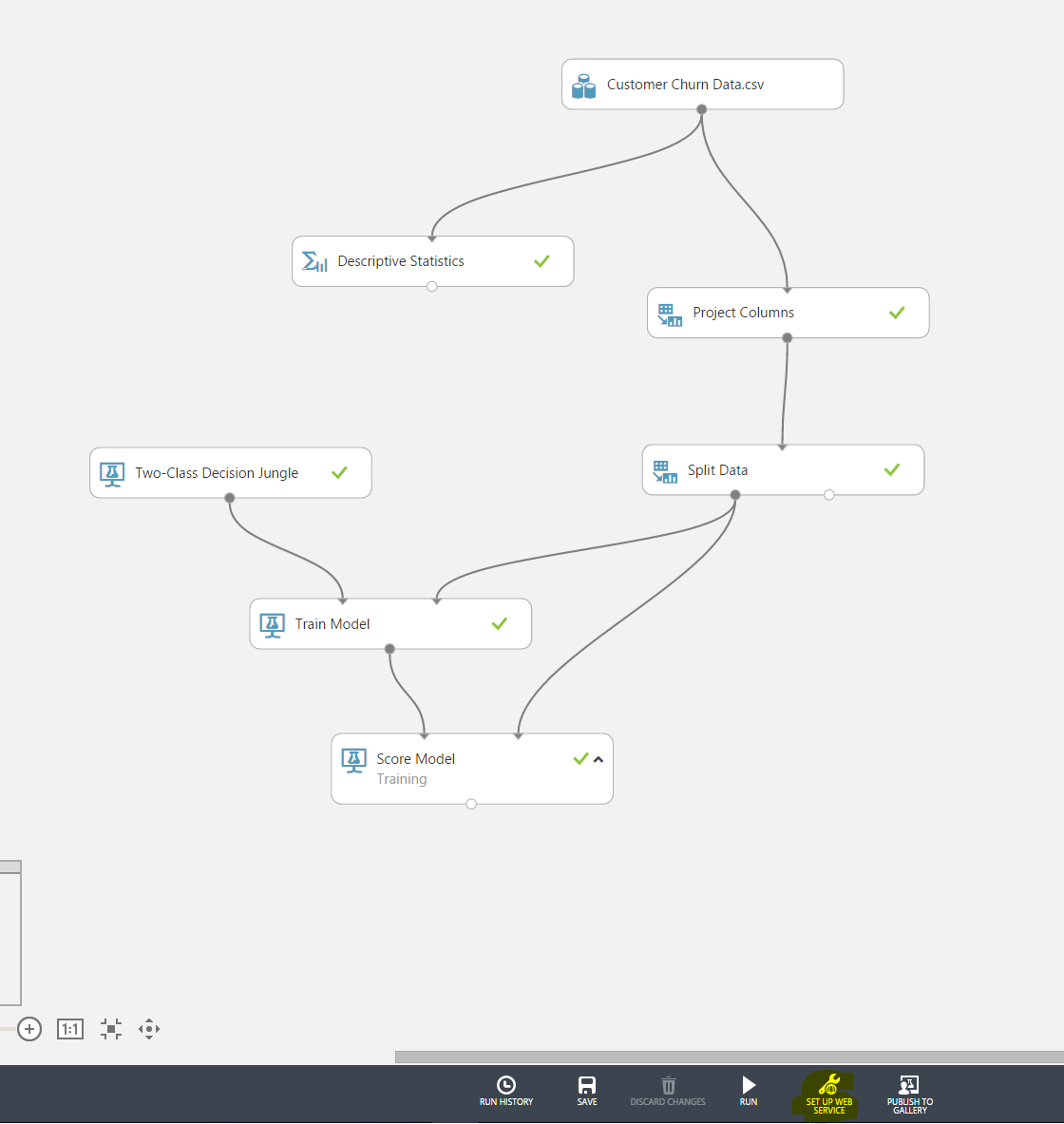
**Key Points**

1. After creating the web service, you will use the test option to verify the RRS (request response service) is operational.
2. Azure ML provides example code snippets in different languages (C# etc.) that can be used to call the service from within different custom applications.

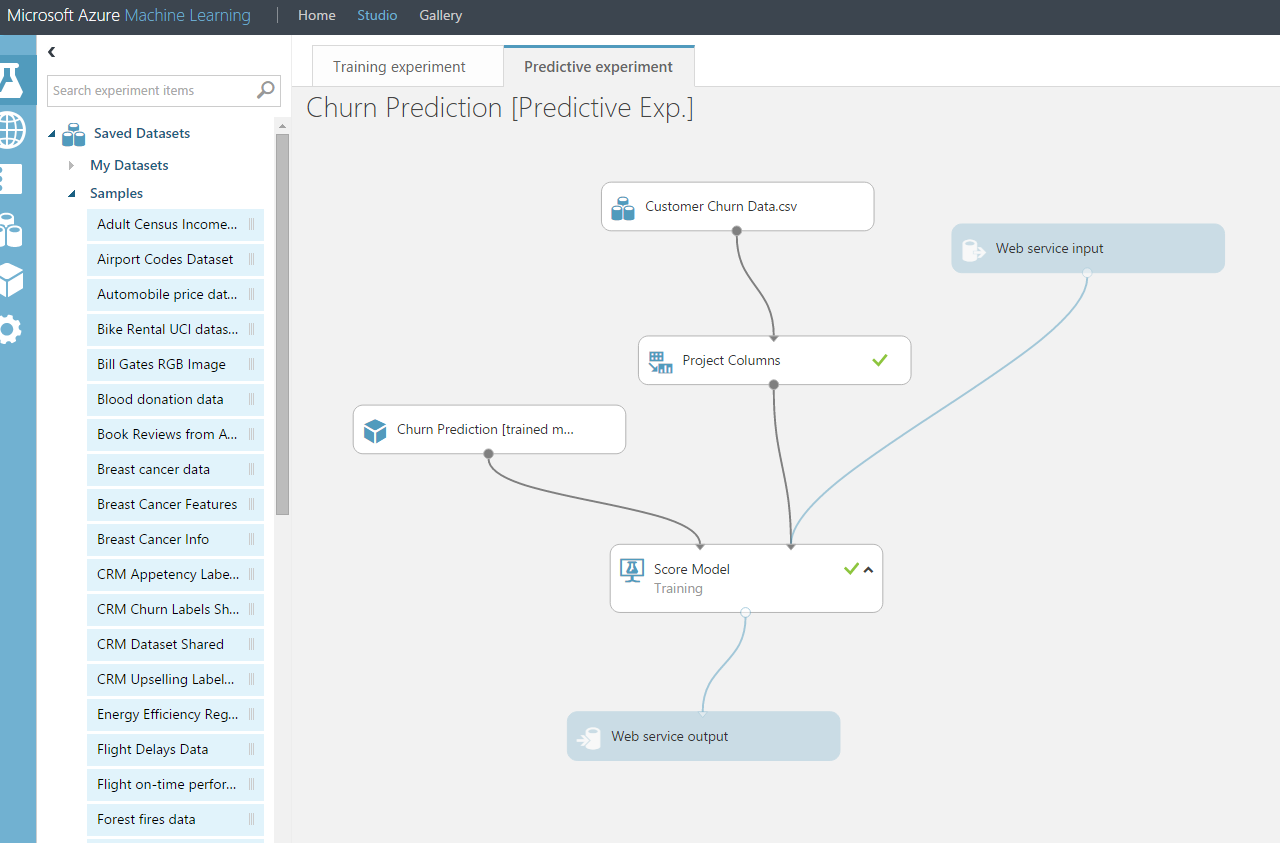
**Step-by-step**

This step takes about 15 minutes to complete.

1. Return to the binary classification experiment named **Churn Prediction**, make sure the status of the experiment in the upper right hand corner of the page is **Finished**. Open the experiment by clicking on the name of the experiment.
2. Click on **SET UP WEB SERVICE** button at the bottom of the screen, then select **Predictive Web Service [Recommended].**



1. A new tab will automatically be created for you, and you will now see a **Web Service Input** module and a **Web Service output** module on the canvas. Think of this automatic experiment creation as the starting point to setting up the web service. In this experiment, the model is pre-trained (taken from the previous experiment), so that when the web service is called, it will simply score using the model rather than training every time.
2. As we don’t end up using some of the feature in the raw data and to simplify the web service, move the connection that goes from the **Web Service Input** to the **Project Columns** to instead go from **Web Service Input** to the right input port of **Score Model.** This changes the web service input to only take the relevant features for the model. Click **Run** to run the experiment.



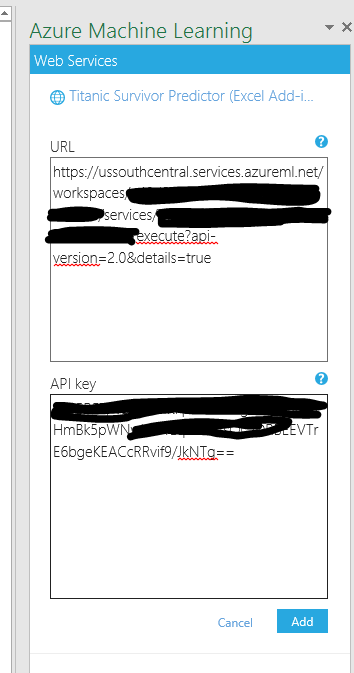
1. Click on **Deploy Web Service** to deploy this as a web service. It will open a new page with the web service details.
2. To test the service, click on the blue **Test** button in the second column of the Default Endpoint table at the bottom of the page. When **Enter data to predict** window pops up, enter the following values: State: NJ, Account\_Lenght: 127, Intl\_Plan: no, Vmail\_plan: no, Vmail\_message: 0, Day\_mins: 245, Day\_calls: 91, Eve\_mins: 217, Eve\_calls: 92, Night\_mins: 243, Night\_calls: 128, Intl\_mins: 14, Intl\_calls: 6, Custserv\_calls: 0, Churn\_: unchecked.
3. Click on the checkmark, then click on the green stacked bars at the bottom right corner of the screen then click **DETAILS**. You will see a test return confirmation message with the **Result: NJ, 127, no, no, 0, 245, 91, 217, 92, 243, 128, 14, 6, 0, False, False, 0.2689**. The last two values represent the churn model prediction for this customer where the predicted probability for this customer’s likelihood to churn (TRUE) is approximately 26.89%.

# Azure ML Web service call from Excel

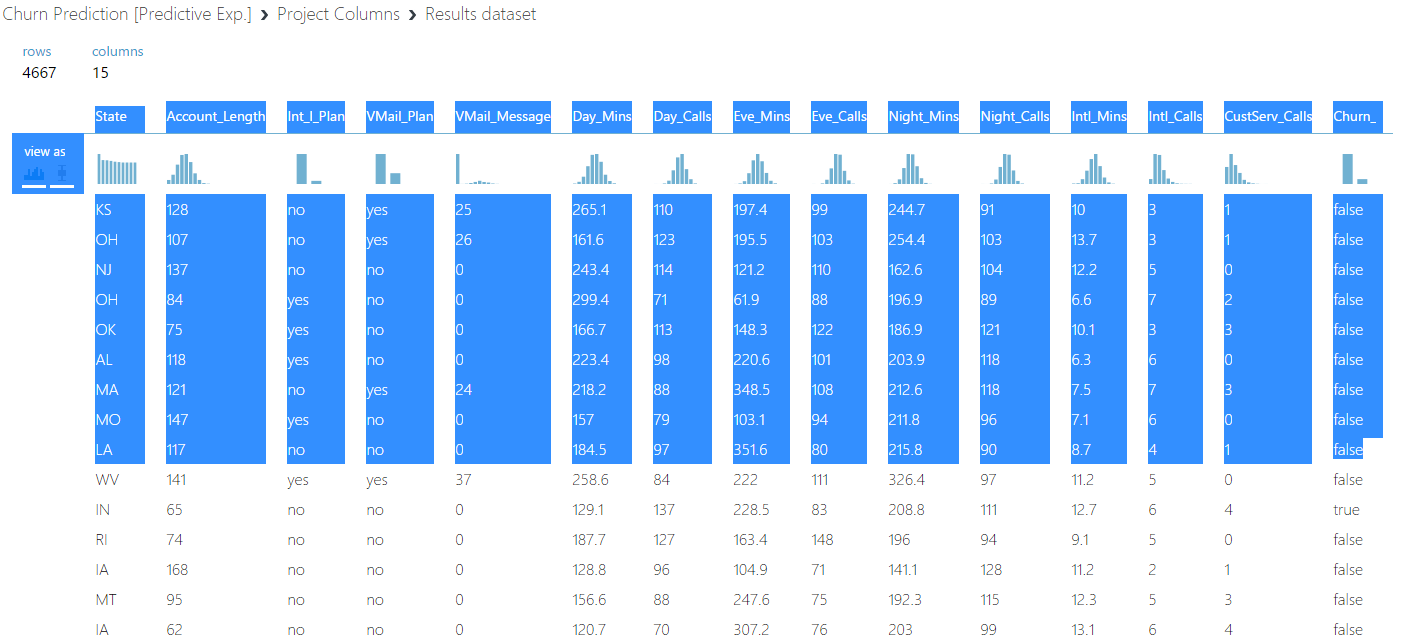
In this exercise, you will use Excel to make a call to the web service you created earlier. For details of how this works, see here: <http://blogs.technet.com/b/machinelearning/archive/2015/09/01/excel-add-in-for-azure-ml.aspx>. This is just one example way of how you can consume the machine learning model. In a production scenario, you would likely use something like Azure Data Factory to schedule regular calls to the machine learning model and putting the results into a database, or connect the web service to a web site where users can enter values, or call out to the machine learning model using Azure Stream Analytics and alerting on results for example.

**Step-by-step**

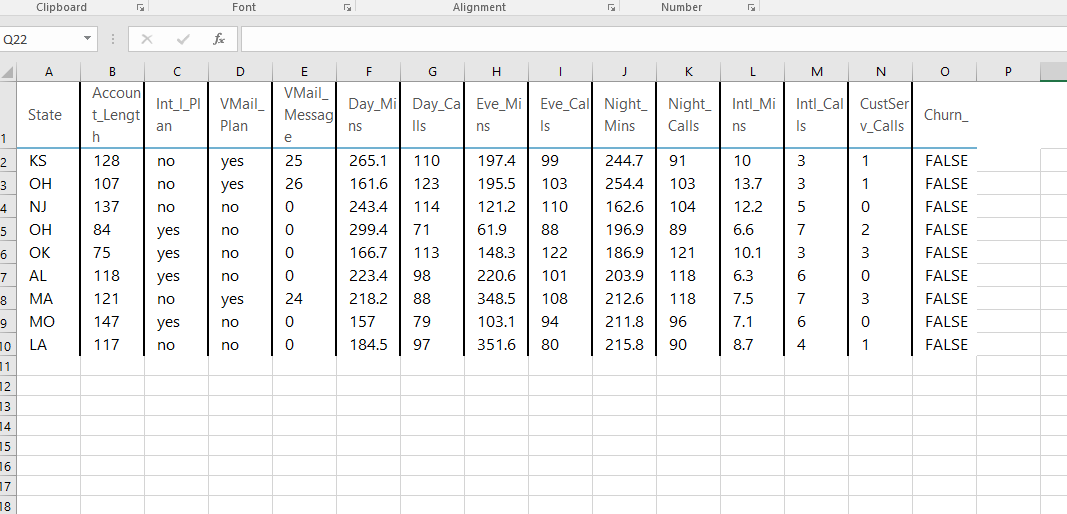
1. Go back to your browser to the web service dashboard. Click the **REQUEST/RESPONSE** link under the **API HELP PAGE** heading. This will direct you to the Request Response API Document for your service. You will find your Web service URI on this page along with Sample Code to call the web service using C#, Python and R when you scroll down to the bottom.
2. Open the excel file **ML Add-in** that was downloaded as part of the materials at the beginning of the lab. You may use the add-in directly in the browser using Excel Online or opening the file in Excel 2013 or later on Windows.  Copy the file to your own OneDrive account if you want to edit it. If you are able to edit it, follow the instructions below. Otherwise you can see how it would work by opening **Example ML Add-in with Web Service Done.**
3. Click on **+Add Web Service** in the Excel document and copy/paste the **Request URI** found next to **POST** in the **REQUEST/RESPONSE** documentation page into the URL spot on the Excel add-in. Copy/paste the API key from the main web service page into the API key of the Excel add-in. Click **Add** to add this web service.



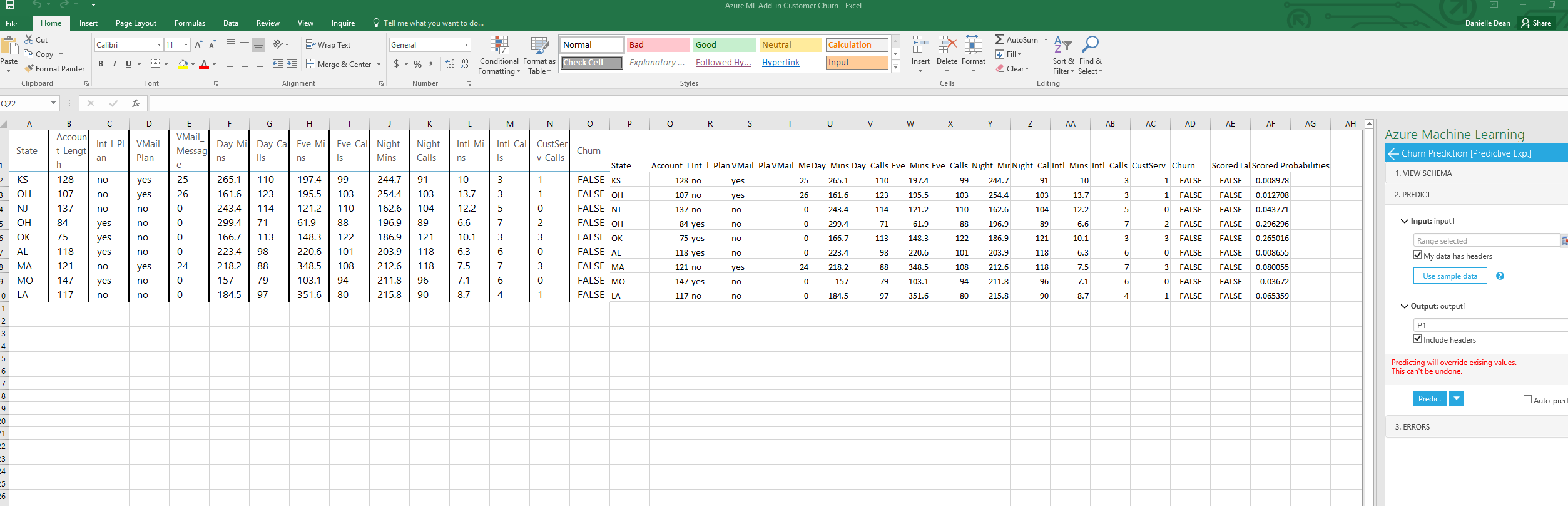
1. Go back to the **Churn Prediction [Predictive Exp.]** experiment and right click the output port of **Project Columns** and **Visualize** to see example data that could be fed into the model.
2. Copy/paste a few rows of data from this view to Excel, including the header.



1. Delete rows that don’t have any relevant data (if you copy/pasted the header you will likely need to delete rows 2/3). Make sure the data and headers are lined up (if you copy/pasted the header you will likely need to cut/paste the data within Excel to match up with the headers). The final dataset should look something like the screenshot below.



1. Type the range (in the case above A1:O10) or select it using the button, and make sure **My data has headers** is selected if you copy/pasted the header.
2. Put **P1** as the output cell (make sure your data is copy/pasted starting at cell **A1** in order for this to not override your data and make sure **Include headers** is selected.
3. Select **Predict**.
4. Columns **P** through **AF** will now be populated with results from the web service, with column **AE** representing the predicted value for whether the customer will churn and **AF** representing the associated probability of churn.



# Visualizations for classification using Power BI

In this section, you will use Power BI Desktop to visualize the scored labels of the classification model you have developed earlier.

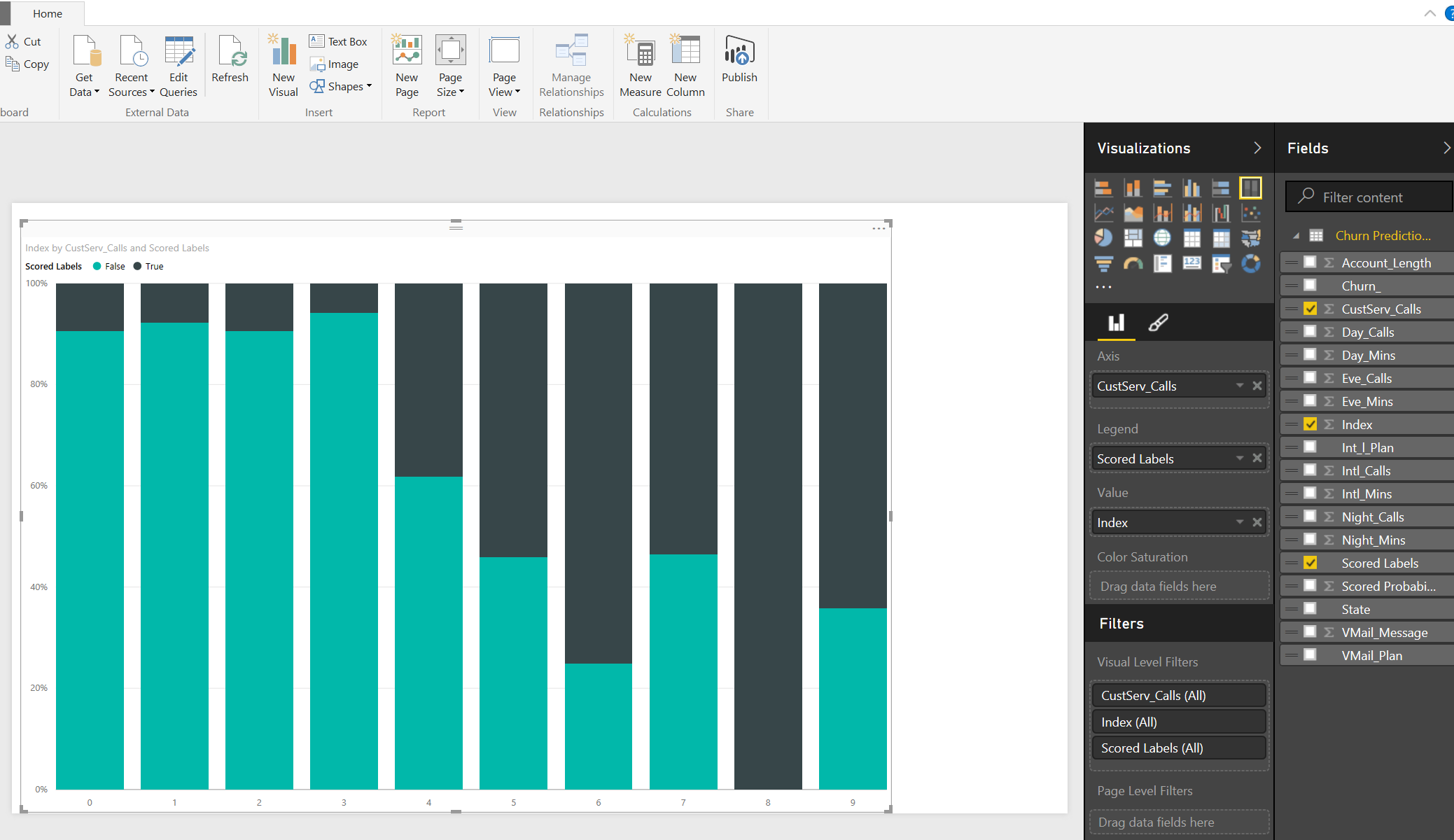
**Key Points**

1. Power BI and other visualization tools and techniques can be used to visualize the results of the models like the scored datasets by importing the data into Power BI Desktop.
2. Power BI Desktop is a free tool, and the details of the Power BI online reporting is here: <https://powerbi.microsoft.com/en-us/pricing>. You can take a tour of Power BI here: <https://powerbi.microsoft.com/en-us/tour>.

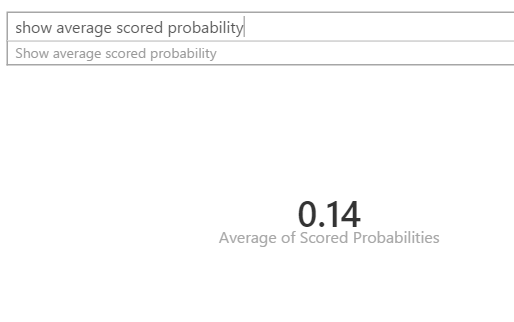
**Step-by-step**

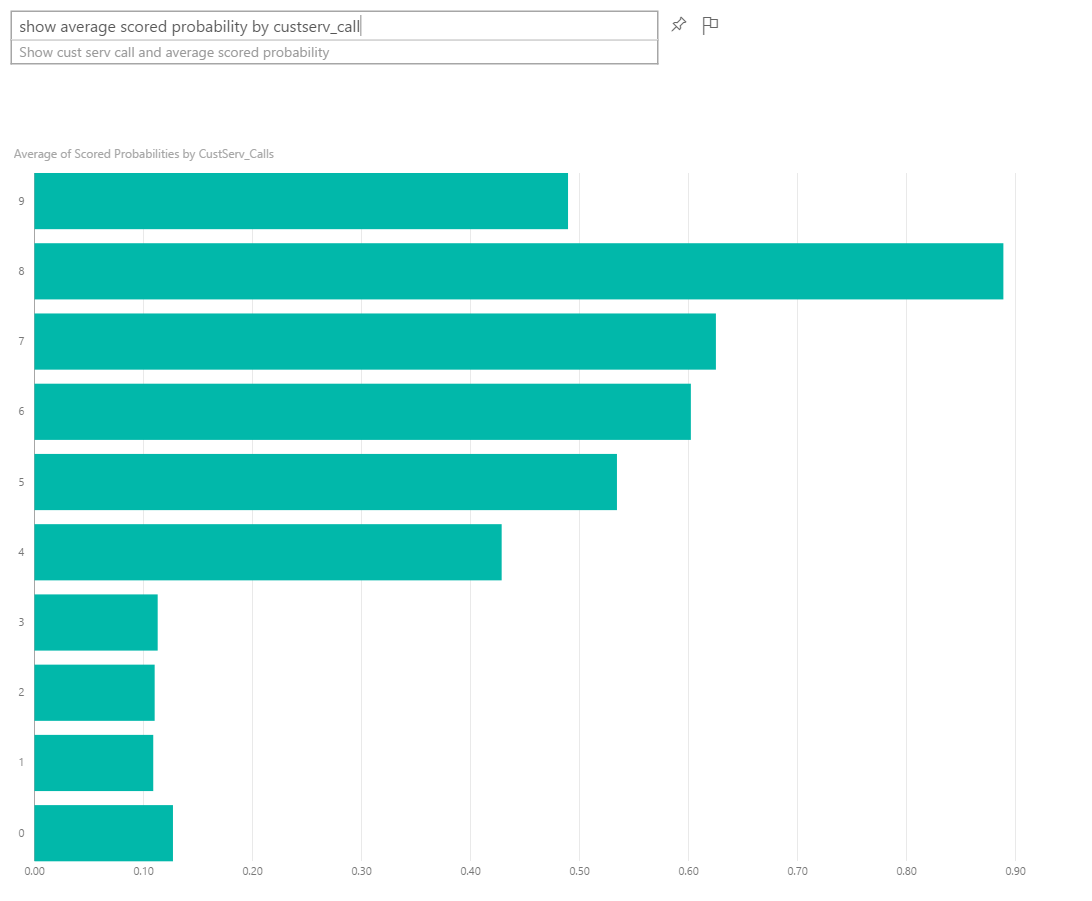
This step takes about 20+ minutes to complete.

1. If you haven’t downloaded Power BI already, please do so from: <https://powerbi.microsoft.com/en-us/desktop>
2. Optionally, you can sign up to Power BI online so you can publish your report as interactive dashboards: <https://powerbi.microsoft.com/en-us/>.
3. Go back to the **Churn Prediction** experiment. Search for **Convert to CSV** in the search box and add the module onto the canvas. Connect the output port of **Score Model** to the input port of **Convert to CSV**. Click run.
4. Right click the output port of the **Convert to CSV** module and click **Download**.
5. Open Power BI Desktop. Click **Get Data** then **CSV** then open the file that was just downloaded. Click **Load** to load this data into Power BI Desktop.
6. Click on **Edit Queries** then **Add Column** then **Add Index Column** to make it easier to do queries where you are counting the number of individuals. Click **Home** then **Close & Apply**.
7. Start building visualizations by clicking on a visualization type and selecting columns you want to visualize. And example is below.



1. If you have a Power BI online account, you can publish this by clicking on the **Publish** button.
2. Once published, Power BI also gives the option to ask natural language queries against the data. An example is in the screenshot below:





# Clustering

In this section, you will create a new series of experiments by clustering the Customer Churn data set. Clustering is an unsupervised learning technique that doesn’t require labels in the data in order to apply the algorithm. In the Customer Churn data set, labels are supplied, but we are going to ignore those labels for our current purposes.

The clustering algorithm you will use is called k-means, where k refers to the number of clusters you would like to have. In k-means, each sample in the data set is mapped into an n-dimensional feature space (where n is the number of columns, or features, you are asking the algorithm to consider). Depending on the specific implementation of the algorithm, k initial centroids, or center points of the clusters are established and then an iterative process begins. In each iteration of the process, the centroids are moved so that they are closer (using a distance formula such as Euclidean distance) to the center of a group of n / k nearest points. The iterations continue until either a static state is reached or the maximum number of iterations is reached.

This will be done two ways – once through Azure ML Studio experiments and once through IPython Notebooks.

**Key Points**

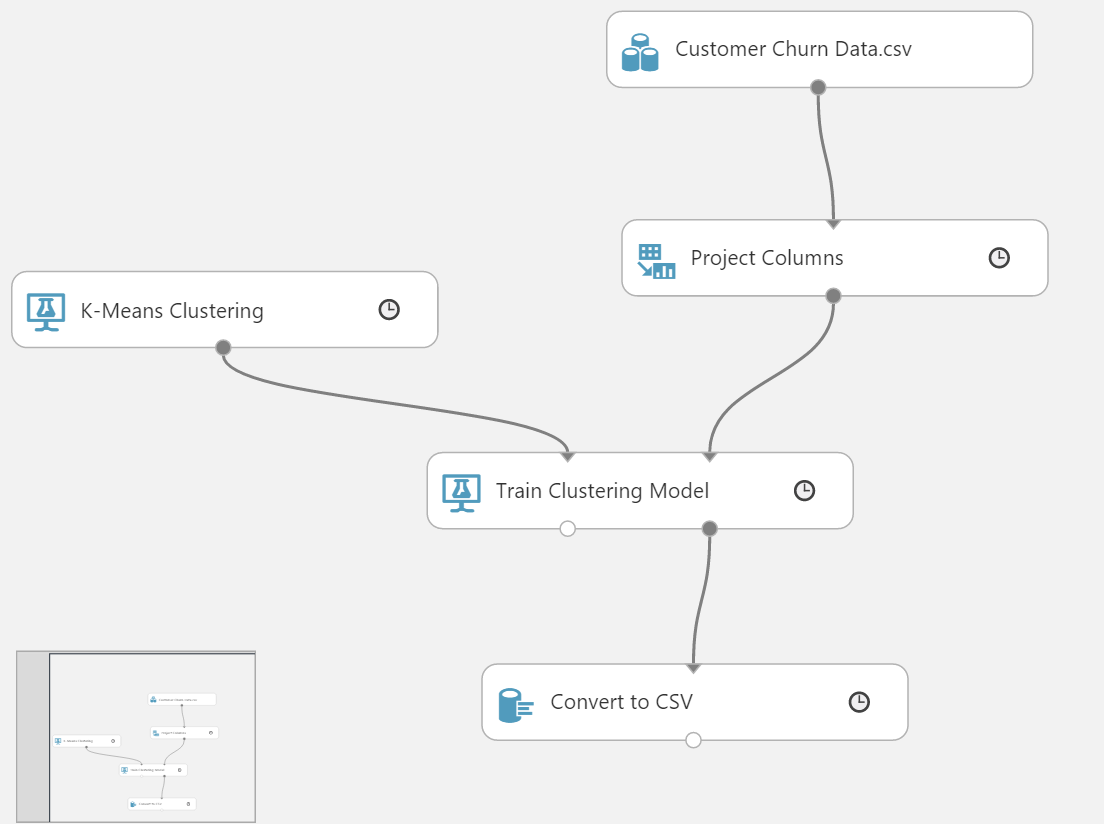
1. The k-Means clustering algorithm classifies unlabeled data.
2. Be careful to only include significant columns (features) in your clustering model and exclude highly-correlated features. This way, redundant, highly-correlated features will not skew your model.

## Through Experiment Canvas

**Step-by-step**

This step takes about 15 minutes to complete.

1. From literally anywhere in Azure ML Studio, create a new experiment, by clicking on the **+ NEW** button in the lower left corner of the window and then click on **Blank Experiment** box.
2. Rename your experiment by typing **Segmentation** into the title field.
3. In the Search box, search for Customer Churn Data and drag the data set onto the canvas. You will prepare your clustering experiment against this data set.
4. Before clustering, you will first eliminate columns that are not necessary, as in the previous experiment. Search for **Project Columns** and drag that module onto the canvas.
5. Connect the **ChurnData** data set to the **Project Columns** module.
6. With the **Project Columns** module selected, click the **Launch column selector** button in the **Properties** pane and include the following columns **X\_dataobs\_, State, Account\_Length, Int\_l\_Plan, VMail\_Plan, VMail\_Message, Day\_Mins, Day\_Calls, Eve\_Mins, Eve\_Calls, Night\_Mins, Night\_Calls, Intl\_Mins, Intl\_Calls, CustServ\_Calls,** and **Churn\_** as in the screenshot below. Note that some highly- correlated columns are being excluded, as in the previous experiments. Also note that **X\_dataobs\_** (the observation number) and **Churn\_** (the label) will not be used in the clustering itself, but will be used later for visualization.
7. Click the check mark and use the Search box to find the **K-Means Clustering Model** and the **Train Clustering Model** modules and drag them onto your canvas and connect them.
8. Select the **K-Means Clustering** module, go to the **Properties** pane and set the **Number of Centroids** (K) to 2.
9. With the **Train Clustering** module selected, select the columns that you will include in clustering and the resulting visualization, by pressing the **Launch column selector** button in the **Properties** pane.
10. Select all columns except **X\_dataobs\_** and the **Churn\_** columns that will not be included in the clustering.
11. Use the **Search** box to locate the **Convert to CSV** module and wire it up, as shown below. **Run** the experiment.



1. Once the experiment has completed, right click on the output node of the Convert to CSV module and select **Download**.
2. Save the resulting file as **K – Means 2 .csv** where 2 is the number of clusters that you have requested.
3. Repeat the experiment by changing the cluster **Number of Centroids** parameter in the properties pane for the **K-Means Clustering Model** module to **4**.
4. Download and save the resulting CSV file as **K – Means 4.csv**. You can visualize this with Power BI Desktop like you did the other model.

## Through IPython Notebook

**Step-by-step**

This step takes about 15 minutes to complete.

1. Go back to the IPython Notebook and run the code starting at **Data Preprocessing.**
2. Replace the workspace id and authorization token in relevant sections.
3. These steps also include how to publish a web service from IPYNB.

# Summary

This lab was intended to introduce you to the basic concepts of data science such as binary classification, segmentation, training and testing a model using Azure Machine Learning. A web service is created to operationalize the model and Excel was used to consume the web service, and then visualized with Power BI.