[](http://www.calstatela.edu/centers/hipic) CIS5560 Term Project Tutorial

#### Authors: [Ruchi Singh](https://www.linkedin.com/in/ruchi-singh-68015945/); [Kulwinder Kaur](https://www.linkedin.com/in/kulwinder-kaur-528a1b129/); [Ovanes Mikaelian](https://www.linkedin.com/in/hovik-mikaelian-93a257a3/)

#### Instructor: [Jongwook Woo](https://www.linkedin.com/in/jongwook-woo-7081a85)

#### Date: 05/16/2017

**Lab Tutorial**

Ruchi Singh ([rsingh26@calstatela.edu)](mailto:rsingh26@calstatela.edu))

Kulwinder Kaur ([kkaur7@calstatela.edu)](mailto:kkaur7@calstatela.edu))

Ovanes Mikaelien ([omikael@calstatela.edu](mailto:omikael@calstatela.edu))

05/20/2017

**Applications of Machine Learning Models for Yelp Local Business Data**

**Objectives**

**List what your objectives are.** In this hands-on lab, you will learn how to implement the following machine learning algorithms:

* Match-box Recommender
* Compare Two Class Logistic Regression and Two Class Boosted regression
* K-means Clustering( 3 and 5 clusters)
* Text- Analysis using n-gram and uni-gram

**Platform Spec**

* Microsoft Azure Machine Learning Studio
* # of nodes: 1
* Total Memory Size: 10 GB
* # of modules per experiment: 100

**Four steps to create an experiment using ML studio:**

1. Create a model

* [Step 1: Upload the data](https://azure.microsoft.com/en-us/documentation/articles/machine-learning-create-experiment/#step-1-get-data)
* [Step 2: Preprocess and clean data](https://azure.microsoft.com/en-us/documentation/articles/machine-learning-create-experiment/#step-2-preprocess-data)

1. Train the model

* [Step 3: Choose and apply a learning algorithm](https://azure.microsoft.com/en-us/documentation/articles/machine-learning-create-experiment/#step-4-choose-and-apply-a-learning-algorithm) from the set.

1. Score and test the model

* Step 4: Evaluate the model

**Creating a Recommender**

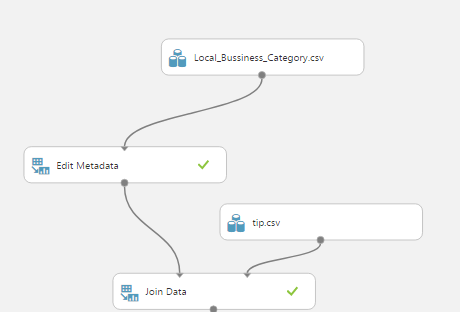
This tutorial shows how to create a Matchbox Recommender using AzureML

**Data Preparation**

In this step the data is uploaded into AzureML, then it is transformed and cleaned so that it is suitable for the recommender algorithm.

1. Open a browser and browse to [https://studio.azureml.net.](https://studio.azureml.net/) Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment, and give it the title **Yelp** **Category Recommendation**.
3. Download the *business.csv* file and drag it to canvas
4. Search for the **Edit Metadata (Metadata Editor)** module and drag it onto the canvas.
5. Connect the output of the **Business** dataset to the **Dataset** input of the **Edit Metadata (Metadata Editor)**.
6. Configure the properties of the **Edit Metadata (Metadata Editor)** to ensure the **Stars** column is of **Integer** type as required by the **Matchbox Recommender** module:
   * **Lunch Column selector** and select the column: Stars
   * **Data type**: Integer
   * **Categorical**: Unchanged
   * **Fields**: Unchanged
   * **New column names**: blank
7. Download the *tip.csv* dataset and drag it into canvas. This dataset maps user\_ IDs to business\_IDs and provides an info about the likes and the tips about the business.
8. Search for the **Join Data (Join)** module and drag it onto the canvas.
9. Connect the **Results dataset** output of the **Edit Metadata (Metadata Editor)** to the **Dataset1** (left) input of the **Join Data (Join)** module.
10. Connect the output of the **Tip** dataset to the **Dataset2** (right) input of the **Join Data (Join)** module.
11. Configure the properties of the **Join Data (Join)** module as follows:
    * **Column Selector for L** (left): business\_id
    * **Column Selector for R** (right): business\_id
    * **Match case**: Checked
    * **Join type**: Inner Join
    * **Keep right key column**: Unchecked

Our experiment looks like this:



1. Search for the **Select Columns in Dataset (Project Columns)** module and drag it onto the canvas.
2. Connect the **Results dataset** output of the **Join Data (Join)** module to the **Dataset input** of the **Select Columns in Dataset (Project Columns)** module.
3. Launch and Configure the **Column Selector** of the **Select Columns in Dataset (Project Columns)** module. As shown in the below figure, select the **Allow duplicates and preserve column order in selection** box, and then select the following columns in the order shown below
   * **User\_id, categories, stars**
4. Search for the **Clean Missing Data** module and drag it into the canvas
5. Connect the **Results dataset** output of the **Select Columns in Dataset (Project Columns)** module to the input of the **Clean Missing Data** module.
6. Configure the properties of the **Clean Missing Data** module as follows:
   * **Begin with: No Columns**
   * **Column Selector**: User\_Id, categories , stars
   * **Retain first duplicate row**: checked
7. Search for the **Remove Duplicate Rows** module and drag it onto the canvas.
8. Connect the **Cleaned dataset** output of the **Clean Missing Data** module to the input of the **Remove Duplicate Rows** module.
9. Configure the properties of the **Remove Duplicate Rows** module as follows:
   * **Begin with: No Columns**
   * **Column Selector**: User\_Id, categories
   * **Retain first duplicate row**: checked

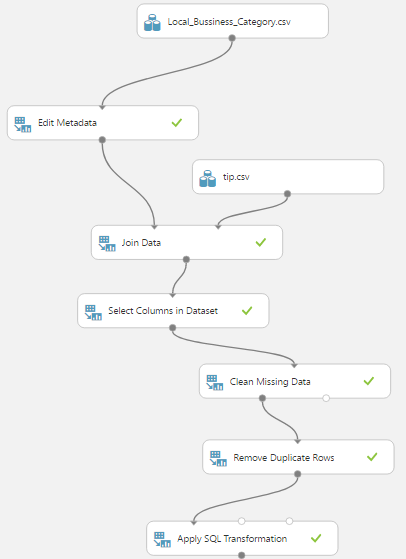
1. Search for the **Apply SQL Transformation** module and drag it onto the canvas.

In the SQL Query Command type the following:  
SELECT user\_id, categories, avg(stars) FROM t1

GROUP BY user\_id, categories;

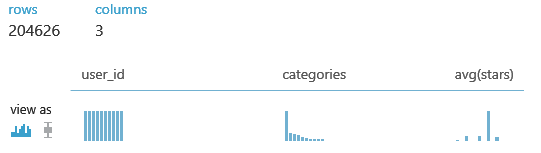
(The SQL command calculates the average star rating each user gives to a particular category.)

Our experiment looks like this:



1. Save and run the experiment. When the experiment has finished, visualize the output of the **Apply SQL Transformation** module.

It should look like this:





1. Search for the **Convert to CSV** module and drag it onto the canvas
2. Connect the output of **Apply SQL Transformation** module to input of **Convert to CSV** module
3. Download the csv. file and save it as reviewstar.csv. This file will be used for SparkML in another project

**Train a Recommender**

Now that the data is prepared, you can train a recommender.

1. Search for the **Split Data (Split)** module and drag it onto the Canvas.
2. Connect the **Results dataset** output of the **Remove Duplicate Rows** module to the input of the **Split Data (Split) module**.
3. On the properties pane of the **Split Data (Split) module**, configure the properties as follows:
   * **Splitting mode**: Recommender Split
   * **Fraction of training-only users**: 0.75
   * **Fraction of test user ratings for training**: 0.25
   * **Fraction of cold users**: 0
   * **Fraction of cold items**: 0
   * **Fraction of ignored users**: 0
   * **Fraction of ignored items**: 0
   * **Remove occasionally produced cold items**: unchecked
   * **Random seed for Recommender**: 5432
4. Search for the **Train Matchbox Recommender** module and drag it onto the canvas.
5. Connect the **Results dataset1** (left) output of the **Split Data (Split) module** to the **Training dataset of user- item-rating triples** (left) input of the **Train Matchbox Recommender** module.
6. On the properties pane for the **Train Matchbox Recommender** module, configure the properties as follows:
   * **Number of traits**: 20
   * **Number of recommendation algorithm iterations**: 10
   * **Number of training batches**: 4

**Evaluating a Recommender**

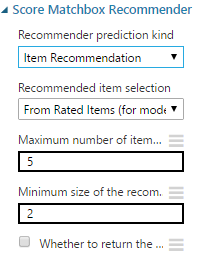
Since there is always a user bias towards a particular business or category, several methods and metrics are used to get a more accurate picture. In this model we will build the following recommenders:

**Item recommendation, Related Items, Rating Prediction, Related Users**

**Evaluate by Item (Category) Recommendation**

This recommender provides recommendations for a category based on user’s rating. Results are evaluated by NDCG (Normalized Discounted Cumulative Gain). An ideal result has a value of 1.0.

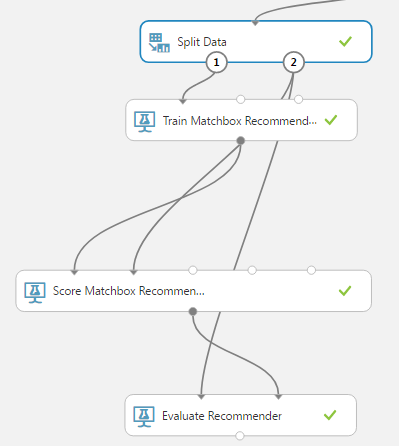
1. Search for the **Score Matchbox Recommender** module and drag it onto the canvas.
2. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left) input of the **Score Matchbox Recommender** module.
3. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the **Score Matchbox Recommender** module; look at the connect in the figure below.
4. On the properties pane for the **Score Matchbox Recommender** module, ensure that the following properties are specified:
   * **Recommender prediction kind**: Item Recommendation
   * **Recommended item selection**: From Rated Items (for model evaluation)
   * **Maximum number of items to recommend to a user**: 5
   * **Minimum size of the recommendation pool for a single user**: 2



5. Search for the **Evaluate Recommender** module and drag it onto the canvas.

1. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left) input of the **Evaluate Recommender** module.
2. Connect the **Scored dataset** (right) output of the **Score Matchbox Recommender** module to the **Scored dataset** (right) input of the **Evaluate Recommender** module.
3. On the properties pane of the **Evaluate Recommender** module, verify that the properties are set as follows:
   * **Minimum number of items that the query user and the related user must have rated in common:** 2
   * **Minimum number of users that the query item and the related item must have been rated by in common:** 2

After training and evaluating the recommender our screen should be like this:::



1. Save and run the experiment.
2. When the experiment has finished, Visualize the output form the **Evaluate Recommender** module. Note that the **NDCG** is about 0.98. This is an encouraging result, not too far from the ideal.

**Evaluate by Related Items (Categories)**

This metric predicts stars for one category based on the stars of other categories. Here we consider related pairs of categories that a group of users has rated. Results are evaluated by the similarity of the ratings using both L1 and L2 NDCG. Ideal recommender will give a value of 0.0 if the ratings are identical in all cases.

1. Copy the **Score Matchbox Recommender** module and the **Evaluate Recommender** module.
2. Paste these modules onto the canvas and drag them to one side.
3. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left most) input of the new **Score Matchbox Recommender** module.
4. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the new **Score Matchbox Recommender** module.
5. On the properties pane of the new **Score Matchbox Recommender** module configure the following properties:
   * **Recommender prediction kind**: Related Items
   * **Related item selection**: From Rated Items (for model evaluation)
   * **Maximum number of related items to find for an item**: 5
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2
   * **Minimum size of the related item pool for a single user**: 2

1. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left hand) input of the new **Evaluate Recommender** module.
2. Ensure the **Scored dataset** (right) output of the new **Score Matchbox Recommender** module is connected to the **Scored dataset** (right hand) input of the new **Evaluate Recommender** module.
3. On the properties pane of the new **Evaluate Recommender** module configure the parameters as follows:
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2

1. Save and run the experiment. 

12. When the experiment has finished, visualize the output form the **Evaluate Recommender** module. Note that the **L1 Sim NDCG** is about 0.94 and the **L2 Sim NDCG** is about 0.95. These values aren’t very good, since the scale is 0 to 5.

**Evaluate by Rating Predictions**

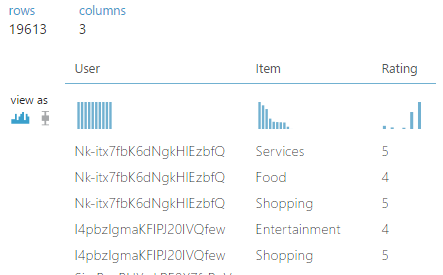
Rating Prediction compare predicted stars to actual star values using mean absolute error (MAE) and root mean square error (RMSE). Ideal results are 0.0 in both cases.

1. Copy the newest **Score Matchbox Recommender** module and the newest **Evaluate Recommender** module.
2. Paste these modules onto the canvas and drag them to one side.
3. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left most) input of the newest **Score Matchbox Recommender** module.
4. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the newest **Score Matchbox Recommender** module.
5. On the properties pane of the newest **Score Matchbox Recommender** module, set the

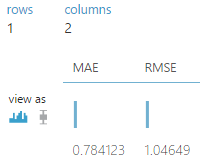
Recommender prediction kind property to Rating Prediction

1. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left hand) input of the newest **Evaluate Recommender** module.
2. Ensure the **Scored dataset** (right) output of the newest **Score Matchbox Recommender** module is connected to the **Scored dataset** (right hand) input of the newest **Evaluate Recommender** module.
3. On the properties pane of the newest **Evaluate Recommender** module configure the parameters as follows:
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2
4. Save and run the experiment.

Visualize the output of



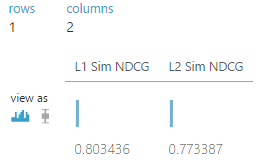
1. When the experiment has finished, visualize the output form the Evaluate Recommender module.



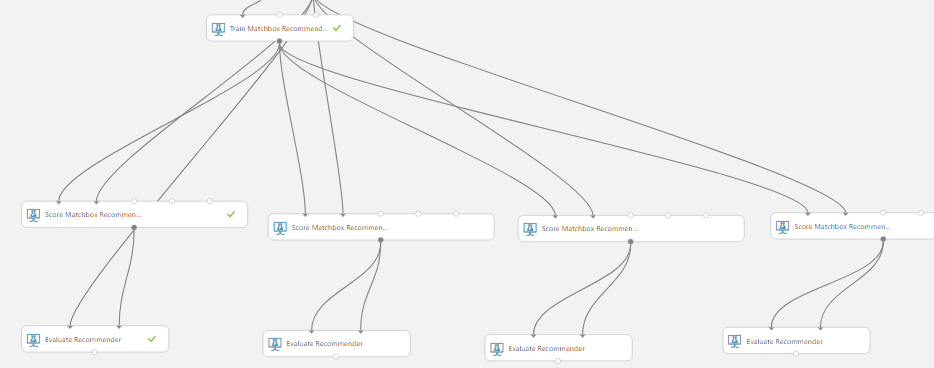
**Evaluate by Related Users**

For pair of users who have rated the same categories, we can predict ratings of one user based on the ratings of the other user. Results are evaluated by the similarity of the ratings using both L1 and L2 average normalized discounted cumulative gain (NDCG) averaged over all the pairs selected. In both cases an ideal value is 0.0.

1. Copy the newest **Score Matchbox Recommender** module and the newest **Evaluate Recommender** module.
2. Paste these modules onto the canvas and drag them to one side.
3. Connect the **Trained Matchbox recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox recommender** (left) input of the newest **Score Matchbox Recommender** module.
4. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Dataset to score** (second from left) input to the newest **Score Matchbox Recommender** module.
5. On the properties pane of configure the following properties:
   * **Recommender prediction kind**: Related Users
   * **Related user selection**: From Users That Rated Items (for model evaluation)
   * **Maximum number of related Users to find for a User**: 5
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum size of the related user pool for a single user**: 2
6. Connect the **Results dataset2** (right) output of the **Split Data (Split) module** to the **Test dataset** (left hand) input of the newest **Evaluate Recommender** module.
7. Ensure the **Scored dataset** (right) output of the newest **Score Matchbox Recommender** module is connected to the **Scored dataset** (right) input of the newest **Evaluate Recommender** module.
8. On the properties pane of the newest **Evaluate Recommender** module configure the parameters as follows:
   * **Minimum number of items that the query user and the related user must have rated in common**: 2
   * **Minimum number of users that the query item and the related item must have been rated by in common**: 2
9. Save and run the experiment.
10. When the experiment has finished, visualize the output form the Evaluate Recommender module. The numbers can be better, but aren’t too bad either.



After all four metrics are added, the bottom part of our screen looks like the following:

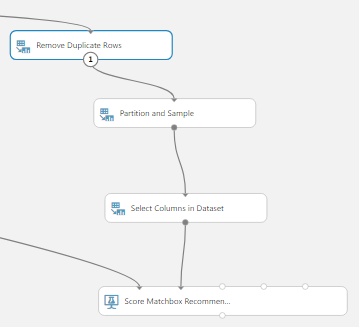


**Computing Category Recommendations**

Now that we have constructed the recommender and tested the four metrics, we can compute category recommendations for selected users.

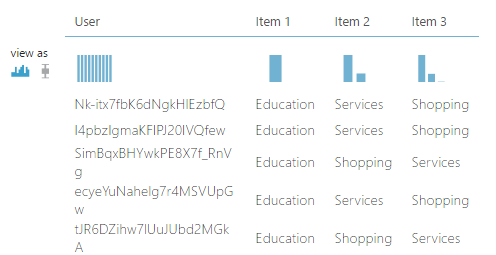
**Compute Recommendations**

1. Search for the **Remove Duplicate Rows** module, and drag a new one onto the canvas.
2. Connect the output of the existing **Remove Duplicate Rows** module to the input of the new **Remove Duplicate Rows** module.
3. In the properties pane of the new **Remove Duplicate Rows** module set the following:
   * **Column selector**: user\_id
   * **Retain first duplicate row**: checked
4. Search for the **Partition and Sample** module, and drag it onto the canvas.
5. Connect the output of the new **Remove Duplicate Rows** module to the input of the **Partition and Sample** module.
6. On the properties pane of the **Partition and Sample** module set the following parameters:
   * **Partition or sample mode**: Head
   * **Number of rows to select**: 100
7. Search for the **Select Columns in Dataset (Project Columns)** module, and drag it onto the canvas.
8. Launch the **column selector** of the **Select Columns in Dataset (Project Columns)** module and select only the **user\_id** column. Connect **Partition and Sample** to **Select Columns in Dataset (Project Columns)** module.
9. Search for the **Score Matchbox Recommender** module, and drag a new one onto the canvas.
10. Connect the **Trained Matchbox Recommender** output of the **Train Matchbox Recommender** module to the **Trained Matchbox Recommender** input of the **Score Matchbox Recommender** module.
11. Connect the output of the newest **Select Columns in Dataset (Project Columns)** module to the **Dataset to Score** (second from the left) input of the **Score Matchbox Recommender** module.
12. On the properties pane of the **Score Matchbox Recommender** module set the following parameters:
    * **Recommender prediction kind**: Item Recommendation
    * **Recommended item selection**: From All Items
    * **Maximum number of items to recommend to a user**: 3
13. The new parts of our experiment resemble the lower right part of this diagram:



1. Save and run the experiment.
2. When the experiment has finished running visualize the output of the **Score Matchbox**

**Recommender** module. Examine the output:

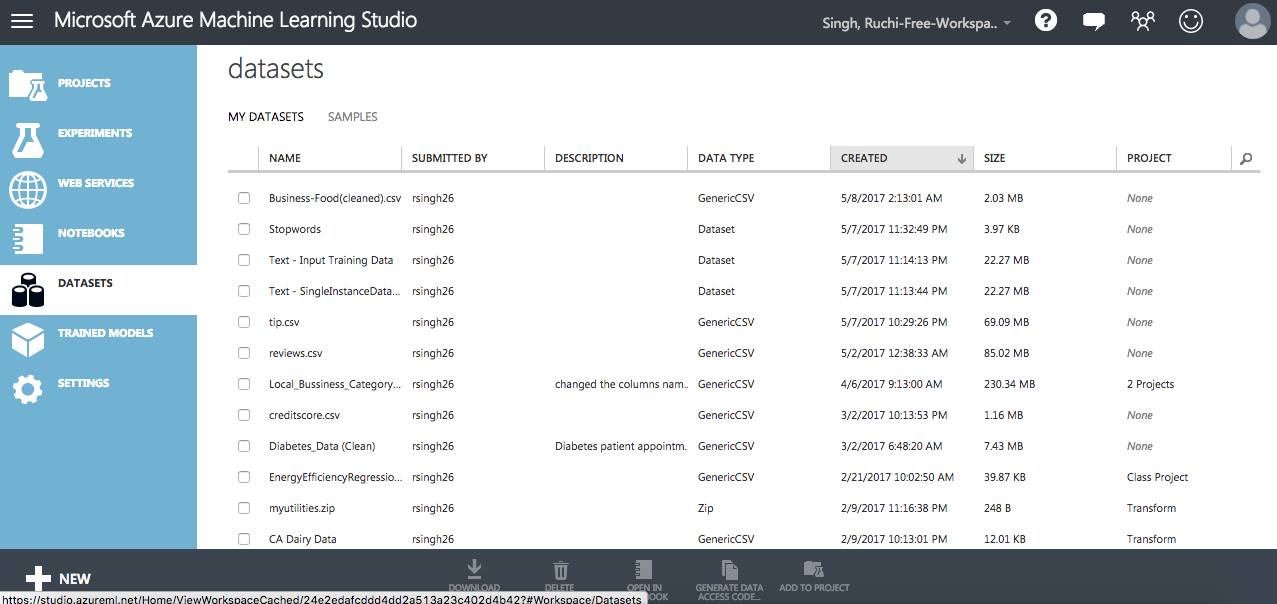


There are 100 rows, one for each of the users. Each row contains category recommendation for each user.

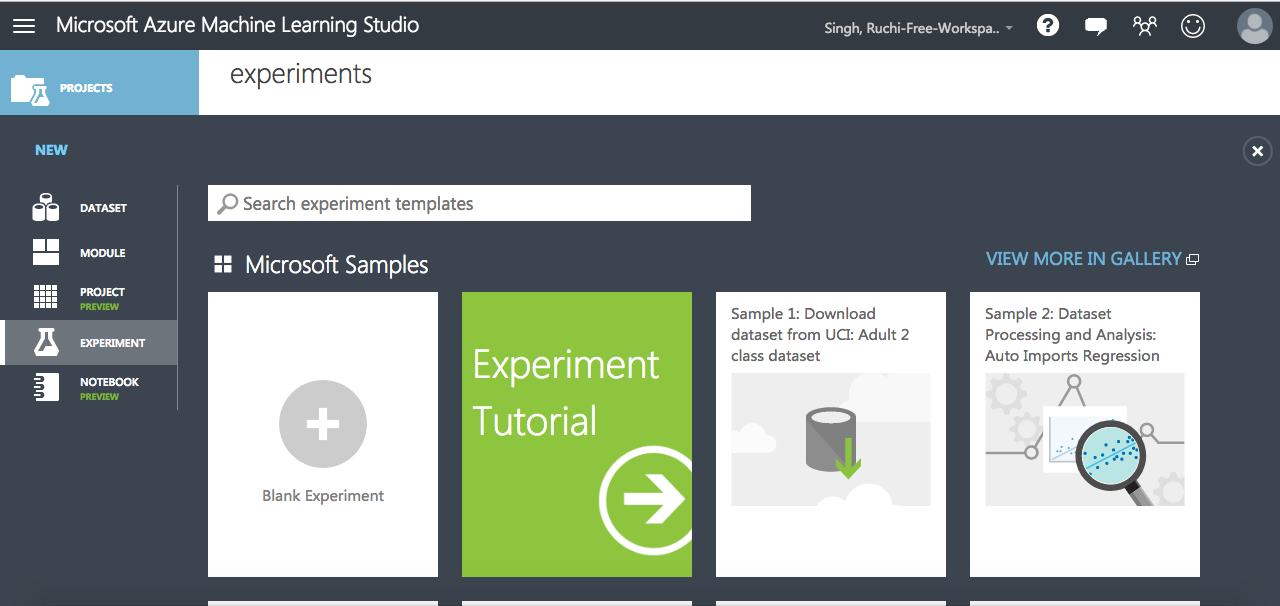
Step 1: Upload data

**Explain what this step is for.** This step is to get data manually….

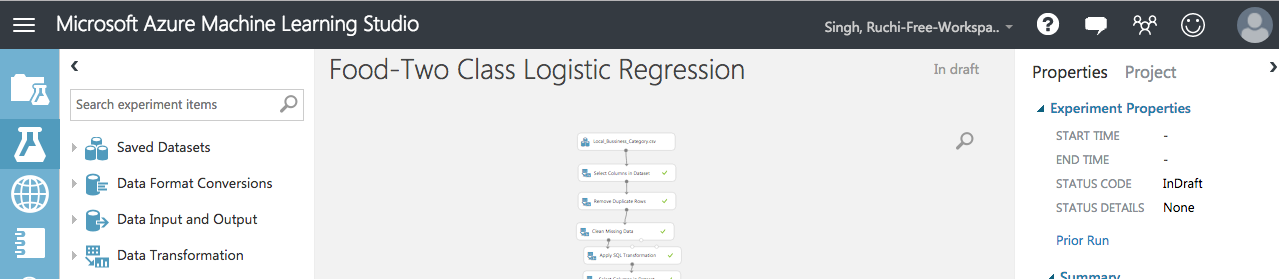
1. Create an account for Azure Machine Learning Studio.
2. Upload the Yelp dataset to Azure. This dataset includes entries like business ID, Name, Location, Ratings, Review Count, Category of the business etc.
3. Go to Dataset tab on left hand side, click on **+ New** sign and upload the dataset file. When you click on my datasets then the dataset you have uploaded will appear.



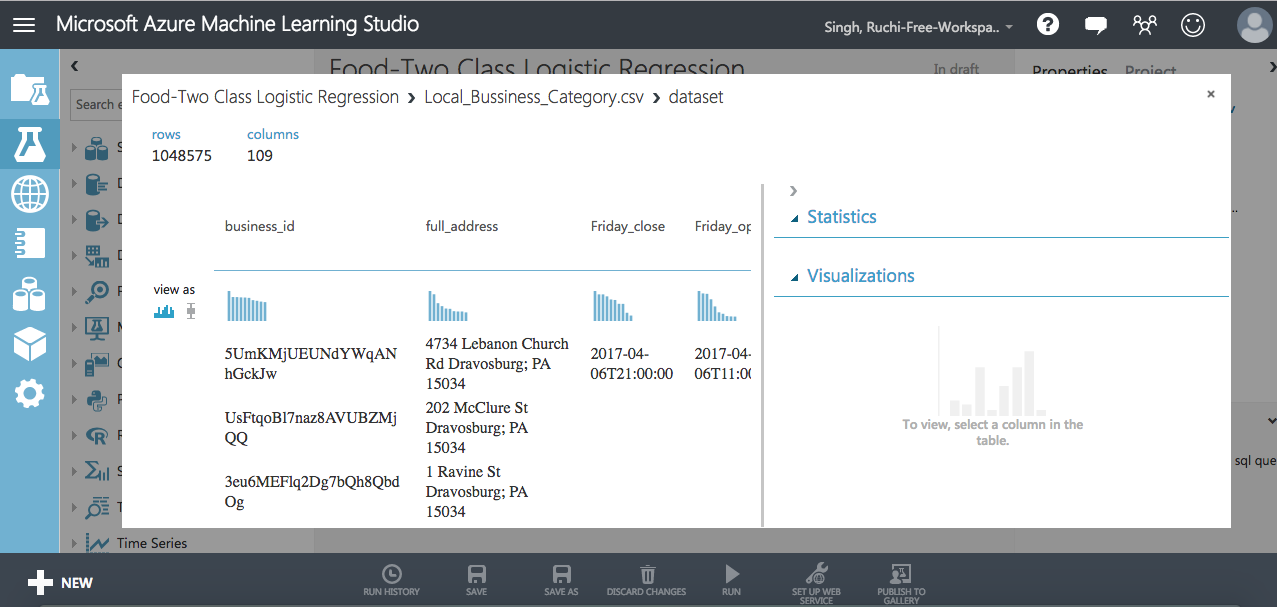
1. Start a new experiment by clicking +NEW at the bottom of the Machine Learning Studio window, select EXPERIMENT, and then select Blank Experiment. Select the default experiment name at the top of the canvas and rename it to something meaningful, for example, **Business Ratings prediction** or **Yelp dataset analysis** etc.



1. To the left of the experiment canvas is a palette of datasets and modules. Search for the dataset you want to use for the experiment.
2. Drag the dataset to the experiment canvas. In this case, upload Yelp dataset which we have uploaded earlier in my datasets.



1. To see what this data looks like, click the output port at the bottom of the Yelp dataset, and then select **Visualize**.

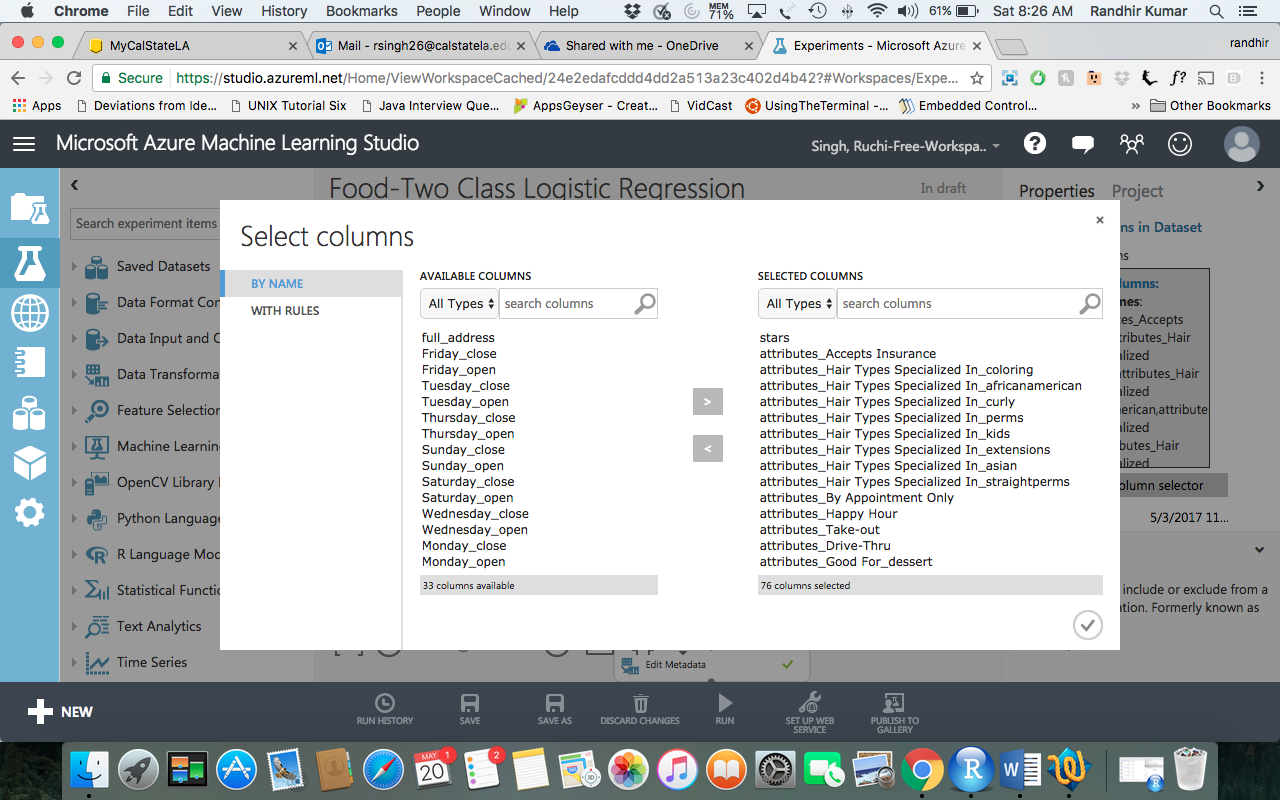


Step 2: Preprocess and Clean the Data

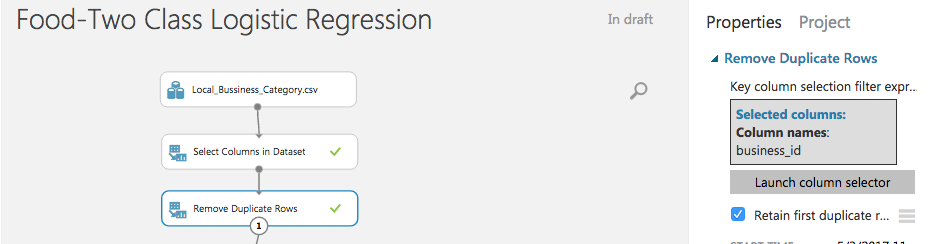
A dataset usually requires some preprocessing before it can be analyzed. You might have noticed the missing values present in the columns of various rows. These missing values need to be cleaned so the model can analyze the data correctly. In our case, we'll remove any rows that have missing values.

First we'll remove the normalized-losses column, and then we'll remove any row that has missing data.

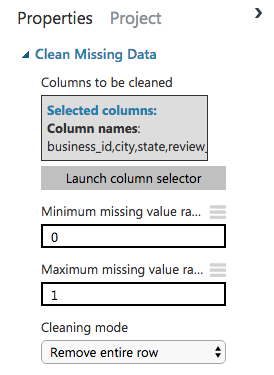
1. Search for project columns in the Search box at the top of the module palette to find the [Project Columns](https://msdn.microsoft.com/library/azure/1ec722fa-b623-4e26-a44e-a50c6d726223/) module, then drag it to the experiment canvas and connect it to the output port of the Yelp (Raw) dataset. This module allows us to select which columns of data we want to include or exclude in the model.
2. Select the [Project Columns](https://msdn.microsoft.com/library/azure/1ec722fa-b623-4e26-a44e-a50c6d726223/) module and click Launch column selector in the Properties pane.
3. Make sure all the relevant columns are selected in the filter drop-down list, Begin With. This directs [Project Columns](https://msdn.microsoft.com/library/azure/1ec722fa-b623-4e26-a44e-a50c6d726223/) to pass through all the columns (except those we're about to exclude).[ stars,attributes\_Accepts Insurance,attributes\_Hair Types Specialized In\_coloring,attributes\_Hair Types Specialized In\_africanamerican,attributes\_Hair Types Specialized In\_curly,attributes\_Hair Types Specialized In\_perms,attributes\_Hair Types Specialized In\_kids,attributes\_Hair Types Specialized In\_extensions,attributes\_Hair Types Specialized In\_asian,attributes\_Hair Types Specialized In\_straightperms,attributes\_By Appointment Only,attributes\_Happy Hour,attributes\_Take-out,attributes\_Drive-Thru,attributes\_Good For\_dessert,attributes\_Good For\_latenight,attributes\_Good For\_lunch,attributes\_Good For\_dinner,attributes\_Good For\_brunch,attributes\_Good For\_breakfast,attributes\_Caters,attributes\_BYOB,attributes\_Corkage,attributes\_Noise Level,attributes\_Takes Reservations,attributes\_DietaryRestrictions\_dairy-free,attributes\_DietaryRestrictions\_glutenfree,attributes\_DietaryRestrictions\_vegan,attributes\_DietaryRestrictions\_kosher, attributes\_DietaryRestrictions\_halal, attributes\_DietaryRestrictions\_soyfree, attributes\_DietaryRestrictions\_vegetarian,attributes\_Delivery,attributes\_DogsAllowed,attributes\_CoatCheck,attributes\_Smoking,attributes\_Ambience\_romantic,attributes\_Ambience\_intimate, attributes\_Ambience\_classy,attributes\_Ambience\_hipster,attributes\_Ambience\_divey,attributes\_Ambience\_touristy,attributes\_Ambience\_trendy,attributes\_Ambience\_upscale,attributes\_Ambience\_casual,attributes\_Parking\_garage,attributes\_Parking\_street,attributes\_Parking\_validated,attributes\_Parking\_lot,attributes\_Parking\_valet,attributes\_AgesAllowed,attributes\_WheelchairAccessible,attributes\_Open24Hours,attributes\_Music\_dj,attributes\_Music\_background\_music,attributes\_Music\_jukebox,attributes\_Music\_live,attributes\_Music\_video,attributes\_Music\_karaoke,attributes\_HasTV,attributes\_OutdoorSeating,attributes\_Attire,attributes\_Alcohol,attributes\_WaiterService,attributes\_Wi-Fi,attributes\_OrderatCounter,attributes\_AcceptsCreditCards,attributes\_BYOB/Corkage,attributes\_GoodforKids,attributes\_GoodForGroups,attributes\_GoodForDancing,business\_id,Category,city,state,review\_count]



1. Click the check mark (OK) button to close the column selector. The properties pane for Project Columns shows that it will pass through all the selected columns from the dataset.
2. Next module we need to add is Remove Duplicate rows. This will remove all the duplicate rows based on the Business id.

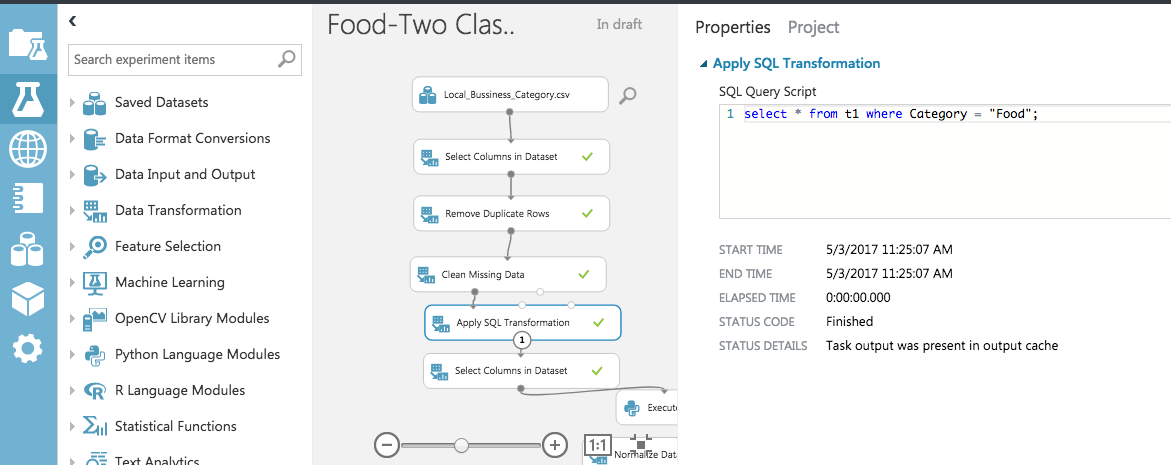


1. Now search for the module named clean missing data. Drag that to the experiment. Set the Properties pane. Fill minimum missing value as 0 and maximum missing value as 1 and select the columns business\_is, city state,review\_count, stars and category



Now that the data is clean our next steps will be to process the data.

1. Next we use an SQL Transformation module for writing an SQL query script. In this experiment I wish to predict the popularity of Food business. Thus lets write a query to select rows with category Food.



1. Now we need to select columns relevant to Food category. Use a Column Selector again and select the columns, review\_count,stars,attributes\_Good For\_lunch,attributes\_Good For\_dinner,attributes\_Good For\_breakfast,attributes\_Take-out,attributes\_Takes Reservations,attributes\_Parking\_lot, attributes\_Delivery, attributes\_WheelchairAccessible,attributes\_Alcohol,attributes\_WaiterService,attributes\_Wi-Fi,attributes\_Noise Level
2. Next we write a Python script int Execute Python Script module to categorize stars < 3 as 0 and stars > 3 as 1.

def set\_readmit\_class(x):

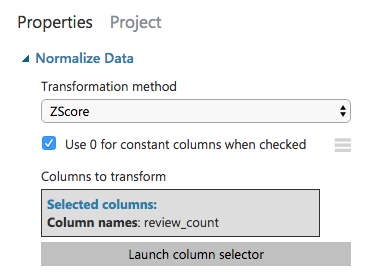
return [0 if (y < 3) else 1 for y in x]

def azureml\_main(df):

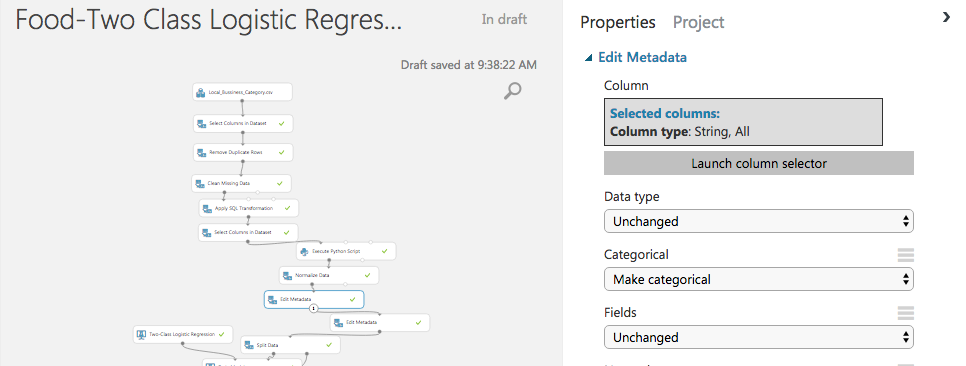
df['stars'] = set\_readmit\_class(df['stars'])

return df

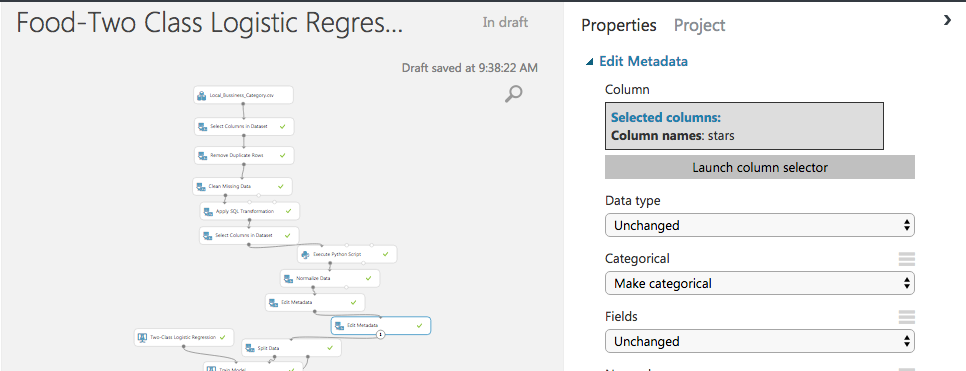
1. Now we need to normalize the review\_count using the Normalize Data module.



1. Change all String type columns to Category using Edit Metadata.



1. We also need to convert stars to Category using another Edit Metadata.



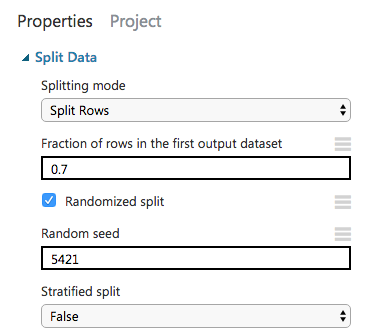
Step 3: Choose and apply machine learning algorithms:

Now that the data is ready, constructing a predictive model consists of training and testing. I am using data to train the model and then test the model to see how close it's able to predict ratings.

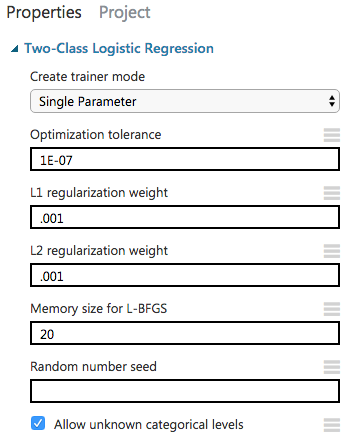
***Classification*** and ***Regression*** are 2 types of supervised machine learning techniques. We want to predict the popularity of the business, which can be value 0 meaning unpopular (stars < 3) and 1 meaning popular (stars > 3), so we'll use a Classification model.

Our data for both training and testing by splitting it into separate training and testing sets. Select and drag the [Split Data](https://msdn.microsoft.com/library/azure/70530644-c97a-4ab6-85f7-88bf30a8be5f/) module to the experiment canvas and connect it to the output of the last Edit Metadata module. Set **Fraction of rows in the first output dataset** to 0.7 and Random Seed of 5421. This way, we'll use 70 percent of the data to train the model, and hold back 30 percent for testing.

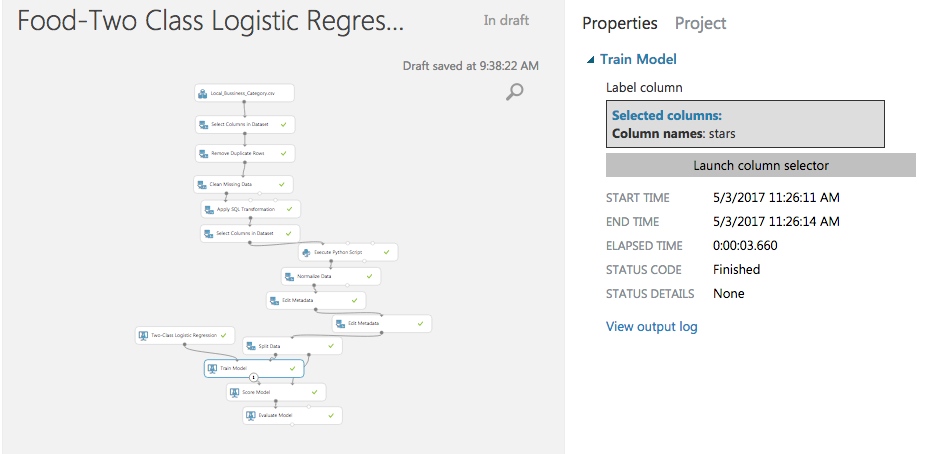
1. Run the experiment. This allows the Clean and processed data and then [Split Data](https://msdn.microsoft.com/library/azure/70530644-c97a-4ab6-85f7-88bf30a8be5f/) modules to pass column definitions to the modules we'll be adding next.



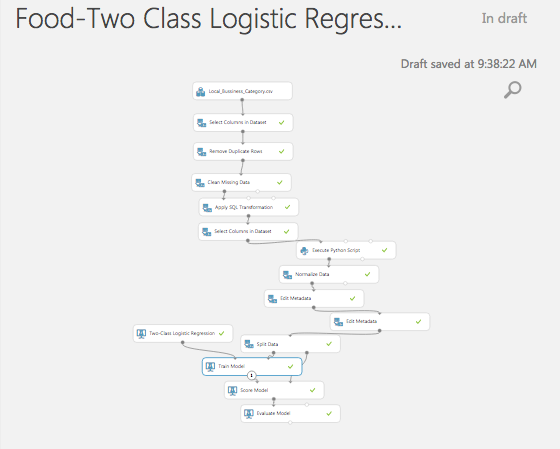
1. To select the learning algorithm, expand the Machine Learning category in the module palette to the left of the canvas, and then expand Initialize Model. This displays several categories of modules that can be used to initialize machine learning algorithms. For this experiment, select the Two Class Logistic Regression module under the Classification category and drag it to the experiment canvas.



1. Find and drag the [Train Model](https://msdn.microsoft.com/library/azure/5cc7053e-aa30-450d-96c0-dae4be720977/) module to the experiment canvas. Connect the left input port to the output of the Two class Logistic Regression module. Connect the right input port to the training data output (left port) of the [Split Data](https://msdn.microsoft.com/library/azure/70530644-c97a-4ab6-85f7-88bf30a8be5f/) module.

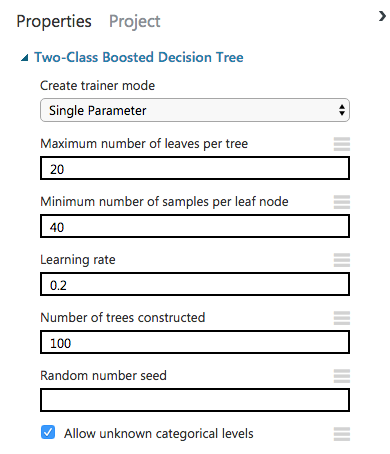


1. Select the [Train Model](https://msdn.microsoft.com/library/azure/5cc7053e-aa30-450d-96c0-dae4be720977/) module, click Launch column selector in the Properties pane, and then select the Stars column. This is the value that our model is going to predict.
2. Add Score Module and an Evaluate Module to see the evaluation of Two Class logistic regression.
3. Run the experiment.

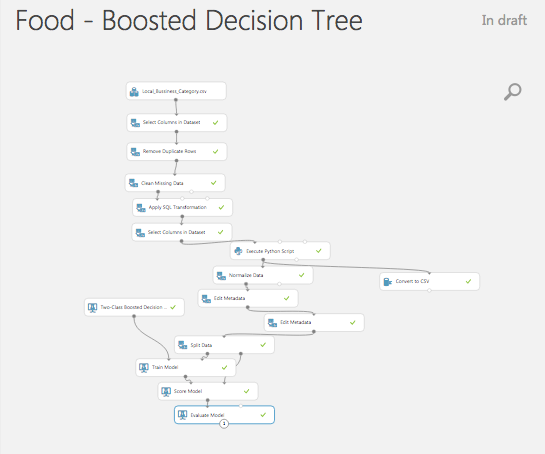


We can try another classification machine learning algorithm and evaluate it to see which model gives a better score. To do so we will need to follow the bellow steps after removing Two class Logistic regression module from the experiment.

1. To select the learning algorithm, expand the Machine Learning category in the module palette to the left of the canvas, and then expand Initialize Model. This displays several categories of modules that can be used to initialize machine learning algorithms. For this experiment, select the Two Class Boosted Decision Tree module under the Classification category and drag it to the experiment canvas.



1. Find and drag the [Train Model](https://msdn.microsoft.com/library/azure/5cc7053e-aa30-450d-96c0-dae4be720977/) module to the experiment canvas. Connect the left input port to the output of the Two class Logistic Regression module. Connect the right input port to the training data output (left port) of the [Split Data](https://msdn.microsoft.com/library/azure/70530644-c97a-4ab6-85f7-88bf30a8be5f/) module.
2. Select the [Train Model](https://msdn.microsoft.com/library/azure/5cc7053e-aa30-450d-96c0-dae4be720977/) module, click Launch column selector in the Properties pane, and then select the Stars column. This is the value that our model is going to predict.
3. Add Score Module and an Evaluate Module to see the evaluation of Two Class Boosted Decision Tree.
4. Run the experiment.

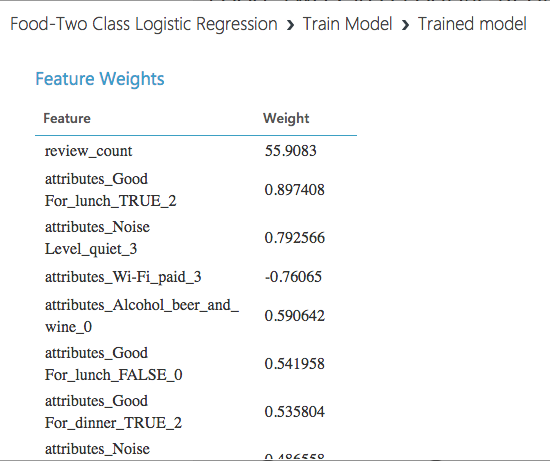


Step 4: Visualization:

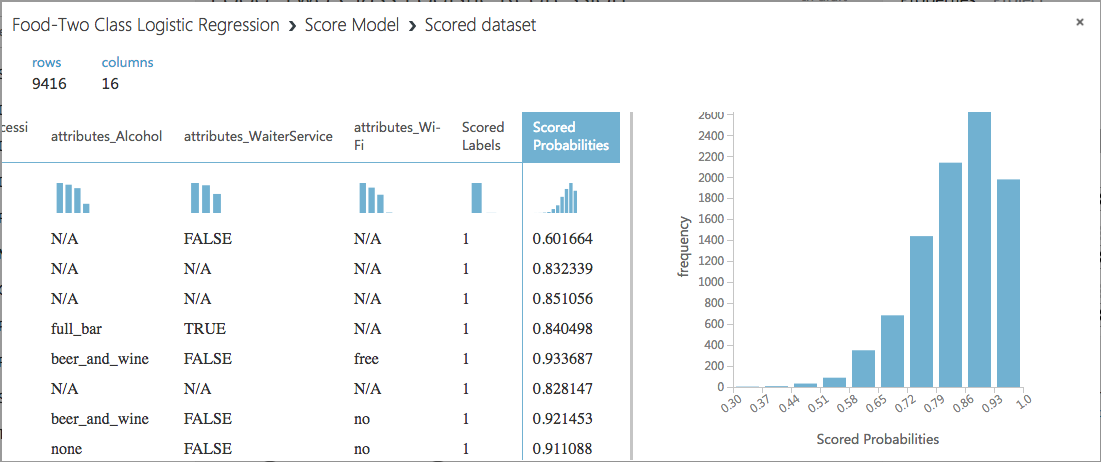
To compare the two models we need to visualize the output of Training module, Score module and Evaluate module first for Two class Logistic regression and then for Two class Boosted decision tree.

Two Class Logistic Regression

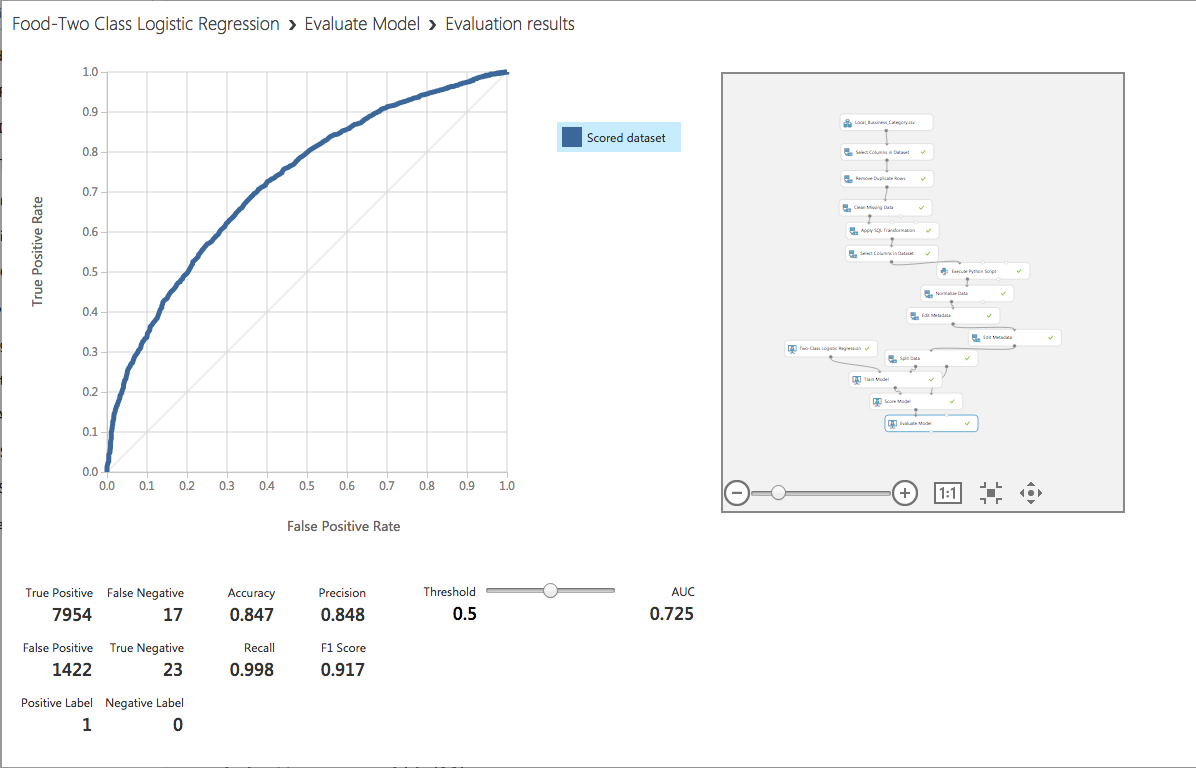
1. The Visualize the output of the train module. It shows the weight of the features for the prediction.



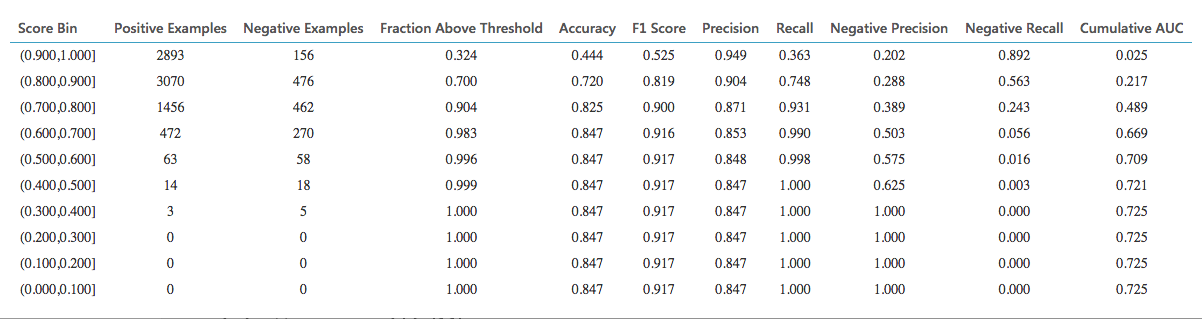
1. Visualize the Score model to see the Scored label and the Scored probabilities. Select the Scored probability column in the table to see the probability distribution of the prediction. It can be seen thet the frequency of the probability 0.7-1 is very high, which indicates that the probability of our perdic to be correct is very high.



1. Visualize the Evaluate model to of see the final result of our experiment. As expected it shows a very good accuracy of 0.8, recall of 0.9 and AUC of 0.7

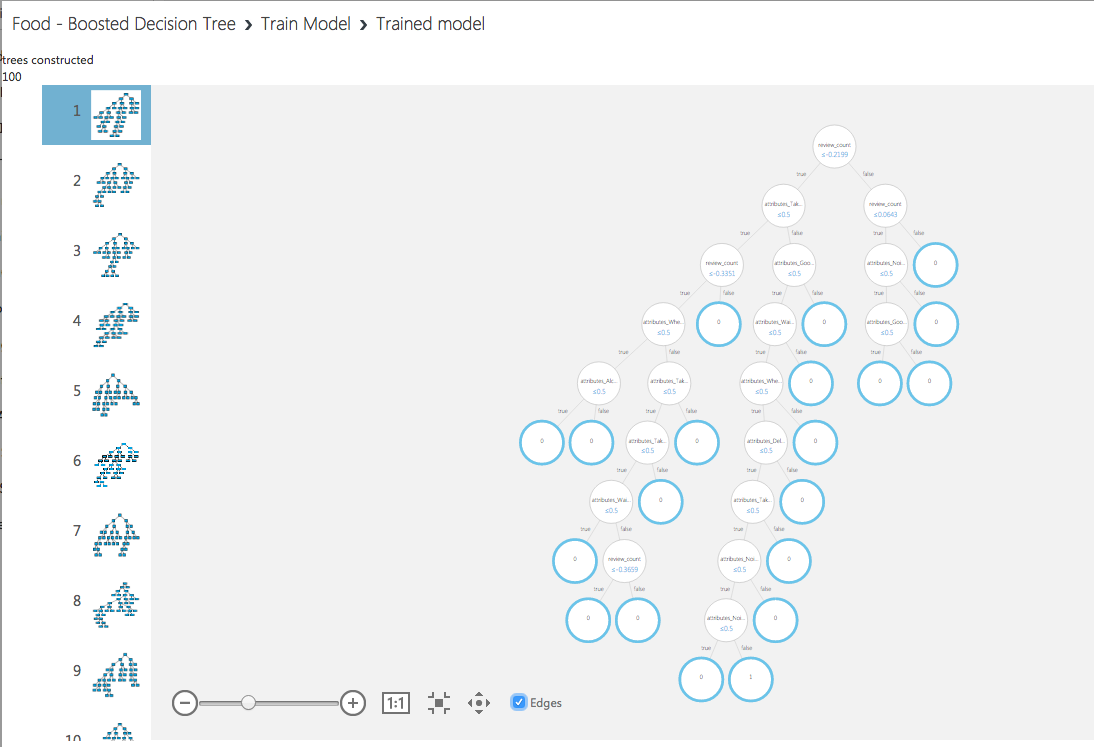


This the detail of every score bin and its prediction.

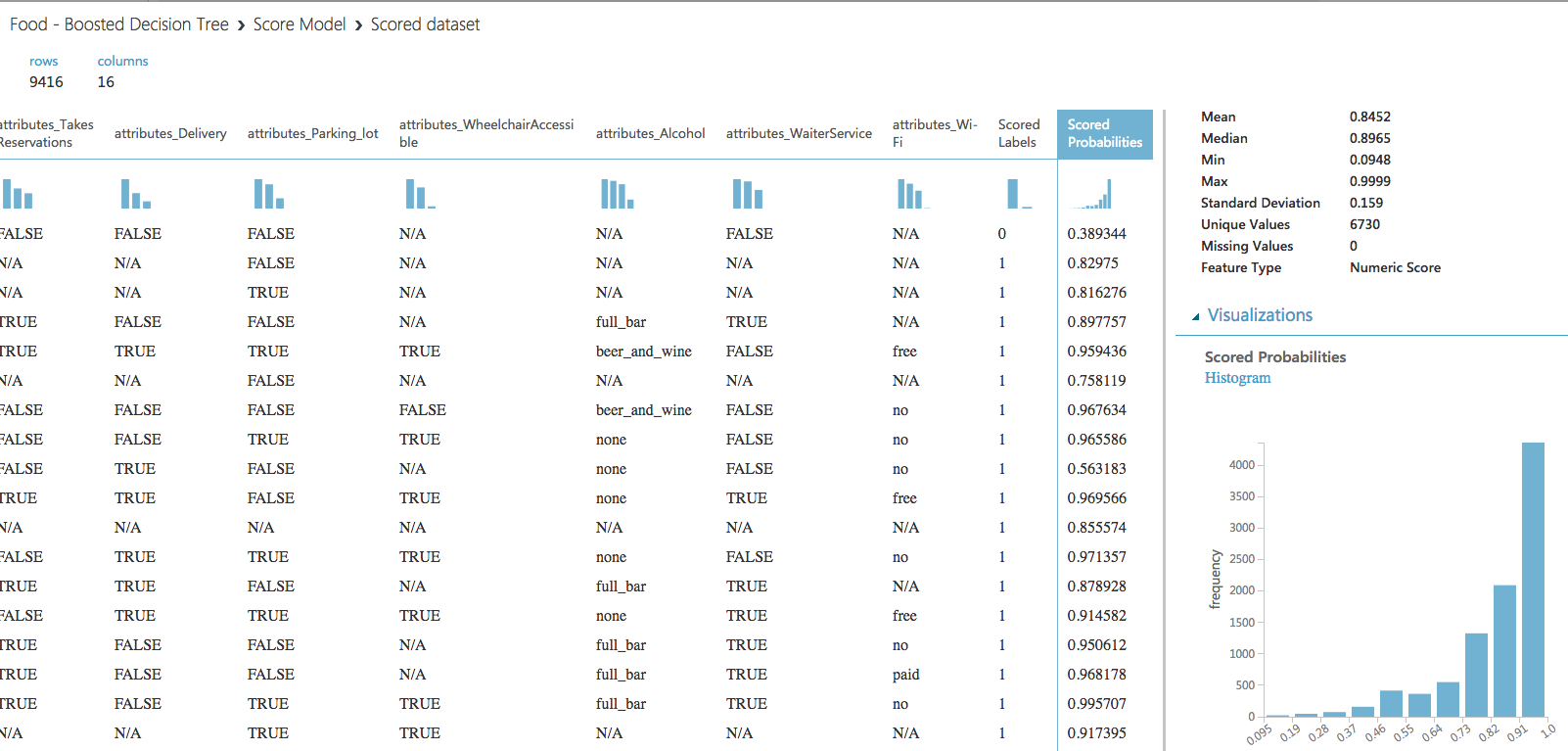


Two Class Boosted Decision Tree

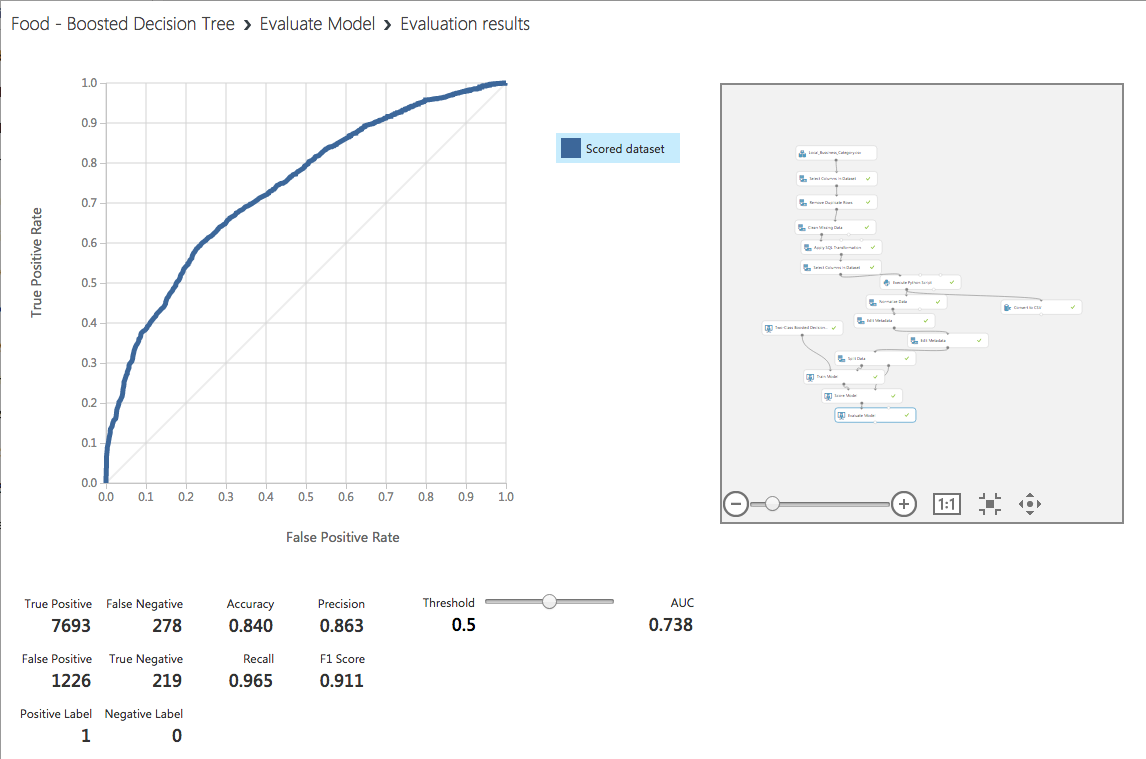
1. The Visualize the output of the train module. It shows the weight of the features for the prediction.



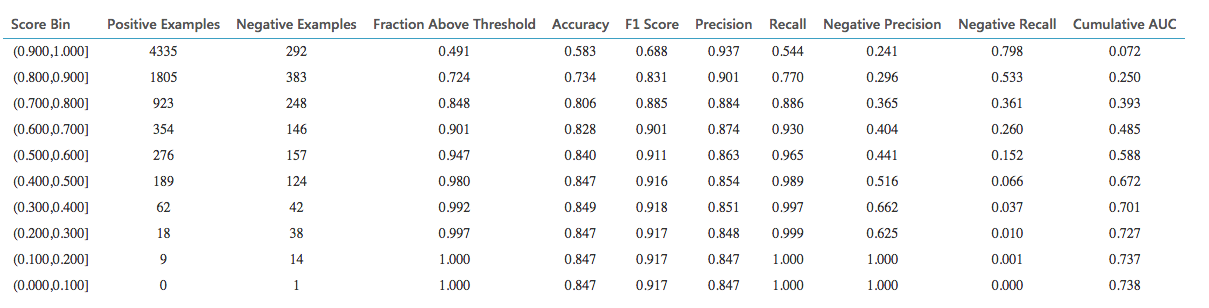
1. Visualize the Score model to see the Scored label and the Scored probabilities. Select the Scored probability column in the table to see the probability distribution of the prediction. It can be seen thet the frequency of the probability 0.7-1 is very high, which indicates that the probability of our perdic to be correct is very high.



1. Visualize the Evaluate model to of see the final result of our experiment. As expected it shows a very good accuracy of 0.8, recall of 0.9 and AUC of 0.7



This is the detail of every score bin and its prediction.

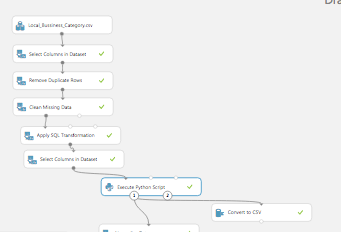


Step 5: Comparison of models to Predict popularity of business:

The two Class Logistic Regression and Two class Boosted Decision tree both give a good score of AUC = 0.7 which indicates that either of the two models can be used to predict the popularity of the business. The Two class Boosted Decision Tree has slightly better score than the logistic regression.These classification models are apt to predict whether a busibess will get less than 3 stars and will be unpopular or will get more that 3 stars and will be popular , depending upon the attributes of the business like, good for breakfast, good for lunch, good for dinner, parking lot, takeout, delivery, alcohol, waiter service, wi-fi and noise level. Therefore either od the two models in our case can be used to have a good prediction accuracy.

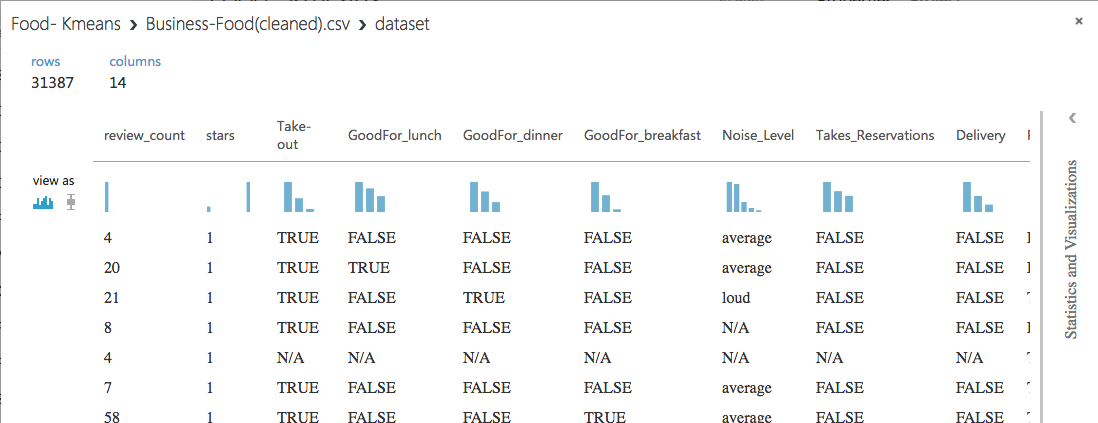
Step 1: Upload data

We saved the clean dataset as CSV from the Classification experiment to be used in the Clustering experiment for Food category.



The saved dataset in Azure is called Business-Food(Cleaned).csv.

1. Start a new experiment by clicking +NEW at the bottom of the Machine Learning Studio window, select EXPERIMENT, and then select Blank Experiment. Select the default experiment name at the top of the canvas and rename it to Food- Kmeans .
2. To the left of the experiment canvas is a palette of datasets and modules. Search for the dataset Business-Food(Cleaned).csv to use for the experiment.
3. Drag the dataset to the experiment canvas.
4. To see what this data looks like, click the output port at the bottom of the Business-Food(Cleaned) dataset, and then select **Visualize**.

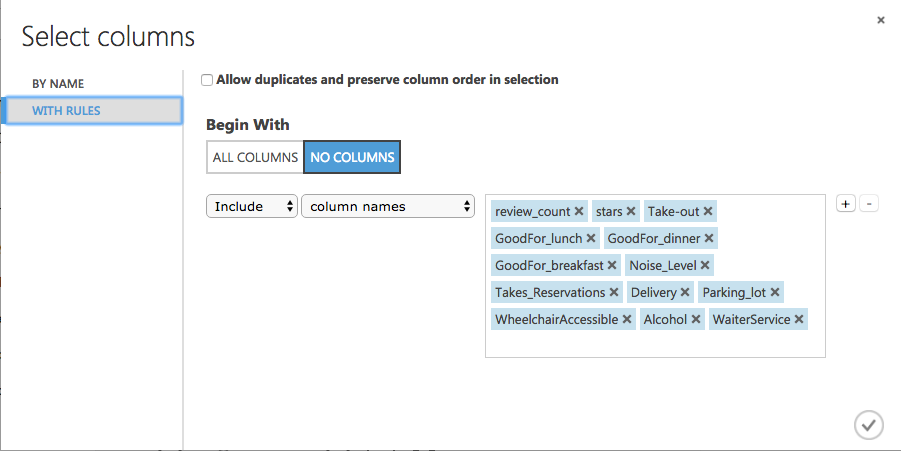


Step 2: Preprocess and Clean the Data

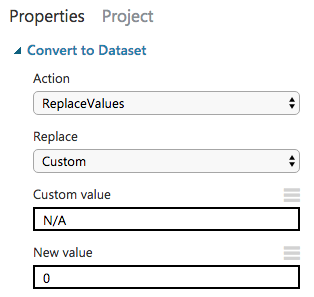
A dataset usually requires some preprocessing before it can be analyzed. You might have noticed the missing values present in the columns of various rows. These missing values need to be cleaned so the model can analyze the data correctly. In our case, we'll remove any rows that have missing values.

First, we'll remove the not required columns, and then we'll remove any row that has missing data.

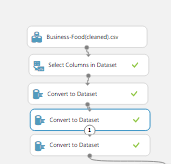
1. Search for project columns in the Search box at the top of the module palette to find the [Project Columns](https://msdn.microsoft.com/library/azure/1ec722fa-b623-4e26-a44e-a50c6d726223/) module, then drag it to the experiment canvas and connect it to the output port of the Yelp (Raw) dataset. This module allows us to select which columns of data we want to include or exclude in the model.
2. Select the [Project Columns](https://msdn.microsoft.com/library/azure/1ec722fa-b623-4e26-a44e-a50c6d726223/) module and click Launch column selector in the Properties pane.
3. Make sure all the relevant columns to Food Category are selected in the filter drop-down list, Begin With. This directs [Project Columns](https://msdn.microsoft.com/library/azure/1ec722fa-b623-4e26-a44e-a50c6d726223/) to pass through all the columns [review\_count,stars,Take-out,GoodFor\_lunch,GoodFor\_dinner,GoodFor\_breakfast,Noise\_Level,Takes\_Reservations,Delivery,Parking\_lot,WheelchairAccessible,Alcohol,WaiterService].



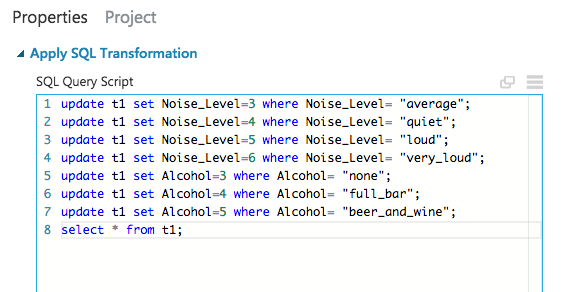
1. Click the check mark (OK) button to close the column selector. The properties pane for Project Columns shows that it will pass through all the selected columns from the dataset.
2. Next select the Convert to Dataset module to replace the N/A values to 0.



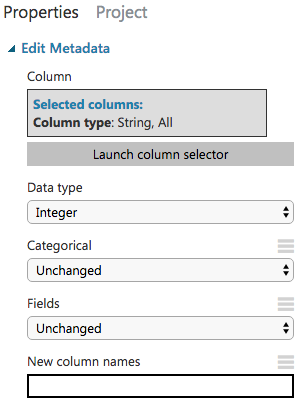
1. Add two more such modules to convert False to 1 and True to 2. After that the Experiment should look like below.



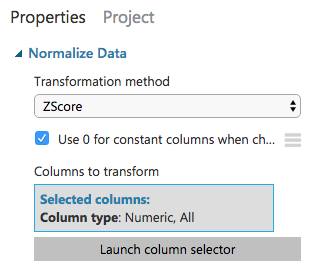
1. Next, we use an SQL Transformation module for writing an SQL query script. This query should update the various values for Noise level and Alcohol to numeric values.



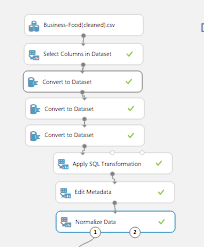
1. Add edit metadata to change all String type columns to Category.



1. Now we need to normalize all the numeric columns using the Normalize Data module.



1. After the completion of all the data cleaning and processing steps the experiment should look like below.



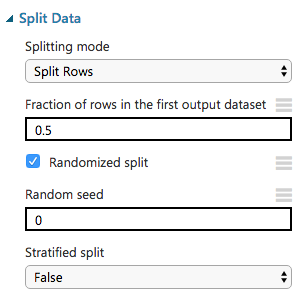
Step 3: Choose and apply machine learning algorithms:

Now that the data is ready, constructing two clustering model based on the intuitive grouping of the data. I am using data to train the model and then evaluate the model to understand which is better and why.

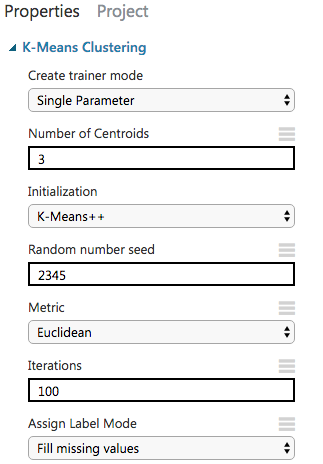
***Clustering*** is an un-supervised machine learning technique. Using the k-means clustering model we shall be able to classify the food business based on their attributes. It can help the business owners understand why a food business has low or high review count. It can also suggest the investor or new business owners, the combination of food business attributes that could lead to increase in review count.

Every column in our clean dataset is categories, thus there is no need for cleaning the outliers from the dataset. We shall use 3 cluster and 5 cluster K-means algorithms to compare which is better algorithm for our dataset.

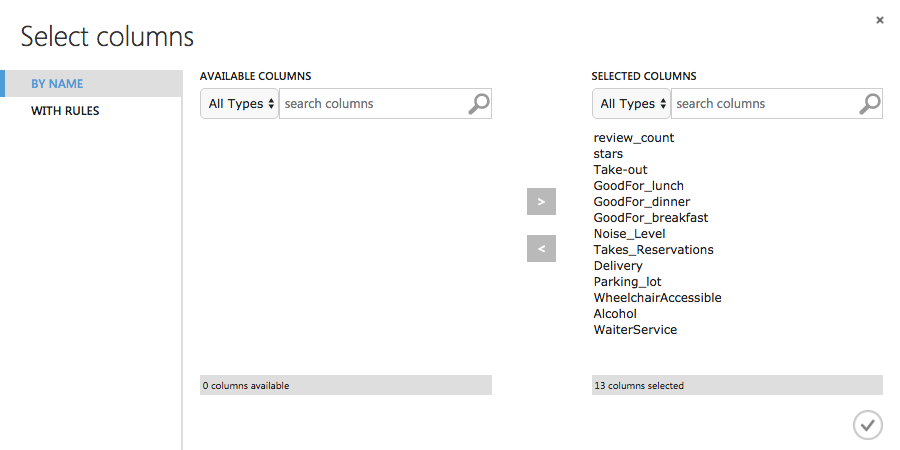
1. Run the experiment. This allows the Clean and processed data and then [Split Data](https://msdn.microsoft.com/library/azure/70530644-c97a-4ab6-85f7-88bf30a8be5f/) modules to pass column definitions to the modules we'll be adding next. We split the data to 0.5 to that we can used 50% of the data for 3 cluster K-means and 50% for 5 cluster K means.



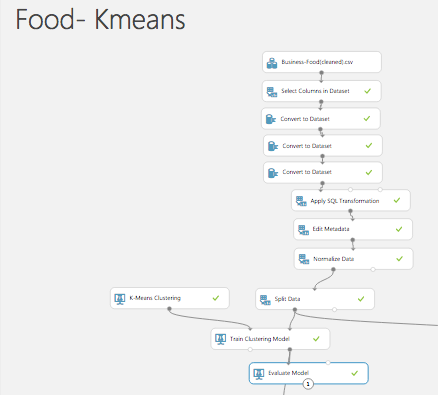
1. To select the learning algorithm, expand the Machine Learning category in the module palette to the left of the canvas, and then expand Initialize Model. This displays several categories of modules that can be used to initialize machine learning algorithms. For this experiment, select the K-Means Clustering module under the Clustering category and drag it to the experiment canvas. Configure it as below.



1. Find and drag the Train Clustering Model module to the experiment canvas. Connect the left input port to the output of the K-Means Clustering module. Connect the right input port to the training data output (left port) of the [Split Data](https://msdn.microsoft.com/library/azure/70530644-c97a-4ab6-85f7-88bf30a8be5f/) module.
2. Select the Train Clustering Model module, click Launch column selector in the Properties pane, and then select all the feature column. Based on these columns the model will cluster all the food business.

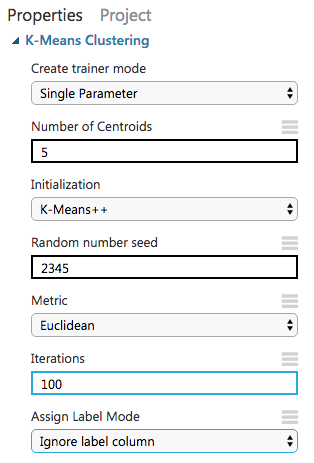


1. Add Evaluate Module to see the evaluation of 3 cluster K-Means Clustering algorithm.
2. Make sure the experiment looks like below and then run the experiment.

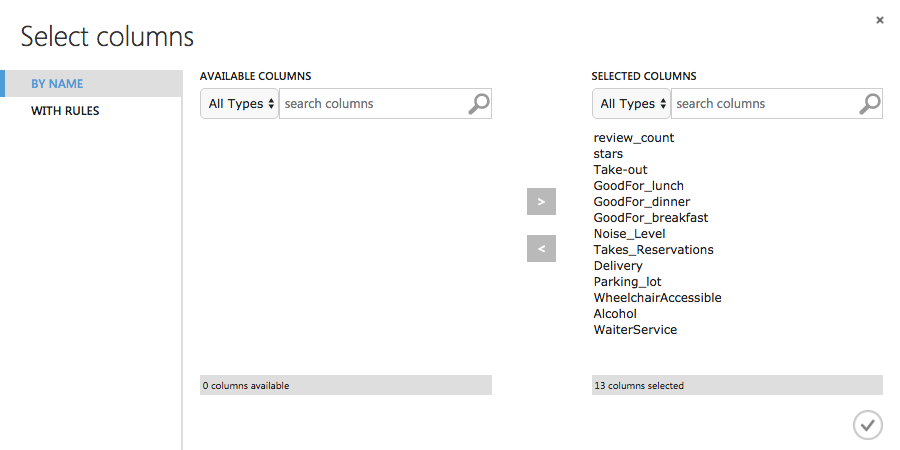


We can try another K-Means Clustering module with 5 cluster and evaluate it to see which model gives a better score. To do so we will need to follow the bellow steps.

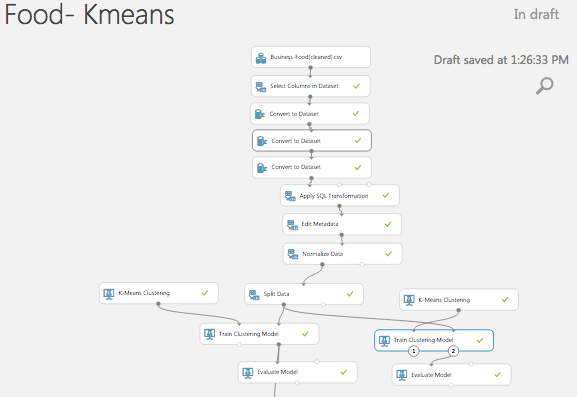
1. Select another the K-Means Clustering module under the Clustering category and drag it to the experiment canvas. Configure it as below.



1. Find and drag the Train Clustering Model module to the experiment canvas. Connect the left input port to the output of the K-Means Clustering module. Connect the right input port to the training data output (left port) of the [Split Data](https://msdn.microsoft.com/library/azure/70530644-c97a-4ab6-85f7-88bf30a8be5f/) module.
2. Select the Train Clustering Model module, click Launch column selector in the Properties pane, and then select all the feature column. Based on these columns the model will cluster all the food business.



1. Add Evaluate Module to see the evaluation of 3 cluster K-Means Clustering algorithm.
2. Make sure the experiment looks like below and then run the experiment.

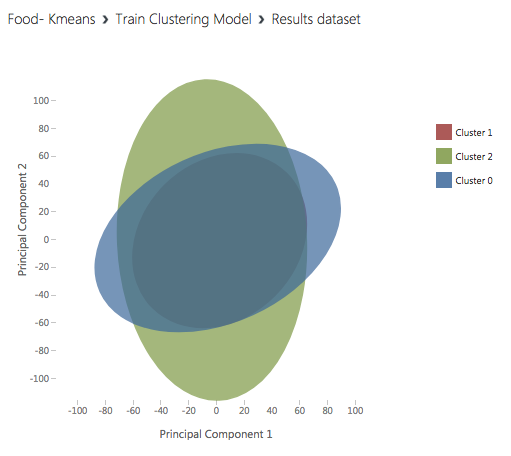


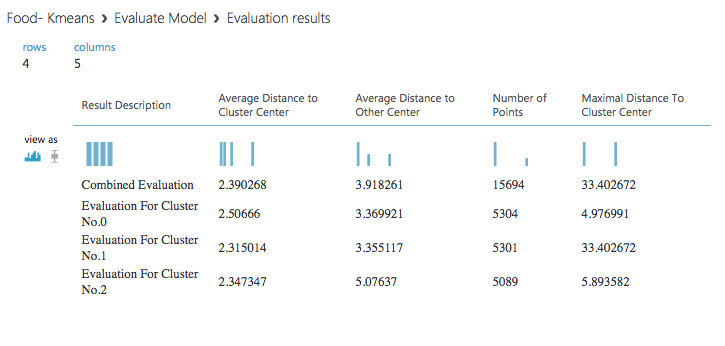
Step 4: Visualization:

To compare the two models, we need to visualize the output of Training module and Evaluate module first for 3 Cluster K-Means Clustering and then for 5 Cluster K-Means Clustering.

3 Cluster K-Means Clustering

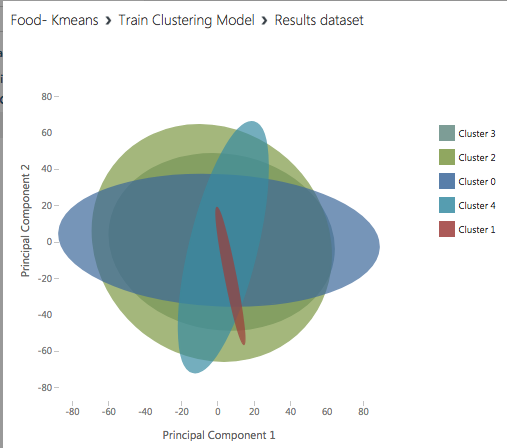
1. The Visualize the right output (result dataset) of the train clustering module. It shows the 3 clusters created by the algorithm. There are two distinct ellipses shown on this projection of the first two principle components. The major axes (long dimension) of each ellipse are in a distinct, nearly orthogonal, indicating the two clusters have good separation. The last cluster is circular in shape and is completely overlapped.

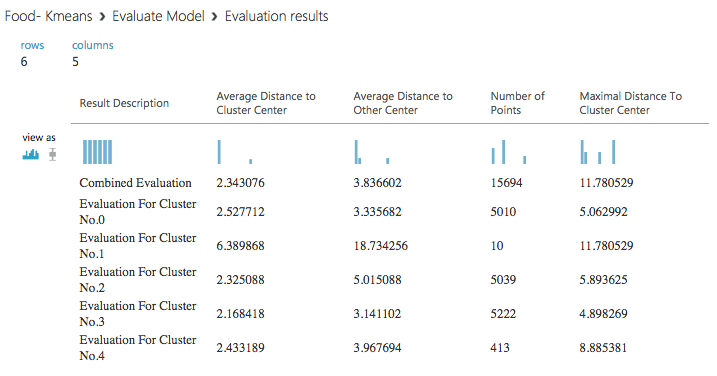


1. Visualize the Evaluate model to of see the final result of our experiment. This shows that there are 3 cluster centers and the Maximum Distance to Cluster Center for cluster. Of all the 3 clusters 33.4 is the maximum distance.

5 Cluster K-Means Clustering

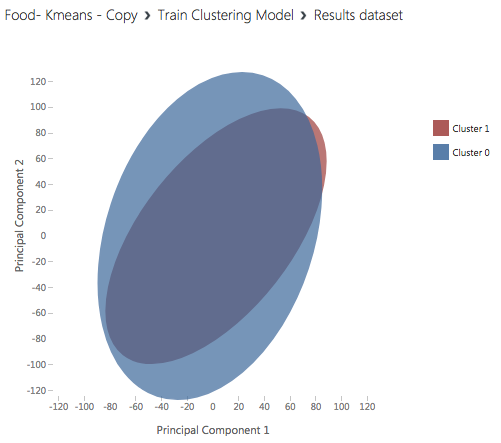
1. The Visualize the right output (result dataset) of the train clustering module. It shows the 3 clusters created by the algorithm. The ellipse for three clusters (Cluster 0, Cluster 1 and Cluster 4) looks about the same. However, the ellipses for the other two clusters (Cluster 2 and Cluster 3) are nearly circular. Further the direction of the major axes of the ellipses for Cluster 2 and Cluster 3 are close to that of Cluster 4. The separation of the clusters in this projection is poor and it appears that five clusters are too many for this dataset.



1. Visualize the Evaluate model to of see the final result of our experiment. This shows that there are 5 cluster centers and the Maximum Distance to Cluster Center for cluster. Of all the 5 clusters 11.7 is the maximum distance.

Step 5: Comparison of models for clustering of Food business:

From the Train Clustering module, we find out that 5 cluster has less distinct ovals compared to the 3 cluster, making 5 cluster less favorable for our dataset. From the Evaluation module, we find out that the maximum distance of the points plotted from the cluster center is grater in 3 cluster as compared to the 5 cluster. This means that the in 5 cluster the points are better clustered together. To conclude a better cluster, we tested a 2-cluster medal and following was the result on training the model.



The major axes of the ellipses for Cluster 0 and Cluster 1 are close and the separation of the clusters in this projection is poor. This helps us conclude that cluster 3 and cluster 5 have better results. Reducing the number of clusters will not improve the results further.

**Creating a Model:**

This tutorial shows how to create a Model in AzureML using text analysis modules. Applications of text classification include categorizing newspaper articles and news wire contents into topics, organizing web pages into hierarchical categories, filtering spam email, sentiment analysis, predicting user intent from search queries, routing support tickets, and analyzing customer feedback.

**Each instance in the data set has 6 fields:**

* business\_id – unique id for the business
* date – date of the tip
* likes – number of likes given to a tip range from 0 to 10.
* text- text of the users’s thoughts about business
* type- named as tip for the text provided
* user\_id - the user who posted the tip

Two columns text and likes are used in text analysis.

**Step 1: Data Preparation**

In this step the data is uploaded into AzureML, then it is transformed and cleaned so that it is suitable for the text analysis algorithms.

1. Open a browser and browse to [https://studio.azureml.net.](https://studio.azureml.net/) Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment, and give it the title **Text analysis**.
3. Download the tip*.*csvfile and drag it to canvas.
4. Search for the **Select Columns in dataset(Project Column)**module and drag it onto the canvas.
5. Connect the output of the **tip** dataset to the **Dataset**input of **Select Columns in dataset(Project**

**Column).**

1. Select columns to be required by the experiment by clicking **Launch Column selector By name:**

* text
* likes
* Business\_id
* User\_id
* Date
* type

1. The records with missing text values are removed using **Clean Missing Values** module. Search for **Clean Missing data module** and drag it to the canvas. Connect the output of **Select Columns in dataset module** to the input of the **Clean missing data module**. **Select Launch Column Selector**. With Rules all Columns and select following:

**Minimum missing value ratio – 0**

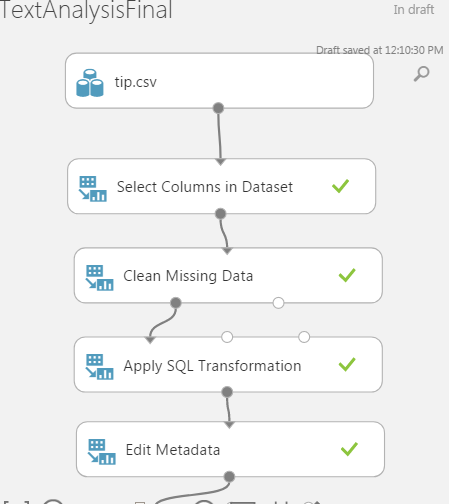
**Maximum missing value ratio – 1**

**Cleaning mode – Remove entire row**

1. Search for **Apply SQL Transformation module** and drag it to canvas. Connect the output of **Clean missing data** to the input of the **Apply SQL Transformation**. Copy and paste the SQL query below to select likes from 1 to 10 in default table t1 made for dataset:

select \* from t1 where likes IN ('1','2','3','4','5','6','7','8','9','10');

1. Search for **Edit Metadat(Metadata Editor) module** and drag it to the canvas. Connect the output of the **Apply SQL transformation module** to the input of the **Edit metadata module**. **Launch Column selector** By name and select likes columns only.
2. Run the experiment.



1. Search for **Apply SQL Transformation module** and drag it to the canvas. Copy and paste the Sql query below to delete unnecessary business\_id and user\_id in t1:

DELETE FROM t1

WHERE business\_id = '#NAME?';

DELETE FROM t1

WHERE user\_id = '#NAME?';

select \* from t1;

1. Search for **Select Columns in dataset(Project Columns)** module and drag it to the canvas. Connect the output of the **Apply SQL Transformation module** to the input of the **Select Columns in dataset(Project Columns)** module. Launch Column selector and with Rules No columns include text and likes columns.
2. Use the **Partition and Sample** module to select the top record in the dataset. Connect the output of the **Select Columns in dataset(Project Columns)** module to the input of the **Partition and Sample** module.
3. Search for Split Data module and drag it to the canvas. Connect the output of **Select Columns in dataset(Project Columns)** to the input of Split data module and apply following:

**Splitting mode : Split Rows**

**Fraction of rows in first output dataset : 0.2**

**Randomized Split : selected**

**Random Seed : 0**

**Stratified split : False**

**Step 2: Creating R scripts to Data Preprocessing:**

Unstructured text such as tips, tweets, product reviews, or search queries usually requires some preprocessing before it can be analyzed. This experiment includes a number of optional text preprocessing and text cleaning steps, such as replacing special characters and punctuation marks with spaces, normalizing case, removing duplicate characters, removing user-defined or built-in stop-words, and word stemming. By preprocessing the text, you can more easily create meaningful features from text. For example, the **Preprocess Text** module supports these common operations on text. These steps are implemented using the R programming language. We’ll use default dataset named Stopwords in the Azureml to compare the text and remove the stopwords in our dataset.

1. Search for **Execute R script module** and drag it to the canvas. Connect the first output of split data module to the first input od Execute R script, Copy and paste the R script below:

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

# Please determine the required text preprocessing steps using the following flag

replace\_special\_chars <- TRUE

remove\_duplicate\_chars <- TRUE

replace\_numbers <- TRUE

convert\_to\_lower\_case <- TRUE

remove\_default\_stopWords <- FALSE

remove\_given\_stopWords <- TRUE

stem\_words <- TRUE

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

# Map 1-based optional input ports to variables

dataset1 <- maml.mapInputPort(1) # class: data.frame

# get the label and text columns from the input data set

text\_column <- dataset1[["text"]]

label\_column <- dataset1[["likes"]]

stopword\_list <- NULL

result <- tryCatch({

dataset2 <- maml.mapInputPort(2) # class: data.frame

# get the stopword list from the second input data set

stopword\_list <- dataset2[[1]]

}, warning = function(war) {

# warning handler

print(paste("WARNING: ", war))

}, error = function(err) {

# error handler

print(paste("ERROR: ", err))

stopword\_list <- NULL

}, finally = {})

# Load the R script from the Zip port in ./src/

source("src/text.preprocessing.R");

text\_column <- preprocessText(text\_column,

replace\_special\_chars,

remove\_duplicate\_chars,

replace\_numbers,

convert\_to\_lower\_case,

remove\_default\_stopWords,

remove\_given\_stopWords,

stem\_words,

stopword\_list)

data.set <- data.frame(

label\_column,

text\_column,

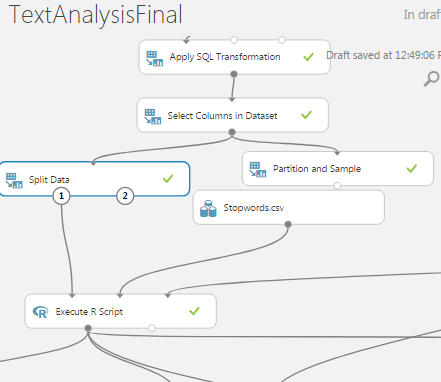
stringsAsFactors = FALSE

)

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

maml.mapOutputPort("data.set")

1. Load the dataset that contains the stopwords.
2. Attach the loaded dataset to the second port of the **Execute R Script** module.
3. Run the experiment.



1. Search for the text.oreprocessing.zip dataset that already exist in Azure and drag it to the canvas. Connect the output of the Dataset to the third output of the **Execute R script module.**
2. Search for the **Execute R script module** and drag it to the canvas. Connect the first output of first **Execute R script module** with first input of the new **Execute R script module**. Connect the output of the text.preprocessing.zip to the third input of new **Execute R script module**. We’ll make word cloud with this R script. Its taking label and text columns from the first input data set. The source is **text.preprocessing.R**. The script will draw word cloud with max words 50. Copy and paste the r script below :

# Map 1-based optional input ports to variables

dataset1 <- maml.mapInputPort(1) # class: data.frame

# get the label and text columns from the first input data set

text\_column <- dataset1[,"text\_column"]

label\_column <- dataset1[,"label\_column"]

# Load the R script from the Zip port in ./src/

source("src/text.preprocessing.R");

#freq = data.frame(sort(colSums(as.matrix(text\_column), decreasing=TRUE))

#wordcloud(rownames(freq), freq[,1], max.words=50, colors=brewer.pal(1, "Dark2"))

drawWordCloud(text\_column,label\_column, maxWords=50)

data.set <- dataset1

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

maml.mapOutputPort("data.set")

1. Go to the second output port of the last **Execute R Script** module named and select visualize if you need to see the most frequent words for each class. In the adopted use case, the first word cloud represents the top positive words and the second word cloud shows the most frequent negative words in the input training corpus.
2. Search for **Edit Metadata Module(Metadata Editor) module** and drag it to the canvas. Connect the first output of the first Execute R script to the input of the **Edit Metadata Module(Metadata Editor)** module. Launch Column Selector with Rules No cloumns select text\_column with following options:

**Data type: string**

**Categorical: Make non-categorial**

**Fields: unchanged**

**New column names: blank**

**Step 3: Creating Models:**

* **First Model :  N-grams TF feature extraction**

In the sample experiment, we set the number of hashing bits to 15, and set the number of n-grams to 2. With these settings, the hash table can hold 2^15 or 32,768 entries, in which each hashing feature represents one or more n-gram features and its value represents the occurrence frequency of that n-gram in the text instance. For many problems, a hash table of this size is more than adequate, but in some cases, more space might be needed to avoid collisions.

1. Search **Feature Hashing module** to transform a stream of English text into a set of features represented as integers. You can then pass this hashed feature set to a machine learning algorithm to train a text analysis model. The feature hashing functionality provided in this module is based on the Vowpal Wabbit framework. For more information, see [Train Vowpal Wabbit 7-4 Model](https://msdn.microsoft.com/en-us/library/azure/dn905861.aspx) or [Train Vowpal Wabbit 7-10 Model](https://msdn.microsoft.com/en-us/library/azure/mt674683.aspx). Connect the output of the **Edit Metadata Module(Metadata Editor**) module to theinput of the **Feature Hashing module.** Launch column selector and select text\_column with following options:

**Hashing bit size: 15**

**N- grams: 2**

The classification time and complexity of a trained model depends on the number of features (the dimensionality of the input space). For a linear model, such as a support vector machine, the complexity is linear with respect to the number of features. For text classification tasks, the number of features resulting from feature extraction is high because each word in the vocabulary and each n-gram is mapped to a feature. To select a more compact feature subset from the exhaustive list of extracted hashing features, we used the **Filter Based Feature Selection** module. The aim is to avoid the effects of the curse of dimensionality and to reduce the computational complexity without harming classification accuracy.

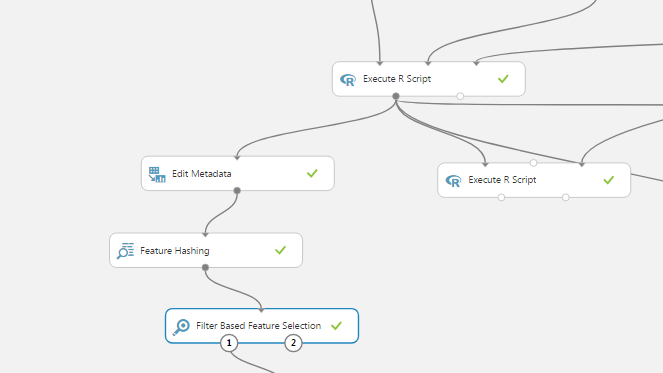
1. Search **Filter based Feature selection module** and drag it to the canvas. Connect the output of the **Feature Hashing module** to the input of **Filter based Feature selection module**. To get the top 1,000 most relevant features with respect to the sentiment label out of the 2^15 extracted features, we used the Chi-squared score function to rank the hashing features in descending order. Launch Column selector andselect label\_column. Select following:

**Feature Scoring Method : Chi Squared**

**Operate on Feature columns only : Selected**

**Number of desired features : 1000**

1. Run the experiment.



* **Second Model: Unigrams TF-IDF feature extraction**

**Create the Word Dictionary**

First, extract the set of unigrams (words) that will be used to train the text model. In addition to the unigrams, the number of documents where each word appears in the text corpus is counted (DF). It is not necessary to create the dictionary on the same labeled data used to train the text model. TF-IDF Calculation. When the metric word frequency of occurrence (TF) in a document is used as a feature value, a higher weight tends to be assigned to words that appear frequently in a corpus (such as stop-words). The inverse document frequency (IDF) is a better metric, because it assigns a lower weight to frequent words. You calculate IDF as the log of the ratio of the number of documents in the training corpus to the number of documents containing the given word. Combining these numbers in a metric (TF/IDF) places greater importance on words that are frequent in the document but rare in the corpus. This assumption is valid not only for unigrams but also for bigrams, trigrams, etc.

This experiment converts unstructured text data into equal-length numeric feature vectors where each feature represents the TF-IDF of a unigram in a text instance.

1.Search for **Execute R script module** and drag it to the canvas. Connect the output of the very first **Execute R script module** to the first input of the **Execute R script module**. Connect the output of the text.preprocessing.zip to the last input of the new **Execute R script module.**

Specify the following parameters in script for a word to be included in the dictionary created from the input dataset:

**a.** the minimum minWordLen and maximum maxWordLen length of a word.

**b.** the minimum minDF and the maximum maxDF document occurrence frequency.

Copy and paste the script below:

# Map 1-based optional input ports to variables

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

##################################################

# Determine the following input parameters:-

# minimum length of a word to be included into the dictionary.

# Exclude any word if its length is less than \*minWordLen\* characters.

minWordLen <- 3

# maximum length of a word to be included into the dictionary.

# Exclude any word if its length is greater than \*maxWordLen\* characters.

maxWordLen <- 25

# minimum document frequency of a word to be included into the dictionary.

# Exclude any word if it appears in less than \*minDF\* documents.

minDF <- 9

# maximum document frequency of a word to be included into the dictionary.

# Exclude any word if it appears in greater than \*maxDF\* documents.

maxDF <- Inf

##################################################

# we assume that the text is the second column in the input data frame

text\_column <- dataset[[2]]

# Contents of optional Zip port are in ./src/

source("src/text.preprocessing.R");

# the output dictionary includes each word, its DF and its IDF

input.voc <- create.vocabulary(text\_column, minWordLen,

maxWordLen, minDF, maxDF)

# the output dictionary includes each word, its DF and its IDF

data.set <- calculate.IDF (input.voc, minDF, maxDF)

# Select the dictionary to be sent to the output Dataset port

maml.mapOutputPort("data.set")

2. Search second **Execute R Script module** and drag it to the canvas. Connect the first output of the last **Execute R Script module** to the second input of the new **Execute R Script module.** Connect thefirst output of the very first **Execute R Script module** to the first input of the new **Execute R Script module.** Connect the output of the text.preprocessing.zip dataset to the last input of the new **Execute R Script module**. Copy and paste the code below:

In this module, we’ll do TF-IDF Calculation, make sure to specify the same values for minWordLen and maxWordLen.

# Map 1-based optional input ports to variables

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

input.dictionary <- maml.mapInputPort(2) # class: data.frame

##################################################

# Determine the following input parameters:-

# minimum length of a word to be included into the dictionary.

# Exclude any word if its length is less than \*minWordLen\* characters.

minWordLen <- 3

# maximum length of a word to be included into the dictionary.

# Exclude any word if its length is greater than \*maxWordLen\* characters.

maxWordLen <- 25

##################################################

# we assume that the text is the second column in the input data frame

label\_column <- dataset[[1]]

text\_column <- dataset[[2]]

# Contents of optional Zip port are in ./src/

source("src/text.preprocessing.R");

data.set <- calculate.TFIDF(text\_column, input.dictionary,

minWordLen, maxWordLen)

data.set <- cbind(label\_column, data.set)

# Select the document unigrams TF-IDF matrix to be sent to the output Dataset port

maml.mapOutputPort("data.set")

3.Search for **Filter based Feature Selection module** and drag it to the canvas.  specify the feature scoring method and the number of desired features. In the sample experiment, we selected the top 1,000 most relevant features. You may increase the number of desired features to get better classification performance. Connect the first output of the last **Execute R Script module** to the input of **Filter based Feature Selection module.** Launch Column selector and select label\_column.

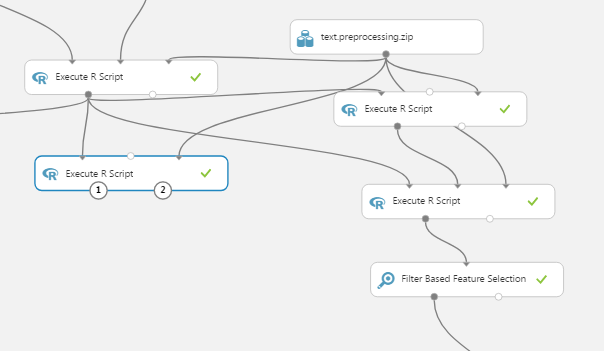
**Select:**

**Feature Scoring Method : Chi Squared**

**Operate on Feature columns only : Selected**

**Number of desired features : 1000**

4.Run the experiment.



**Step 4: Train and evaluate models:**

Use the first **Split module** to split the data into two subsets. The first subset will be used to train the model and the second subset will be split in the next step into development/validation set and test set. In the sample experiment, we split the data into 70% and 30% respectively.

1.Search for **Split module** and drag it to the canvas. Connect the first output of first **Filter based Feature Selection module** to the input of **Split module.** Launch Column selector and label\_column. Select:

Splitting mode : Split Rows

Fraction of the rows in first output dataset : 0.7

Randomized split : Selected

Random seed : 0

Stratified split : True

Use the second **Split** module to split the data into two subset. The first subset will be used later by the **Sweep Parameters** module. The second subset is used as test set to evaluate the performance of the trained model. In the sample experiment, we split the 30% data sample into two halves. That is, each of the development set and the test set represents 15% of the input data.

2. Search for **Split module** and drag it to the canvas. Connect the second output of the first **Split module** to the input of second **Split module.** Launch Column selector and label\_column. Select:

Splitting mode : Split Rows

Fraction of the rows in first output dataset : 0.5

Randomized split : Selected

Random seed : 0

Stratified split : True

3.Search for **Tune Model Hyperparameters module** to get the optimal values for the underlying learning algorithm parameters. Connect the first output of the first **Split module** to the second input of **Tune Model Hyperparameters module**. Connect the first output of second **Split module** to the last input of **Tune Model Hyperparameters module.** Launch Column selector and label\_column. Select:

**Specify parameter sweeping mode: Random Sweep**

**Maximum number of runs on sweep parameters: 5**

**Random seed: 0**

**Metric for measuring performance for classification: AUC**

**Metric for measuring performance for regression: Mean Absolute Error**

4.Search for **Two-Class Logistic Regression** **module** for binary-class classification tasks. Connect the output of the **Two-Class Logistic Regression** **module** to thefirst input of the **Tune Model Hyperparameters module.** Select default settings with:

**Create trainer mode: Parameter Range**

5.Copy and paste the **Split modules, Tune Model Hyperparameters module, Two-Class Logistic Regression** **module** on the other side of the canvas. Connect the first output of the second Filter based Feature selection to the new **Split module** pasted.

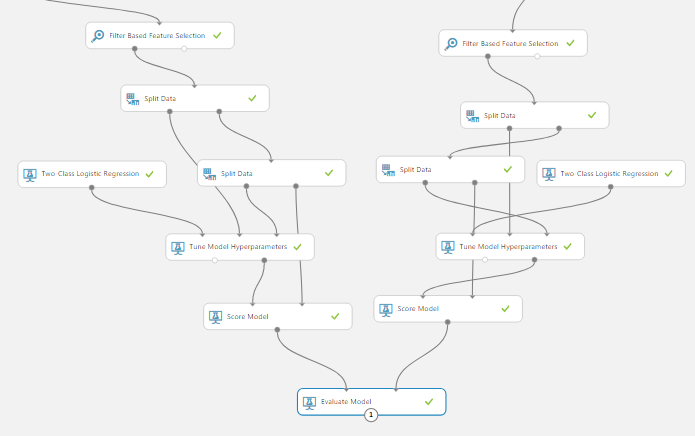
**Step5: Score Models**

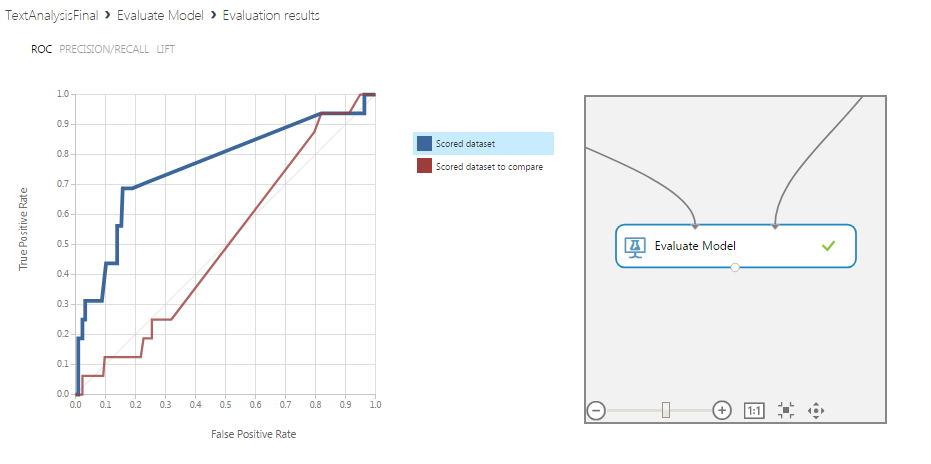
1.Search for **Score model module** and drag it to the canvas. Connect the second output **Tune Model Hyperparameters module** to the first input of the **Score model module.** Connect the second output of the second **Split module** to the second input of the **Score model module** Select Append score column.

2. Search for **Score model module** and drag it to the canvas. Connect the second output **Tune Model Hyperparameters module** to the first input of the **Score model module.** Connect the second output of the second **Split module** to the second input of the **Score model module** Select Append score column.

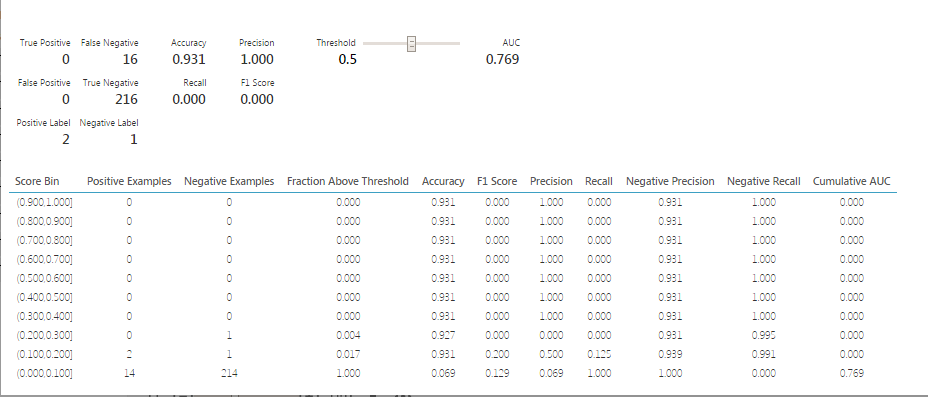
**Step 6: Evaluate Model**

Search for the **Evaluate Model module.** Connect the outputs of the Score models to the input of the **Evaluate Model module.**

****



Click the output port of the **Evaluate Model** module and visualize the results of comparison between the two trained models: N-grams model (in blue color in the graphs below) and unigrams model (in red color in the graphs below)



References

* 1. URLs of Data Source: https://s3.amazonaws.com/hipicdatasets/yelp\_raw\_fall\_2016.csv

https://www.yelp.com/dataset\_challenge/dataset

* 1. URL of our Github: https://github.com/rsingh26/DataScience/tree/master/MachineLearning
  2. URL of Refereces :

https://courses.edx.org/asset-v1:Microsoft+DAT203x+1T2016+type@asset+block/DAT203x-

Lab\_5A.pdf