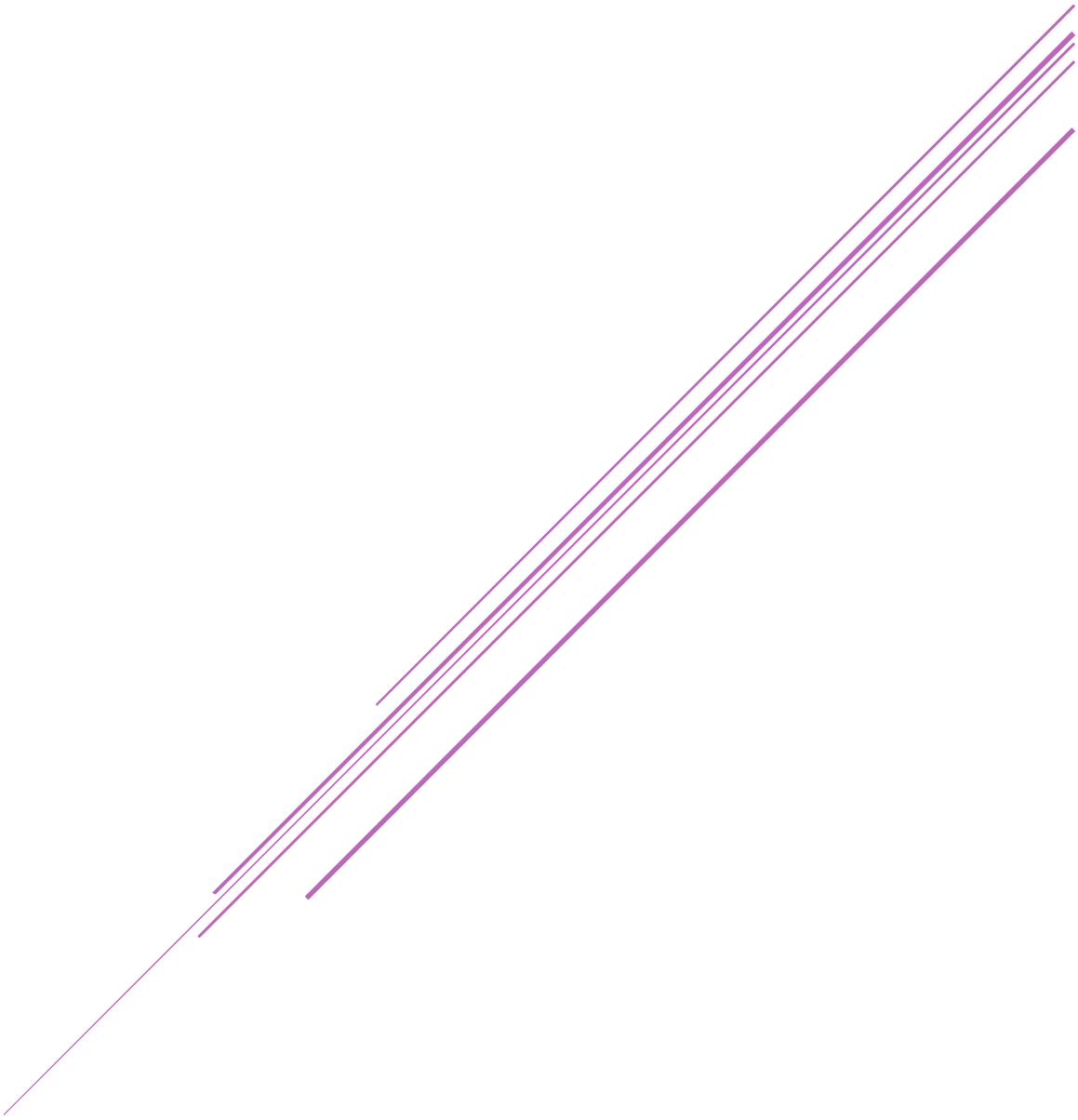
# HOTEL REVIEW ANALYSIS

## Sentimental Analysis Using Azure ML Studio



Huong Pham

## INTRODUCTION

From a business perspective, have you ever wonder what rating a customer will give your business, based on their review? This type of data has brought new opportunity to businesses in order to gain a deeper understanding of the customer experience. I found a dataset from Kaggle, and wanted to perform sentimental analysis on the reviews, using a cloud-based analytics tool – Azure. The dataset contains 515,000 customer reviews and scoring of 1493 luxury hotels across Europe.

The goal of this text-mining project is to generate an algorithm model to predict the Reviewer Score of the review. With this prediction, an e-commerce marketer, travel agencies or a hotel owner can estimate the customer review score. Based on this kind of analysis, they can get a better understanding about the demographics and preferences of their customers.

## DATA EXAMINATION

The initial csv file contains 17 fields. The description of each field is as below:

| Column Names | Metadata |
|---|---|
| Hotel_Address | Address of hotel. |
| Review_Date | Date when reviewer posted the corresponding review |
| Average_Score | Average Score of the hotel, calculated based on the latest comment in the last year |
| Hotel_Name | Name of Hotel |
| Reviewer_Nationality | Nationality of Reviewer |
| Negative_Review | Negative Review the reviewer gave to the hotel. If the reviewer does not give the negative review, then it should be: 'No Negative' |
| Review_Total_Negative_Word_Counts | Total number of words in the negative review |
| Positive_Review | Positive Review the reviewer gave to the hotel. If the reviewer does not give the negative review, then it should be: 'No Positive' |
| Review_Total_Positive_Word_Counts | Total number of words in the positive review |
| Reviewer_Score | Score the reviewer has given to the hotel, based on his/her experience |
| Total_Number_of_Reviews_Reviewer_Has_Given | Number of Reviews the reviewers has given in the past |
| Total_Number_of_Reviews | Total number of valid reviews the hotel has |
| Tags | Tags reviewer gave the hotel |
| days_since_review | Duration between the review date and scrape date |
| Additional_Number_of_Scoring | There are also some guests who just made a scoring on the service rather than a review. This number indicates how many valid scores without review in there |
| lat | Latitude of the hotel |
| lng | longtitude of the hotel |

I determined that I only needed the following features for the analysis, including Average_Score, Reviewer_Score, Total_Number_of_Reviews_Reviewer_Has_Given, Total_Number_of_Reviews, Additional_Number_of_Scoring.

## DATA PREPARATION

Based on the visualization (Fig.1), there are a lot more reviews with high rating of 9 and 10 than other ratings.

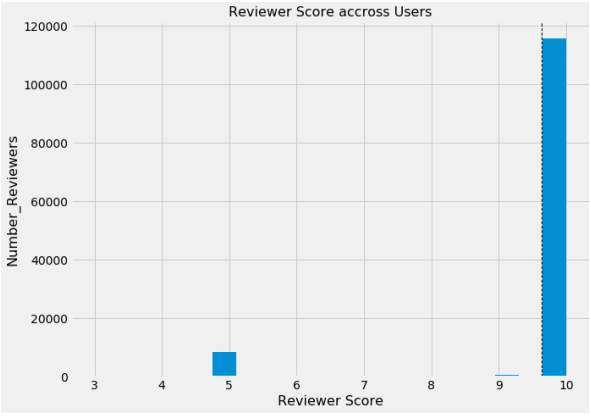



Fig.1: Reviewer Score across Users

In Excel, I manually converted the decimal values of ratings to integers. For example, 2.5 and 2.9 are converted to 3. After prepping data in Excel, I had all round numbers of ratings for my dataset. In addition, I decided to keep the analysis simple by predicting only bad reviews versus good reviews. As a result, I discarded all neutral reviews with ratings of 6,7,8 (the lowest rating is 3).

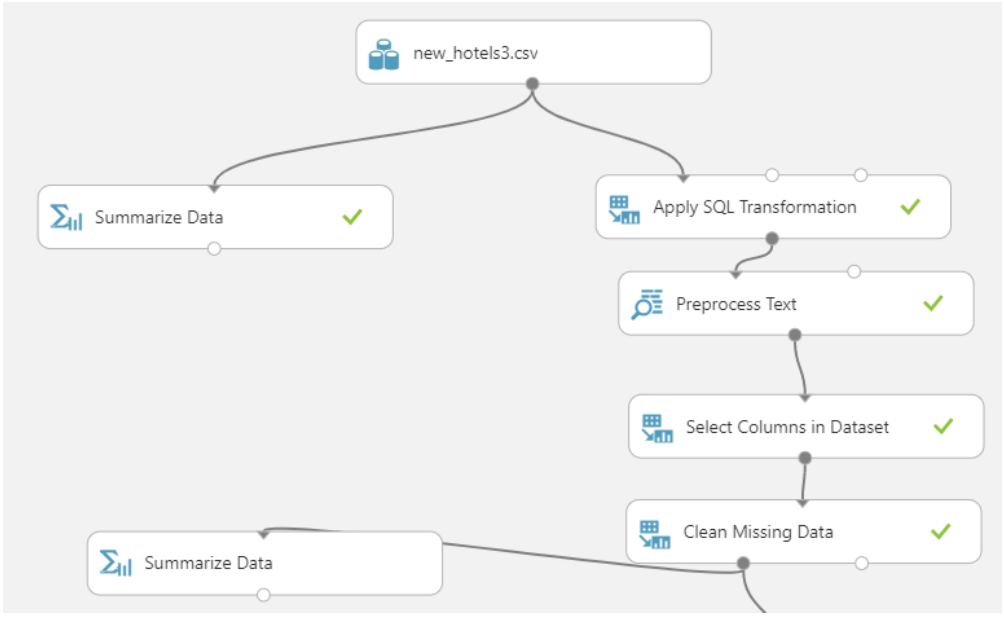



*Fig.2: Data Prep Workflow in Azure*

**Step 1**: Upload my data to the Azure ML Studio. Then summarized the data to see what my features looked like in Azure.


*Fig.3: Upload data to Azure*

**Step 2:** I applied SQL transformation for ratings. I coded the ratings of 3, 4 and 5 as "1". Ratings of 9 and 10 to be "2". This transformation provided a new rating system. Rating of 1 indicates that the reviews are bad, 2 means good reviews. The goal of the project is to create an algorithm that can predict the ratings of the reviews: 1 – bad reviews, and 2 – good reviews.

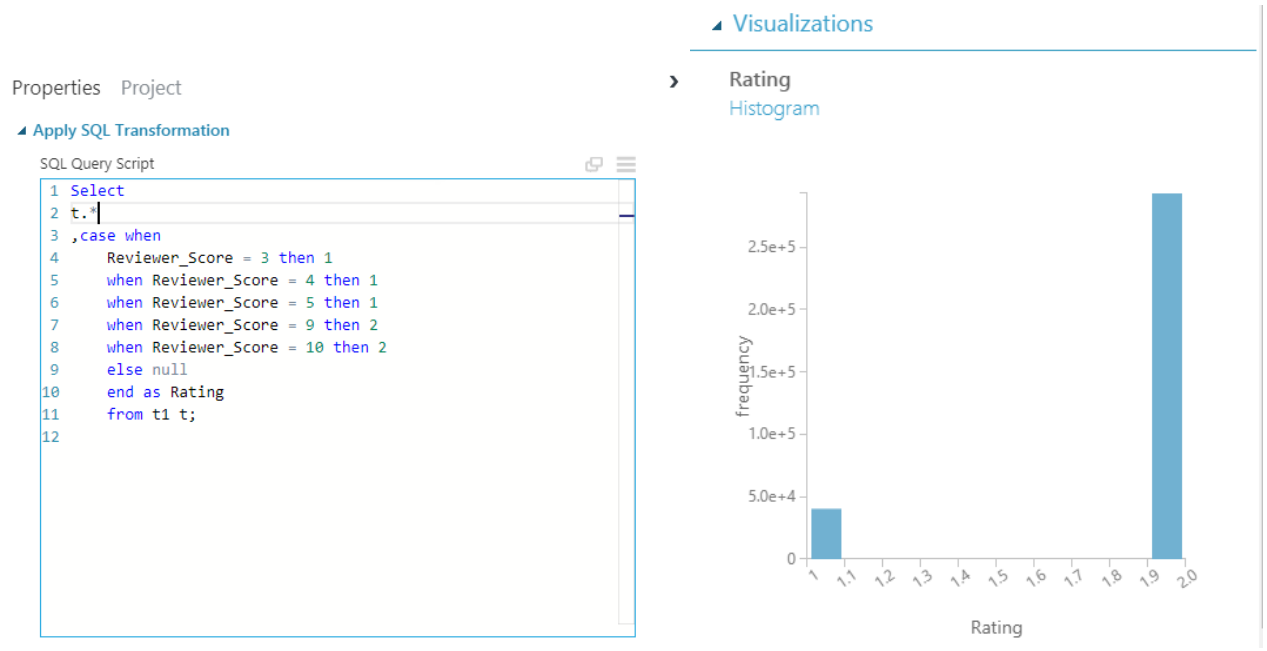



*Fig.4: SQL Transformation in Azure*

**Step 3**: I connected my data flow to the Preprocess Text module. I really enjoyed using Azure's Preprocessed Text feature. This feature took care of all the preprocess steps. I selected my Review text attribute to be preprocessed. You can see how easily I could remove stop words, make all letters lowercase, remove special characters, and much more. It was easier using this module in Azure than inputting all the code in Python. It also created a new attribute called "Preprocessed Review."

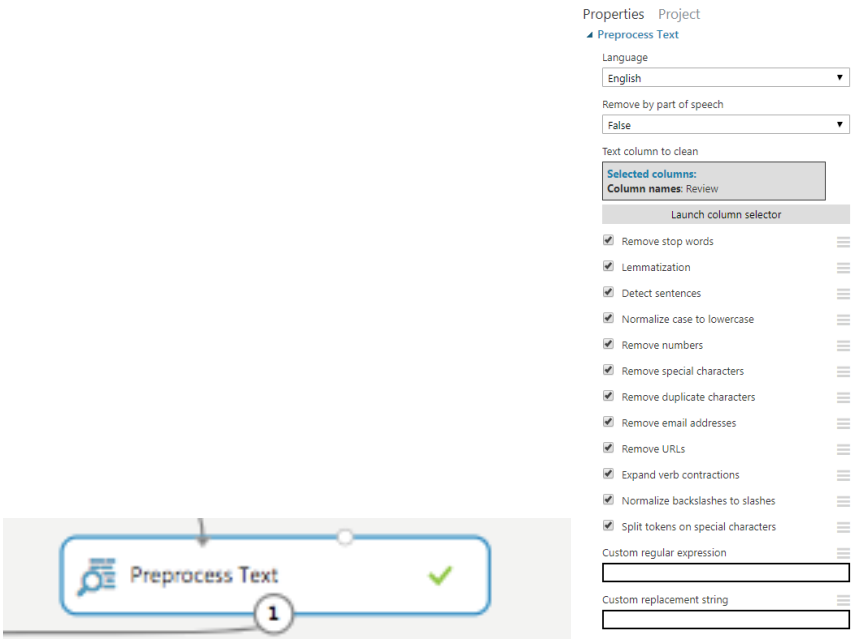



*Fig.5: Preprocess Text Settings in Azure*

**Step 4**: I connected the Preprocessed Review to a Select Column in Dataset module. This module allowed me to select which attributes I wanted to perform my analysis on.


The selected columns are:

- Average_Score: Average Score of the hotel, calculated based on the latest comment in the last year.
- Total_Number_of_Reviews_ReviewIr_Has_Given: Number of Reviews the reviews has given in the past.
- Total_Number_of_Reviews: Total number of valid reviews the hotel has.
- Additional_Number_of_Scoring: There are also some guests who just made a scoring on the service rather than a review. This number indicates how many valid scores without review in there.
- Preprocessed Review – New attribute
- Rating

**Step 5**: I also included a Clean Missing Data module. The module ensured that any missing data would be removed, so it would not become an error in the text processing.

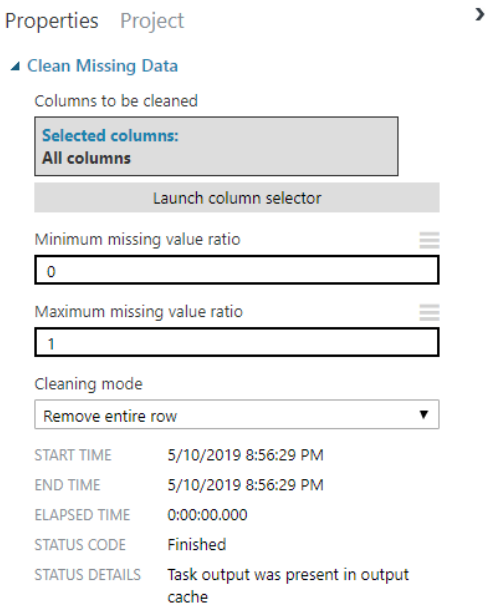



*Fig.6: Clean Mising Data Module in Azure*

After prepping the data, I created two data flows in Azure. The first one is to use the Feature Hashing module. The second flow is to use the Extract N-grams Features module.

## FIRST DATA FLOW

I connected the cleaned data flow to the Feature Hashing module. Feature Hashing works by converting unique tokens into integers. It operates on the exact string that I provide as an input and does not perform any linguistic analysis or preprocessing on its own. I wanted to use my review text to build a model, but I needed to select my Preprocessed Review to ensure that the model was using the text that had been preprocessed. Therefore, I set my selected column to be: Preprocessed Review. The Feature Hashing module creates a dictionary of n-grams. I can control the size of the n-grams by using the n-grams property. I experimented several of parameters for this module

After the dictionary has been built, the Feature Hashing module converts the dictionary terms into hash values and computes whether a feature was used in each case. For each row of text data, the module outputs a set of columns, one column for each hashed feature. If the value in the column is 0, the row did not contain the hashed feature, and if the value is 1, the row did not contain the feature. In my data the first row one does not include the first or second features, but it does have the third and fourth features.

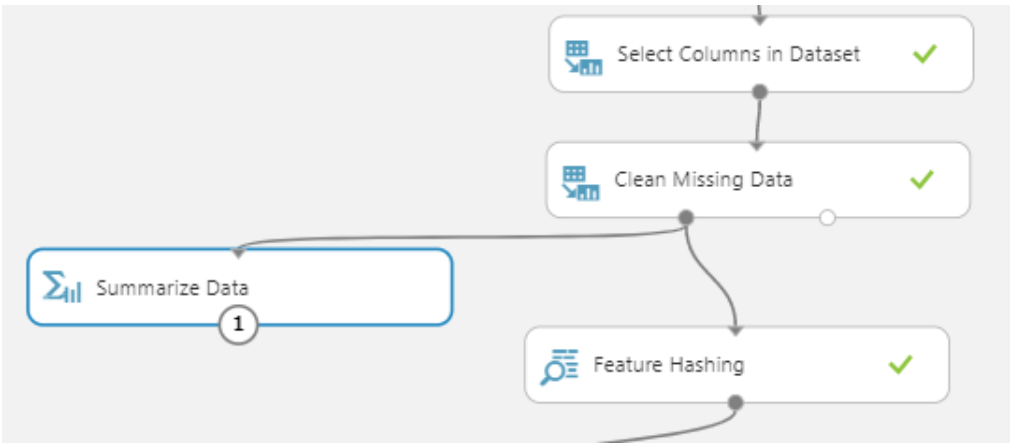



*Fig.7: Connect the Cleaned Data to Feature Hashing Module in Azure*

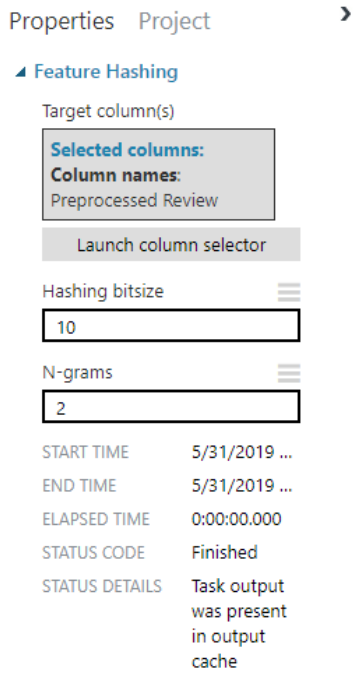



*Fig.8: Feature Hashing Module in Azure*

After this module, I split data .7 and .3. Then I applied different models to train the data, including:

- Two-Class Logistic Regression
- Two-Class Decision Forest
- Two-Class Bayes Point Machine
- Two – Class Support Vector Machine

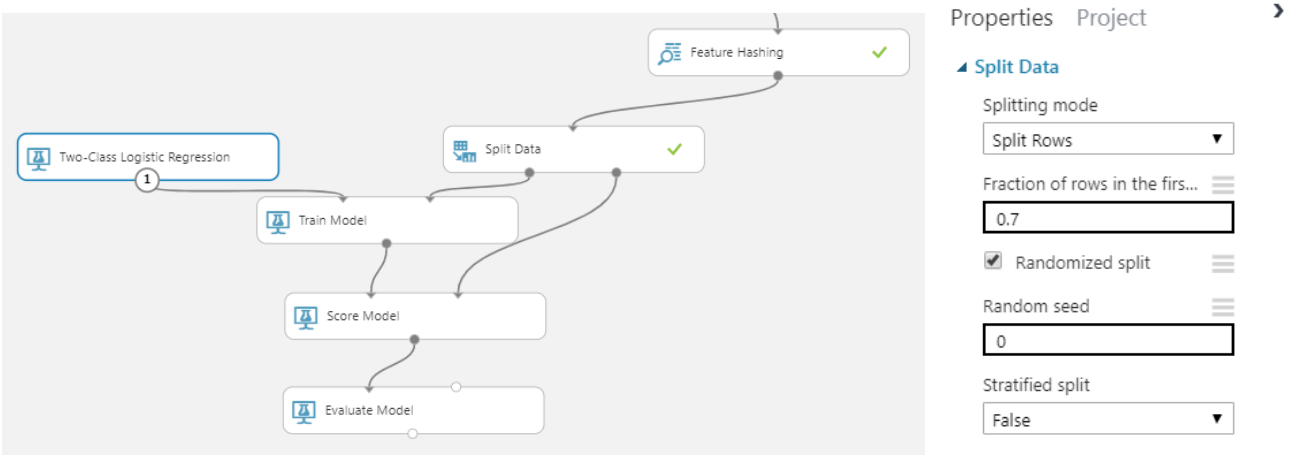



Fig 9: Using Two-Class Logistic Regression to train data

The steps to train all models are the similar. I repeated this workflow for other models.

## SECOND DATA FLOW

I wanted to try one more module in Azure called: Extract N-gram Features from Text. I wanted to try this module because feature hashing was able to extract key phrases, but it was unable to show me what those key phrases were. This new module: Extract N-gram Features from Text was able to label what the key phrases are.

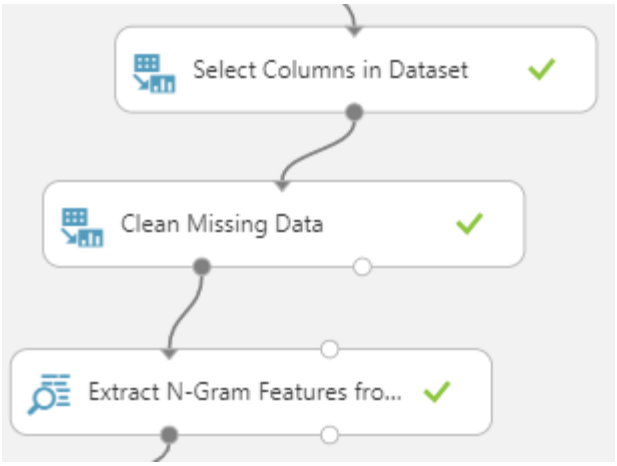



Fig 10: Connect the Cleaned Data to Extract N-Gram Features from Text

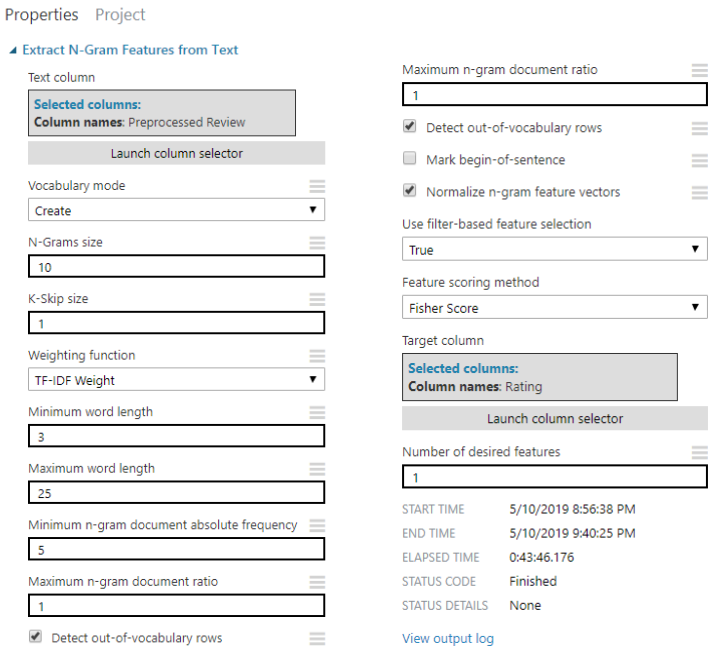



Fig 11: Parameters for Extract N-Gram Features from Text

The process of training models is similar to the first data flow. I will only show the visualizations for the model evaluations.

The Multiclass Logistic Regression model is overfitting.

a. Multiclass Decision Forest

## DATA ANALYSIS

| | First Data Flow – Feature Hashing | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Two-class Logistic Regression** | 0.929 | 0.939 | 0.982 | 0.960 |
| **Two-Class Decision Forest** | 0.879 | 0.879 | 1.00 | 0.936 |
| **Two-class Bayes Point Machine** | 0.889 | 0.889 | 0.998 | 0.941 |
| **Two – Class SVM** | 0.922 | 0.935 | 0.979 | 0.957 |

*Fig 12: Results from First Data Flow*

| | Second Data Flow –Extract N-gram Features | | | |
|---|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Two-class Logistic Regression** | 0.885 | 0.892 | 0.988 | 0.938 |
| **Two-Class Decision Forest** | 0.879 | 0.879 | 1.00 | 0.936 |
| **Two-class Bayes Point Machine** | 0.879 | 0.880 | 0.999 | 0.936 |
| **Two – Class SVM** | 0.875 | 0.898 | 0.967 | 0.931 |

*Fig.13: Results from Second Data Flow*

Fig 12 and 13 show that two-class logistic regression has the highest scores for both data flows. Both Feature Hashing and Extract N-grams Features have high results. Feature Hashing has slightly higher results by an average of 0.000337%.

One thing I noticed is that Logistic Regression is consistently the best fit model for this data set, regardless Python or Azure.

Azure offers very informative visualizations to evaluate the models, including ROC curve.

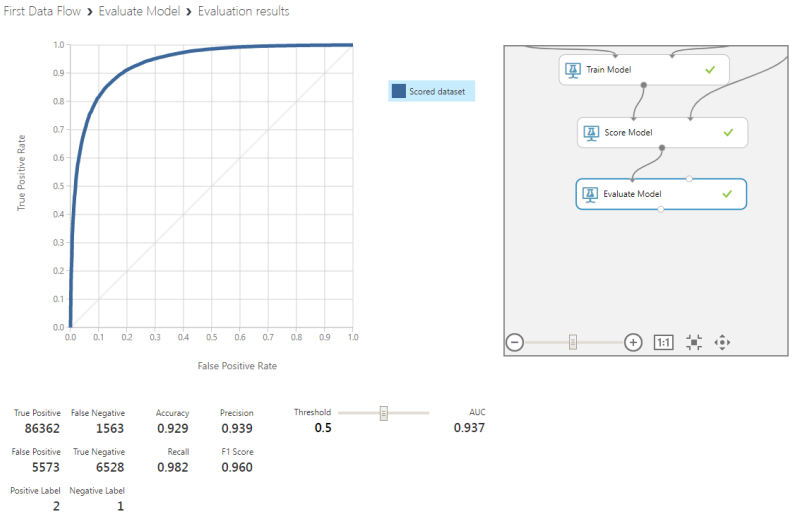



*Fig.14: First Data Flow. Two-Class Logistic Regression*

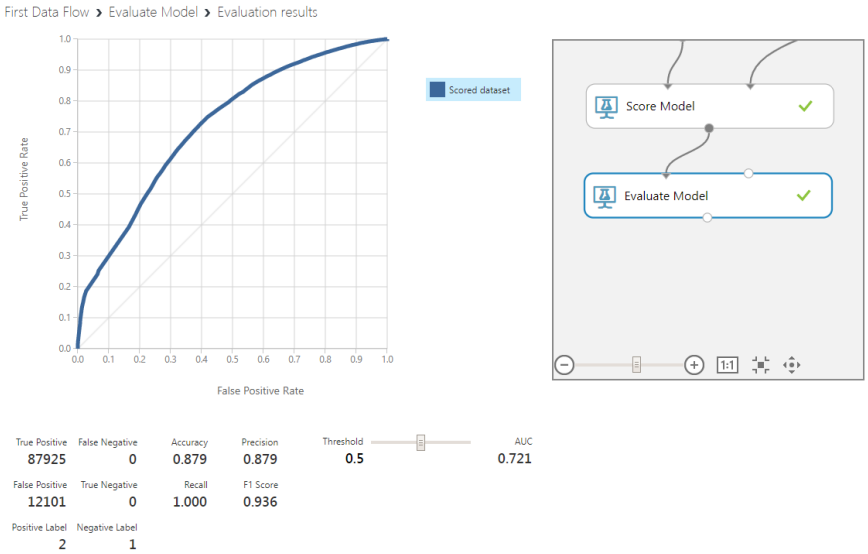



*Fig.15: First Data Flow. Two-Class Decision Forest*

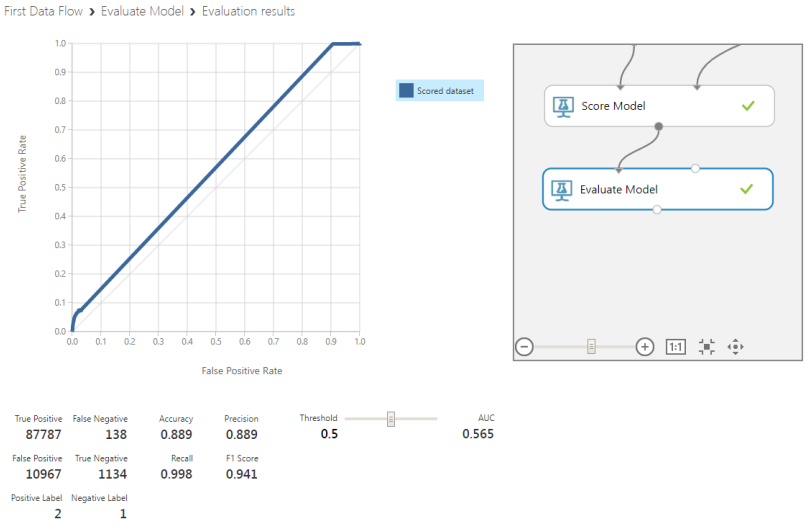



*Fig.16: First Data Flow. Two-Class Bayes Point Machine*

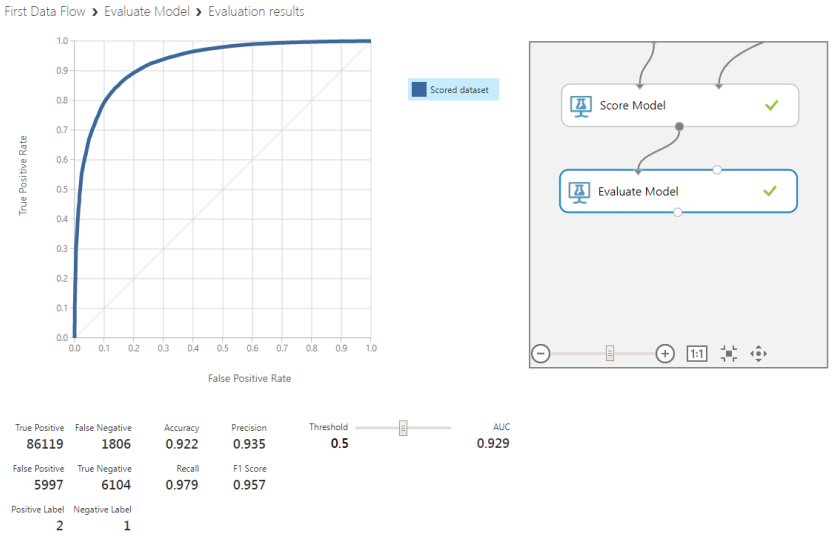



*Fig.17: First Class. Two-class SVM*

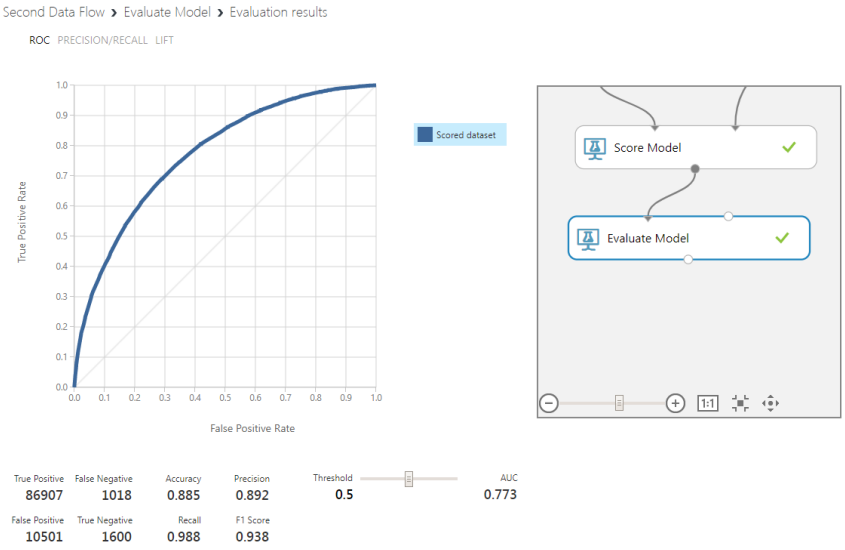



*Fig.18: Second Data Flow. Two-class Logistic Regression*

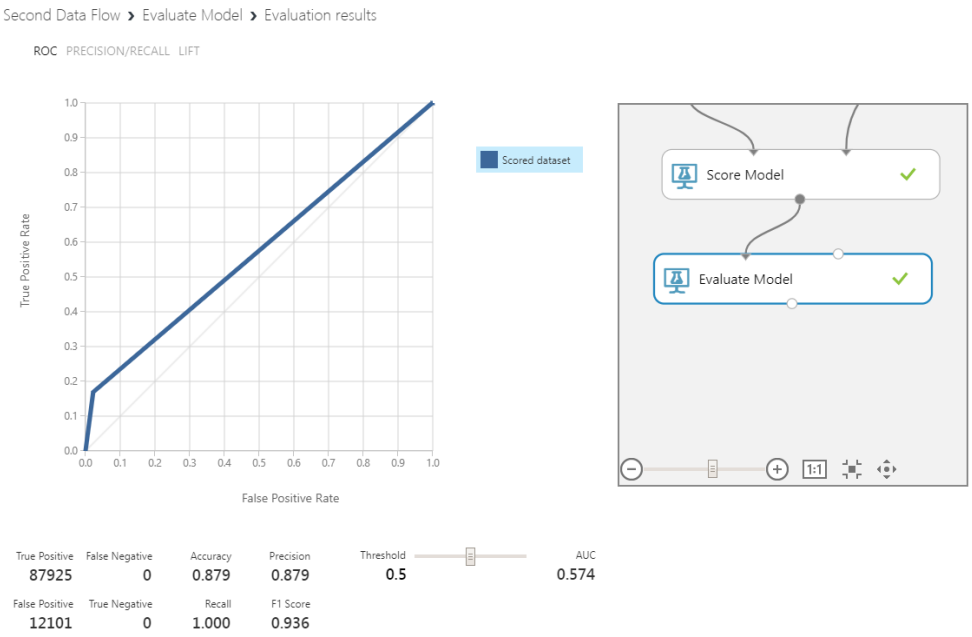



*Fig.19: Second Data Flow. Two-class Decision Forest*

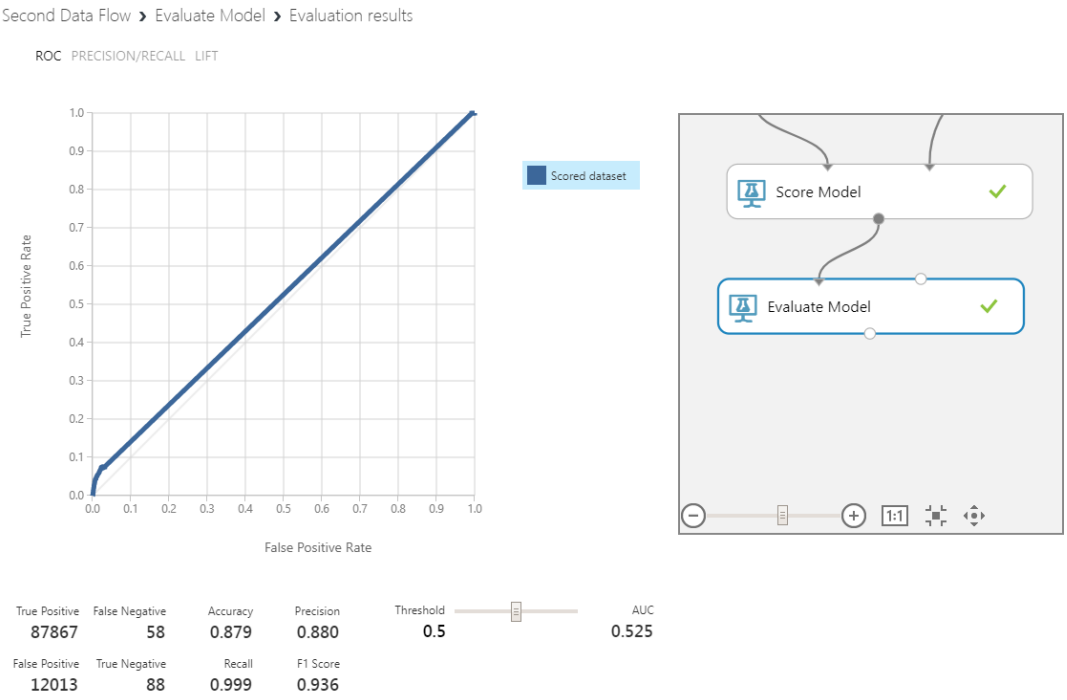



*Fig.20: Second Data Flow. Two-class Bayes Point Machine*

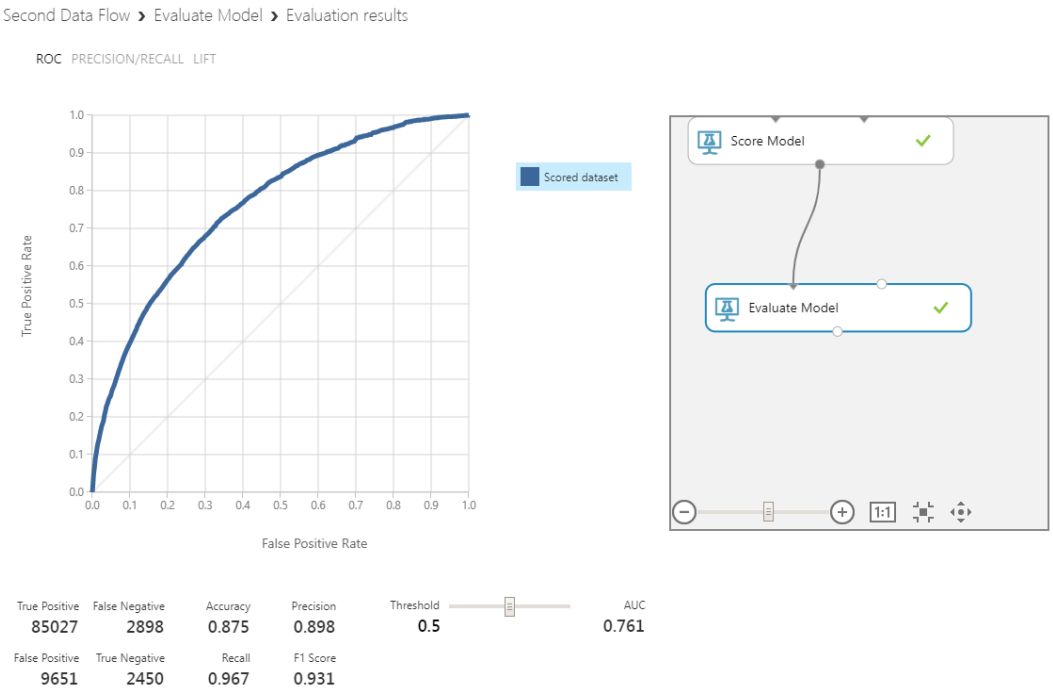



*Fig.21: Second Data Flow. Two-class SVM*

## FINAL THOUGHTS

### WHAT QUESTION WAS ANSWERED?

The question that was answered was that I could predict with about 93% accuracy which reviews have a rating of 1 or 2. To match with the original format of the dataset, this means.

- 1 = Bad ratings of 3,4,5
- 2 = Good ratings of 9,10

### WHAT SEEMED TO WORK?

What worked well was implementing thorough pre-processing techniques on the data. The Preprocessing Text module in Azure was extremely helpful and simple, compared to writing Python scripts for the same task.

### WHAT TO EXPLORE IN FUTURE?

- Azure also has modules to support R-script and Python-script. I spent some time on experimenting Execute Python Script module in Azure. However, I kept getting errors since the script needed to follow a certain format in Azure. If I had more time with this project, I would definitely explore the Execute R Script in Azure. This will help me learn a new language. Overall, Azure' advantage is that the built-in modules are very user friendly. However, I think coding languages like Python and R provide more helpful insights when it comes to text analysis. Although it is harder to write the scripts from scratch, I think it is easier to track the execution process and analyze what worked and not worked in the mining process in Python than Azure.
- I would like to try the algorithms on my own set of data, or maybe on a different dataset to see if they work.