

Amadation - The Amazon Reviews Expert Status Report 2

Team : C

Date : 11 / 8 / 2020

Team Members:

Jaineel Vyas, Mahlika George, Pranita Menavlikar, Sylvia DeMarree, Bhavin Jethra

Summarized Journey:

Attempts:

- **Identify Problems:**
 - Skewness of data
 - Text analysis module a good option?
 - Is Neural Network Implementation proper?
- **What We Tried:**
 - Preprocessing the Input Text
 - Turning on “parts of speech” filter
 - Remove nouns
 - Remove adjectives
 - Remove verbs
 - Remove all combinations of nouns, adjectives and verbs
 - Not to remove duplicates
 - Remove Numbers
 - Undersampling
 - Partition and Sample Module
 - Oversampling
 - SMOTE with Binary data for class values less than 1 and equal to 1
 - SMOTE with Binary data for each class with respect to class value reviewRating 1
- **Observations:**
 - Basic preprocessing was done to ensure only the necessary and filtered words of the sentence go to the LDA model. It would be more useful for determining the sentiment of the text.
 - Model only ever predicts class 1.
 - LDA was not working properly, the LDA is classifying the text into ‘N’ number of most prominent topics. ‘N’ was set to 5, as we had 5 classes to classify into, but the LDA seemed to attach every word to the same topic.
 - With data resampling the stratified split option could only reduce to the minimum number of examples for a particular class.
 - SMOTE only works with Binary data.
 - Decision trees with boosting/bagging are often known to perform better in some classification scenarios, but did not help in this case.

Working Model:

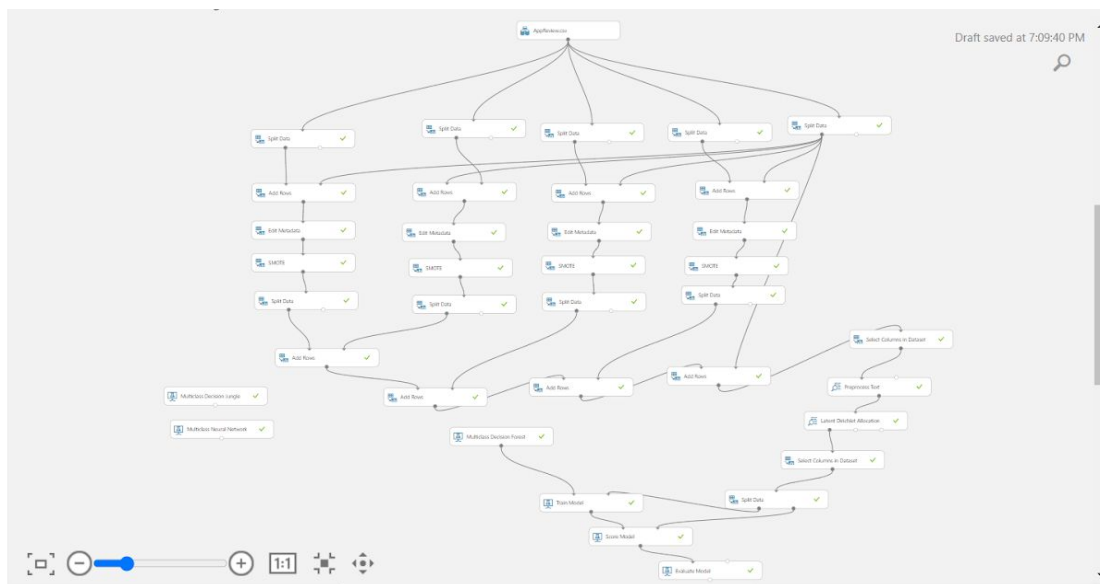
- **Access the Model:**

<https://gallery.cortanaintelligence.com/Experiment/DSCI644-TermProject-ForSubmission>

- **Model Accuracy:**

Overall accuracy	0.884049
Average accuracy	0.95362
Micro-averaged precision	0.884049
Macro-averaged precision	0.887102
Micro-averaged recall	0.884049
Macro-averaged recall	0.884453

- **Model Structure:**



- **How We Got There:**

- Fine-tuned the model by learning more about how LDA works.
- LDA module functionality was explored and the other options to train the LDA model were changed and tested.
- Set the N-Gram to 3, as trigrams have more cohesion than bigrams.
- The rho parameter and alpha parameter are the prior probabilities to signify the sparsity of topic distribution and per topic weights. They were kept low to start with.
- Initial value of iteration count was set to 3.
- Subsample the class data using SMOTE to remove the skewness.
- Multiclass decision forest was tested and it gave an accuracy of 88.4% with the parameters [Number of decision trees - 20, Max depth - 50, Random splits - 256, Min samples in a node - 1 and Bagging as resampling method]

Ongoing/Future Work:

- Generalizing of the model
 - Currently, the model has almost equal number of instances for each class using the preprocessing techniques. This is not a regular real world scenario where you'll have equal instances for all classes in your data. That might also be the reason for getting a higher accuracy. We ought to try another sampling technique where we intend to generalise this scenario.
- Fine Tuning of the model parameters
- Deploying the model on a webpage for user friendly access to the Review Recommender System.

In-Depth Journey:

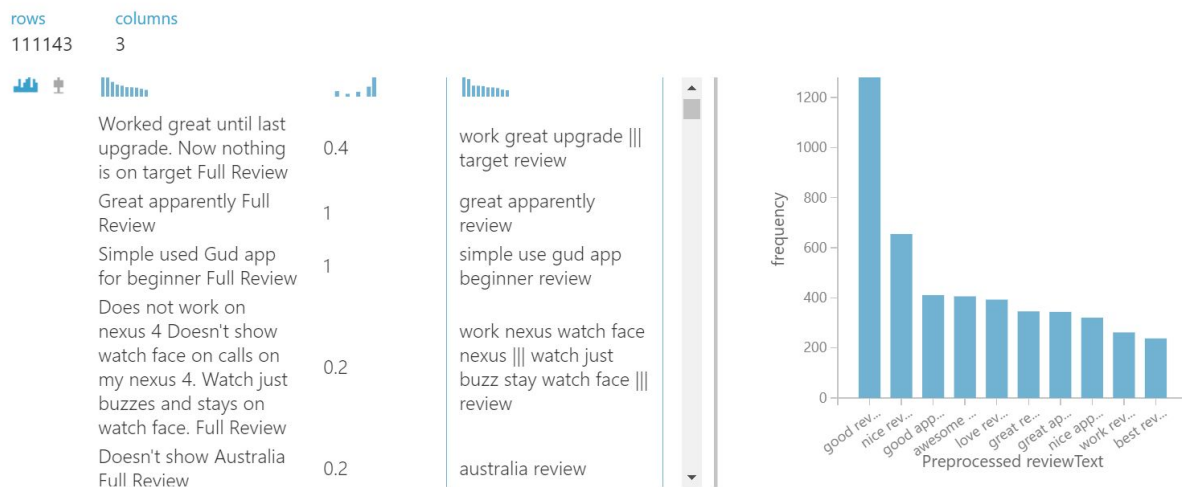
Text Analytics (NLP):

After identifying that the problem could be mainly the skewness of the data, we thought to also investigate whether the text analysis module is proper.

We then played around with preprocessing of the input text by turning on the “parts of speech” filter. Removed nouns, adjectives and verbs in all combinations, yet there wasn't a significant improvement. We turned on and off other options, like removing duplicates, removing numbers. No improvement.









However, the basic preprocessing was done to ensure only the necessary and filtered words of the sentence go to the LDA model, that would be more useful to determine the sentiments involved with the text.

DSCI644:TermProject > Preprocess Text > Results dataset



We found that no matter what, it always predicts class 1.

The LDA classifies the text into 'N' numbers of most prominent topics. 'N' was set to 5, as we had 5 classes to classify into. LDA seemed to attach every word to the same topic as shown in the image below, which Isn't how it should work.

reviewText	reviewerRating	Preprocessed reviewText	Topic1	Topic2	Topic3	Topic4	Topic5
 Won't load new game I finish a game and there's no clear way to move on the the next one. And the screen freezes when looking at the other players tiles. Full Review	 0.4	 load new game finish game 's clear way screen freeze look player tile review	 0.999612	 0.000097	 0.000097	 0.000097	 0.000097
Very good app. Use it all the time Full Review	1	good app. use time review	0.998623	0.000344	0.000344	0.000344	0.000344
Good one. Full Review	1	good review	0.996935	0.000766	0.000766	0.000766	0.000766
Can't figure out how it works, duh! Help! I can't download, where is a button? Full Review	0.2	figure work duh help download button review	0.999345	0.000164	0.000164	0.000164	0.000164
It crashes It crashes after watching a couple of seconds of a video Full Review	0.2	crash crash watch couple second video review	0.999071	0.000232	0.000232	0.000232	0.000232

So we fine-tuned the model by learning more about how LDA works. Then the LDA module functionality was explored and the other options to train the LDA model were changed and tested.

The N-Gram value was set to 3, as trigrams have more cohesion than bigrams. The rho parameter and alpha parameter are the prior probabilities to signify the sparsity of topic distribution and per topic weights. They were kept low to start with. Also, the initial value of iteration count was set to 3. Which both of these should be fine-tuned to check for further improvements.

After these changes, each word was classified into different topics unlike the earlier case. It got randomized showing signs of improvement as shown in the image below.

reviewText	reviewerRating	Preprocessed reviewText	Topic1	Topic2	Topic3	Topic4	Topic5
							
Excellent. This may be a stupidly simple app, but works so fine. Full Review	1	excellent simple	0.252892	0.000019	0.084302	0.66277	0.000019
Good app for indian weather systems Full Review	1	indian	0.522795	0.434351	0.000042	0.000042	0.042771
Akbar Full Review	0.2		0.999334	0.000167	0.000167	0.000167	0.000167
Very nice, works fine and looks good. Motorola defy. Full Review	1	nice	0.508838	0.241833	0.184557	0.064746	0.000026
Good & thanks! As							

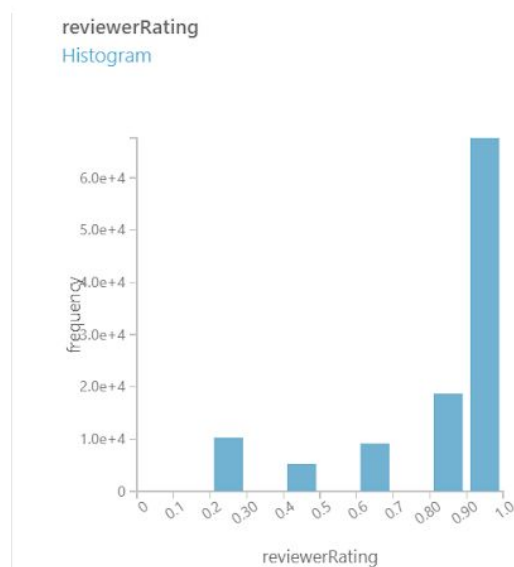
Data Preprocessing:

As additional data provided by the client was not an option, for handling the skewness of our data or for handling the data imbalance, we tried multiple ways for data resampling. Our goal was data for every class nearly equal to each other.

1. Partitioning, Sampling:

The basic "partition and sample" module was explored to identify ways of resampling by using the stratified split option. It could only reduce to the minimum number of examples for a particular class, i.e. perform undersampling. This resulted in the overall low accuracy of the model.

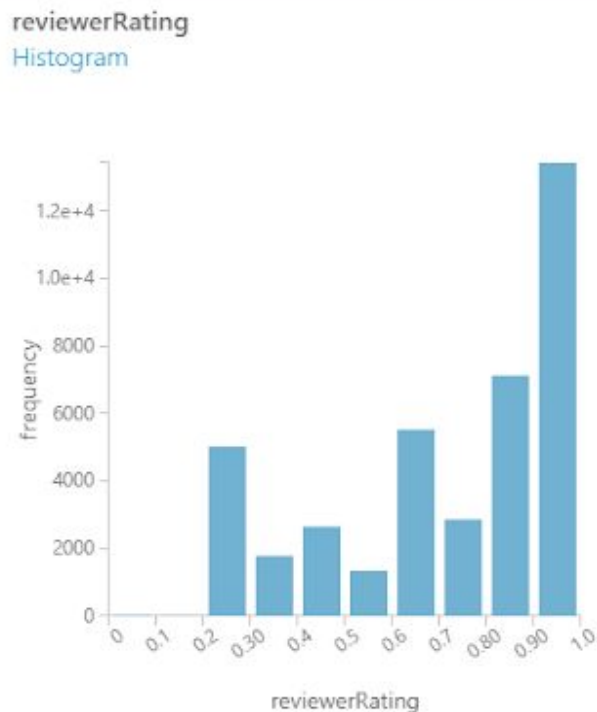
Before SMOTE:



2. Synthetic Minority Over-sampling Technique(SMOTE):

SMOTE, another module of Azure, was used to perform oversampling, that could equalize each class records to match with the class having maximum number of records. However, SMOTE only works with Binary data. Therefore, we created a new column in the data, that would binarize rating below 1 as 0, and rating of 1 as 1. The reason for doing this was the skewness towards instances with class 1. This enabled to boost every other rating which was less than 1 to be increased in the same proportion. It did not perform well, as the data was still not equal as shown in figure.

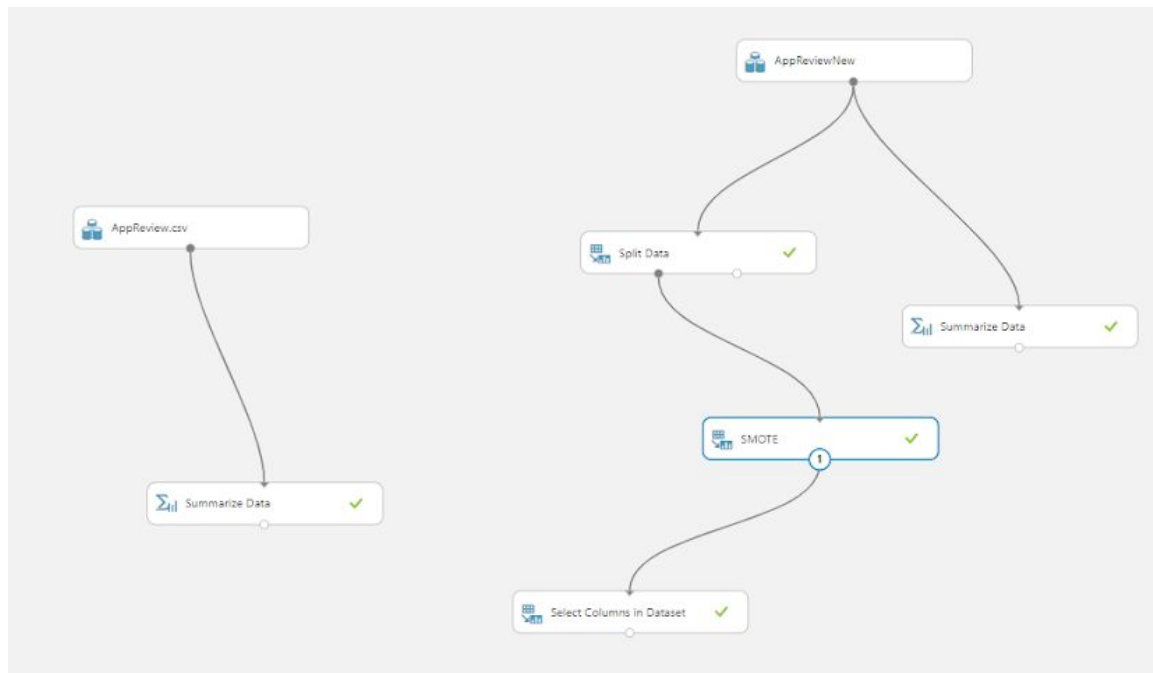
After SMOTE:



Tutorial followed for learning SMOTE:

<https://www.c-sharpcorner.com/article/cleansing-data-using-azure-machine-learning-studio-part-4/>

Model For SMOTE:



As this still didn't improve much accuracy, we handled each class records separately by processing data based on each class. For example, records with 0.2 rating were SMOTE with records of class 1 rating. This equalized the number of records for class 0.2 to the number of records of class 1. Iteratively, for all the class, the records were equalized. Following is the approach followed for tuning the model after this step.

Model Tuning:

The initial model which had multiclass neural network had approx 60% accuracy which was quite hard to improve on with just changing the parameters of the NN. The parameter which we tried changing were :

1. Number of hidden nodes : 20, 50, 150, 500, 1000 - accuracy was still in range (60,61)
2. Learning rate : 0.01, 0.1, 0.001, 0.05 - accuracy was still in range (60,61)
3. Learning iterations : 500, 1000, 3000 - accuracy improved by 1% and reached 62.3% with 3000 iterations
4. Normalizers : Binning, Gaussian and Min-max - best accuracy was with min-max but still 61%

After going through some research papers on test classification and blogs on review classification we thought of trying out decision trees with boosting/bagging as they are often known to perform better in some classification scenarios.

1. Multiclass decision jungle : Which makes an ensemble of DAGs
 - a. Initial setting : Number of DAGs - 8, Max depth - 32, Max width - 128 and number of optimizing steps - 2048 gives an accuracy of 24%.

Overall accuracy	0.243377
Average accuracy	0.697351
Micro-averaged precision	0.243377
Macro-averaged precision	NaN
Micro-averaged recall	0.243377
Macro-averaged recall	0.249472

- b. Tuning tried : Number of DAGs - 8, 20, 50, 100, Max depth - 10, 20, 50, 100, Max width - 50, 128, 512 and number of optimizing steps - 500, 1000, 4000, 100 with accuracy still in range(24,27) [Best 27.4% - 20,100,512,1000].
 2. Multiclass decision forest : Which makes an ensemble of decision trees
 - a. Initial setting : Number of decision trees - 8, Max depth - 8, Random splits - 128, Min samples in a node - 1 gave accuracy was 26%].
 - b. Tuning tried : Number of decision trees - 8, 50, 100, 150, Max depth - 8, 5, 10, 20, Random splits - 128, 32, 256, 512, Min samples in a node - 1, 5, 10, 20, 100 gave accuracy in range(25, 28) [Best 28.2% - 50, 20, 32, 10].

All these model testing was done before the subsampling of class data using SMOTE to remove the skewness. After removing the skewness the multiclass decision forest was tested and it gave an accuracy of 88.4% with the parameters [Number of decision trees - 20, Max depth - 50, Random splits - 256, Min samples in a node - 1 and Bagging as resampling method]

Our next course of action would be to optimize the model because the current model takes 38+ minutes to execute. We will try removing other columns which are not used before all the split and sampling steps to reduce some execution time. Also since right now the models parameters are a bit high and takes up more computational time, so we are going to find less valued combination of parameters with almost the same accuracy i.e as stated by Occam's razor if less number of attributes / simpler model can give almost same accuracy as more complex model then we use the simple one. So if possible then we will try to improve the accuracy more than 88.4% but if that is the maximum accuracy we can achieve then we will try to simplify the model to give at least 88% accuracy.

Number of rows after resample for each class

