# Naive Bayes Implementation - COMPSCI 762

## 1 Motivation behind Solutions

The motivation behind the presented solution was establishing an improved model pipeline that iteratively improved performance and robustness. The first step taken was developing a baseline model that had a predictive accuracy similar to the baseline model accuracy on Kaggle. This allowed me to gain an understanding of how the classifier functions and what was required to improve its accuracy. This provided me with a sense of direction in terms of model improvement approaches to improve performance.

## 2 Preprocessing Techniques and Data Representation

The review and name data are both document data and require preprocessing for noise reduction, and to restructure the data into a more meaningful format. The data must also be modified to be more functional as an input into the classifier (Data Representation). The following steps were taken to achieve this:

- The removal of punctuation from the review and name attributes. This normalises the structure of each document as punctuation marks convey meaning in written communication, but when considering them as features can be considerably noisy. I also ensured all the words contained lower case characters to reduce variability in that manner.
- The removal of stopwords or common words from the review and name attributes. This is a common procedure in text analysis, and in natural language they provide meaning. But when considering them as individual features they hold little semantic value, add significant computational complexity to the model (additional dimensionality), and are considerably noisy.
- Stemming words involves reducing words to their root form. This helps handle misspelt instances of words, and reduces dimensionality by reducing variations of the same word. This was done after the data was partially preprocessed from the methods above.
- Using Bag of Words (BoW) to represent the text document data as numerical input features. This involves creating a vocabulary containing the unique word occurrences. Each feature column is an individual word and each row corresponds to the frequency of the word within the given review or name attribute data.

All the stated techniques aim to improve model accuracy and functionality by more meaningfully representing both the data, and the quality of the data.

## 3 Implementation Overview

The solution to the first assignment task consists of improving a given benchmark model by only utilising the review attribute, and this involves:

1. Preprocessing the review string data through common texting cleaning and normalisation techniques.
2. Using Bag of Words (NLP technique) to represent the text data as numerical input features for the Naive Bayes model.
3. Extending the Naive Bayes model by implementing feature selection (Extension to the Naive Bayes classifier). This involved reducing the number of words accepted into the vocabulary list of

the Bag of Words feature. This removes words with significantly low occurrences across all examples, essentially noise reduction.

4. Evaluate cross validation accuracy as an unbiased estimator of the test accuracy
5. Tune alpha hyperparameter (Laplace smoothing constant value) and train the model with the optimal parameter values.
6. Evaluate performance on unseen data through Kaggle submission, and the expectation is that it outperforms the benchmark model.

The solution to the second assignment task consists of building onto the task 1 model by implementing additional features with the intention of improving accuracy:

1. Re-engineer the name attribute using preprocessing techniques and Bag of Words (word count frequency with a unique word vocabulary for each instance).
2. Re-engineer the review attribute using word count frequency vectors representing bigrams (pair of consecutive words).
3. Use feature selection to reduce dimensionality. This was done within the code by setting an appropriate value for min_df (0.005), and this removed words from the vocabulary list with a document frequency below the threshold min_df value. This is because words that occur very rarely across reviews/names may not carry much useful information for distinguishing between classes and may actually add noise to the model.
4. Evaluate cross validation accuracy as an unbiased estimator of the test accuracy
5. Tune alpha hyperparameter (Laplace smoothing constant value) and train the model with the optimal parameter values.
6. Evaluate performance on unseen data through Kaggle submission, and the expectation is that it outperforms the task 1 model.

## 4 Evaluation Procedure

The evaluation procedure involved using k-fold cross validation (where k = 5) to both tune the alpha hyperparameter (Laplace smoothing constant value), and evaluate the following:

1. An unbiased estimate of the test accuracy by computing the average of the validation accuracy measures across 5 different models.
2. The best model evaluated from 5 iterations (best validation score). This reduces the effect of the variability or bias caused by partitioning the data.

The multinomial Naive Bayes model is then retrained using the optimal alpha hyper-parameter value.

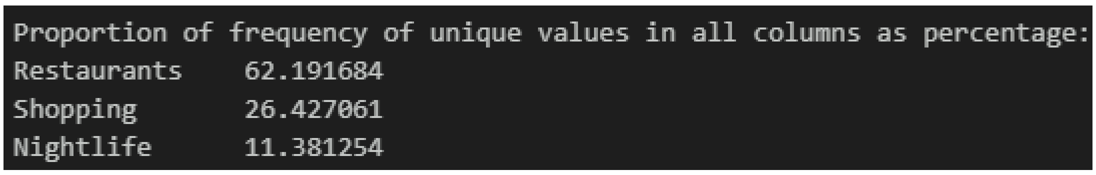
## 5 Performance

After establishing the best model for a given trial of the machine learning pipeline, I would document the cross-validation score and the specifications of the model to determine if changes made positively or negatively affected the predictive ability of the model. The most significant thing to note was the removal of the bigrams feature as it reduced performance heavily (only predicting majority class), and this was to a degree expected as the frequency of bigrams were low across the training examples. So the final model consisted of BoW features re-engineered from the name and review attributes, and resulted in a cross-validation score of 0.8865 and Kaggle test accuracy of 0.92201. This was an improvement relative to both my task 1 model (CV Score: 0.87843, Kaggle Test Score: 0.90366), and the benchmark model (Kaggle Test Score: 0.89449).

## 6 Future Implementations

Potentially in the future to improve performance further I can test the implementation of the following:

- Addressing the class imbalance within the training set by using resampling methods, either oversampling the minority class or undersampling the majority class. Keeping in mind the potential concern of overfitting as a result of this.



```
Proportion of frequency of unique values in all columns as percentage:
Restaurants    62.191684
Shopping       26.427061
Nightlife      11.381254
```

**Figure 1: Class Distributions**