Faith Kilonzi
Natural Language Processing: Lab 4- Report
Logistic Regression and Naïve Bayes Classifiers 11th November 2018

In this lab 4, I explored the use of naïve Bayes classifier and logistic regression to predict sentiment analysis and their implementation. I made use of different python libraries for text preprocessing, normalization, algorithm implementation, testing and evaluation.

Among the imported libraries included the following:

```
In [1]: ## Importing Libraries

from sklearn.linear_model import LogisticRegression
from sklearn import naive_bayes
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
import nltk
import numpy as np
import pandas as pd
import re
from io import StringIO

import sys
```

Across the two classifiers, *sklearn* library was used. This is because it has a clean API, is free and open source, easy to use and has a comprehensive documentation and therefore technical support is assured. *sklearn* is also very robust and compatible when implementing machine learning algorithms such as classification and regression in python.

The logistic Regression model was built on the *sklearn.linear_model* which uses a linear multinomial regression format for analysis. This is because the sentiment labels were dichotomous in nature and hence the need to use logistic regression's cross-entropy loss function.

The Naïve Bayes model was built on the *sklearn* Naïve Bayes *MultinomialNB* classifier. This is because the multinomial classifier is suitable for processing sparse data through word counts and frequency. It also works best especially in text classification using in-built library functions.

Other libraries used in the program include *nltk*, *numpy* and *pandas* for text reading, feature extraction, preprocessing, vectorization and visualization. Python's *io* library was also imported for file processing.

The training dataset was split by a ration of 0.75:0.25 for both training and testing (quarter (25%) of it as a test data). The accuracy was then calculated.

To evaluate the classifiers, I used the *tdifVectorizer* which makes use of the bag of words assumption to represent the given data features into a representational matrix. The matrix is used to describe the performance of classification model on a set of test dataset for which the

true values are known. For prediction, the test dataset uses the *predict_proba* function for both Naïve Bayes and Logistic Regression. The model accuracy on the test dataset is the calculated using '*score*' function which computes and prints the results as shown below.

Results

```
Unnormalized data, NaiveBayes Classifier
Accuracy:  0.8910244548197916

Normalized data, NaiveBayes Classifier
Accuracy:  0.9062250598563447

Unnormalized data, Logistic Regression Classifier
Accuracy:  0.887290184921764

Normalized data, Logistic Regression Classifiers
Accuracy:  0.9049740770505443
```

From the above results, it is evident that with normalization, the accuracy increases, both in the logistic regression and the naïve Bayes classifier. This could be because with normalization, the given data is preprocessed well by the help of different libraries such as *feature vectorizer* and therefore classification function errors are minimized.