## 1 Abstract

This report gives a brief about Classification task in Natural Language Processing. I have also tried fitting Random Forest classifier, K nearest neighbour, SVM and Logistic regression on Dataset to get better accuracy. The Dataset used is a movie review dataset with 2 classes, Positive and Negative.

## 2 Introdutcion

Natural Language Processing is a part of Computer science which deals with interactions between human language and computer. Whenever we work on text data, computer can not understand that directly, meaning of that word won't reach to computer, so it will not do the required task. So, first task is to convert text data into something that computer can understand.

### 2.1 Tokenization

Tokenization is the process to convert a sentence into tokens. This could be done in many ways, for example:

- On the **basis of White space**, but here punctuation marks will remain with the word so 2 similar words will be treated differently just because of punctuation, so "it" and "it?" will be treated differently.
- On the basis of **Punctuation**, here punctuation mark will be treated as a different token, so "isn't" will break into "isn", " ' ", "t". Now "isn" and "t" have no meaning.

Next We can do is something so that, "isn't" breaks into "is", "n't", This makes much more sense.

**Token Normalization** Now we have tokens with us, we need to normalise them i.e. sometimes a word in used in different forms, like "wolf" and "wolves", computer will think of them as 2 different words with different meanings, that will unnecessarily increase the vocabulary, so we will try to reduce these words and count them as a single word in vocabulary list.

There are 2 ways to do this: Stemming and Lemmatization

- **Stemming** removes suffixes to get root form, "talks" changes to "talk" and "ponies" to "poni", but here poni has no meaning.
- **Lemmatization** returns base form of word, "feet" to "foot" and "wolves" to "wolf" and "talked" to "talked", here talked didn't changed. So, we need to see which one will be better for our dataset.

Further, we can also try lowercasing letters so that "Feet" and "feet" will be considered same, but that also involves some cases, as "us" and "US" are different, so we can do all this according to needs of dataset.

### 2.2 Tokens to features

For each word we will have a feature column(one hot vector) called text vectorization but here we will loose the word order, So to solve that we will use **N grams**, N is a positive integer, if N = 2, the words will be counted as pairs, if N = 3, then words will be counted as triplets and so on. Now this will lead to too many number of features. So we can Remove stop words like : "and", "is", "are", "but" etc. Now, we have a matrix "X".

## 2.3 Frequency

- **Term Frequency** = TF(t, d) = frequency of term 't' in document 'd'.
- **Inverse Document Frequency** = IDF(t, D) = log $(N/( \ |d \in D : t \in d| \ ))$
- **TF-IDF** = TF(t,d)* IDF(t, D)

A high weight is reached by a high frquency term and a low document frequency of the term in whole collection of documents. Now we get a final X matrix, earlier matrix was sparse, this will be a dense matrix.

## 3 Experiment

The **dataset** that has been used in this NLP classification problem consisted of 2 classes with labels as positive and negative, positive classes consists of documents with positive movie reviews and negative classes consists of documents containing negative movie reviews, Each containing 1000 documents. We need to train a model so that it can classify the unseen documents correctly as positive or negative. 4 different Classifiers have been used, namely: **Random forest Classifier, K nearest neighbour, Support Vector Machine, Logistic Regression**.
For **Data preprocessing**, i have removed all special characters, numbers, unwanted spaces, single characters and lower cased all letters then did lemmatization of the data and then used Bag of Words to convert it to numbers from text, also used 1500 most occurring words as features for training our classifier, included only those words that occur in at least 5 documents and in a maximum of 70% of all the documents. Now Bag of words assigns a score to a word based on its occurrence in a particular document. It doesn't take into account the fact that the word might also be having a high frequency of occurrence in other documents as well, so to tackle this issue, we use TF-IDF and finally get our matrix X, and then feed it into Classifier.
The accuracy Measure used in experiments is Classification Accuracy Score.

## 4 Results

Table 1: Random Forest Classifier

| **No. of trees** | **Max Depth** | **Min Sample split** | **Accuracy** | **F1 score** |
|---|---|---|---|---|
| 1000 | 40 | 2 | 0.7663230 | 0.83 |
| 800 | 50 | 4 | 0.7731958 | 0.85 |
| 800 | 70 | 2 | 0.776632302 | 0.85 |
| 700 | 50 | 4 | 0.780068 | 0.86 |
| 700 | 70 | 2 | 0.84 | 0.84 |

Table 2: K Nearest Neighbour

| **K value** | **Accuracy** | **F1 score** |
|---|---|---|
| 10 | 0.7250859 | 0.81 |
| 5 | 0.71134020 | 0.89 |
| 20 | 0.7079037 | 0.81 |
| 30 | 0.7285223367 | 0.82 |

Table 3: Support Vector Machine

| **Kernel** | **C value** | **Accuracy** | **F1 score** |
|---|---|---|---|
| Linear | 1 | 0.84192439 | 0.89 |
| Linear | 1.5 | 0.8384879 | 0.89 |
| rbf | 1.5 | 0.810996 | 0.87 |
| rbf | 1 | 0.78350515 | 0.86 |

Table 4: Logistic Regression

| **Penalty** | **Accuracy** | **F1 score** |
|---|---|---|
| l2 | 0.841924 | 0.69 |
| None | 0.8419243 | 0.69 |

So, we can conclude that we get best Classification accuracy score on Validation set of 84% with Logistic regression and SVM classifiers and Random Forest
But Since the F1 score of Logistic regression Model is Low, we can't consider it as a good Model, So we get SVM and Random Forest, performing best on our Dataset.