Honours Degree in Computing

Text Analyse Assessment:

Analyse a dataset containing text from 50 articles

Submitted by: Derek McCarthy, B00007439

Submission date: 04/04/2019

Declaration

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, except where otherwise stated. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged, and the source cited are identified in the assignment references.

I/We understand that plagiarism, collusion, and copying are grave and serious offences and accept the penalties that would be imposed should I/we engage in plagiarism, collusion or copying. I acknowledge that copying someone else's assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism. I/We have read and understood the colleges plagiarism policy 3AS08 (available here).

This material, or any part of it, has not been previously submitted for assessment for an academic purpose at this or any other academic institution.

I have not allowed anyone to copy my work with the intention of passing it off as their own work.

Name: Derek McCarthy   Dated: 04/04/2019

Contents

Project Overview

The purpose of this project was to build a dataset from fifty different articles from the internet. Half of the dataset will contain articles on English football and the other half will consist of articles relating to Brexit, Donald Trump, Space and football. To build the dataset we first had to create a web scrapper to retrieve the relevant text from the web site, such as text placed within html <p> and <h1-6> tags. Once we have received the relevant text the article was then saved to a csv file along with its label/category. The English football articles were given the label “englishFootball” and the rest of the data was given the label “notEnglishFootball”. Once we have saved the data to csv we will then read this csv file into the program. We then carried out several different cleaning techniques to reduce the data feature space. After this we created different word vectors from the data. Then we carried out 4 different model techniques 2 clustering and 2 classification. By doing this we will be able to establish which modelling technique is best at predicting the category/label for the data.

Data Understanding

In this section of the report we will look at the three of the main characteristics of Data Understanding,

- Gathering the Data
- Describing the Data
- Do an initial survey of the data Quality

Gathering the Data

The first stage of data understanding is to gather the data. Firstly, we obtained fifty links for the two categories of the data, which were stored within a list We then had to have some way of retrieving the relevant data from these links. This was achieved by building a web scrapper which extracts text placed within html <p> and <h1-6> tags from websites.

```python
def retriveDataFromUrls(url):
    source = urllib.request.urlopen(url).read()
    soup = bs.BeautifulSoup(source,'lxml')
    return soup

def retriveMainText(soup):
    titles = soup.find_all(re.compile('^h[1-6]$'))
    h1s = [str(h1) for h1 in soup.find_all('h1')]
    h1s = [re.sub(r'</?h1>', '', h1) for h1 in h1s]
    paragraphs = [str(p) for p in soup.find_all('p')]
    mainParagraphs = [p for p in paragraphs if p.startswith('<p>')]
    mainParagraphs = [re.sub(r'<.+?>', '', p) for p in mainParagraphs]
    return ' '.join(mainParagraphs)
```

Figure 1 - Web scrapper

In the above screenshot figure 1 we can see the two functions used for scraping the websites for the data. The first function was passed the websites URL which then took all the text from the website including html tags. The data acquired from this was then passed to the second function which stripped the html tags from the data.

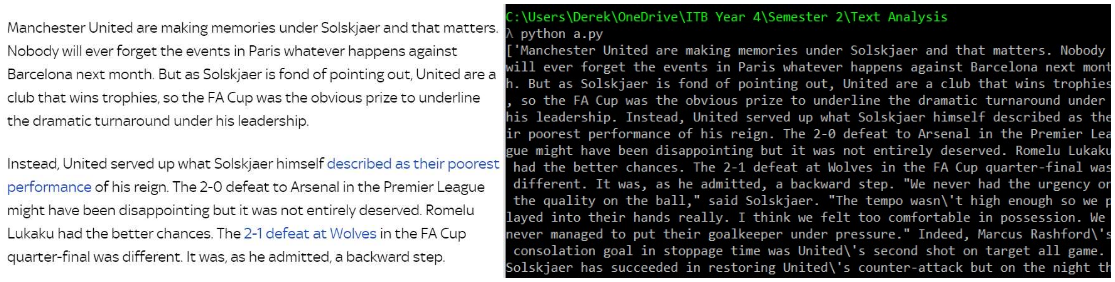


Figure 2 – On the left text from website on the right the text retrieved from our web scrapper

Once the html tags were removed we analysed the data by comparing the resulting data to the data on the website see figure 2. Once we were happy with the data we then saved it to a csv file along with its label.

Describing the Data

To get a better understanding of the data we created a word cloud of each category of data.

Figure 3 - English Football Word Cloud

In figure 3 we can see the word cloud produced from the ‘englishFootball’ data in our dataset. By analysing this word cloud, it will help us identify potential stop words within the dataset.

Figure 4 - Not English Football Word Cloud

In figure 4 we can see the word cloud produced from the 'notEnglishFootball' data in our dataset. By analysing this word cloud, it will help us identify potential stop words within the dataset.

Exploring the data

In this section we will explore the data we have scrapped from the websites.

	A	B	C	D	E	F
1	label	doc				
2	englishFoo	Manchester United are making memories under Sc				
3	englishFoo	City remain in the hunt for the quadruple this seas				
4	englishFoo	Second-half goals from Raul Jimenez and Diogo Jo				
5	englishFoo	The Scot was appointed as United manager follow				
6	englishFoo	Chelsea are facing a transfer ban for the next two				
7	englishFoo	The pitch invasions occurred at Premier League fixt				
8	englishFoo	Chelsea dominated the first half at Goodison Park				
9	englishFoo	Mane produced another piece of brilliance for Live				

Figure 5 - Data Stored in CSV file

In figure 5 we can see the data stored in the CSV file. Within this file we have 2 columns and 50 rows. In the first column we have the label of the data and in the second column we have the data itself.

```
40         NaN    NaN  NaN    NaN     NaN   NaN      NaN   3.0   3.0     NaN  13.0
41         NaN    NaN  NaN    NaN     NaN   NaN      NaN   3.0   7.0     NaN  14.0
42         NaN    NaN  1.0    NaN     NaN   NaN      NaN  10.0   1.0     NaN  14.0
43         NaN    NaN  2.0    NaN     NaN   NaN      NaN   6.0   2.0     NaN  15.0
44         NaN    NaN  NaN    NaN     NaN   NaN      NaN   NaN   2.0     NaN  11.0
45         3.0    3.0  NaN    NaN     NaN   NaN      NaN  10.0   7.0     NaN  22.0
46         5.0    1.0  1.0    2.0     NaN   NaN      NaN  14.0  25.0     NaN  45.0
47         NaN    2.0  NaN    NaN     NaN   1.0      4.0   4.0   3.0     NaN  12.0
48         NaN    3.0  1.0    NaN     NaN   NaN      1.0   9.0   NaN     NaN  11.0
49         NaN    2.0  NaN    1.0     NaN   NaN      NaN   7.0   2.0     NaN   9.0

[50 rows x 3897 columns]
```

Figure 6 - Data before Cleaning

In figure 6 we see the data before any data preparation techniques have being applied. We can see that the dataset contains 50 rows and 3897 columns of data.

Data Preparation

The data preparation phase of the project requires conducting several different phases. The initial phase was to do some basic data cleaning techniques such as removing spaces before a comma, replacing numeric digits with the word digits and replacing punctuation with a space etc.

```
# replace ISO-8859-1 encodings, references, quotes, and multiple white spaces with a single space
docs = clean_corpus(docs, [r'\x09', r'\xa0', r'\x96', r'\[\d+\]', r'[\'"]+', r'\s{2,}'], ' ')
# remove space before a comma
docs = clean_corpus(docs,[r'\s+,'],',')
# replace actual digits with the word DIGITS
docs = [re.sub(r'\d+', " DIGITS ", row) for row in dataCopy]
# replace punctuation with a space
docs = [re.sub(r'[\'\.\?!,;:\\/å£*"]+', " ", row) for row in dataCopy]
# replace repeating white spaces with a single space
docs = [re.sub(r'\s{2,}', " ", row) for row in dataCopy]
```

Figure 7 - Initial Data Cleaning

In figure 7 we can see the code snippet of the initial cleaning phase of the data. By carrying out these initial cleaning techniques this helps reduce the size of the data feature space.

After this we tokenized the data and applied part of speech POS filter. This filter removes the POS's that are unlikely to help in the model prediction. The filter works by removing determiners, modals, conjunctions, prepositions, pronouns and punctuation from the data corpus.

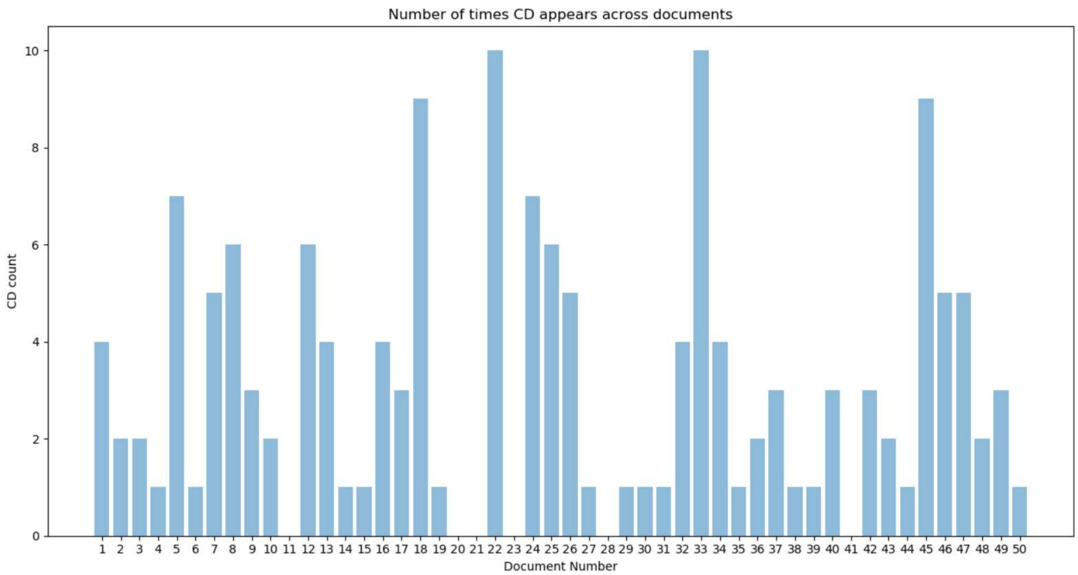


Figure 8 - CD Part of Speech

In figure 8 we can see the graph of the CD part of speech across the fifty documents. As you can see on the graph it appears on all but 6 of the documents. Because of the frequency of CD part speech through the data set we decided to keep it.

From carrying out the previous cleaning techniques we can build a bag of words from our data.

38	0.0	0.0	0.0	0.0	0.0
39	0.0	0.0	0.0	0.0	0.0
40	0.0	0.0	0.0	0.0	0.0
41	0.0	0.0	0.0	0.0	0.0
42	0.0	0.0	0.0	0.0	0.0
43	0.0	0.0	0.0	0.0	0.0
44	0.0	0.0	0.0	0.0	0.0
45	3.0	3.0	0.0	0.0	0.0
46	5.0	1.0	2.0	0.0	0.0
47	0.0	2.0	0.0	0.0	1.0
48	0.0	3.0	0.0	0.0	0.0
49	0.0	2.0	1.0	0.0	0.0

[50 rows x 3551 columns]

Figure 9 - Initial Cleaning Results

In figure 9 we can see the results of implementing the initial cleaning techniques discussed above. From carrying out this initial cleaning process we have reduced the data feature space from 50 rows and 3897 columns to 50 rows 3551 columns. In addition to the cleaning techniques applied we also replace NaN values in the dataset with a zero.

The next phase in the data preparation was the removal of stop words. This is to filter out words which contain little information or are not helpful in the predictive ability. Firstly, we removed the common English stop words and then removed the custom stop words which we identified from the word cloud.

```python
def removeStopWords(newTokenizedCorpus):
    stopWords = nltk.corpus.stopwords.words('english')
    newTokenizedCorpus = [[word for word in doc if word not in stopWords]
        for doc in newTokenizedCorpus]

    # list of custom stop words to be removed
    customStopWords = ['solskjaer', 'premier', 'manchester', 'united', 'league', 'said',
    'first', 'will', 'players', 'chelsea', 'city', 'wolves','team', 'get', 'president',
    'prime', 'deal', 'brexit', 'minister', 'trump', 'mps','house', 'mrs', 'government',
    'vote', 'support']
    newTokenizedCorpus  = [ [ word for word in doc if word not in customStopWords ]
        for doc in newTokenizedCorpus ]
    return newTokenizedCorpus
```

Figure 10 - Stop Word Removal

In figure 10 we can see the function which receives the tokenized data and removes the stop words from the data corpus. Again, this helps reduce the feature space and improve the accuracy of the model. After removing the stop words from the dataset, we reduced the data feature space to 50 rows and 3257 columns. The custom stop words were identified from our word cloud and removed as they were deemed not predictive of the data.

The next cleaning technique we conducted was Porter Stemming to reduce words to their root form.

Stop Words Bow

	making	memories	matters	nobody	ever	forget
0	2.0	1.0	1.0	1.0	1.0	1.0
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	0.0	1.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0

Stemmed Bow

	make	memori	matter	nobodi	ever	forget
0	2.0	1.0	1.0	1.0	1.0	1.0
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	0.0	1.0	0.0	0.0	0.0
8	1.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	1.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0

Figure 11 - Un-Stemmed & Stemmed Dataset

In figure 11 we can see how the words in the dataset looked before stemming was applied to the dataset. In this figure we can see the word ‘making’ was reduced to its root form ‘make’, ‘memories’ was reduced to ‘memori’ and ‘matters’ was reduced to ‘matter’ etc. Although stemming can produce incorrect words this method was preferred over lemmatization as its less computational heavy on computing hardware resources. From applying the porter stemming technique we have reduce the dataset to 50 rows and 2548 columns.

	count	mean	std	min	25%	50%	75%	max
make	50.0	0.52	0.788696	0.0	0.0	0.0	1.00	3.0
memori	50.0	0.02	0.141421	0.0	0.0	0.0	0.00	1.0
matter	50.0	0.08	0.274048	0.0	0.0	0.0	0.00	1.0
nobodi	50.0	0.02	0.141421	0.0	0.0	0.0	0.00	1.0
ever	50.0	0.12	0.385450	0.0	0.0	0.0	0.00	2.0
forget	50.0	0.02	0.141421	0.0	0.0	0.0	0.00	1.0
event	50.0	0.12	0.385450	0.0	0.0	0.0	0.00	2.0
pari	50.0	0.06	0.313636	0.0	0.0	0.0	0.00	2.0
whatev	50.0	0.04	0.197949	0.0	0.0	0.0	0.00	1.0
happen	50.0	0.36	0.692820	0.0	0.0	0.0	0.75	3.0

Figure 12 - Summary Statistics

In figure 12 we can see the summary statistic of the data. In this we can see statistics such as the mean, standard deviation and quartiles of the data.

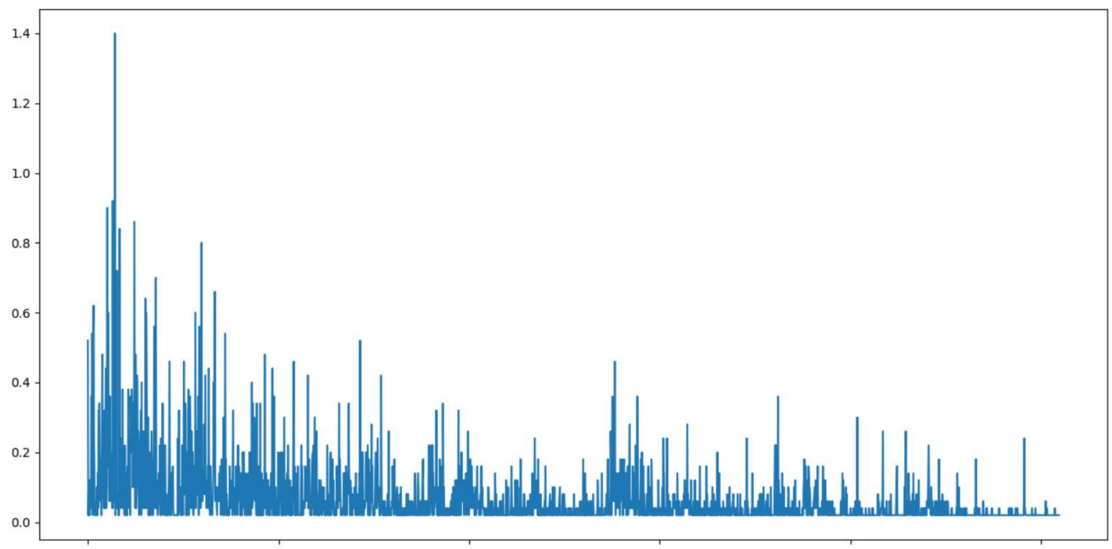


Figure 13 - Summary Statistics Mean

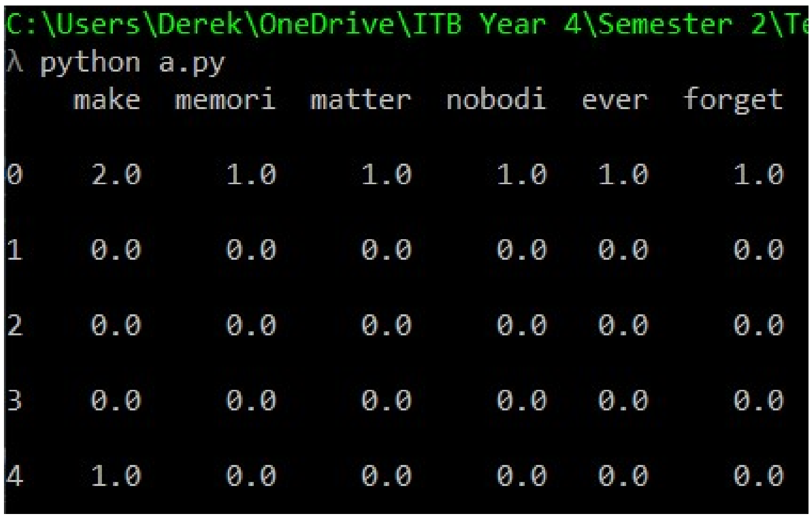
In the figure 13 we can see the summary statistic mean of the data on the graph.

Data Vectorization

As part of the data vectorization we will create 3 types of word vectors simple counts, normalized counts and TFIDF.

Simple Counts

The simple counts vector just counts the number of times a word is present in the dataset.



```
C:\Users\Derek\OneDrive\ITB Year 4\Semester 2\Te
λ python a.py
   make  memori  matter  nobodi  ever  forget

0   2.0     1.0     1.0     1.0   1.0     1.0

1   0.0     0.0     0.0     0.0   0.0     0.0

2   0.0     0.0     0.0     0.0   0.0     0.0

3   0.0     0.0     0.0     0.0   0.0     0.0

4   1.0     0.0     0.0     0.0   0.0     0.0
```

Figure 14 - Simple Counts Vector

In figure 14 we can see how frequently a word appears in each document. For example, the word make appears in row 0/Document 1 two times and document 5 one time.

Normalized Counts

Unlike simple counts normalized counts doesn't just count the frequency of a word appearing in the data. It determines how relevant a word is to the data by looking at the total number of words in the dataset.

	make	memori	matter	nobodi	ever
0	0.000398	0.000199	0.000199	0.000199	0.000199
1	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.000000
4	0.000305	0.000000	0.000000	0.000000	0.000000
5	0.000000	0.000000	0.000000	0.000000	0.000000
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	0.000250	0.000000	0.000250	0.000000	0.000000
8	0.000622	0.000000	0.000000	0.000000	0.000000
9	0.000000	0.000000	0.000000	0.000000	0.000000
10	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 15 - Normalized Count Vector

In figure 15 we can see the normalized count of each word in the document.

TFIDF

TFIDF determines how important a word is in the data by using a weighting metric.

	make	memori	matter	nobodi	ever
0	2.791857	5.643856	3.643856	5.643856	3.321928
1	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.000000
4	1.395929	0.000000	0.000000	0.000000	0.000000
5	0.000000	0.000000	0.000000	0.000000	0.000000
6	0.000000	0.000000	0.000000	0.000000	0.000000
7	1.395929	0.000000	3.643856	0.000000	0.000000
8	1.395929	0.000000	0.000000	0.000000	0.000000
9	0.000000	0.000000	0.000000	0.000000	0.000000
10	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 16 - TFIDF Vector

In figure 16 we can see the TFIDF value of each word in the dataset. The higher the value the more important the word is to the dataset.

Data Mining

Clustering

In this section we will look at the two clustering algorithms we implemented and analyse the results from them. The two clustering algorithms that we implemented were K-Means clustering and Agglomerative clustering.

K-Means Clustering

For the K-Means clustering we used the library K-Means from sklearn. We got the TFIDF values and passed these to the algorithm. We also set the algorithm to predict the labels of the data.

```
[50 rows x 2548 columns]
Cluster Distance Ratio:  0.5560277313624659
```

Figure 17 - Cluster Distance Ratio

In figure 17 we can see the cluster distance ration of the K-Means algorithm using a K value of 2. With the K value set to 2 we found this gave the most accurate prediction. With the cluster distance ratio at 0.5 this indicates the model has good accuracy as the lower the distance ratio the better the accuracy.

Agglomerative clustering

The Agglomerative clustering model is a hierarchical clustering technique which builds a hierarchy of cluster from the data.

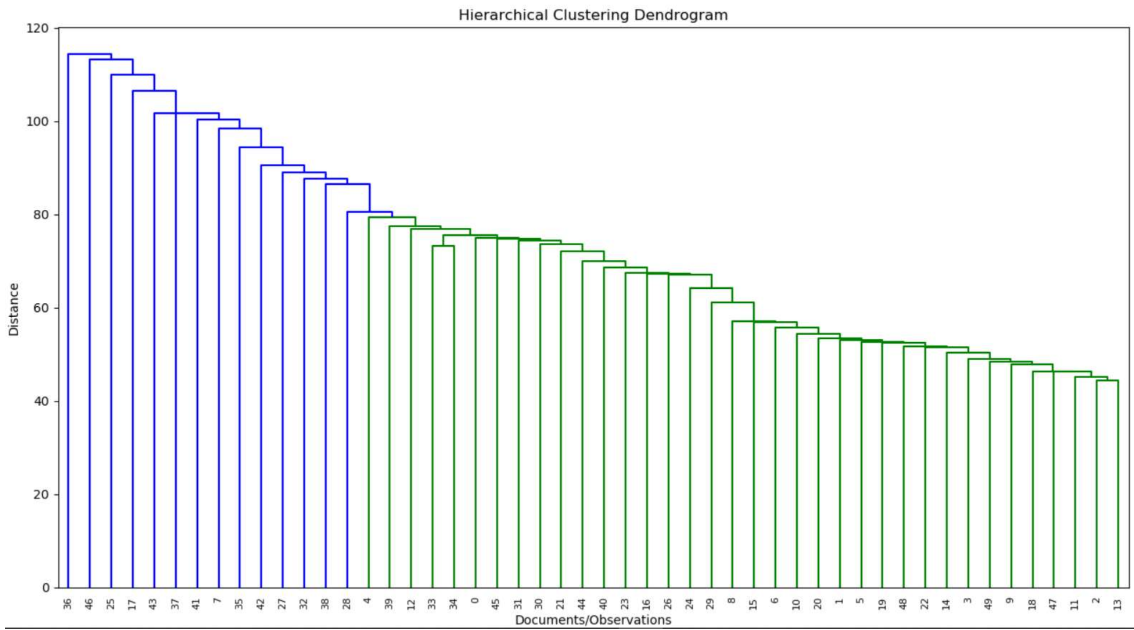


Figure 18 - Agglomerative Clustering Dendrogram

In figure 18 we can see the results of the dendrogram which was created from the data.

Classification

In this section we will look at the two classification algorithms we implemented and analyse the results from them. The two clustering algorithms that we implemented were SVM and Naïve Bayes. With each algorithm we will analyse the results of using the 3 different word vectors we created.

SVM

Support Vector Machine (SVM) is a binary classifier algorithm which uses algebraic methods to classify data.

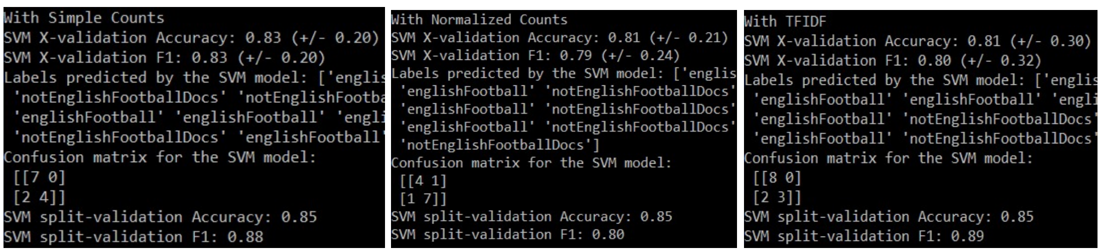


Figure 19 – SVM Results With the 3 Different Vectors

In figure 19 we can see the results of SVM algorithm being performed on the 3 different word vectors Simple counts, Normalized counts and TFIDF respectively. The algorithm was setup to split the data 75% for training and 25% for testing. It used 5-fold cross validation as this gave best results. Both Normalized counts and TFIDF produced the same X-validation/Standard deviation at 0.81 while Simple counts had an X-validation value of 0.83. The split validation accuracy was 0.85 across all three-word vectors. As for the SVM split validation F1 TFIDF had the highest accuracy at 0.89 while simple counts had 0.88 and Normalized counts had 0.80.

Naïve Bayes

Naïve Bayes is a binary classifier algorithm which uses probabilistic methods to classify data.

```
λ python a.py
With Simple Counts
NB X-validation Accuracy: 0.84 (+/- 0.13)
NB X-validation F1: 0.83 (+/- 0.13)
Labels predicted by the NB model: ['englis
 'notEnglishFootballDocs' 'notEnglishFoot
 'notEnglishFootballDocs' 'notEnglishFoot
 'englishFootball' 'englishFootball' 'engl
 'notEnglishFootballDocs' 'englishFootball
Confusion matrix for the NB model:
 [[5 0]
 [1 7]]
NB split-validation Accuracy: 0.92
NB split-validation F1: 0.91
```

```
With Normalized Counts
NB X-validation Accuracy: 0.92 (+/- 0.14)
NB X-validation F1: 0.91 (+/- 0.14)
Labels predicted by the NB model: ['englis
 'englishFootball' 'notEnglishFootballDoc
 'notEnglishFootballDocs' 'notEnglishFoot
 'englishFootball' 'englishFootball' 'engl
Confusion matrix for the NB model:
 [[6 0]
 [3 4]]
NB split-validation Accuracy: 0.77
NB split-validation F1: 0.80
```

```
With TFIDF
NB X-validation Accuracy: 0.81 (+/- 0.06)
NB X-validation F1: 0.81 (+/- 0.06)
Labels predicted by the NB model: ['notEng
 'englishFootball' 'englishFootball' 'notE
 'englishFootball' 'englishFootball' 'engl
 'notEnglishFootballDocs' 'englishFootball
 'englishFootball']
Confusion matrix for the NB model:
 [[9 0]
 [1 3]]
NB split-validation Accuracy: 0.92
NB split-validation F1: 0.95
```

Figure 20 - Naive Bayes Results With the 3 Different Vectors

In figure 20 we can see the results of Naive algorithm being performed on the 3 different word vectors Simple counts, Normalized counts and TFIDF respectively. The algorithm was setup to split the data 75% for training and 25% for testing. It used 3-fold cross validation as this gave best results. Normalized counts produced the best X-validation accuracy with 0.92 while Simple counts and TFIDF had a X-validation value of 0.84 and 0.81 respectively. For the X-validation F1 score again Normalized counts produced the best results with 0.91 while Simple counts and TFIDF had an F1 score of 0.83 and 0.81 respectively. As for the split validation accuracy both Simple counts and TFIDF had an accuracy of 0.92 while Normalized counts had a split validation accuracy of 0.77. And finally, the TFIDF produced the best split validation F1 accuracy with 0.95 while Simple count and Normalized counts had a split validation F1 accuracy of 0.91 and 0.80 respectively.

Overall the Naïve Bayes with TFIDF produced the best model at predicting the correct labels. While the Naïve Bayes with Simple Count was second best. The only SVM model to make it to the top three algorithm models was the SVM model with TFIDF.

Appendix

```
import bs4 as bs
import urllib.request
import re
import csv
import nltk
import math
from nltk.stem.porter import PorterStemmer
from nltk.corpus import wordnet
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
import sklearn
from sklearn import cluster
from pandas.plotting import parallel_coordinates
from pandas import DataFrame
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.model_selection import train_test_split as tt

import ClasificationAlgorithms as ca

def retriveDataFromUrls(url):
	source = urllib.request.urlopen(url).read()
	soup = bs.BeautifulSoup(source,'lxml')
	return soup

def retriveMainText(soup):
	titles = soup.find_all(re.compile('^h[1-6]$'))
	h1s = [str(h1) for h1 in soup.find_all('h1')]
	h1s = [re.sub(r'</?h1>', '', h1) for h1 in h1s]
	paragraphs = [str(p) for p in soup.find_all('p')]
	mainParagraphs = [p for p in paragraphs if p.startswith('<p>')]
	mainParagraphs = [re.sub(r'<.+?>', '', p) for p in mainParagraphs]
	return ' '.join(mainParagraphs)

def writeToCSV(textData):
	# write the data to a csv file
  with open('data.csv', 'w', newline='', encoding='utf-8') as csvFile:
    attributes = ['label', 'doc']
    writer = csv.DictWriter(csvFile, fieldnames=attributes)
    writer.writeheader()
    writer.writerows(textData)
  csvFile.close()
```

```
def englisFootballURLs():
        englishFootball = [
    "https://www.skysports.com/football/news/11667/11669109/ole-gunnar-solskjaer-could-rue-manchester-united8217s-fa-cup-exit",
    "https://www.skysports.com/football/news/11679/11669236/pep-guardiola-hopes-his-man-city-players-return-from-international-break-unscathed",
    "https://www.skysports.com/football/news/11667/11668024/ole-gunnar-solskjaer-says-manchester-uniteds-fa-cup-exit-was-poorest-performance-since-he-took-over",
    "https://www.skysports.com/football/news/11667/11669275/manchester-united-had-helicopter-ready-for-gareth-bale-says-david-moyes",
    "https://www.skysports.com/football/news/29328/11668881/eden-hazard-situation-creates-big-conundrum-for-chelsea-says-graeme-souness",
    "https://www.skysports.com/football/news/11095/11669256/football-association-investigating-three-new-incidents-of-supporter-pitch-invasions",
    "https://www.skysports.com/football/news/11095/11668461/maurizio-sarri-baffled-by-strange-shift-in-chelsea-mentality",
    "https://www.skysports.com/football/news/11669/11668747/sadio-mane-has-stepped-up-for-liverpool-in-the-title-run-in",
    "https://www.skysports.com/football/news/11675/11668955/tottenham-chairman-daniel-levy-says-stadium-costs-will-not-affect-transfer-spending",
    "https://www.skysports.com/football/news/11095/11668751/tottenham-to-play-crystal-palace-in-first-game-at-new-stadium",
    "https://www.skysports.com/football/news/11670/11668215/arsenal-an-option-for-antonio-valencia-after-manchester-united-exit-claims-agent",
    "https://www.skysports.com/football/news/11667/11669087/nemanja-matic-urges-manchester-united-not-to-let-top-four-finish-slip-away",
    "https://www.skysports.com/football/wolves-vs-man-utd/406343",
    "https://www.skysports.com/football/news/11699/11668028/nuno-espirito-santo-on-fa-cup-win-over-manchester-united-wolves-boss-pleased-to-give-back-joy",
    "https://www.skysports.com/football/news/11095/11667410/javi-gracia-says-andre-grays-winner-was-no-surprise-and-tips-watford-to-reach-fa-cup-final",
    "https://www.skysports.com/football/news/11685/11667730/manuel-pellegrini-hails-west-ham-character-after-huddersfield-victory",
    "https://www.skysports.com/football/news/11095/11667898/brendan-rodgers-toasts-incredible-leicester-win",
    "https://www.skysports.com/football/burnley-vs-leicester/391059",
    "https://www.skysports.com/football/news/11671/11668455/everton-boss-marco-silva-admits-he-never-doubted-philosophy-after-chelsea-win",
    "https://www.skysports.com/football/news/11678/11668196/peter-beardsley-says-hes-not-a-bully-not-a-racist-amid-fa-investigation",
    "https://www.skysports.com/football/news/11706/11667426/roy-hodgson-unsure-on-wilfried-zaha-return-after-international-break",
    "https://www.skysports.com/football/news/34651/11668343/neil-harris-believes-millwall-were-better-than-brighton-in-fa-cup-loss",
    "https://www.skysports.com/football/news/11700/11665481/yan-valery-signs-southampton-contract-extension-until-2023",
```

```
        "https://www.skysports.com/football/news/11708/11667929/harry-maguire-red-card-shocked-burnley-says-sean-dyche",
        "https://www.skysports.com/football/news/11681/11668283/scott-parker-gutted-for-fulham-players-after-liverpool-defeat"
    ]

            return englishFootball

def notEnglishFootballURLs():
    notEnglishFootball = [
        "https://news.sky.com/story/brexit-the-seven-options-mps-could-vote-on-and-what-they-mean-11672852",
        "https://news.sky.com/story/theresa-may-set-for-showdown-with-mps-after-hinting-at-third-vote-on-deal-11676551",
        "https://news.sky.com/story/dup-prefer-long-brexit-delay-to-pms-deal-sky-sources-11675614",
        "https://news.sky.com/story/theresa-may-tells-mps-she-cannot-commit-to-alternate-brexit-strategy-11674856",
        "https://news.sky.com/story/pm-fights-to-retain-power-as-mps-look-to-seize-control-of-brexit-11674589",
        "https://news.sky.com/story/may-faces-cabinet-coup-as-ministers-warn-she-has-days-left-11673877",
        "https://news.sky.com/story/theresa-may-could-drop-vote-on-brexit-deal-if-it-lacks-support-11672888",
        "https://news.sky.com/story/us-signals-new-space-race-trump-wants-astronauts-back-on-the-moon-within-five-years-11676176",
        "https://news.sky.com/story/mueller-report-putin-ready-to-improve-us-ties-after-trump-cleared-of-collusion-11674840",
        "https://news.sky.com/story/trump-questions-will-remain-until-full-mueller-report-is-revealed-11674570",
        "https://news.sky.com/story/donald-trumps-golan-policy-change-is-illegal-and-unnacceptable-11672382",
        "https://news.sky.com/story/trump-under-fire-over-bizarre-john-mccain-funeral-comments-11671442",
        "https://news.sky.com/story/donald-trump-attacks-the-husband-of-kellyanne-conway-one-of-his-closest-aides-11670943",
        "https://news.sky.com/story/president-trump-has-vetoed-congress-decision-to-end-his-national-emergency-11666719",
        "https://news.sky.com/story/revealed-what-super-jupiter-129-light-years-from-earth-is-like-11676420",
        "https://news.sky.com/story/anne-mcclain-female-astronaut-at-centre-of-nasa-spacesuit-row-speaks-out-11676621",
        "https://news.sky.com/story/black-hole-radio-jet-pointed-almost-directly-at-earth-11614684",
        "https://news.sky.com/story/far-side-of-the-moon-chinas-lunar-probes-take-pictures-of-each-other-11604801",
```

```
        "https://news.sky.com/story/barnards-star-b-frozen-super-earth-found-six-light-years-away-could-support-life-experts-say-11554317",
        "https://news.sky.com/story/nasa-probe-gets-closer-to-the-sun-than-any-spacecraft-in-history-11546524",
        "https://www.skysports.com/football/news/11661/11676254/louis-van-gaal-says-managing-tottenham-might-have-been-better-than-manchester-united",
        "https://www.skysports.com/football/news/11679/11673434/ilkay-gundogans-manchester-city-form-covering-loss-of-fernandinho",
        "https://www.skysports.com/football/news/11667/11675567/victor-lindelof-keen-for-ole-gunnar-solskjaer-to-stay-at-man-utd",
        "https://www.skysports.com/football/news/11667/11674975/ander-herrera-unsure-of-manchester-united-future",
        "https://www.skysports.com/football/news/11667/11674923/juan-mata-hails-retired-louis-van-gaal-and-thanks-him-for-time-at-manchester-united"
    ]
    return notEnglishFootball

def clean_corpus(corpus, to_replace_list, replacement=''):
    for regex in to_replace_list:
        corpus = [re.sub(regex, replacement, doc) for doc in corpus]
    return corpus

def build_bow (tokenized_docs):
    freq_dists = []
    for tokenized_doc in tokenized_docs:
        d = {}
        for token in tokenized_doc:
            if token in d.keys():
                d[token] += 1
            else:
                d[token] = 1
        freq_dists.append(pd.Series(d))
    return pd.DataFrame(freq_dists)

# read in the data from the csv
def readCsvFile():
    data = pd.read_csv('data.csv')
    return data

# tokenize the data
def tokenizeText(data):
    tokenizedDocs = [nltk.word_tokenize(doc) for doc in docs]
    return tokenizedDocs

# tokenize the data
def tagTokenizeText(data):
    taggedCorpus = [nltk.pos_tag(doc) for doc in data]
```

```
    return taggedCorpus

def removePOS(taggedCorpus, data):
    new_tagged_corpus = []
    for doc in taggedCorpus:
        new_doc = []
        for pair in doc:
            if pair[1] not in data:
                new_doc.append(pair)
        new_tagged_corpus.append(new_doc)
    return new_tagged_corpus

def countCdPartOfSpeech(newTaggedCorpus):
    cdDistributions = []
    for doc in newTaggedCorpus:
        count = 0
        for pair in doc:
            if pair[1] == 'CD':
                count += 1
        cdDistributions.append(count)
    return cdDistributions

def visualizeCdDistributions(cdDistributions):
    plt.rcdefaults()
    # we named the docs according to their order in the corpus
    docs = ['1', '2', '3', '4', '5', '6','7','8', '9', '10', '11', '12', '13','14','15', '16', '17', '18', '19', '20',
    '21','22', '23', '24', '25', '26', '27','28','29', '30', '31', '32', '33', '34','35','36', '37', '38',
    '39', '40', '41','42','43','44', '45', '46', '47', '48', '49','50']
    y_pos = np.arange(len(docs))
    plt.bar(y_pos, cdDistributions, align='center', alpha=0.5)
    plt.xticks(y_pos, docs)
    plt.ylabel('CD count')
    plt.xlabel('Document Number')
    plt.title('Number of times CD appears across documents')
    plt.show()

def buildNewCorpus(newTaggedCorpus):
    newTokenizedCorpus = []
    for doc in newTaggedCorpus:
        newTokenizedCorpus.append([pair[0] for pair in doc])
    #print(newTokenizedCorpus)<<<<<<<<<--------------Add print outs for report
    # apply case transformation:
    newTokenizedCorpus = [ [word.lower() for word in doc]
        for doc in newTokenizedCorpus]
    return newTokenizedCorpus

def removeStopWords(newTokenizedCorpus):
```

```
    stopWords = nltk.corpus.stopwords.words('english')
    newTokenizedCorpus = [[word for word in doc if word not in stopWords]
        for doc in newTokenizedCorpus]

    # list of custom stop words to be removed
    customStopWords = ['solskjaer', 'premier', 'manchester', 'united', 'league', 'said',
    'first', 'will', 'players', 'chelsea', 'city', 'wolves','team', 'get', 'president',
    'prime', 'deal', 'brexit', 'minister', 'trump', 'mps','house', 'mrs', 'government',
    'vote', 'support']
    newTokenizedCorpus  = [ [ word for word in doc if word not in customStopWords ]
        for doc in newTokenizedCorpus ]
    return newTokenizedCorpus

def porterStemming(newTokenizedCorpus):
    stemmer = nltk.PorterStemmer()
    stemmedCorpus = [[stemmer.stem(word) for word in doc]
        for doc in newTokenizedCorpus]
    return stemmedCorpus

def plotMeanOfStats(stats):
    means = stats['mean']
    means.plot()
    plt.show()

def findSynonym(data):
    tokens = tokenizeText(data)
    synonyms = []
    for syn in wordnet.synsets(token):
        for lm in syn.lemmas():
            synonyms.append(lm.name())
    return synonyms

def compute_idfs(bow):
    ## n = the number of rows/docs
    N = len(bow.index)
    ## the list of how many docs a terms occurs in
    nis = []
    ## go through each term/column
    for ti in bow.columns:
        ni = 0
        ## go through each row
        for row in range(N):
            ## check that the entry for that term in the given rowis > 0
            ## which means it occurs in a doc
            if bow[ti].iloc[row] > 0:
                ## if it occurs, simply increase the ni by 1
                ni += 1
```

```
        ## append the ni to the nis list
        nis.append(ni)
    ## compute the logs using the nis list and N
    return [math.log2(N/ni) for ni in nis]

def build_tfidf_bow (bow, idfs):
    ##go through each column and each idf
    for i in range( len(idfs)):
        idf = idfs[i]
        column = bow.columns[i]
        ## go through each value in the row
        for row in bow.index:
            ## replace each value with the tf-idf weight
            bow[column].iloc[row] = bow[column].iloc[row]  * idf
    return bow

def build_tf_bow (bow, total_counts):
    ##go through each row and total
    for i in range( len(total_counts)):
        total = total_counts[i]
        row = bow.iloc[i]
        ## go through each value in the row
        for col in row.index:
            ## replace each value with the tf weight
            row[col] = row[col] / total

def kMeansClustering(data):
    A = data.copy()

    for k in range (1, 11):

        # Create a kmeans model on our data, using k clusters.  random_state helps ensure that
the algorithm returns the same results each time.
        kmeans_model = KMeans(n_clusters=k, random_state=1).fit(A.iloc[:, :])

        # These are our fitted labels for clusters -- the first cluster has label 0, and the second
has label 1.
        labels = kmeans_model.labels_

        # Sum of distances of samples to their closest cluster center
        interia = kmeans_model.inertia_
        print("k:",k, " cost:", interia)
    print("")

def display_dendrogram(df, method='single'):
    Z = linkage(df, method)
    plt.figure(figsize=(25, 10))
```

```
    plt.title('Hierarchical Clustering Dendrogram')
    plt.xlabel('Documents/Observations')
    plt.ylabel('Distance')
    dendrogram(Z, labels=df.index, leaf_rotation=90)
    plt.show()

def clustering_agglomerative(data):
    model = cluster.AgglomerativeClustering(
        affinity='cosine',linkage='single')
    model.fit(data)
    cluster_labels = model.labels_ + 1
    return cluster_labels

def clustering_k_means(data, k=2):
    model = cluster.KMeans(k)
    model.fit(data)
    clust_labels = model.predict(data)
    centers = model.cluster_centers_
    return (clust_labels, centers)

def display_k_centers(df, cluster_centers, cluster_labels):
    centers_df = pd.DataFrame(cluster_centers,
                index=['Means1', 'Means2'],
                columns=list(df.columns))
    centers_df['cluster'] = [1, 2]
    print(centers_df)
    plt.figure(figsize=(7, 5))
    plt.title('Clusters 1 and 2 means along 5 terms')
    parallel_coordinates(centers_df, 'cluster',
                    color=['blue', 'red'], marker='o')
    plt.show()

englishFootball = englisFootballURLs()
notEnglishFootball = notEnglishFootballURLs()

# get the data/soups
englishFootballsoups = [retriveDataFromUrls(url) for url in englishFootball]
notEnglishFootballsoups = [retriveDataFromUrls(url) for url in notEnglishFootball]

# retrieve main paragraphs'texts for both types of docs
englishFootballDocs = [retriveMainText(soup) for soup in englishFootballsoups]
notEnglishFootballDocs = [retriveMainText(soup) for soup in notEnglishFootballsoups]

# put the data into a list of dictionaries
data = [{'label':'englishFootball','doc': doc} for doc in englishFootballDocs] +
[{'label':'notEnglishFootballDocs','doc': doc} for doc in notEnglishFootballDocs]
writeToCSV(data)
```

```
# get the data from file
data = readCsvFile()

dataCopy = data['doc'].copy()
# get the labels from the file
labels = data['label'].copy()

# fill in empty values
docs = [row for row in dataCopy.fillna("")]
# replace ISO-8859-1 encodings, references, quotes, and multiple white spaces with a single
space
docs = clean_corpus(docs, [r'\x09', r'\xa0', r'\x96', r'\[\d+\]', r'[\'"]+', r'\s{2,}'], ' ')
# remove space before a comma
docs = clean_corpus(docs,[r'\s+,'],',')
# replace actual digits with the word DIGITS
docs = [re.sub(r'\d+', " DIGITS ", row) for row in dataCopy]
# replace punctuation with a space
docs = [re.sub(r'[\'\.\?!,;:\\/å£*"]+', " ", row) for row in dataCopy]
# replace repeating white spaces with a single space
docs = [re.sub(r'\s{2,}', " ", row) for row in dataCopy]
# tokenize the data
tokens = tokenizeText(docs)

# keep only tokens longer than 3 chars
tokens = [[token for token in doc if len(token)>3] for doc in tokens]

# attach a POS tag to the token
taggedTokens = tagTokenizeText(tokens)

# POS tags to be removed
posToRemove = ['DT', 'MD', 'CC', 'IN', 'TO', 'PRP', 'PRP$', ',', '.', ')', '(', ':', '']
# corpus with tags we want to retain
newTaggedCorpus = removePOS(taggedTokens, posToRemove)

# count the CD part of speech for each document and place it into a list
cdDistributions = countCdPartOfSpeech(newTaggedCorpus)

# show bar chart of the CD distributions throughout the 50 documents
visualizeCdDistributions(cdDistributions)

# build the bag of words
baselineBow = build_bow(tokens)

# replace NaN values with 0 in the bag of words
refinedBow = baselineBow.fillna(0)
```

```
# build a new corpus of docs (tokenized) by extracted from the
# tagged one only the first element of each tuple in each document, that is, each token
newTokenizedCorpus = buildNewCorpus(newTaggedCorpus)

# remove stop words
stopWordsRemoved = removeStopWords(newTokenizedCorpus)

# new bag of words after removingStopWords
stopWordsRemovedBow = build_bow(stopWordsRemoved)
stopWordsRemovedBow = stopWordsRemovedBow.fillna(0)
print("Stop Words Bow")
print(stopWordsRemovedBow)

# apply Porter stemming, then build a new bow
stemmed = porterStemming(stopWordsRemoved)
stemmedBow = build_bow(stemmed)
stemmedBow = stemmedBow.fillna(0)
print("Stemmed Bow")
print(stemmedBow)

# get the stats of the BOW
stats = stemmedBow.describe()
# transpose stats for better view
stats = stats.T
# plot the mean of the stats
plotMeanOfStats(stats)
print(stats)

# --------------- Word vectors --------------------------

# copy BOW
normalizedBow = stemmedBow.copy()
totalCounts = [len(doc) for doc in docs]

# simple counts
simpleCountBow = stemmedBow.copy()
print(simpleCountBow)

# apply tf weigthing to the BOW
build_tf_bow(normalizedBow, totalCounts)
print(normalizedBow)

# copy BOW
copyBow = stemmedBow.copy()

# compute the idfs for the docs in the collection
idfs = compute_idfs(copyBow)
```

```
print(idfs)

tfidfBow = build_tfidf_bow(copyBow, idfs)
print(tfidfBow)

# ---------------END WORD VECTORS -----------------------

# ----------------Clustering Algorithms -----------------

# k means clustering - Method 1
kMeansClustering(tfidfBow)

# k means clustering - Method 2
df = pd.DataFrame(tfidfBow)

cluster_labels, cluster_centers = clustering_k_means(df)
print(cluster_labels, cluster_centers)

df['cluster'] = list(cluster_labels)
print(df)
print("Cluster Distance Ratio: ",sklearn.metrics.davies_bouldin_score(df, cluster_labels))

collection = tfidfBow.copy()
df = pd.DataFrame(collection)
cluster_labels = clustering_agglomerative(df)
print(cluster_labels)
display_dendrogram(df)


# ---------------- End Clustering Algorithms ------------

# ----------------Clasification Algorithms --------------
# SVM Algorithm
# classify with simple counts
print("With Simple Counts")
ca.classificationVectorisedOperationSVM(labels, simpleCountBow)
print()
# classify with normalized counts
print("With Normalized Counts")
ca.classificationVectorisedOperationSVM(labels, normalizedBow)
print()
# classify with tfidf
print("With TFIDF")
ca.classificationVectorisedOperationSVM(labels, tfidfBow)

print("With Simple Counts")
ca.classificationVectorisedOperationNaiveBayes(labels, simpleCountBow)
```

```
print()
# classify with normalized counts
print("With Normalized Counts")
ca.classificationVectorisedOperationNaiveBayes(labels, normalizedBow)
print()
# classify with tfidf
print("With TFIDF")
ca.classificationVectorisedOperationNaiveBayes(labels, tfidfBow)

# ----------------End Clasification Algorithms ----------

#print(refinedBow)
#print(sorted(list(stemmedBow.columns)))
#print(stats)
#print(tfidfBow)
```

ClassificationAlgorithms.py

```
import numpy as np
import pandas as pd
import string, re, nltk
from nltk.corpus import stopwords as sw

from matplotlib import pyplot as plt

from sklearn.model_selection import train_test_split as tt

from sklearn.svm import SVC
from sklearn.naive_bayes import ComplementNB

from sklearn.metrics import confusion_matrix as cm
from sklearn.metrics import accuracy_score as acc
from sklearn.metrics import f1_score as f1
from sklearn.model_selection import cross_val_score as xval
import warnings
warnings.filterwarnings('ignore')

def classificationVectorisedOperationSVM(labels , bow):
	## split both the input/feature set and the output/label set
	## at around 3/4 for training and 1/4 for testing
	## this function returns 4 datasets:
	## 2 for features and labels training set
	## 2 for features and labels testing set
	labels = list(labels[:2443].values) + list(labels[2444:].values)
	x_train, x_test, y_train, y_test = tt(bow, labels,
					test_size=0.25)
```

```
        ## train an SVM classifier
        svm_model = SVC(kernel='linear', C=1.0, random_state=1)
        svm_model.fit(x_train, y_train)
        ## use cross validation first on the training data
        svm_xval_acc_scores = xval(svm_model, x_train, y_train, cv=5)
        svm_xval_f1_scores = xval(svm_model, x_train, y_train, cv=5,scoring='f1_macro')

        ## average the scores across the 5 folds and get the standard deviation
        print("SVM X-validation Accuracy: %0.2f (+/- %0.2f)" %
(svm_xval_acc_scores.mean(),
                        svm_xval_acc_scores.std() * 2))
        print("SVM X-validation F1: %0.2f (+/- %0.2f)" % (svm_xval_f1_scores.mean(),
                        svm_xval_f1_scores.std() * 2))
        ## predict the labels of the test_bow row using the trained model
        svm_predicted_labels = svm_model.predict(x_test)
        print('Labels predicted by the SVM model:', svm_predicted_labels)
        svm_confus_matr = cm(y_true=y_test, y_pred=svm_predicted_labels)
        print('Confusion matrix for the SVM model:\n', svm_confus_matr)
        svm_acc_score = acc(y_true=y_test, y_pred=svm_predicted_labels)
        svm_f1_score = f1(y_true=y_test, y_pred=svm_predicted_labels,
pos_label='englishFootball')
        print("SVM split-validation Accuracy: %0.2f" % svm_acc_score)
        print("SVM split-validation F1: %0.2f" % svm_f1_score)

def classificationVectorisedOperationNaiveBayes(labels , bow):
        ## split both the input/feature set and the output/label set
        ## at around 3/4 for training and 1/4 for testing
        ## this function returns 4 datasets:
        ## 2 for features and labels training set
        ## 2 for features and labels testing set
        labels = list(labels[:2443].values) + list(labels[2444:].values)
        x_train, x_test, y_train, y_test = tt(bow, labels,
                        test_size=0.25)
        ## train an NB classifier
        nb_model = ComplementNB(alpha=1.0, fit_prior=True, class_prior=None,
norm=False)
        nb_model.fit(x_train, y_train)
        ## use cross validation first on the training data
        nb_xval_acc_scores = xval(nb_model, x_train, y_train, cv=3)
        nb_xval_f1_scores = xval(nb_model, x_train, y_train, cv=3,scoring='f1_macro')
        ## average the scores across the 5 folds and get the standard deviation
        print("NB X-validation Accuracy: %0.2f (+/- %0.2f)" % (nb_xval_acc_scores.mean(),
                        nb_xval_acc_scores.std() * 2))
        print("NB X-validation F1: %0.2f (+/- %0.2f)" % (nb_xval_f1_scores.mean(),
                        nb_xval_f1_scores.std() * 2))
        ## predict the labels of the test_bow row using the trained model
        nb_predicted_labels =nb_model.predict(x_test)
```

```
	print('Labels predicted by the NB model:', nb_predicted_labels)
	nb_confus_matr = cm(y_true=y_test, y_pred=nb_predicted_labels)
	print('Confusion matrix for the NB model:\n', nb_confus_matr)
	nb_acc_score = acc(y_true=y_test, y_pred=nb_predicted_labels)
	nb_f1_score = f1(y_true=y_test, y_pred=nb_predicted_labels,
pos_label='englishFootball')
	print("NB split-validation Accuracy: %0.2f" % nb_acc_score)
	print("NB split-validation F1: %0.2f" % nb_f1_score)
```