Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019

IR PROJECT 2 REPORT

PART 1: Extracting Features

Implementation: The index that stores the terms for part 1 uses an nltk English stemmer, the nltk standard english stop words, and the default word tokenizer to process words into the index. The feature extraction first parses a vocab file of all important words in the news groups directory (roughly 45,000 words) and then adds those to the index as words we are interested in tracking. The class file is then generated by hard coding each of the classes values and outputting them to a file. The feature file is generated by simply looping through the index and outputting the term and the feature ID that it has been associated with. Finally, the training file is generated. This function takes the given newsgroup directory and recursively walks through each subdirectory to parse all of the files. For each file, a dictionary of terms that it contains is created and the subject line is parsed and any new terms are added to the file. Then the line specifying the number of lines in the body is parsed, and any special exceptions are handled if one does not exist. Once we have the number of lines, we simply iterate backwards starting at the end of the file, and add the term count of each word that is encountered into the term dictionary for the current file. Additionally, an IDF dictionary with terms as keys is maintained so that if a new or existing term is encountered, we add it to a list of files that the term has appeared in so that we can calculate the IDF later. Once done parsing the file, we iterate through the current document's dictionary and replace each term's count with its actual TF value. The features are then sorted and added to a master list of dictionaries for each document. Once we've looped through all the files, we simply loop through our list of document dictionaries and and calculate the TFIDF value for each term using the documents TF dictionary and the global IDF dictionary of each term. A flag can be set to output the TF, IDF, or TFIDF value for each of the terms, and then each class of the document and its features are output to the training data file in LibSVM format.

Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019

Experimental results: First we ran test on the Index.py to make sure all the functions are running without any errors. Secondly, we ran test to make sure that all of our functions in the feature-extract file were running without errors. Following are the results of the test.
Index Lookup tests: In this test some of the words were added and then their ID's were checked

```
def tests():
    # Tests to verify that the index is working correctly.
    index = Index(nltk.word_tokenize, EnglishStemmer(),
nltk.corpus.stopwords.words('english'))

    index.add('Industrial Disease')
    index.add('With')
    index.add('BAlls')
    index.add('Ball')
    index.add('Private Investigations')
    index.add('So Far Away')
    index.add('Twisting by the Pool')
    index.add('Skateaway')
    index.add('Walk of Life')
    index.add('Romeo and Juliet')
    index.add('Tunnel of Love')
    index.add('Money for Nothing')
    index.add('Sultans of Swing')

    # Index lookup tests.
    print("Index Lookup tests:")
    print(index.lookup('Industrial Disease'))
    print(index.lookup('Balling'))

    # Expect None here since this term does not exist.
    print(index.lookup('Missing'))
```

Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019

```
    print("\n")
    # Test for that stop words work.
    print("Stop Word Tests:")
    print(index.isStopWord("And") == True)
    print(index.isStopWord("or") == True)
    print(index.isStopWord("Facts") == False)
```

The results were following:

1
2
None
Stop Word Tests:
True
True
True

Then the test was done for the feature extraction.
Running tests for feature extraction.
Found Vocab file.
Parsing vocab file...
Finished parsing vocab file.
Term count is: 44985
Correct number of terms found!!

Index and TFIDF tests:
Jack featureID: 1
Jack TF: 1.0
Jack IDF: 0.3010299956639812
Jack TFIDF: 0.3010299956639812

Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019

Discussion: The training data was generated in Libsvm format with 2000 articles being parsed. The feature space is roughly 45,000 terms.

PART 2: Classification
Implementation: In part II we performed classification using Multinomial Naive Bayes, Bernoulli NB, KNN and SVM,using cross validation we are doing macro-average score and based on score we are plotting graphs. We used TFIDF for all the 4 algorithms.
Experimental results: The scores obtained are following:

MultinomialNB(f1 macro) Accuracy: 0.69 (+/- 0.10)
MultinomialNB(precision macro) Accuracy: 0.78 (+/- 0.20)
MultinomialNB(recall macro) Accuracy: 0.70 (+/- 0.08)
BernoulliNB(f1 macro) Accuracy: 0.33 (+/- 0.12)
BernoulliNB(precision_macro) Accuracy: 0.56 (+/- 0.13)
BernoulliNB(recall macro) Accuracy: 0.33 (+/- 0.09)
KNeighborsClassifier(f1 macro) Accuracy: 0.16 (+/- 0.10)
KNeighborsClassifier(precision macro) Accuracy: 0.57 (+/- 0.21)
KNeighborsClassifier(recall macro) Accuracy: 0.22 (+/- 0.07)
SVC (f1 macro) Accuracy: 0.07 (+/- 0.00)
SVC (precision macro) Accuracy: 0.04 (+/- 0.00)
SVC (recall macro) Accuracy: 0.17 (+/- 0.00)

Discussion: During the classifier evaluation process we observed among all the four algorithms used Multinomial Naive Bayes's performance is the highest with the highest accuracy, the second highest performing algorithm is bernoulli Naive Bayes, the third highest performing algorithm is SVM and fourth highest performing algorithm is K Nearest Neighbour.

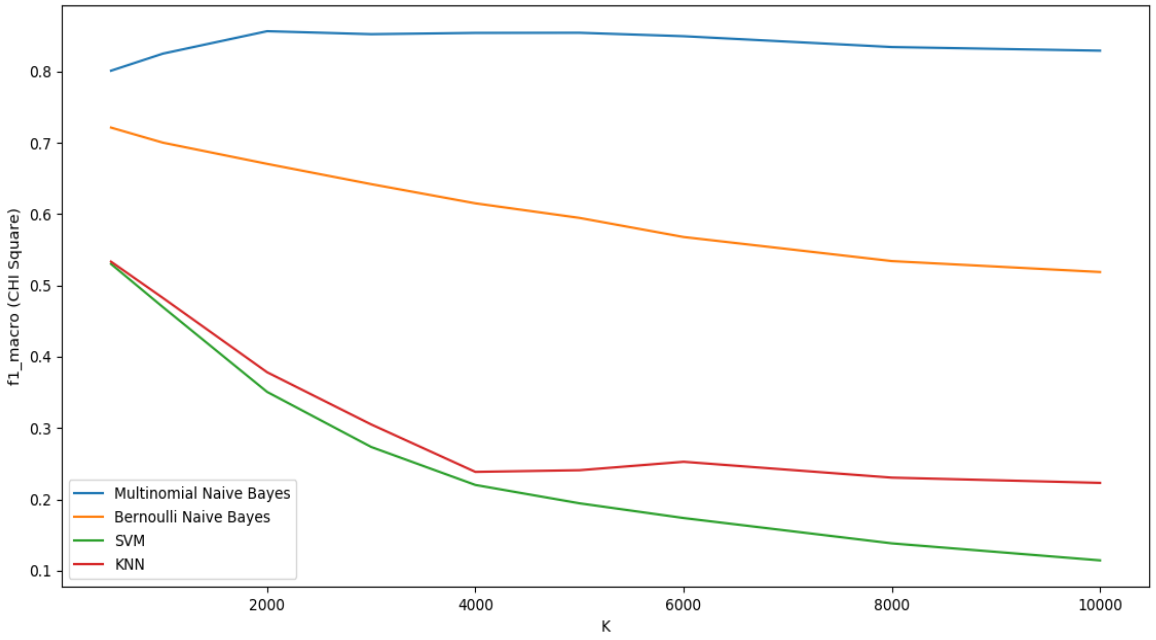
Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019

PART 3: Feature Selection

Implementation: For feature selection we implemented chi-square and mutual information on the dataset using sklearn.

Experimental results:

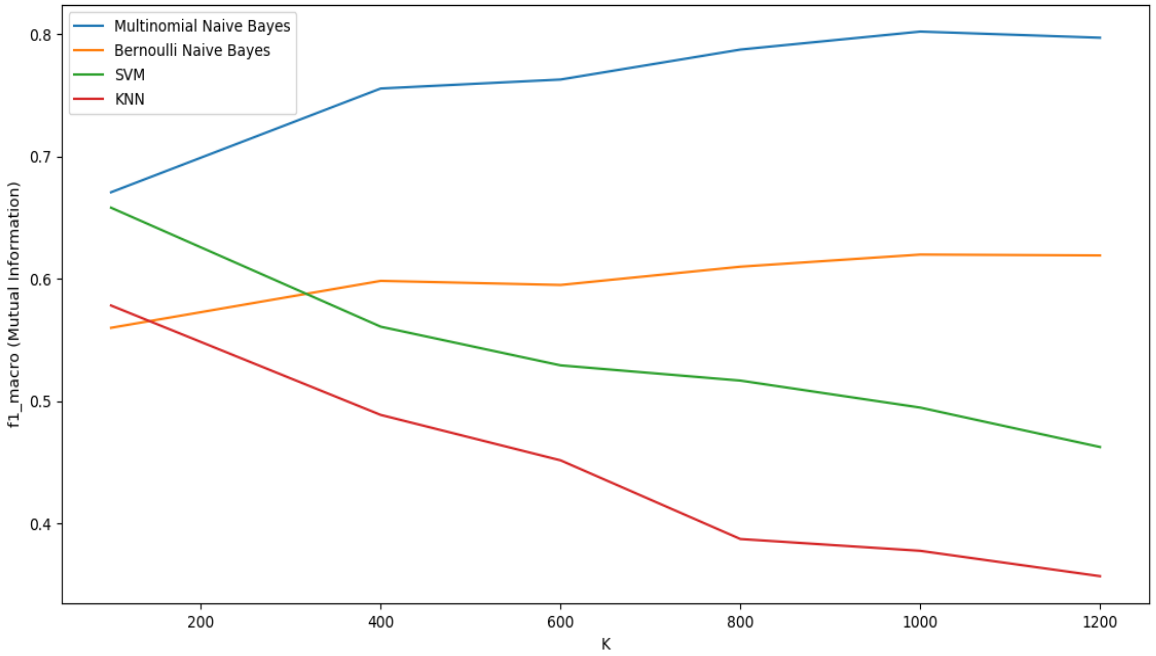
Chi- Sqaure plot for all 4 algorithms, Multinomial NM, Burnolli NB, KNN and SVM.



Multinomial NB performed the best among the 4 algorithms while SVM performed the worst.

Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019

Mutual Information plot for all the 4 algorithms, Multinomial NM, Burnolli NB, KNN and SVM.



Multinomial NB performed the best while KNN performed the worst.

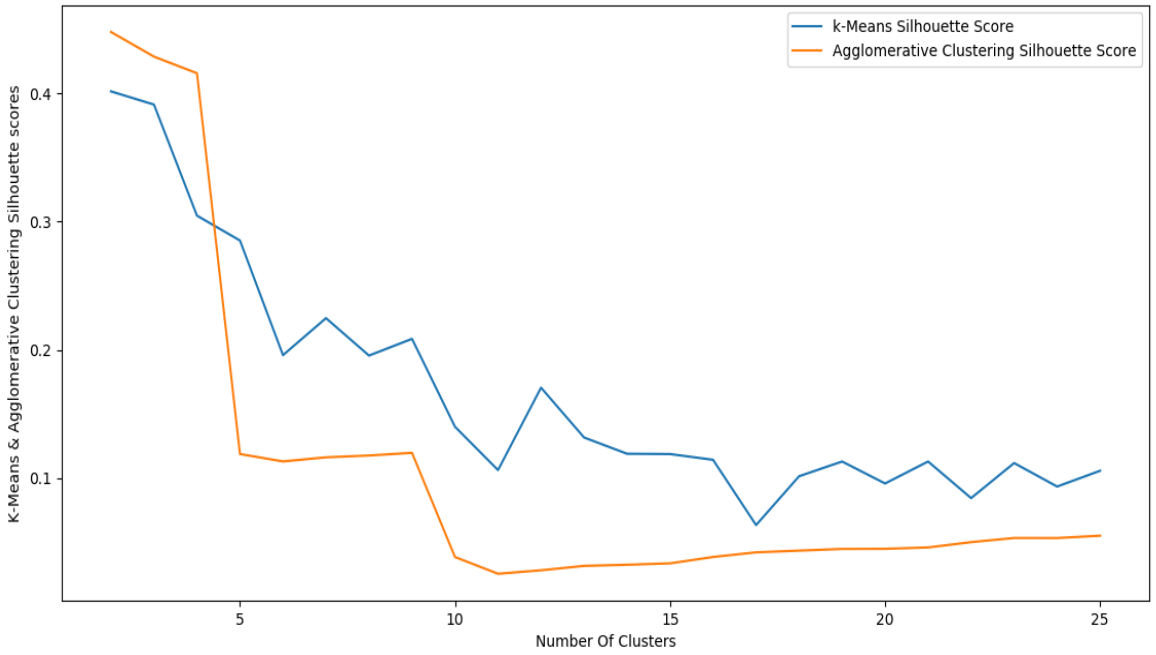
PART 4: Document Clustering

Implementation: In the document clustering part of the project we implemented k Means and hierarchical Clustering. Then we calculated silhouette Coefficient and normalized mutual information score using scikit. The algorithms that are tested are K Means and agglomerative clustering

Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019

Experimental results:

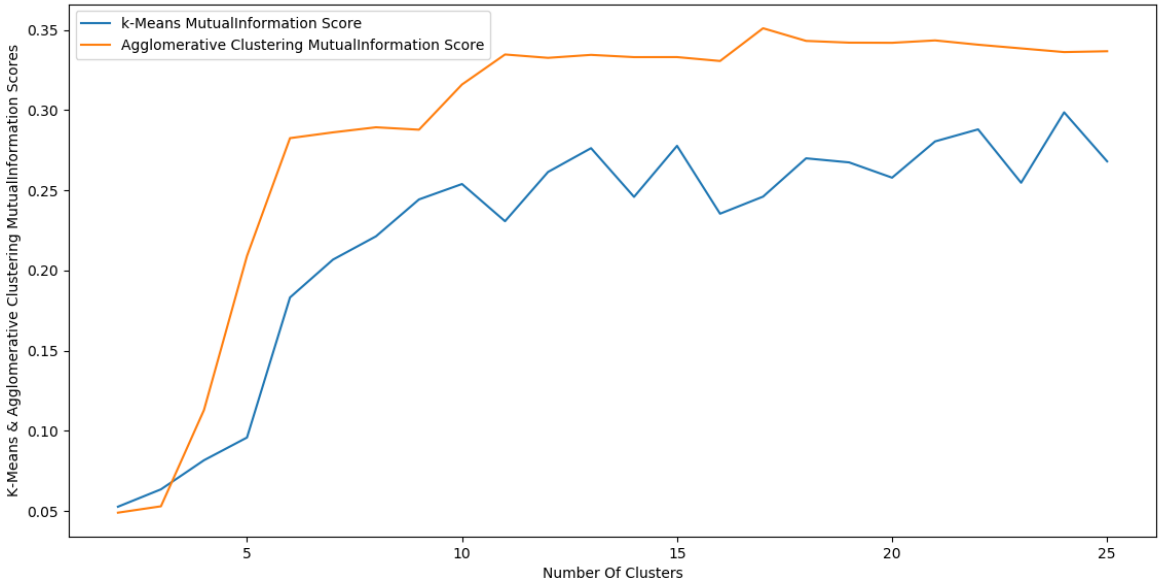
Plot 1 K Means and Agglomerative Clustering against the number of clusters



We observed that after certain number of clusters like 5 or so K Means Silhouette Score outperforms the Agglomerative Clustering Silhouette Score for a larger number of clusters, but when it gets too large it is hard to tell which one performs better,\.

Plot 2. KMeans Mutual Information & Agglomerative Mutual Information against number of clusters.

Team Members : Anthony Edward Drow, UID U00759458
& Syed Waqar Uddin, UID U00765829
Due Date April 21, 2019



Overall the scores of Agglomerative clustering is slightly better than the scores of K Means Mutual Information