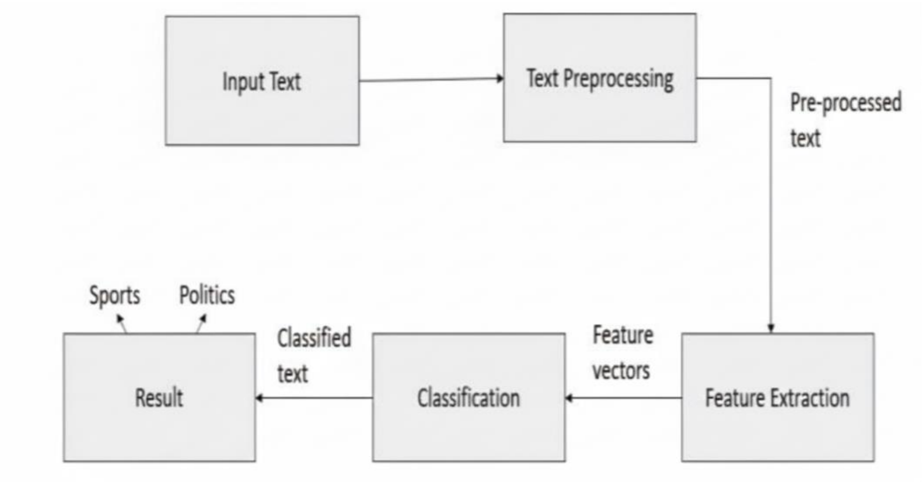
Name: Smithi Sureshan (M25MAC012)

Assignment 1: Problem 4

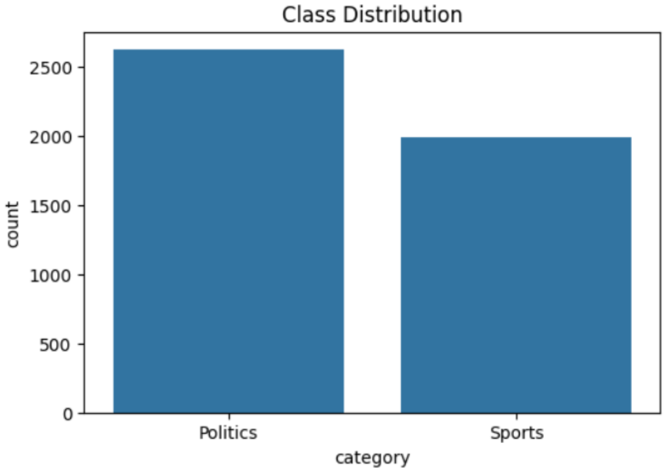
The aim of this problem is to design a classifier that reads text documents and classify it into Sports or Politics using TF-IDF for feature extraction and Machine Learning models for classification.



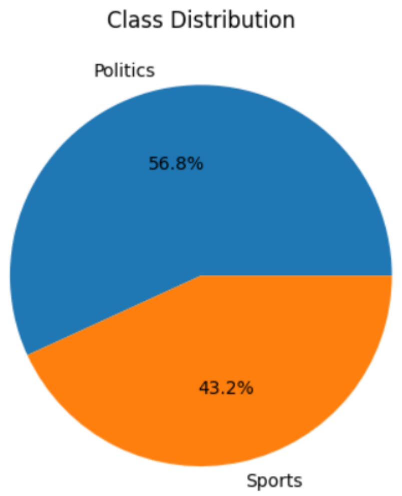
The workflow is shown in the above Fig. which illustrates the process of taking the input text, text preprocessing, feature extraction using TD-IDF algorithm and classification using Machine Learning models.

Data Collection

We used 20 Newsgroups corpus imported from fetch_20newsgroups and then split into training and test subsets. The dataset consists of 4618 documents with 2625 as Politics and 1993 as Sports. We have used two sports newsgroups and three politics newsgroups. Binary label makes it easy to classify so we labelled 0 for Sports and 1 for Politics.

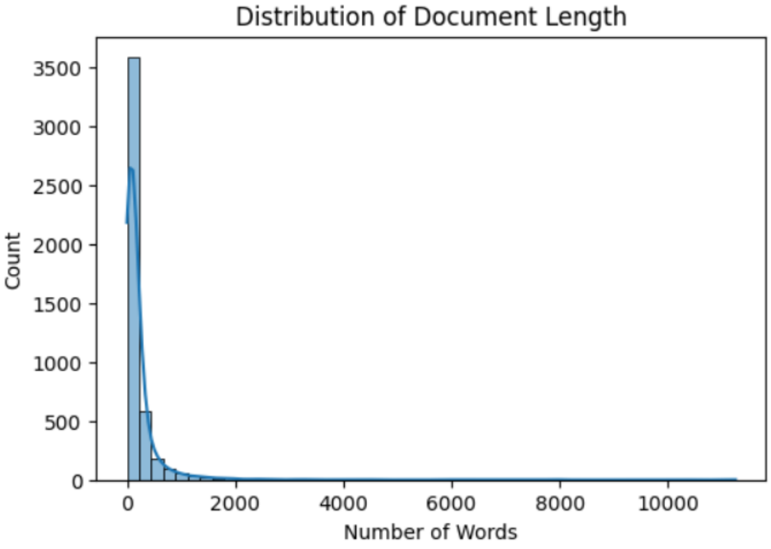


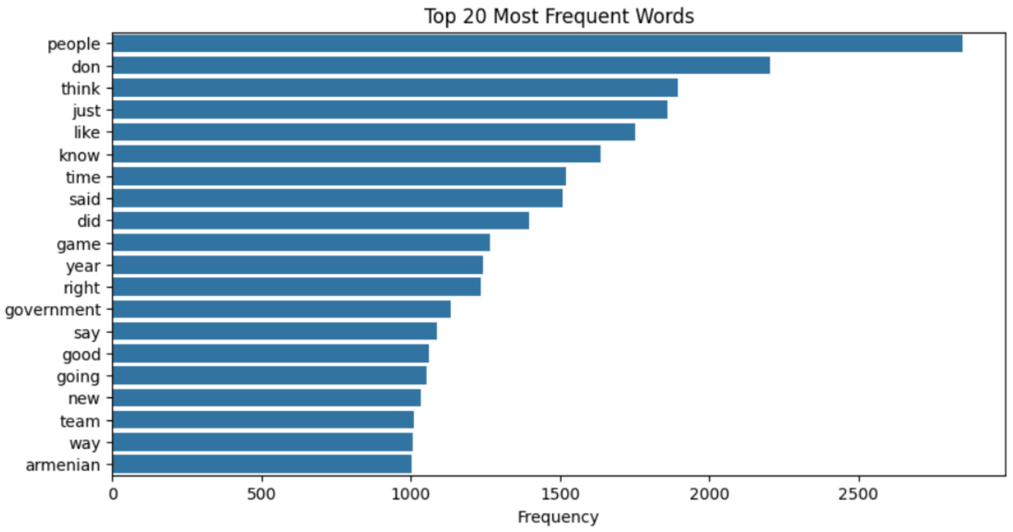
Dataset Description and Analysis



Measure	Value
Total Documents	4618
Average Length (Mean)	223.81 words
Standard Deviation	579.93
Minimum Length	0 words
First Quartile (25%)	43 words
Median Length	96 words
Third Quartile (75%)	204 words
Maximum Length	11,251 words

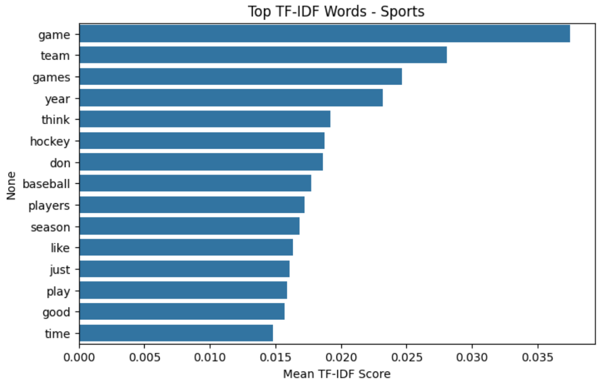
The dataset consists of 4618 documents with each document containing approximately 224 words. The dataset has high variability

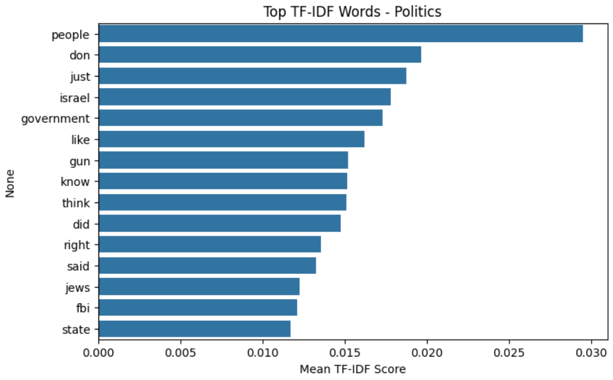




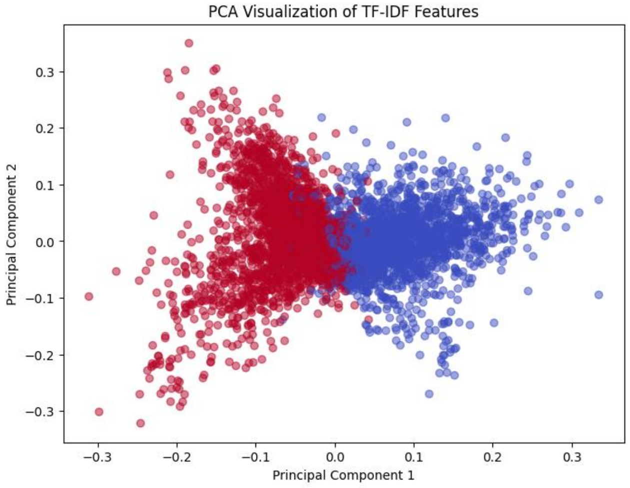
The bar chart shows 20 most frequent words in the dataset. We have done train test split with 70 percent for training and 30 percent for testing. Sports documents contain texts like team names, game scores and Politics documents contain texts like countries, parties, politicians. TF-IDF can easily discriminate these terms and assign higher weights to frequent terms.

Feature Extraction





The above plots shows the highest TF-IDF scoring words for both classes.



PCA plot shows that TF-IDF features effectively separate the two classes. Machine learning models need to be mapped with numerical vector as it cannot work with raw data. For feature extraction we have used TF-IDF. TF-IDF looks for frequency of words in documents and down weights words that occur in multiple documents assuming it is of less importance. We have targeted limit of 4000 frequent terms to speed up training. Then we got matrix in which row corresponds to document and column corresponds to words.

Naive Bayes

It generates document for each class. In this each document is assigned weights. It contains highly sparse data which means most entries are zero and Naive Bayes is computationally highly efficient. If most of words have less number of counts then they contribute very less.

Linear Support Vector Machine (Linear SVM)

Linear SVM has hyperplane which separates two classes with margin. The model tries best to keep hyperplane far for generalisation on new data. Based on topic text can be separated by using hyperplane. Linear SVM can help in getting decision function scores or class labels. But it cannot extract non-linear decision boundaries.

Random Forest

In random forest many decision trees are combined to improve generalisation. In each iteration of splitting it decides which feature and threshold to split. Majority vote from trees is final output. It does randomization thus reducing overfitting. Decision trees keeps splitting on single features in which most of split are not at informative and trees keep becoming more complex. Random Forest performance is comparatively lesser than Linear SVM and Naive Bayes. It is computationally expensive as it mostly have sparse features.

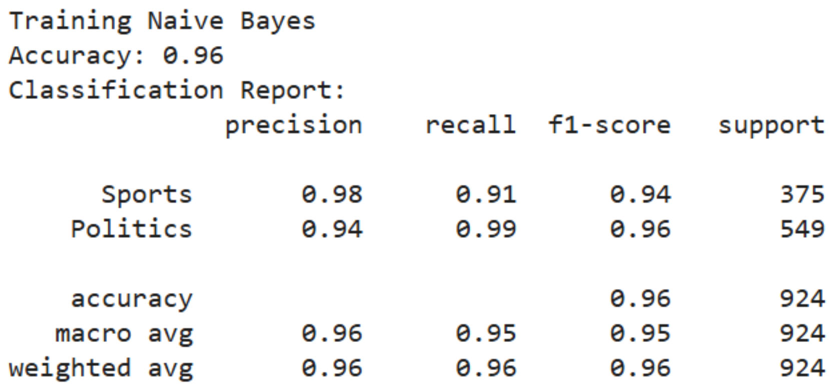
Quantitative Comparison and Results

Model	**Accuracy**	**Precision(Macro)**	**Recall(Macro)**	**F1-Score(Macro)**
Naive Bayes	0.96	0.96	0.95	0.95
Linear SVM	0.95	0.95	0.94	0.94
Random Forest	0.93	0.92	0.92	0.92

Model	**Accuracy**	**Precision (Macro)**	**Recall (Macro)**	**F1-Score (Macro)**
Naive Bayes	0.96	0.96	0.95	0.95
Linear SVM	0.95	0.95	0.94	0.94
Random Forest	0.92	0.92	0.92	0.92

In text classification linear models like SVM outperform Random Forest. It is not performing well because of high dimensions.

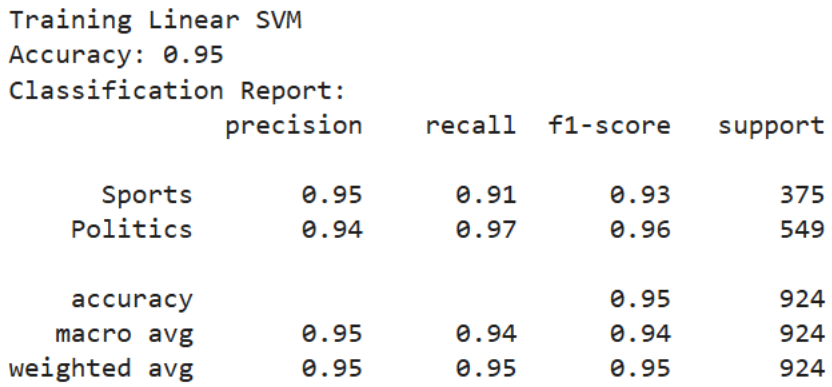
Naive Bayes



```
Training Naive Bayes
Accuracy: 0.96
Classification Report:
              precision    recall  f1-score   support

      Sports       0.98      0.91      0.94       375
    Politics       0.94      0.99      0.96       549

    accuracy                           0.96       924
   macro avg       0.96      0.95      0.95       924
weighted avg       0.96      0.96      0.96       924
```

In this for politics recall, f1-score and support is high than sports. But sports has high precision.

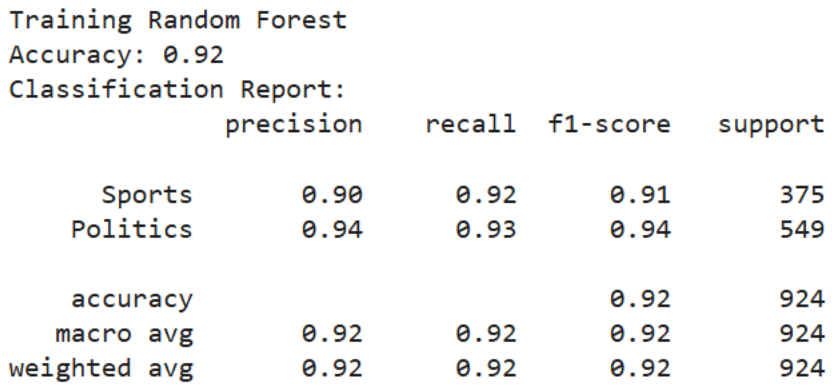
Linear Support Vector Machine (Linear SVM)



```
Training Linear SVM
Accuracy: 0.95
Classification Report:
              precision    recall  f1-score   support

      Sports       0.95      0.91      0.93       375
    Politics       0.94      0.97      0.96       549

    accuracy                           0.95       924
   macro avg       0.95      0.94      0.94       924
weighted avg       0.95      0.95      0.95       924
```

In Linear SVM for sports precision is slightly more than politics.

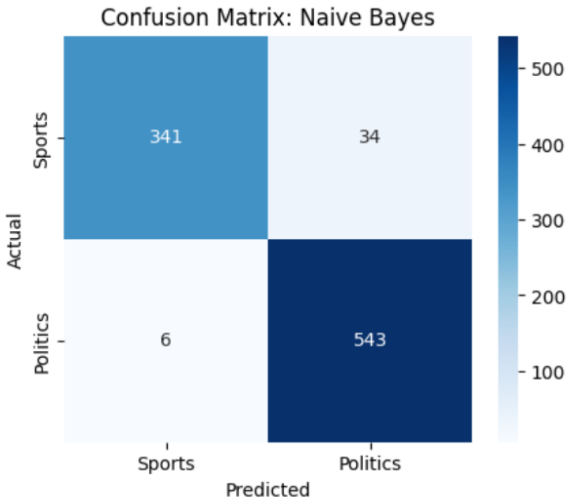
Random Forest

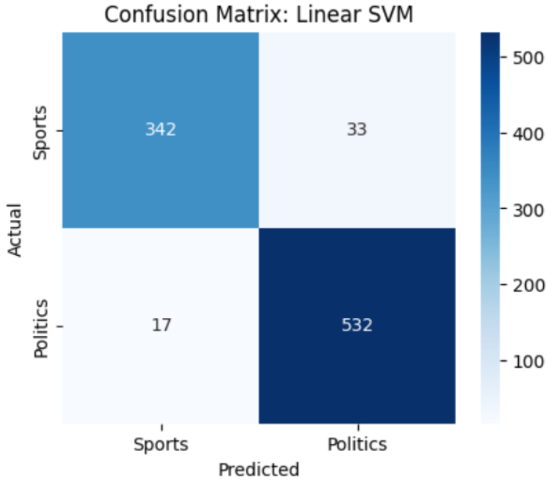


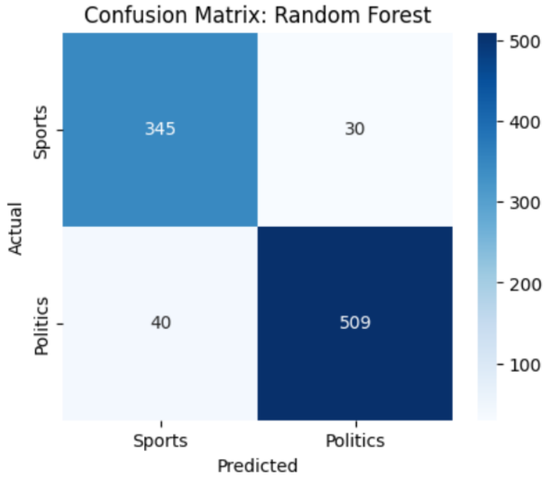
```
Training Random Forest
Accuracy: 0.92
Classification Report:
              precision    recall  f1-score   support

      Sports       0.90      0.92      0.91       375
    Politics       0.94      0.93      0.94       549

    accuracy                           0.92       924
   macro avg       0.92      0.92      0.92       924
weighted avg       0.92      0.92      0.92       924
```

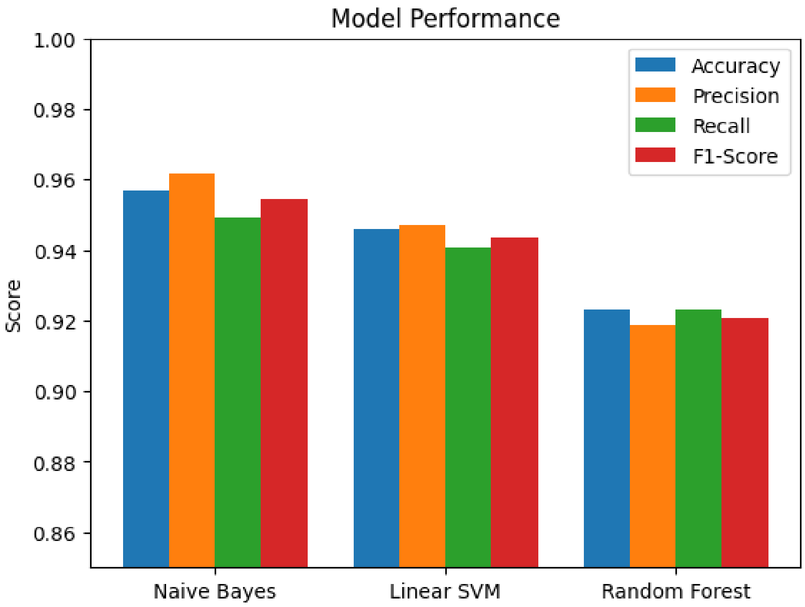
In this politics is having high value for precision, recall, f1-score, support.

Confusion Matrices:


Confusion Matrix: Naive Bayes
Actual
Sports
Politics
341
34
6
543
Predicted
Sports
Politics
500
400
300
200
100


Confusion Matrix: Linear SVM
Actual
Sports
Politics
342
33
17
532
Predicted
Sports
Politics
500
400
300
200
100


Confusion Matrix: Random Forest
Actual
Sports
Politics
345
30
40
509
Predicted
Sports
Politics
500
400
300
200
100



Limitations of system:

TF-IDF ignores word order and context hence it cannot capture meaning or semantic relationships. It also creates very large sparse vectors, which reduces efficiency and increases computational cost. The system is currently handling only two categories and may require retraining for multi class problems .