Flagging terror-related tweets on

Aditi Walia 014516060
Hardik Kumar 014521039

INTRODUCTION

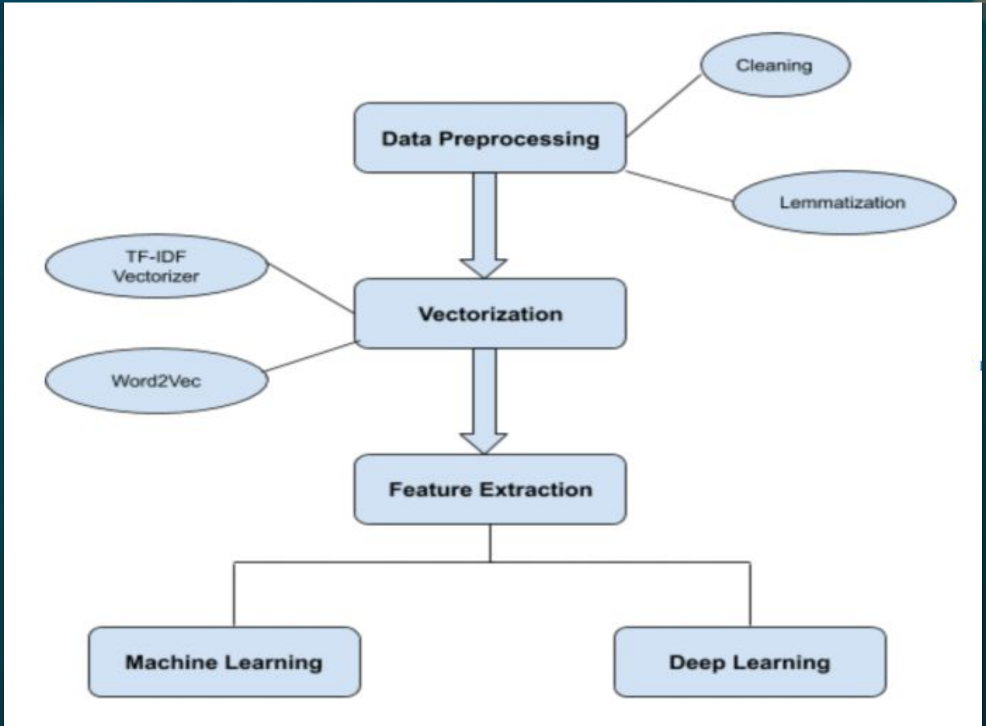
- Social media is one of the main means of communication
- Terrorist organisations like ISIS use Twitter to spread their propaganda
- Removing such accounts is a solution but it can’t be done manually
- Machine Learning can help in flagging tweets posted by accounts if they are radical

DATASET

- Data set is scraped from Twitter through GetOldTweets3 and some of it is self-generated
- Against ISIS tweets were found using hashtags like #NoToISIS. ~800 tweets
- Pro ISIS tweets were filtered out from a dataset found online. ~800 tweets
- Random tweets contain random non-radical content ~4000 tweets

Against ISIS	HT_Felani	I stand with real Iranians, I stand with IR IRAN, I condemn MasihAlinejad for act of terrorism and promoting violence and terrorist activities. #NoToViolence #NoToChaos #NoToTerrorism #NoToPropaganda #NoToFakeNews #NoToGuns #NoToISIS #NoToCyberTerrorism
Pro ISIS	abubakerdimshqi	from the Heart love and respect for all #MUJAHDIN in BILAD #ALSHAM Brothers & Sisters you are our hope to lead us to right way #IS
Random	bado0ouri	Should I sleep and skip my classes today

PROJECT ORGANIZATION

- 

DATA PREPROCESSING

An important phase; clean the data in order to extract the most useful information out of it

- Remove hyperlinks and "RT" (retweets)
- Remove punctuations
- Remove stop words
- Stemming / Lemmatization

LEMMATIZING OVER STEMMING

- Reduces inflections or variant forms to base form
- Offers better precision than stemming, because of meaningful chopping of affixes.

Stemming:

```
print(ps.stem('meanness'))
print(ps.stem('meaning'))
```

```
mean
mean
```

Lemmatization:

```
print(wn.lemmatize('meanness'))
print(wn.lemmatize('meaning'))
```

```
meanness
meaning
```

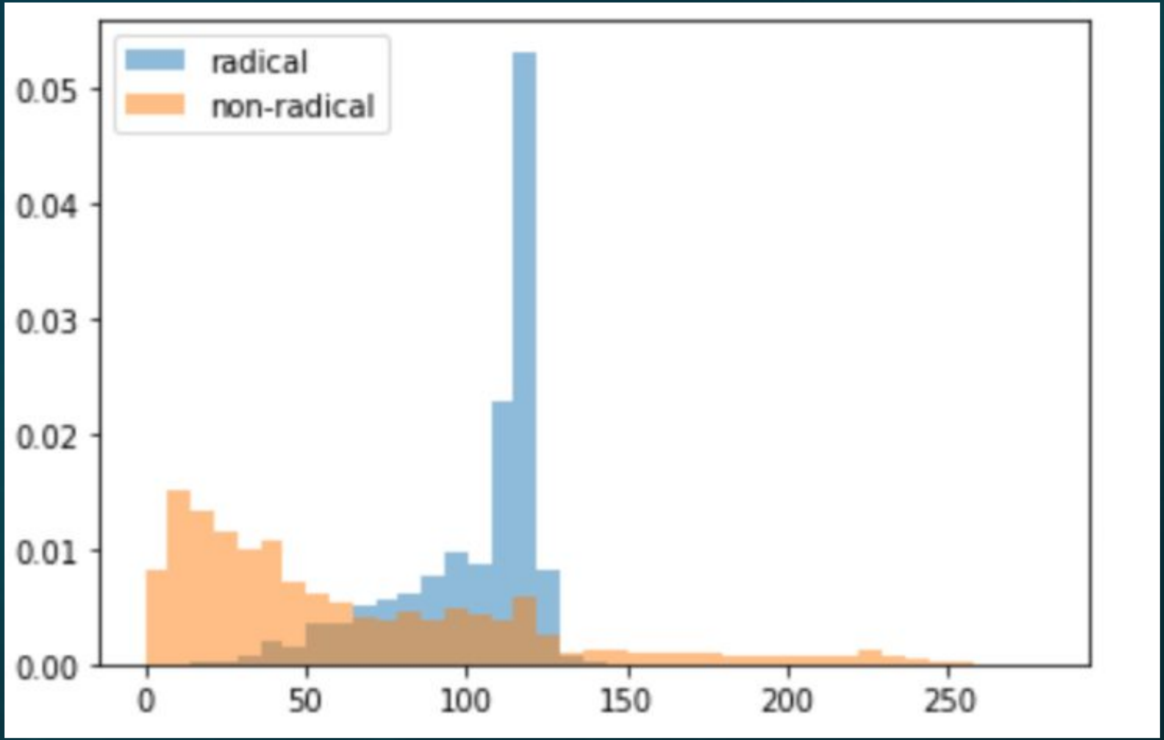
VECTORIZATION

- Word2Vec
- TF-IDF

FEATURE EXTRACTION

- Tweet length
- Presence of radical keywords
- Word embeddings for Word2Vec vectorization
- TF-IDF feature vectors for TF-IDF vectorization

TWEET LENGTH VARIATION IN DATASET



RADICAL KEYWORDS

```
#radical keyworda
radical_keywords = ['KUFFUR', 'IS', 'ISLAMIC STATE', '#ILOVEISIS', 'KUFFARS', 'mujahiddeen',
                    'kuffur', 'KUFFAR', 'kuffars', 'kuffar', 'kafir', 'MUJAHIDEEN', 'KUFAR', 'KAFIR',
                    'KUFR', 'mujahideen', '#IS', 'kufar', 'kufr', 'mujahid', 'JIHAD', 'jihad', 'MUJAHID',
                    'MUJAHIDDEEN', '#ISIS', 'Islamic State', '#ILoveISIS', 'ISIL', 'allah', 'Allah', 'Assad', 'assad',
                    'YPG', '#AleppoIsBurning', 'Aleppo', 'martydom', 'Martyrdom']

def radicalWordPresence(text):
    for key in radical_keywords:
        if key in text:
            return 1
    return 0
```

Word2Vec Embedding

- Embedding words into fixed size vectors
- Common bag of Words
- Skip-n-gram

Word2Vec EMBEDDING

```
In [12]: model.most_similar('kuffar')

Out[12]: [('kafir', 0.6600816249847412),
          ('Kuffar', 0.6480599045753479),
          ('infidels', 0.6436571478843689),
          ('unbelievers', 0.6348938941955566),
          ('polytheists', 0.6023821830749512),
          ('Kafirs', 0.5989149808883667),
          ('Allah', 0.5932642817497253),
          ('infidel', 0.5747469663619995),
          ('jihad', 0.5737059712409973),
          ('kafirs', 0.5729705691337585)]
```

Machine Learning Algorithms
on
Word2Vec

Random Forest

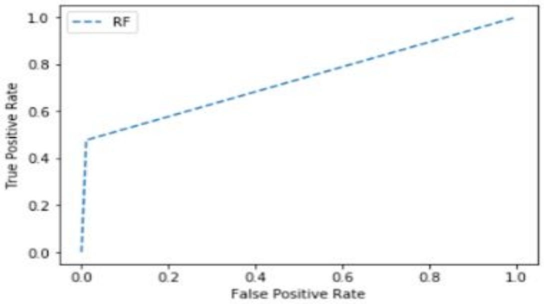
Grid Search Results with 5-Fold cross validation on Word2Vec

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n_estimators	params	split0_test_score	split1_test_score	spl
2	7.117576	0.229126	0.036578	0.004184	None	100	{'max_depth': None, 'n_estimators': 100}	0.911092	0.901408	

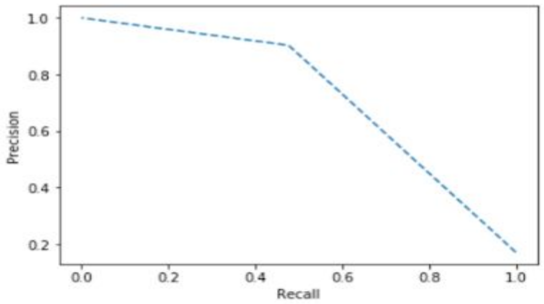
Results

```
Fit Time: 0.964 / Pred Time: 0.111 -------- Precision: 0.903 / Recall: 0.477 / Accuracy: 0.901
Confusion matrix
 [[ 93 102]
 [ 10 931]]
```

ROC Curve



Precision Recall Curve




```
PR AUC: 0.735
```

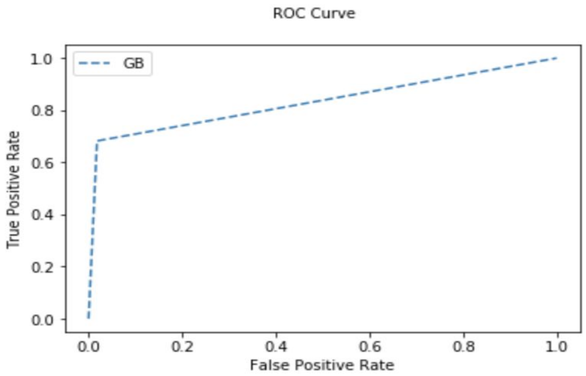
Gradient Boosting

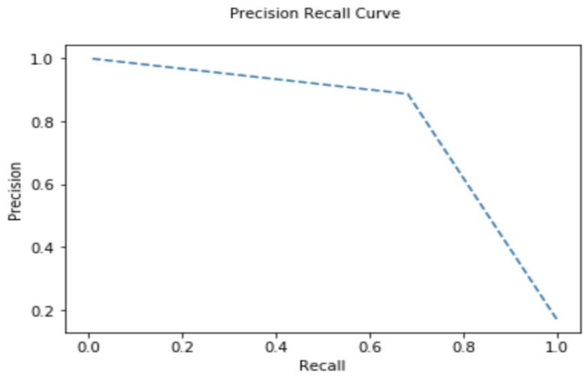
Grid Search Results with 5-Fold cross validation on Word2Vec

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n_estimators	params	split0_test_score	split1_test_score	split
0	129.686026	0.194281	0.021602	0.001401	7	150	{'max_depth': 7, 'n_estimators': 150}	0.931338	0.926937	

Results

```
Fit Time: 106.315 / Pred Time: 0.02 -------- Precision: 0.887 / Recall: 0.682 / Accuracy: 0.93
Confusion matrix
 [[133  62]
 [ 17 924]]
```






```
PR AUC: 0.812
```

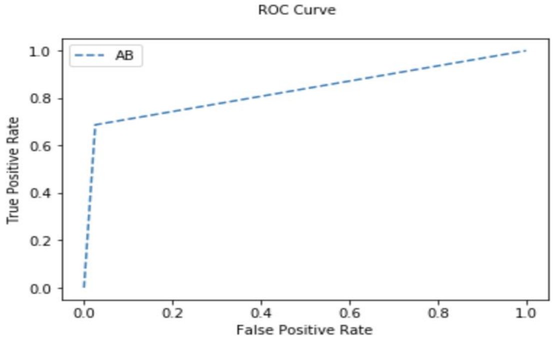
AdaBoost

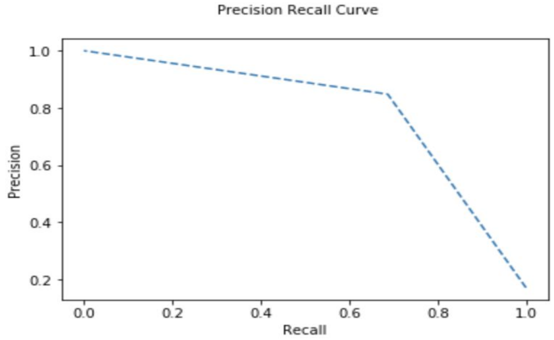
Grid Search Results with 5-Fold cross validation on Word2Vec

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_learning_rate	param_n_estimators	params	split0_test_score	split1_test_score	s
8	151.377902	7.646242	0.146859	0.014226	1	200	{'learning_rate': 1, 'n_estimators': 200}	0.919894	0.902289	

Results

```
Fit Time: 23.471 / Pred Time: 0.11 --------- Precision: 0.848 / Recall: 0.687 / Accuracy: 0.925
Confusion matrix
 [[134  61]
 [ 24 917]]
```






```
PR AUC: 0.794
```

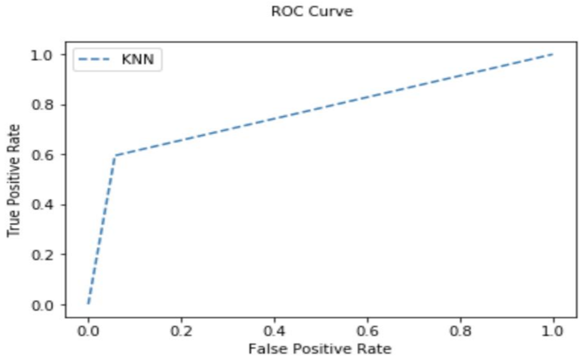
K-Nearest Neighbours

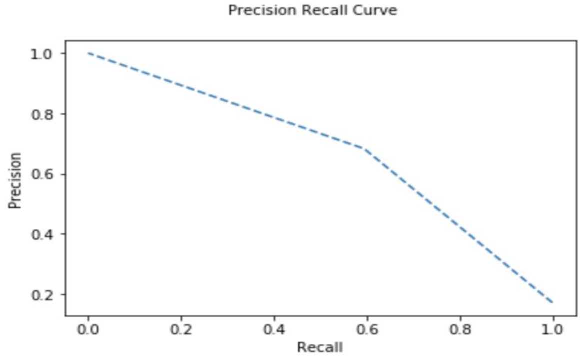
Grid Search Results with 5-Fold cross validation on Word2Vec

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_metric	param_n_neighbors	param_weights	params	split0_test_score	split1_test
15	0.097535	0.006593	0.088721	0.009170	manhattan	19	distance	{'metric': 'manhattan', 'n_neighbors': 19, 'weights': 'distance'}	0.880282	0.

Results

```
Fit Time: 0.11 / Pred Time: 0.126 -------- Precision: 0.682 / Recall: 0.595 / Accuracy: 0.883
Confusion matrix
 [[116  79]
 [ 54 887]]
```






```
PR AUC: 0.673
```

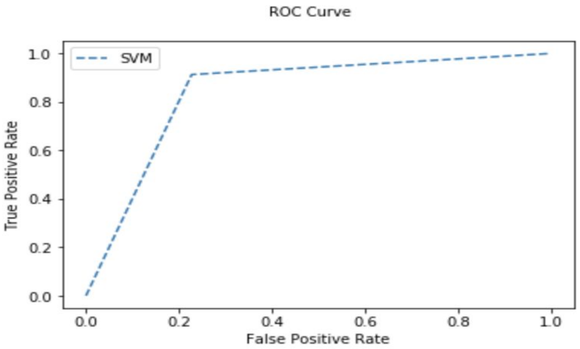
Support Vector Machine (SVM)

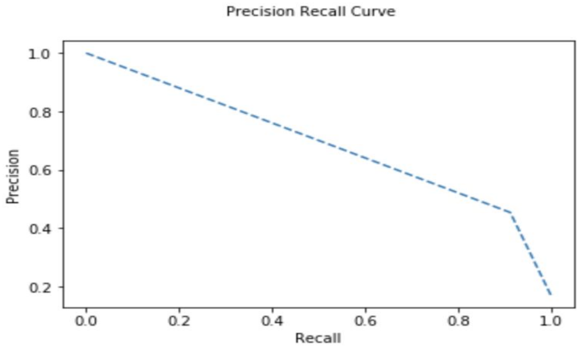
Grid Search Results with 5-Fold cross validation on Word2Vec

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_kernel	params	split0_test_score	split1_test_score	split2_test_score	split3_t
8	4.519594	0.168714	0.631069	0.014725	1000	rbf	{'C': 1000, 'kernel': 'rbf'}	0.872359	0.857394	0.870599	

Results

```
Fit Time: 3.986 / Pred Time: 0.671 --------- Precision: 0.454 / Recall: 0.913 / Accuracy: 0.797
Confusion matrix
 [[178  17]
 [214 727]]
```



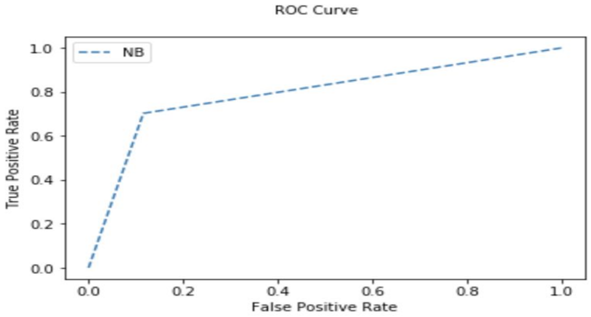


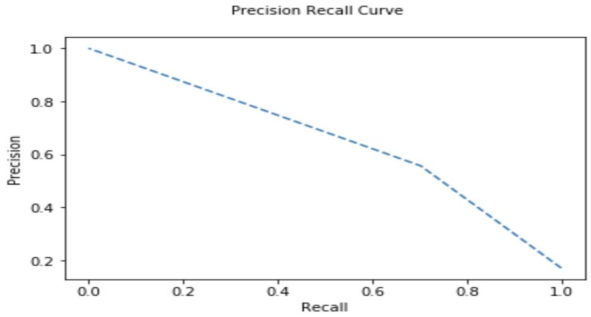

```
PR AUC: 0.691
```

Naive Bayes

Results

```
Fit Time: 0.041 / Pred Time: 0.018 -------- Precision: 0.557 / Recall: 0.703 / Accuracy: 0.853
Confusion matrix
 [[137  58]
 [109 832]]
```






```
PR AUC: 0.655
```

Summary of Results

Classifier	Recall	Precision	Accuracy
Random Forest	0.477	0.903	0.901
AdaBoost	0.687	0.848	0.925
Gradient Boosting	0.682	0.887	0.93
KNN	0.595	0.682	0.883
SVM	0.913	0.454	0.797
Naive Bayes	0.703	0.557	0.853

TF-IDF (Term Frequency Inverse Document Frequency)

- Importance of word in document
- Weight assigned between 0 and 1 to each word according to occurrence and importance

$$W_{ij} = tf_{ij} * \log (N/df_i)$$

- Rarer words occurring in a sentence might be more important to the context of the sentence

Machine Learning Algorithms
on
TF-IDF

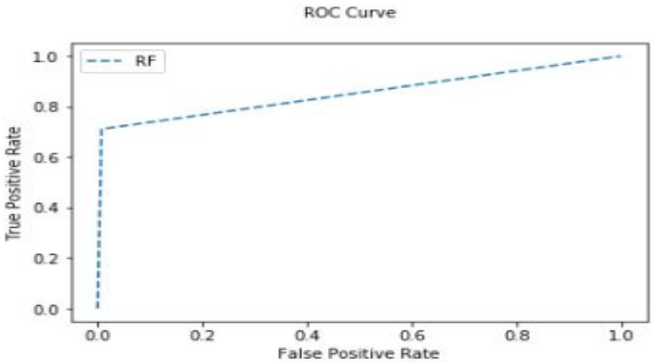
Random Forest

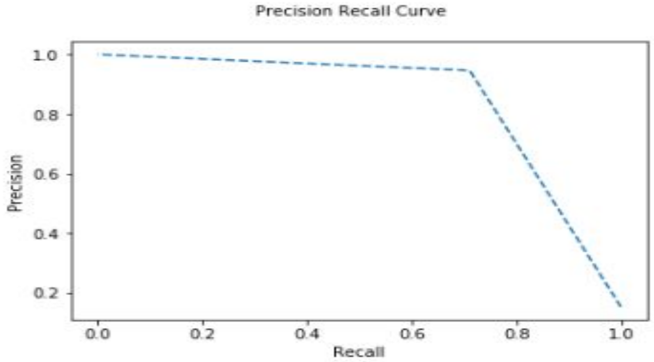
Grid Search Results with 5-Fold cross validation on TF-IDF

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n_estimators	params	split0_test_score	split1_test_score	spl
2	38.610945	1.963792	0.424268	0.085037	None	100	{'max_depth': None, 'n_estimators': 100}	0.943662	0.944542	

Results

```
Fit Time: 2.916 / Pred Time: 0.201 -------- Precision: 0.946 / Recall: 0.711 / Accuracy: 0.95
Confusion matrix
 [[123  50]
 [  7 956]]
```






```
PR AUC: 0.851
```

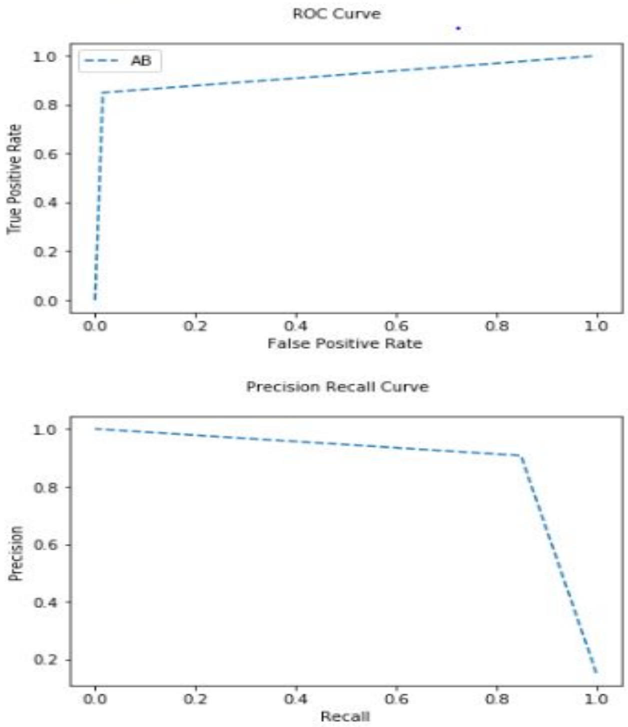
AdaBoost

Grid Search Results with 5-Fold cross validation on TF-IDF

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_learning_rate	param_n_estimators	params	split0_test_score	split1_test_score	s
4	300.254349	4.757440	11.577173	0.351197	0.5	100	{'learning_rate': 0.5, 'n_estimators': 100}	0.961268	0.950704	

Results

```
Fit Time: 53.13 / Pred Time: 3.888 -------- Precision: 0.907 / Recall: 0.85 / Accuracy: 0.964
Confusion matrix
 [[147  26]
 [ 15 948]]
```




```
PR AUC: 0.890
```

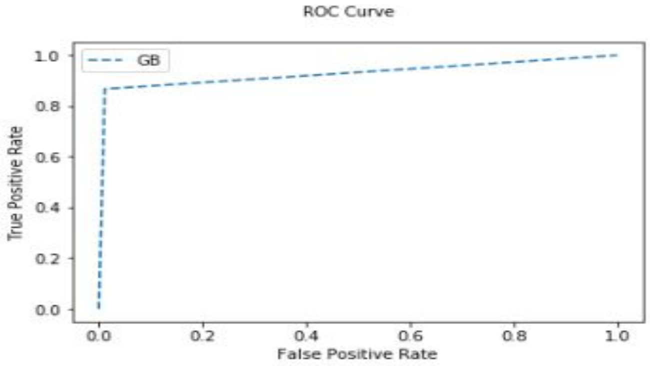
Gradient Boosting

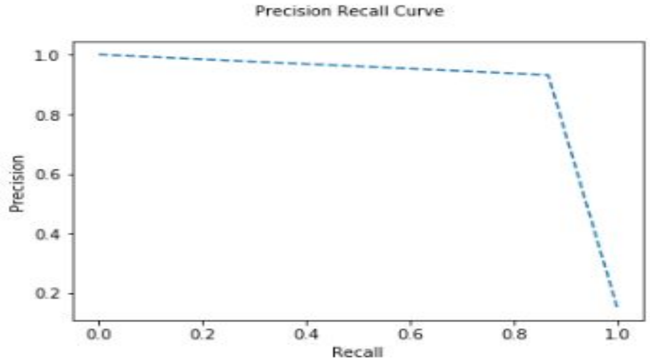
Grid Search Results with 5-Fold cross validation on TF-IDF

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n_estimators	params	split0_test_score	split1_test_score	split
0	210.895003	1.266217	0.393229	0.10199	7	150	{'max_depth': 7, 'n_estimators': 150}	0.960387	0.957746	

Results

```
Fit Time: 348.551 / Pred Time: 0.208 -------- Precision: 0.932 / Recall: 0.867 / Accuracy: 0.97
Confusion matrix
 [[150  23]
 [ 11 952]]
```






```
PR AUC: 0.909
```

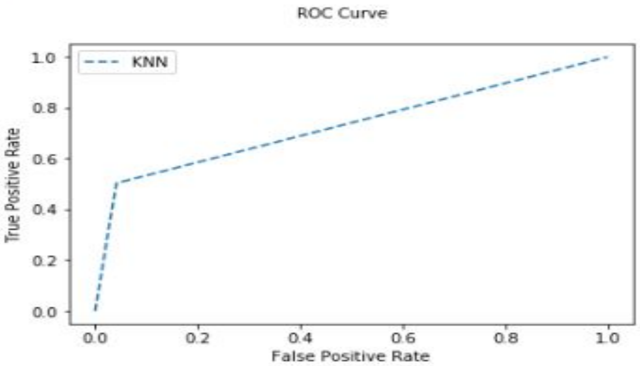
K-Nearest Neighbours

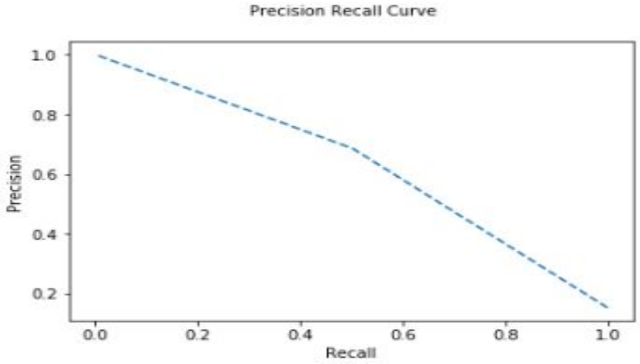
Grid Search Results with 5-Fold cross validation on TF-IDF

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_metric	param_n_neighbors	param_weights	params	split0_test_score	split1_test
5	11.454158	1.269725	10.475139	0.636509	euclidean	11	distance	{'metric': 'euclidean', 'n_neighbors': 11, 'weights': 'distance'}	0.889085	0.

Results

```
Fit Time: 3.033 / Pred Time: 4.522 -------- Precision: 0.685 / Recall: 0.503 / Accuracy: 0.889
Confusion matrix
 [[ 87  86]
 [ 40 923]]
```






```
PR AUC: 0.632
```

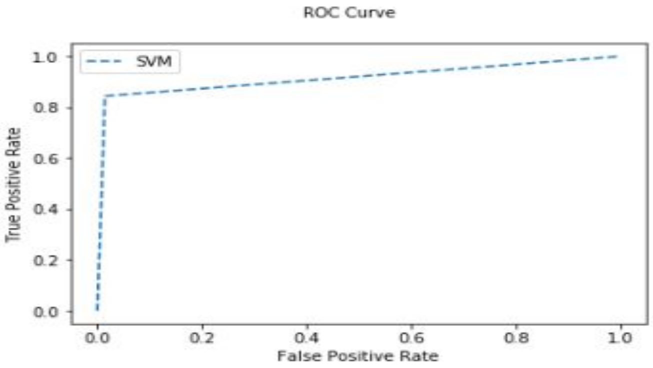
Support Vector Machine (SVM)

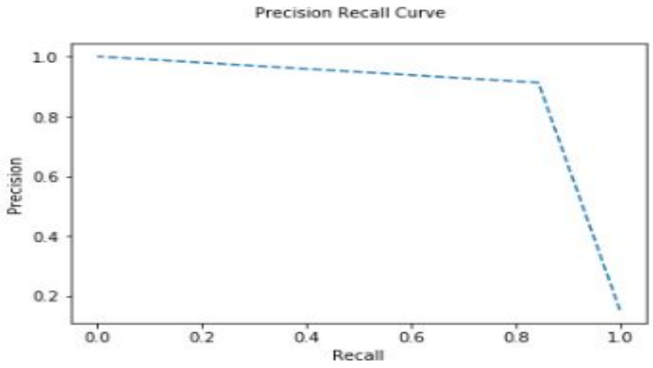
Grid Search Results with 5-Fold cross validation on TF-IDF

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_kernel	params	split0_test_score	split1_test_score	split2_test_score	split3_t
0	787.271341	121.125993	32.54786	1.464494	100	linear	{'C': 100, 'kernel': 'linear'}	0.955106	0.949824	0.959507	

Results

```
Fit Time: 365.59 / Pred Time: 26.645 -------- Precision: 0.912 / Recall: 0.844 / Accuracy: 0.964
Confusion matrix
 [[146  27]
 [ 14 949]]
```



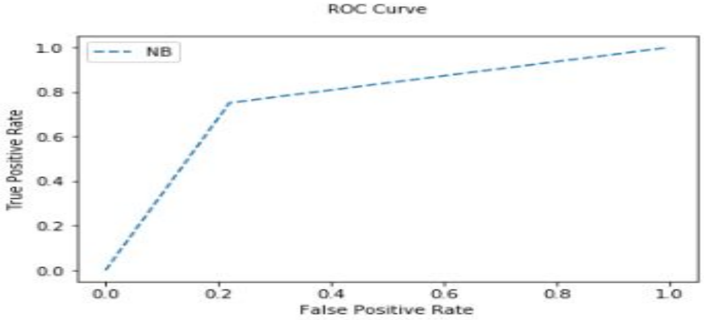


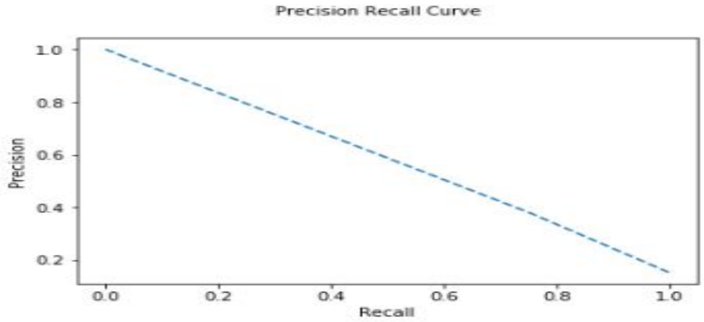

```
PR AUC: 0.890
```

Naive Bayes

Results

```
Fit Time: 1.862 / Pred Time: 0.404 -------- Precision: 0.38 / Recall: 0.751 / Accuracy: 0.776
Confusion matrix
 [[130  43]
 [212 751]]
```






```
PR AUC: 0.585
```

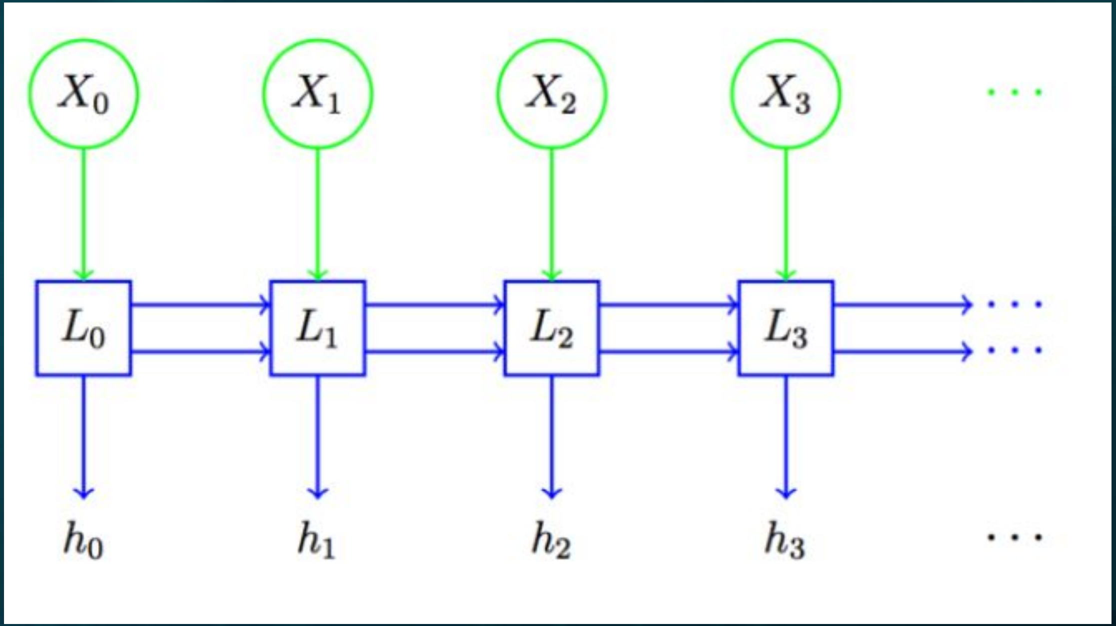
Summary of Results

Classifier	Recall	Precision	Accuracy
Random Forest	0.711	0.946	0.950
AdaBoost	0.850	0.907	0.964
Gradient Boosting	0.867	0.932	0.97
KNN	0.503	0.685	0.889
SVM	0.844	0.912	0.964
Naive Bayes	0.751	0.38	0.776

Deep Learning

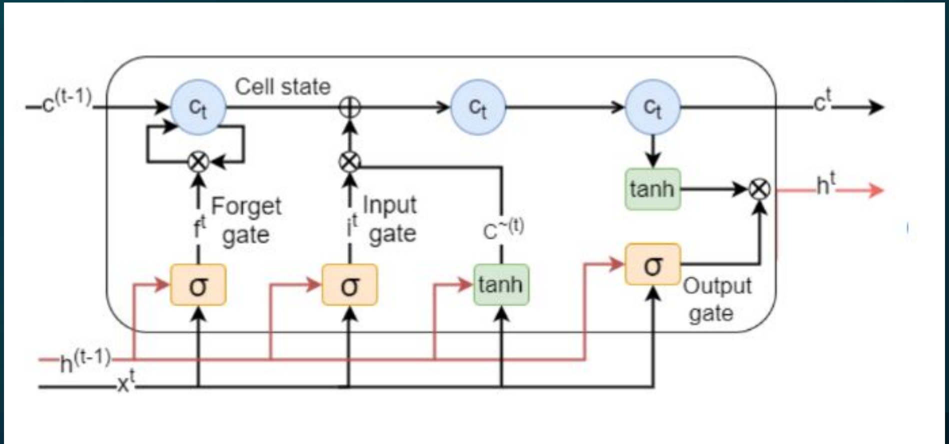
Long Short Term Memory (LSTM)

Grid Search Results with 5-Fold cross validation on TF-IDF



Advantages of LSTM

- Mitigates effect of vanishing, exploding, and unstable gradients
- Achieves this through the use of cell state and gates
- Cell state acts as store of long term info, and network decides what to “remember” and what to “forget”



Results on Word2Vec

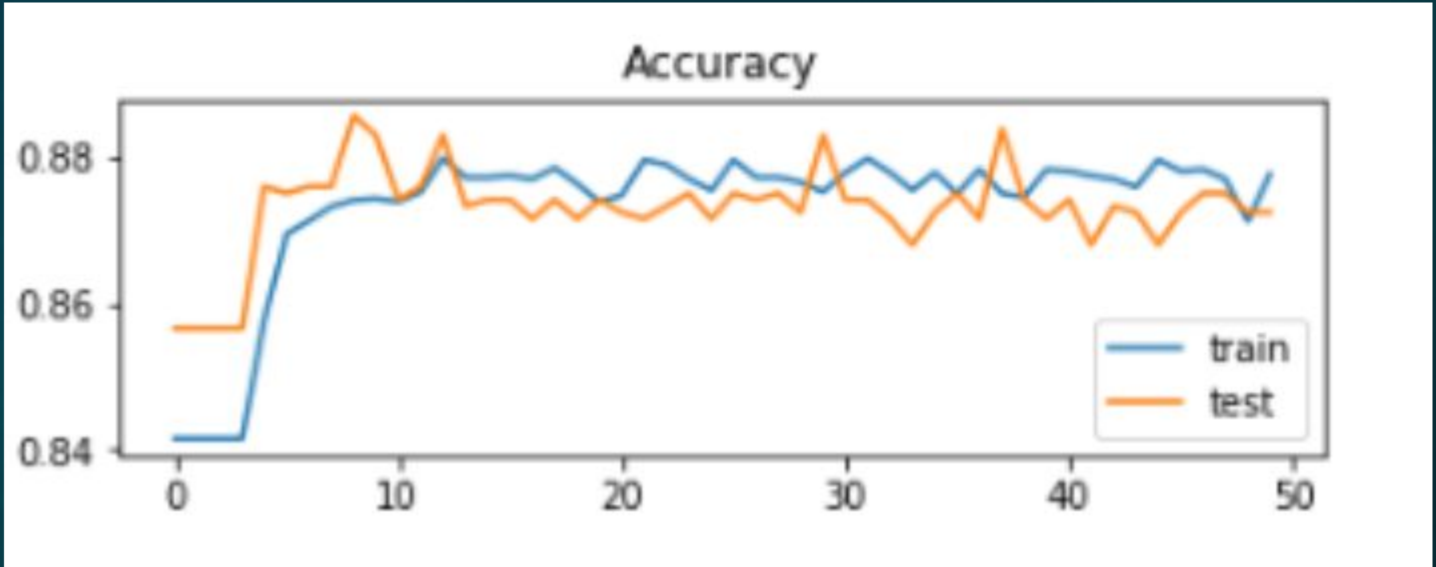
- Used three LSTM layers in the architecture, for 50 epochs
- Focussing on the recall due to the imbalance in data

Accuracy: 0.872359

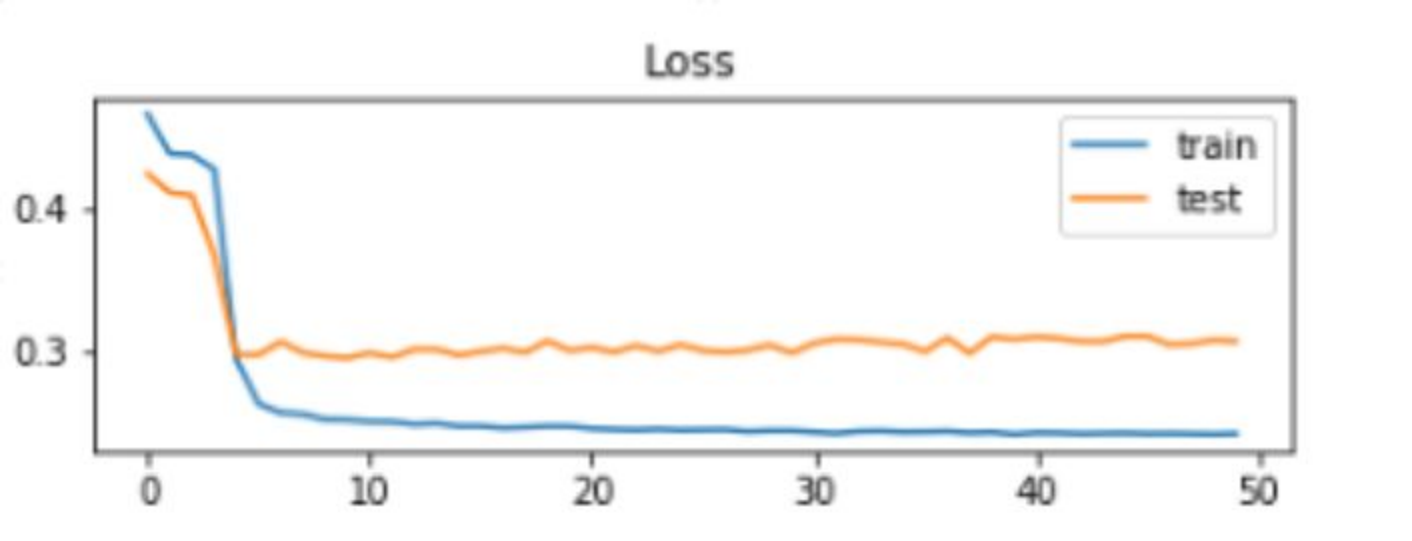
Precision: 0.554217

Recall: 0.564417

Accuracy Graph



Loss Graph



Results on TFIDF

- Used LSTM layer in the architecture, for 10 epochs
- Focussing on the recall due to the imbalance in data

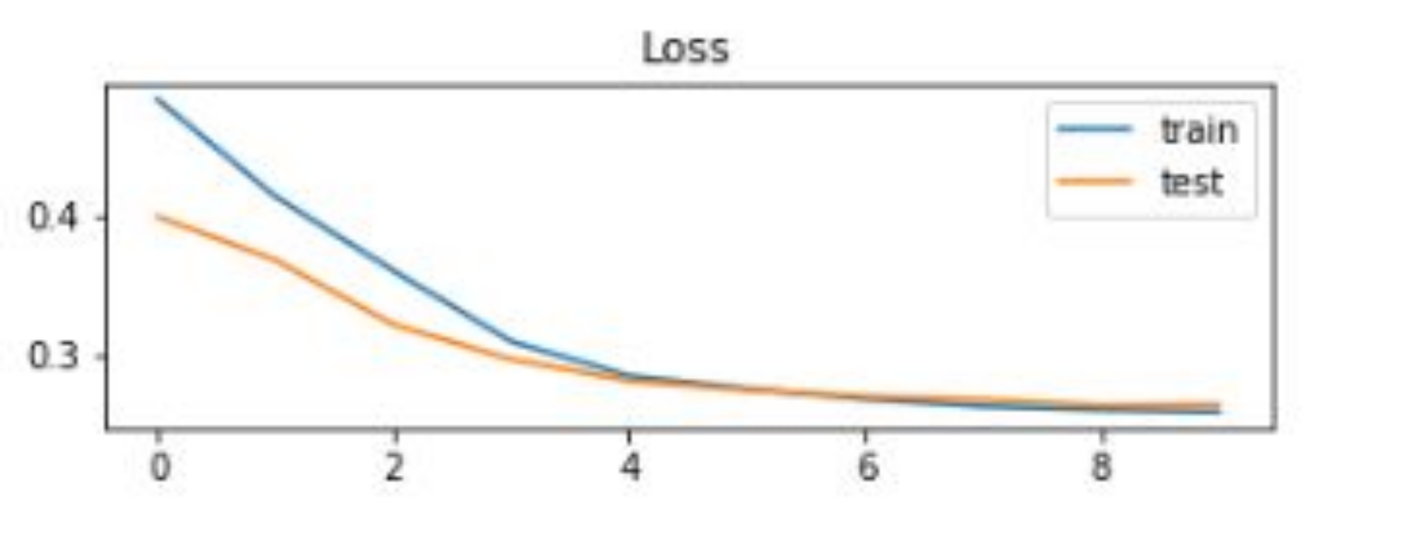
Accuracy: 0.882042

Precision: 0.595890

Recall: 0.537037

Accuracy Graph

Loss Graph



Conclusions

- For Word2Vec features, **SVM** works the best with a recall of ***0.913***
- For TF-IDF features, the boosting algorithms Gradient Boosting and AdaBoost perform the best with recalls of ***0.867*** and ***0.850*** respectively. SVM comes close with recall of ***0.844***
- SVM performs well because data is linearly separable and because of the abundance of features in our data
- Boosting algorithms performed better due to their iteratively greedy approach of boosting the weak classifiers
- With larger and more diverse dataset, and implementation of deeper architectural-models, we believe that he results might be more accurate
- Also, we used Bag of Words approach in Word2Vec. Using Skip-n-grams might give better results considering data set considers some rarely used words.