Pattern Recognition and Machine Learning Major Project: 13

Group Members:
Atharva Ganesh Pade (B21EE014)
Saksham Jain (B21EE059)
Uppala Giridhar (B21EE072)

The project objective is:

The objective of this project is to develop a machine learning model that can predict the probability of six types of toxicity (toxic, severe_toxic, obscene, threat, insult, and identity_hate) for each comment in a large dataset of Wikipedia comments. The objective is to develop a model that can precisely predict the likelihood of each type of toxicity for a given comment after human raters have previously labeled the comments for toxic behavior. This will help prevent cyberbullying.

Preprocessing and EDA:

SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) is basically used to balance imbalanced datasets. The dataset used for this problem is highly skewed. We use SMOTE to synthetically generate data points of the minority class to increase their frequency.

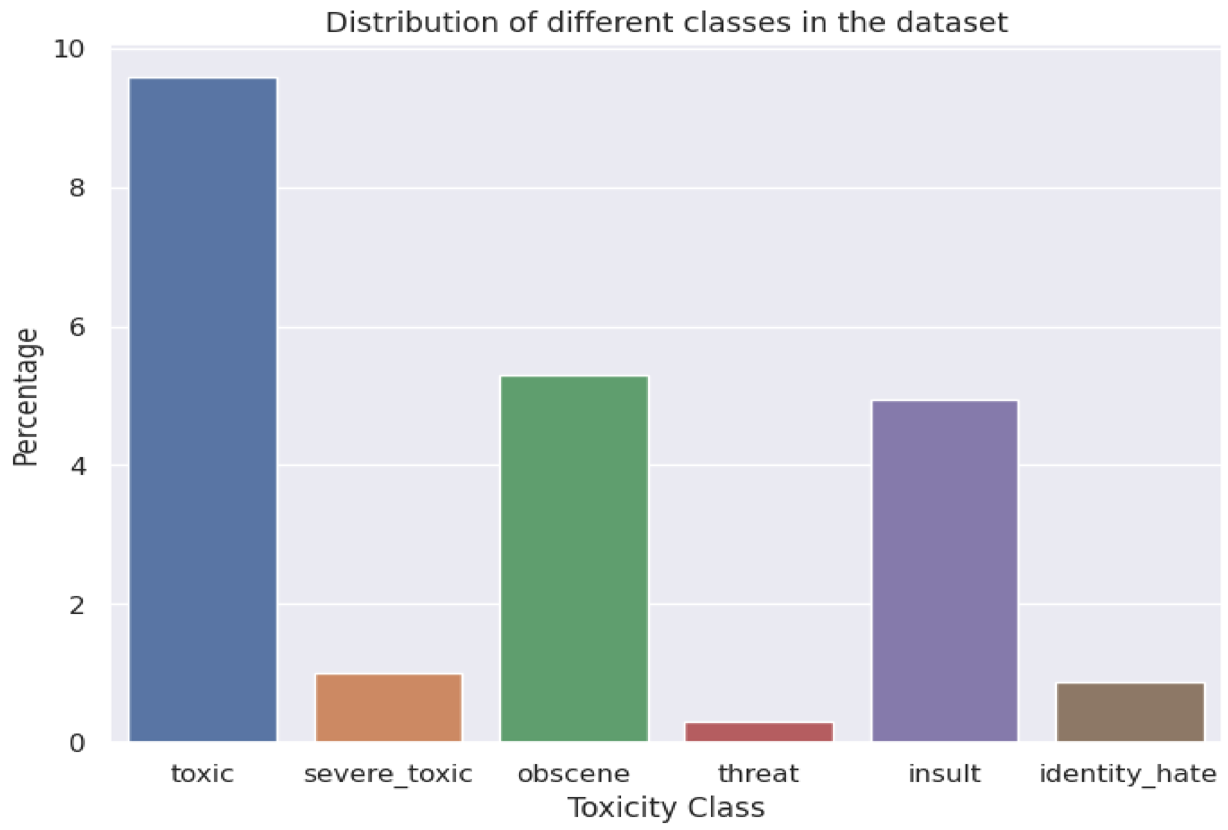
Accuracy Metrics

Since the dataset is highly skewed, we tried to optimize the F1 score and the AUC under the precision recall curve. The precision recall curve is robust to the skewness of the data.

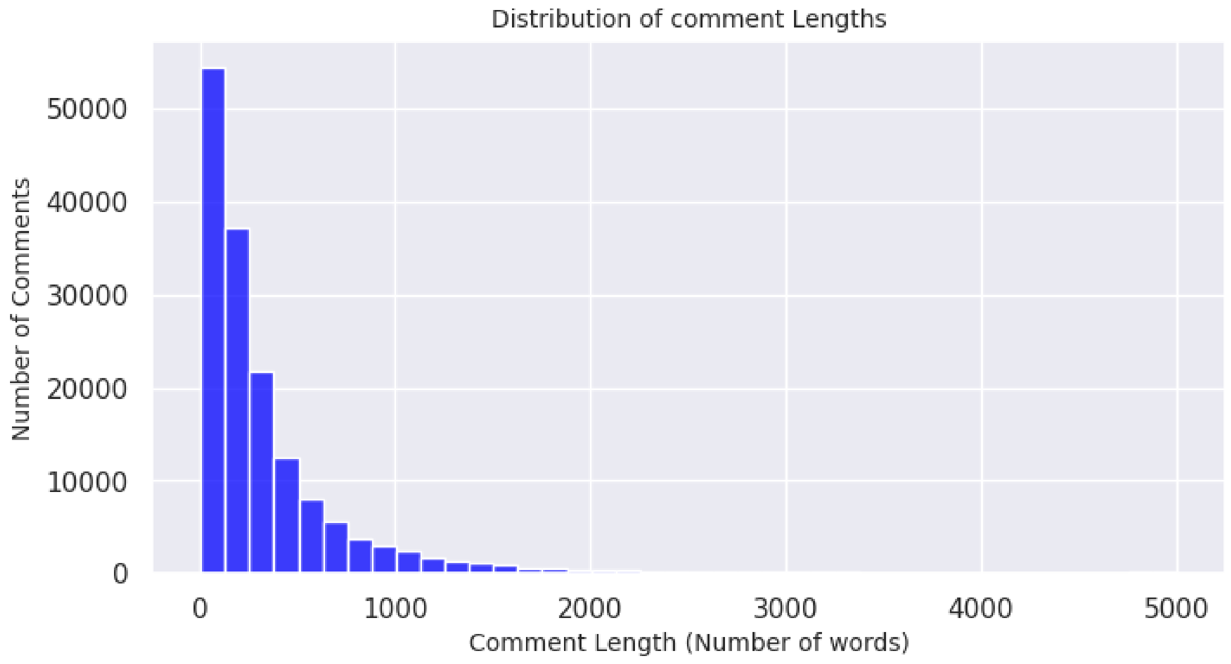
Method

We have six classes to predict, , we will be predicting each class separately using the binary relevance method.

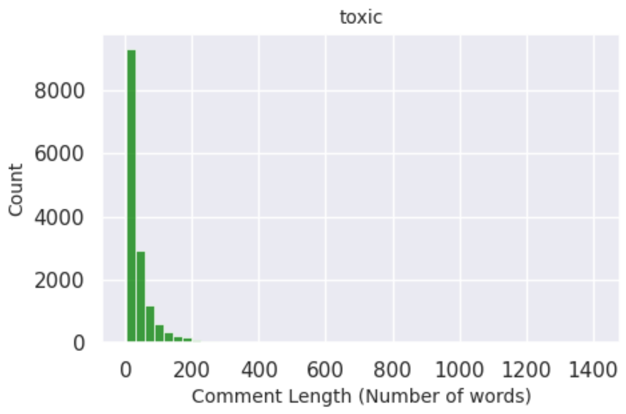
The distribution of different classes in the dataset:



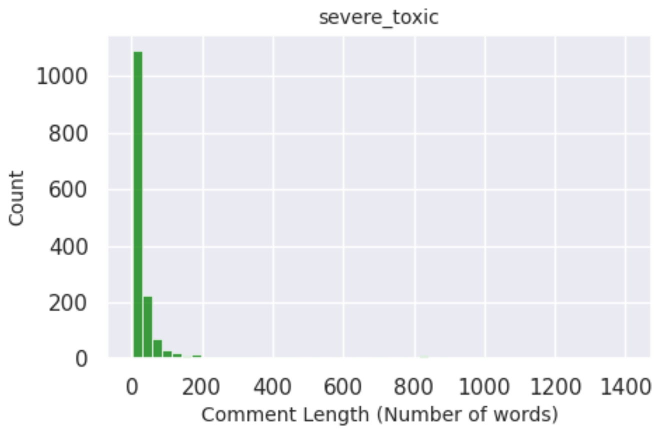
Distribution of comment lengths:



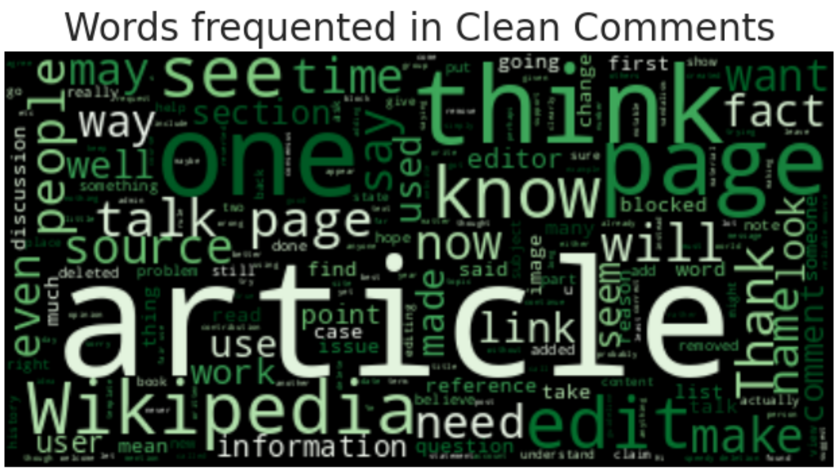
For only Toxic:



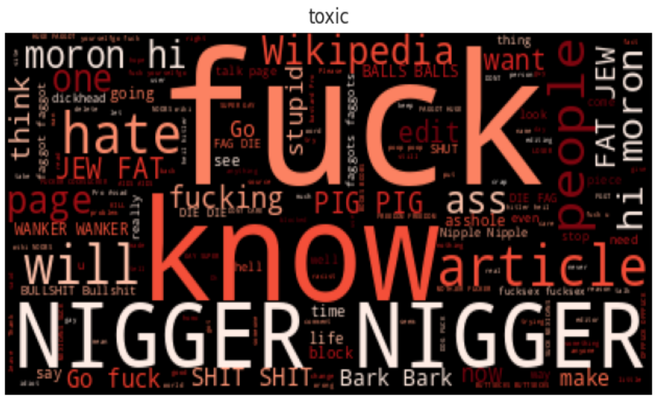
For Severe Toxic:

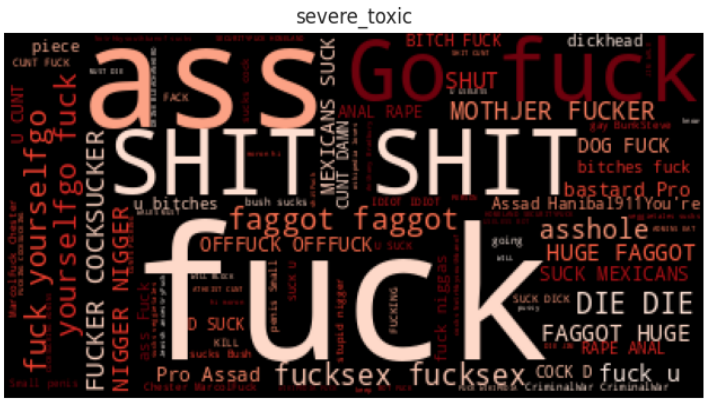


Word cloud for clean comments:



Word Cloud for Toxic and Severe toxic Labels:





<u>Feature extraction:</u>

We used tfidf that is a standard approach for text classification.

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a process of converting text data into numerical representation. It basically counts the frequency of words in the text and assigns numerical values based on them.

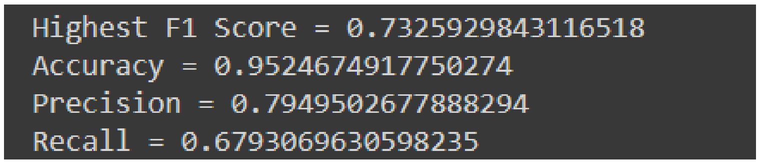
Clean Comment Function:

We first clean the text of the comment_text. This is important because we remove unwanted words, links, noise and make the comments into its base form. We first convert the text into lowercase, then remove links, if any, from the text. We use 'nltk' library to tokenize words, it separates the comments into individual words and then lemmatizes the words. Lemmatize means to convert the words into their base form and then we remove stopwords. Stopwords are most commonly used words and they often do not provide any individual meaning. Then, we return the comment text back by joining the cleaned individual words.

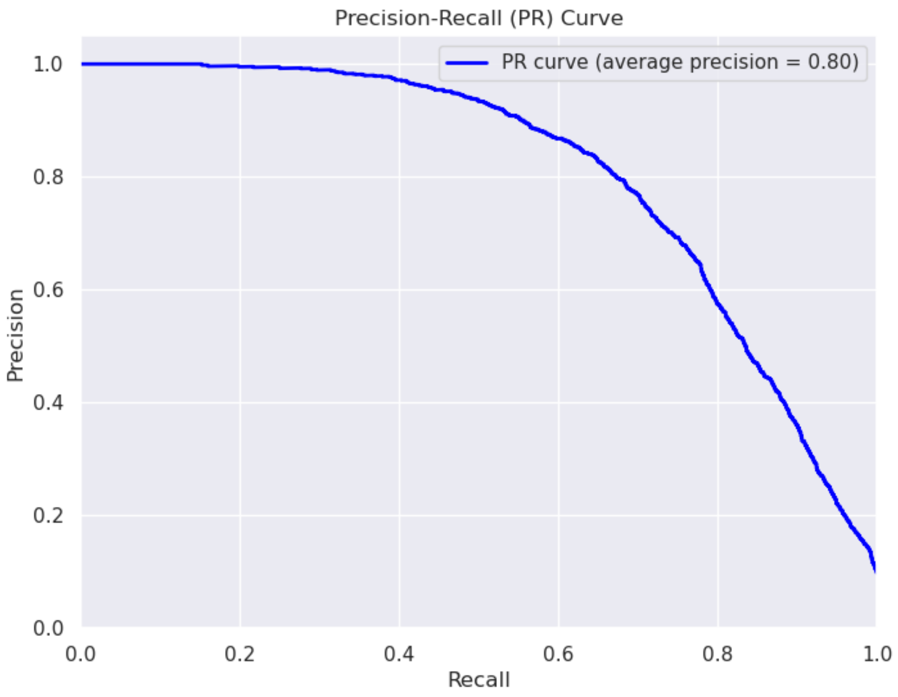
Multinomial Naive Bayes:

We employed the Multinomial Naive Bayes algorithm,. The TF-IDF transformation assigns continuous weights to words, indicating their relative importance within documents, and the Multinomial Naive Bayes algorithm is well-equipped to handle both this continuous representation and the discrete count data commonly found in text.

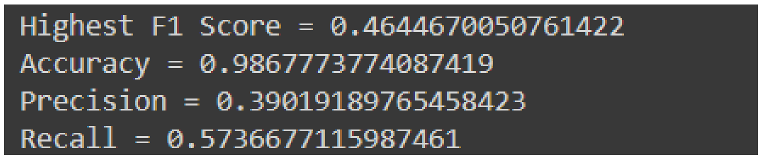
Toxic:



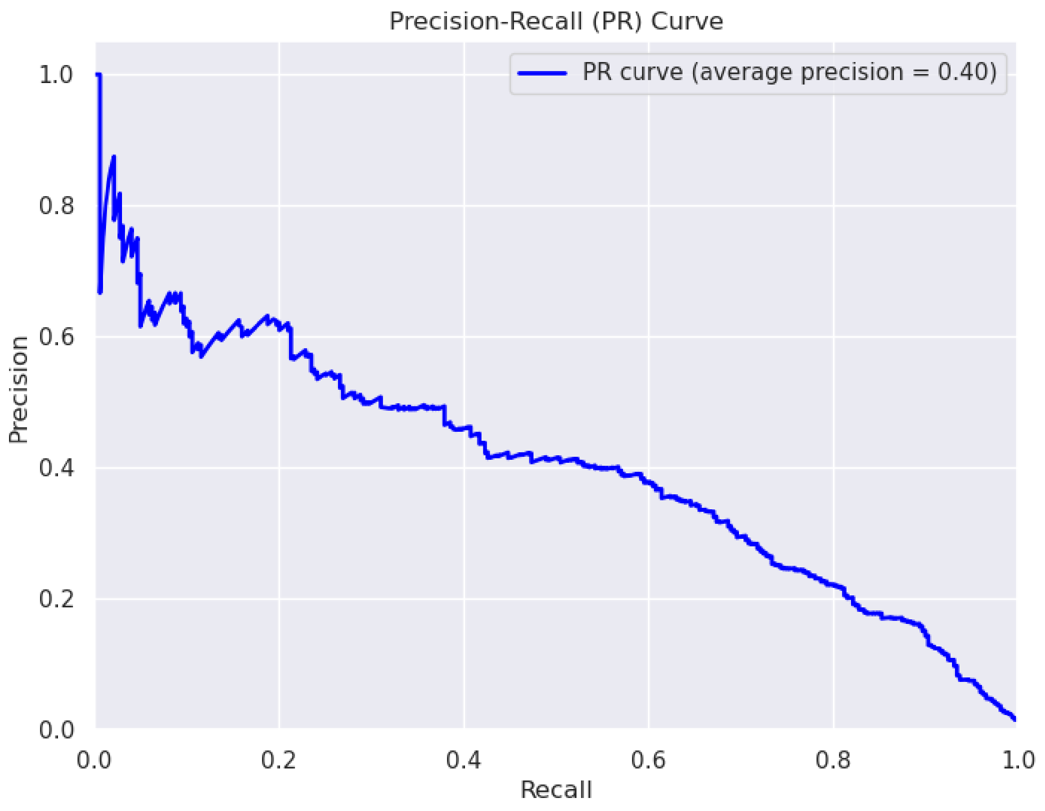
```
Highest F1 Score = 0.7325929843116518
Accuracy = 0.9524674917750274
Precision = 0.7949502677888294
Recall = 0.6793069630598235
```



Severe toxic:

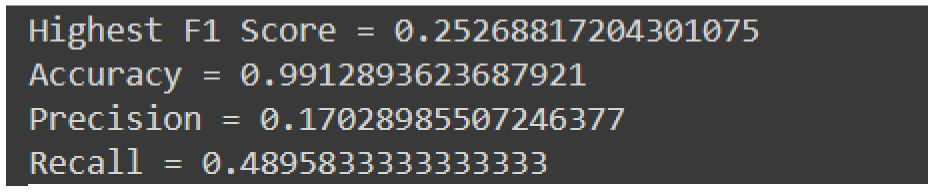


```
Highest F1 Score = 0.4644670050761422
Accuracy = 0.9867773774087419
Precision = 0.39019189765458423
Recall = 0.5736677115987461
```



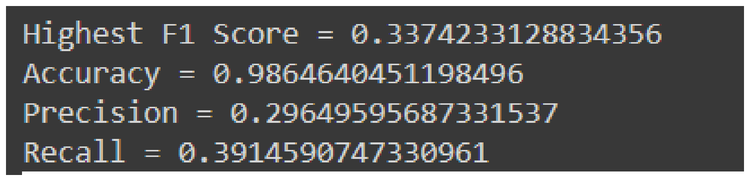
Due to high class imbalance the model doesn't learn to predict severe_toxic properly even afte applying SMOTE.

<u>Threat</u>:



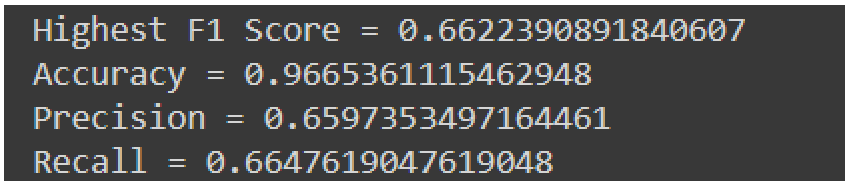
```
Highest F1 Score = 0.25268817204301075
Accuracy = 0.9912893623687921
Precision = 0.17028985507246377
Recall = 0.4895833333333333
```

<u>Identity hate:</u>



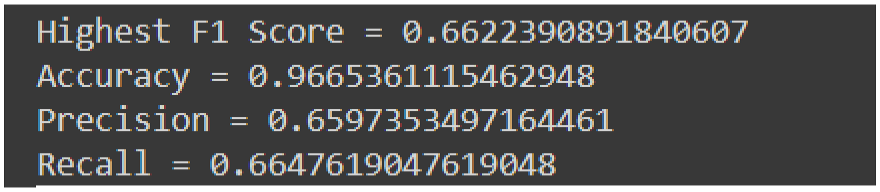
```
Highest F1 Score = 0.33742331288343556
Accuracy = 0.9864640451198496
Precision = 0.29649595687331537
Recall = 0.3914590747330961
```

Obscene:



```
Highest F1 Score = 0.6622390891840607
Accuracy = 0.9665361115462948
Precision = 0.6597353497164461
Recall = 0.6647619047619048
```

Insult:

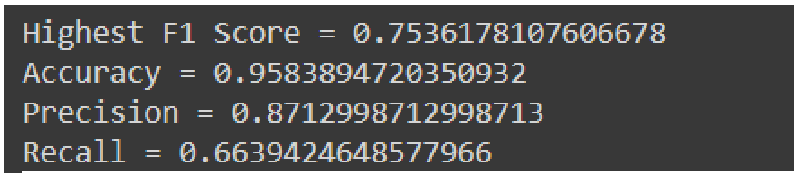


```
Highest F1 Score = 0.6622390891840607
Accuracy = 0.9665361115462948
Precision = 0.6597353497164461
Recall = 0.6647619047619048
```

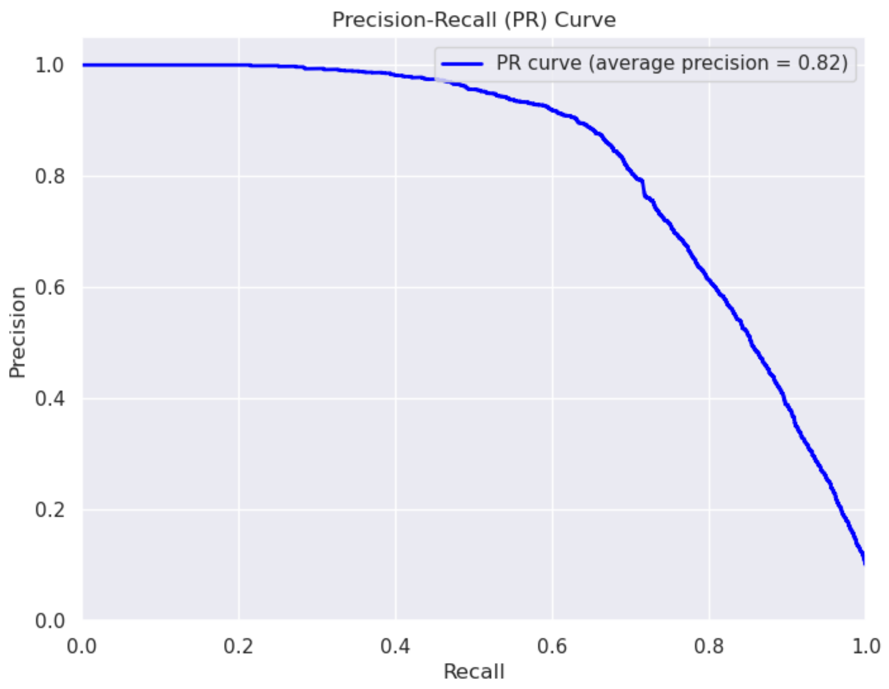
Logistic Regression

Logistic regression is useful for this task because it models the relationship between the words in the text and the probability of belonging to a particular class using a logistic function. The logistic function maps any value to a range between 0 and 1, which can be interpreted as a probability.

Toxic:

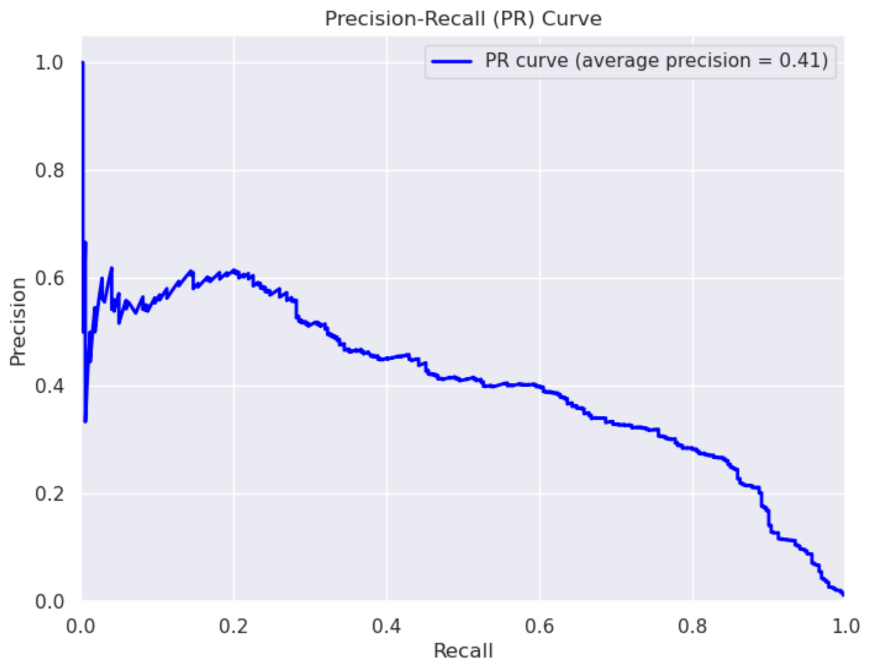


```
Highest F1 Score = 0.7536178107606678
Accuracy = 0.9583894720350932
Precision = 0.8712998712998713
Recall = 0.6639424648577966
```

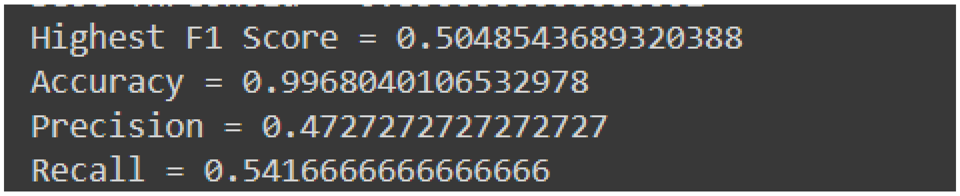


Severe toxic:

```
Highest F1 Score = 0.47595561035758327
Accuracy = 0.9866833777220743
Precision = 0.39227642276422764
Recall = 0.6050156739811913
```

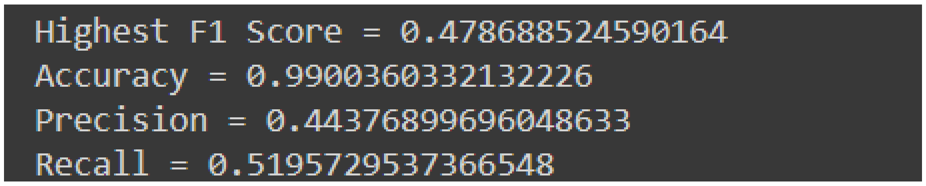


Threat



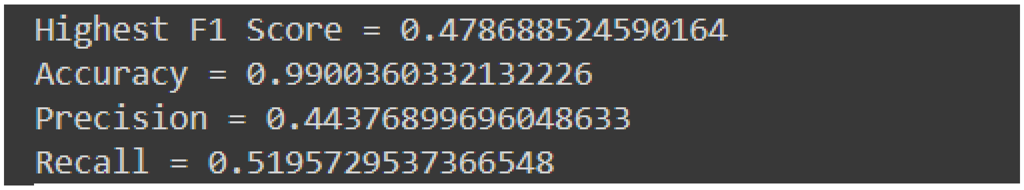
```
Highest F1 Score = 0.5048543689320388
Accuracy = 0.9968040106532978
Precision = 0.4727272727272727
Recall = 0.5416666666666666
```

Identity hate



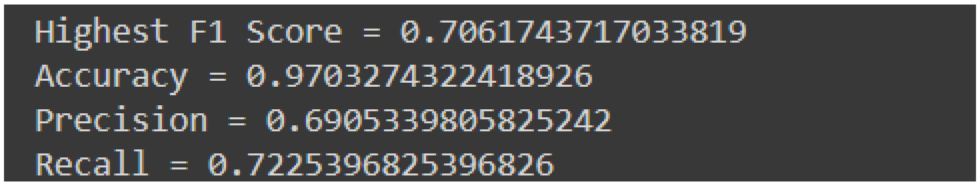
```
Highest F1 Score = 0.478688524590164
Accuracy = 0.9900360332132226
Precision = 0.44376899696048633
Recall = 0.5195729537366548
```

Obscene:



```
Highest F1 Score = 0.478688524590164
Accuracy = 0.9900360332132226
Precision = 0.44376899696048633
Recall = 0.5195729537366548
```

Insult:



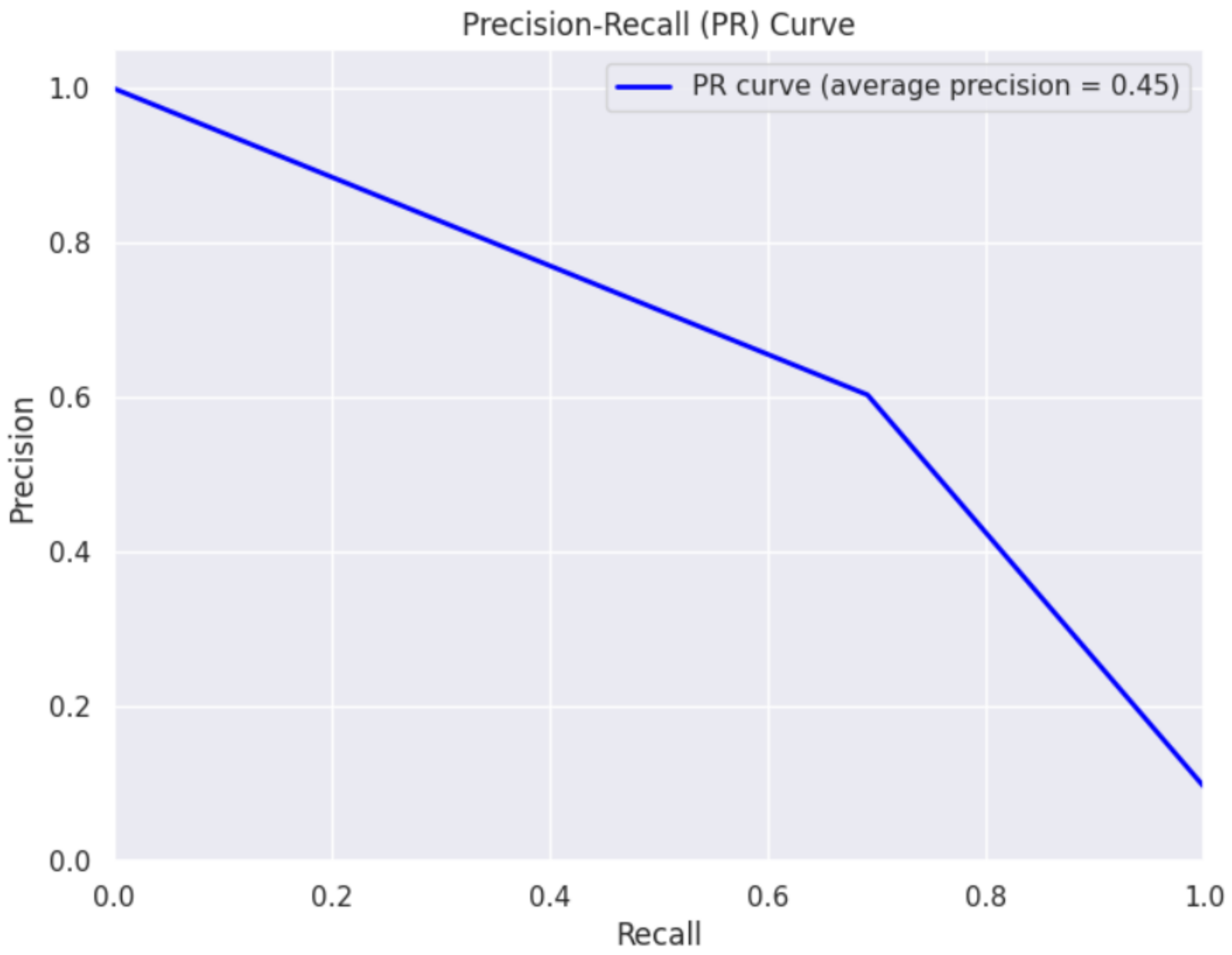
```
Highest F1 Score = 0.7061743717033819
Accuracy = 0.9703274322418926
Precision = 0.6905339805825242
Recall = 0.7225396825396826
```

Decision Tree:

We have chosen DTC and we are using it with SMOTE to balance class distribution. DTC is good because it is non-parametric, which means that they do not make assumptions about the distribution of the data. We also hypertune the parameters first.

Toxic:

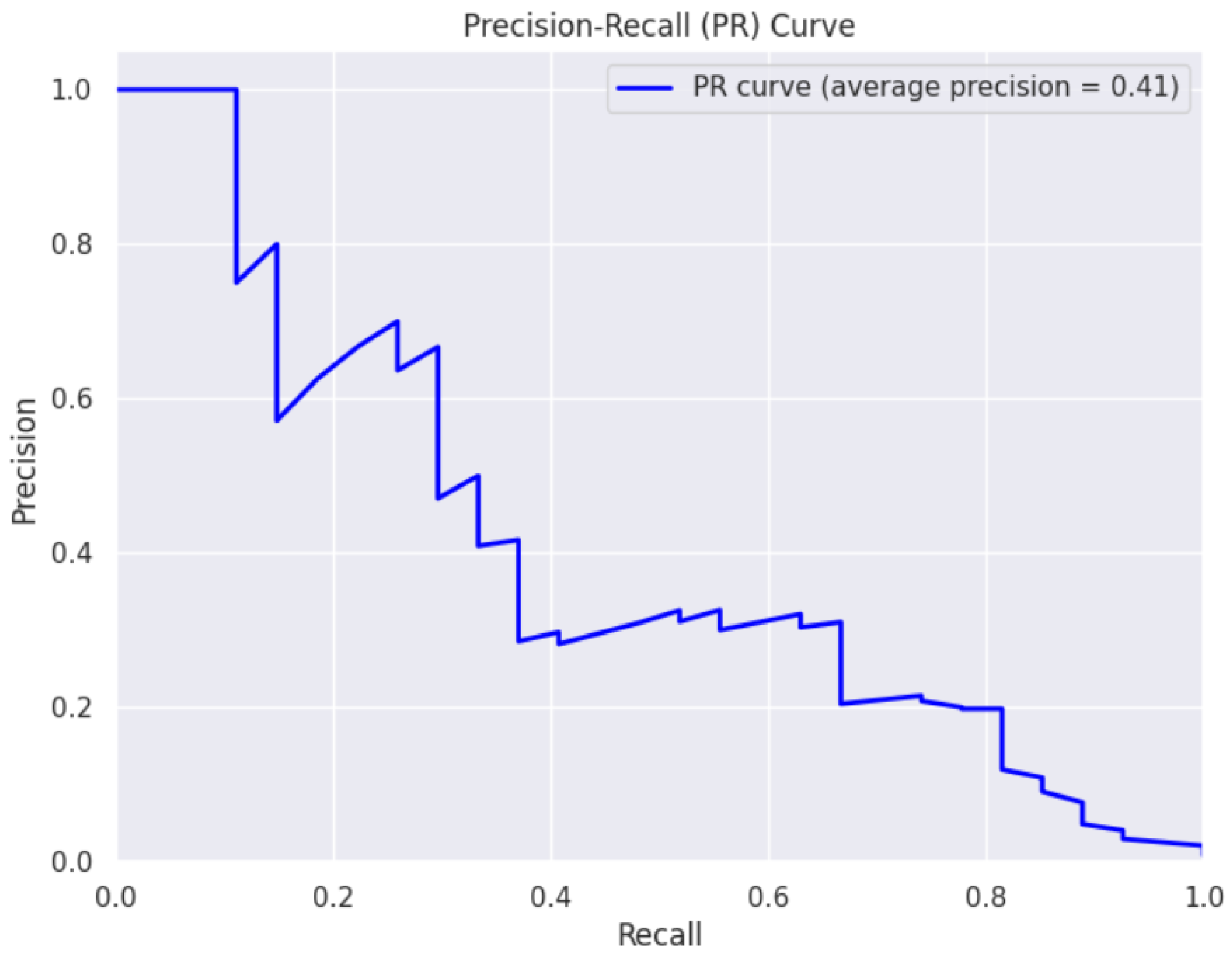
```
Highest F1 Score = 0.6444780635400909
Accuracy = 0.9263784461152882
```



The accuracy of 92.63% is misleading here because we can get accuracy of 90% by just predicting all 1 so here F1 score is a better metric and a F1 score of 0.64 is a good score considering the class imbalance.

Severe toxic:

```
Highest F1 Score = 0.32075471698113206
Accuracy = 0.9774436090225563
```



We obtain poor results for severe toxic comments as clearly SMOTE is not able to generate enough synthetic data that carries any meaningful information.

Obscene:

```
Highest F1 Score = 0.6721311475409836
Accuracy = 0.9624060150375939
```

We get a relatively good F1 score for obscene comments.

Threat:

```
Highest F1 Score = 0.35714285714285715
Accuracy = 0.9943609022556391
```

Insult:

```
Highest F1 Score = 0.6371191135734072
Accuracy = 0.9589598997493735
```

Identity hate:

```
Highest F1 Score = 0.3
Accuracy = 0.9780701754385965
```

Random Forest:

The main advantages of Random Forest is its ability to handle high-dimensional data such as text data represented as tf-idf matrix and it can also handle data imbalance.It can achieve good performance even with limited training data, and is less prone to overfitting compared to other complex models.

Toxic:

```
Highest F1 Score = 0.7152777777777777
Accuracy = 0.9486215538847118
```

We get very good value of F1 score and the accuracy is much better than what we would have gotten by just by predicting all as not being Toxic.

Severe toxic:

```
Highest F1 Score = 0.41379310344827586
Accuracy = 0.9840225563909775
```

Threat:

```
Highest F1 Score = 0.35294117647058826
Accuracy = 0.9965538847117794
```

Obscene:

```
Highest F1 Score = 0.7441860465116279
Accuracy = 0.9724310776942355
```

Insult:

```
Highest F1 Score = 0.7177914110429449
Accuracy = 0.9711779448621554
```

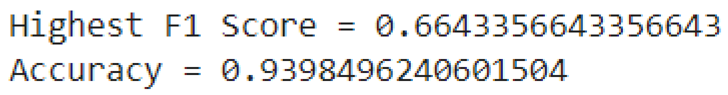
Identity hate:

```
Highest F1 Score = 0.380952380952381
Accuracy = 0.9918546365914787
```

We can observe that overall we are getting much better results than DTC because Random Forest is an ensemble of DTC so it will give much better results than DTC.

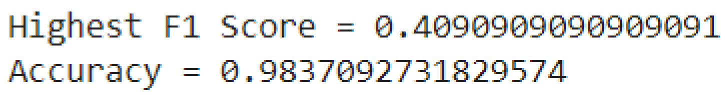
<u>Ada Boost:</u>
AdaBoost is an ensemble method. It combines multiple weak classifiers and it is useful for this as it provides a good generalization and also avoids overfitting. The algorithm effectively handles high-dimensional feature spaces common in text classification.The algorithm's adaptability make it valuable for text classification tasks.

<u>Toxic:</u>



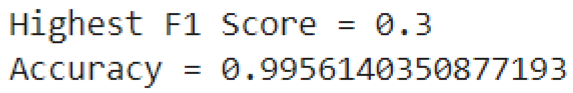
```
Highest F1 Score = 0.6643356643356643
Accuracy = 0.9398496240601504
```

<u>Severe toxic:</u>



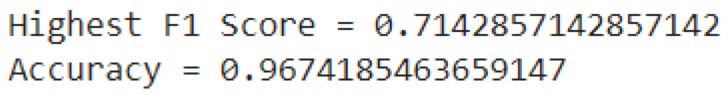
```
Highest F1 Score = 0.4090909090909091
Accuracy = 0.9837092731829574
```

<u>Threat:</u>



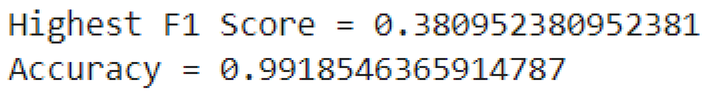
```
Highest F1 Score = 0.3
Accuracy = 0.9956140350877193
```

<u>Obscene:</u>



```
Highest F1 Score = 0.7142857142857142
Accuracy = 0.9674185463659147
```

<u>Identity hate:</u>

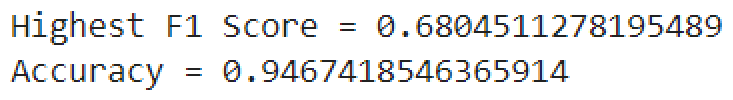


```
Highest F1 Score = 0.380952380952381
Accuracy = 0.9918546365914787
```

XGBoost:

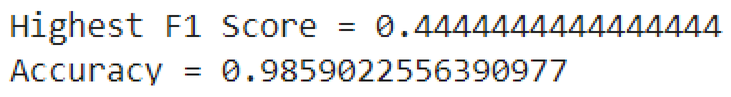
It is well suited for text classification as tfidf is a high dimensional feature. an ensemble method, XGBoost achieves improved accuracy and generalization, making it a strong candidate for text classification.

Toxic:



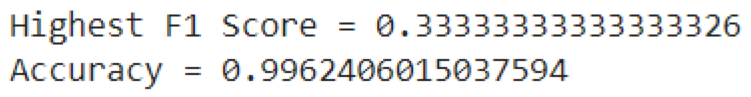
```
Highest F1 Score = 0.6804511278195489
Accuracy = 0.9467418546365914
```

Severe toxic:



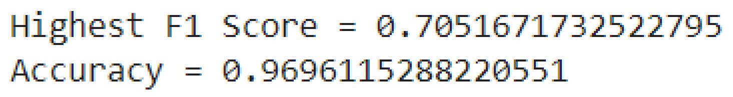
```
Highest F1 Score = 0.4444444444444444
Accuracy = 0.9859022556390977
```

Threat:



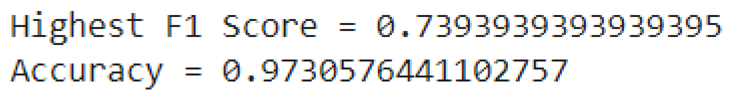
```
Highest F1 Score = 0.33333333333333326
Accuracy = 0.9962406015037594
```

Insult:



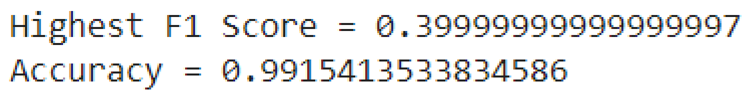
```
Highest F1 Score = 0.7051671732522795
Accuracy = 0.9696115288220551
```

Obscene:



```
Highest F1 Score = 0.7393939393939395
Accuracy = 0.9730576441102757
```

Identity hate:



```
Highest F1 Score = 0.39999999999999997
Accuracy = 0.9915413533834586
```

LightGBM:

Light GBM is used for this task as it han can handle imbalance data and also avoid overfitting. LightGBM has the ability to efficiently handle high-dimensional and sparse data, a common characteristic of text data represented using techniques such as TF-IDF.

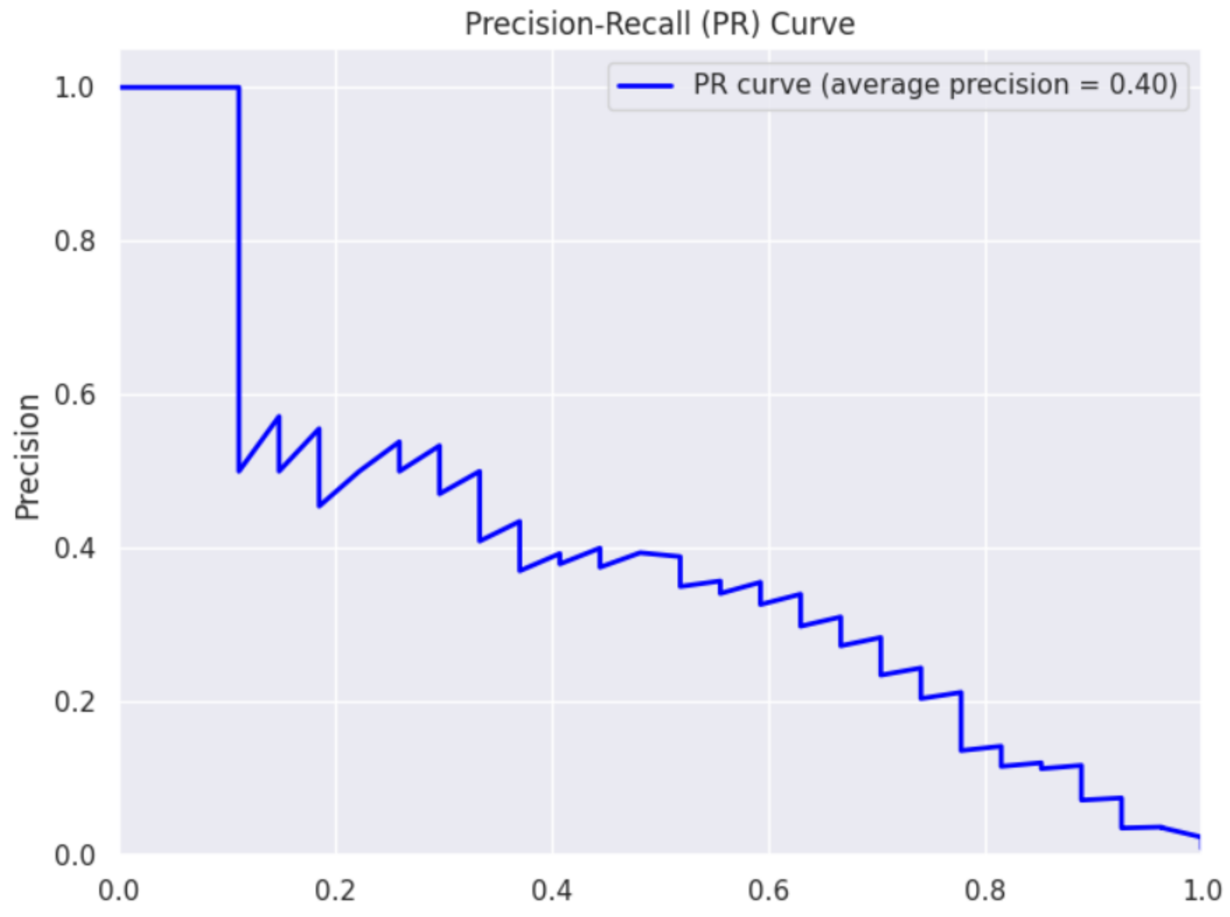
Toxic:

```
Highest F1 Score = 0.7124773960216998
Accuracy = 0.950187969924812
```

We get good accuracy and relatively good f1 score.

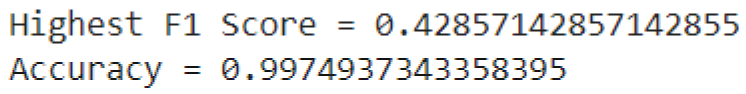
Severe toxic:

```
Highest F1 Score = 0.4444444444444444
Accuracy = 0.9890350877192983
```



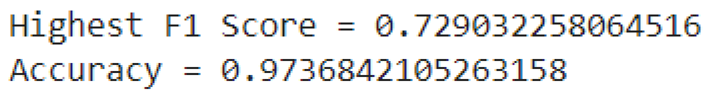
The F1 score obtained is relatively good here.

Threat:



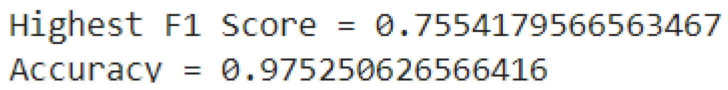
```
Highest F1 Score = 0.42857142857142855
Accuracy = 0.9974937343358395
```

Insult:



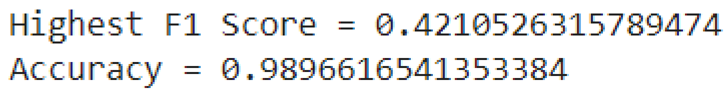
```
Highest F1 Score = 0.729032258064516
Accuracy = 0.9736842105263158
```

Obscene:



```
Highest F1 Score = 0.7554179566563467
Accuracy = 0.975250626566416
```

Identity hate:



```
Highest F1 Score = 0.4210526315789474
Accuracy = 0.9896616541353384
```

Neural Networks (with LSTM)

We used tokeniser and pad sequences to prepare the data for neural network .

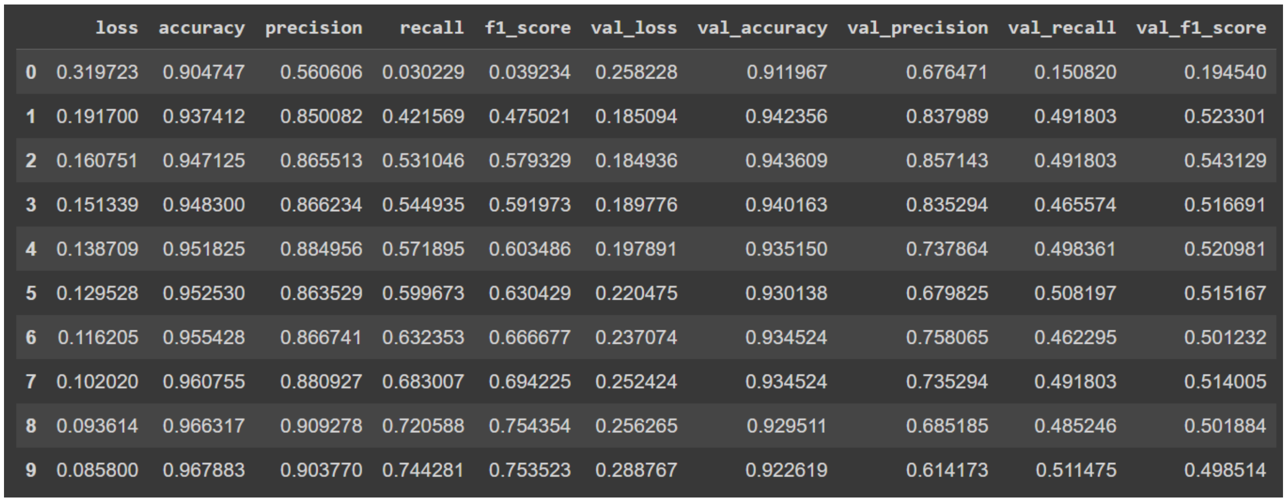
We have used a 6 layer neural network model to predict the probabilities. One layer of embedding, three layers of LSTM, and two layers of ANN.

LSTM: It is a variant of RNN. This is useful for natural language processing as it can model the sequential nature of the data.

We have trained the neural network model individually to predict the probability of each label.

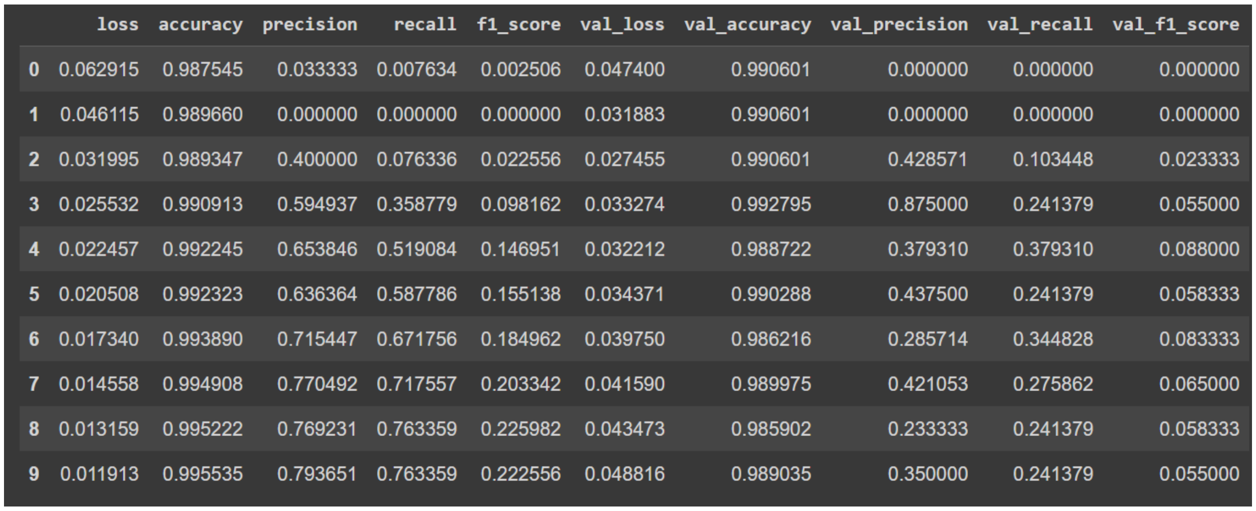
We have used ‘adam’ and ‘binary_crossentropy’ as our optimizer and loss function, respectively.

For label "**toxic**":



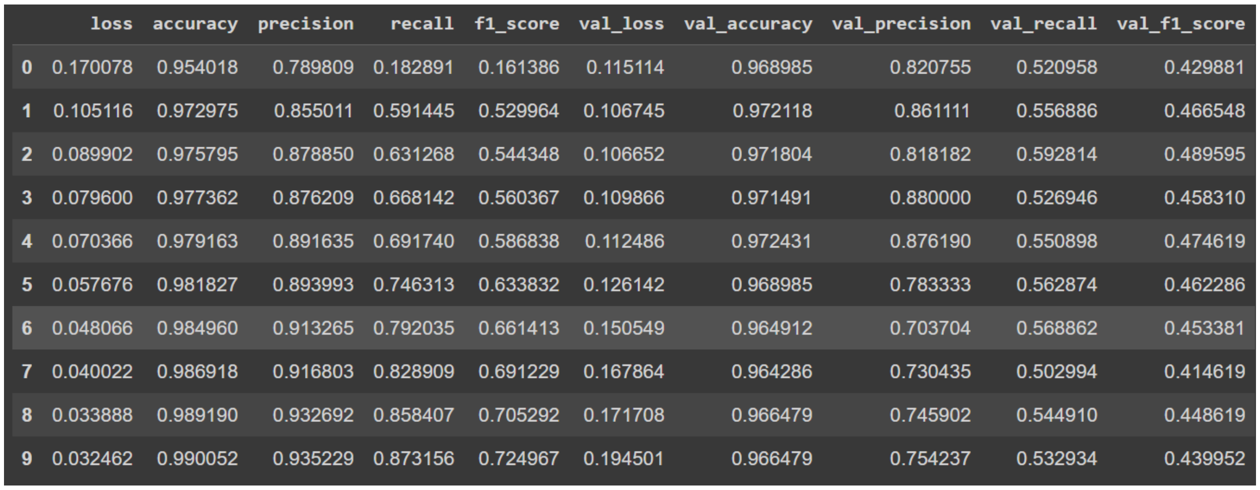
	loss	accuracy	precision	recall	f1_score	val_loss	val_accuracy	val_precision	val_recall	val_f1_score
0	0.319723	0.904747	0.560606	0.030229	0.039234	0.258228	0.911967	0.676471	0.150820	0.194540
1	0.191700	0.937412	0.850082	0.421569	0.475021	0.185094	0.942356	0.837989	0.491803	0.523301
2	0.160751	0.947125	0.865513	0.531046	0.579329	0.184936	0.943609	0.857143	0.491803	0.543129
3	0.151339	0.948300	0.866234	0.544935	0.591973	0.189776	0.940163	0.835294	0.465574	0.516691
4	0.138709	0.951825	0.884956	0.571895	0.603486	0.197891	0.935150	0.737864	0.498361	0.520981
5	0.129528	0.952530	0.863529	0.599673	0.630429	0.220475	0.930138	0.679825	0.508197	0.515167
6	0.116205	0.955428	0.866741	0.632353	0.666677	0.237074	0.934524	0.758065	0.462295	0.501232
7	0.102020	0.960755	0.880927	0.683007	0.694225	0.252424	0.934524	0.735294	0.491803	0.514005
8	0.093614	0.966317	0.909278	0.720588	0.754354	0.256265	0.929511	0.685185	0.485246	0.501884
9	0.085800	0.967883	0.903770	0.744281	0.753523	0.288767	0.922619	0.614173	0.511475	0.498514

For label “**severe_toxic**”:



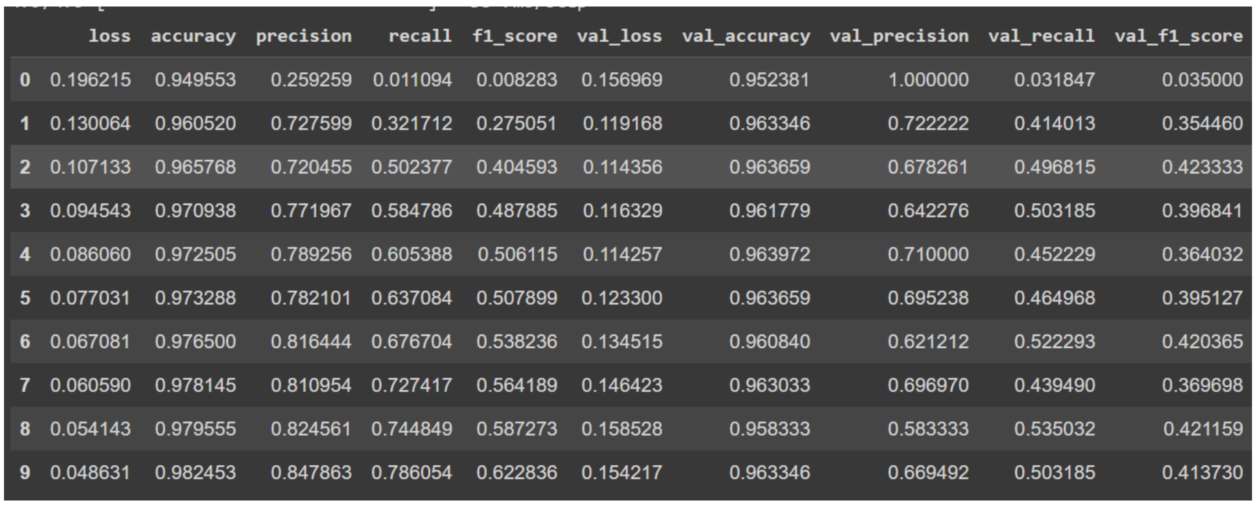
	loss	accuracy	precision	recall	f1_score	val_loss	val_accuracy	val_precision	val_recall	val_f1_score
0	0.062915	0.987545	0.033333	0.007634	0.002506	0.047400	0.990601	0.000000	0.000000	0.000000
1	0.046115	0.989660	0.000000	0.000000	0.000000	0.031883	0.990601	0.000000	0.000000	0.000000
2	0.031995	0.989347	0.400000	0.076336	0.022556	0.027455	0.990601	0.428571	0.103448	0.023333
3	0.025532	0.990913	0.594937	0.358779	0.098162	0.033274	0.992795	0.875000	0.241379	0.055000
4	0.022457	0.992245	0.653846	0.519084	0.146951	0.032212	0.988722	0.379310	0.379310	0.088000
5	0.020508	0.992323	0.636364	0.587786	0.155138	0.034371	0.990288	0.437500	0.241379	0.058333
6	0.017340	0.993890	0.715447	0.671756	0.184962	0.039750	0.986216	0.285714	0.344828	0.083333
7	0.014558	0.994908	0.770492	0.717557	0.203342	0.041590	0.989975	0.421053	0.275862	0.065000
8	0.013159	0.995222	0.769231	0.763359	0.225982	0.043473	0.985902	0.233333	0.241379	0.058333
9	0.011913	0.995535	0.793651	0.763359	0.222556	0.048816	0.989035	0.350000	0.241379	0.055000

For label “**obscene**”:



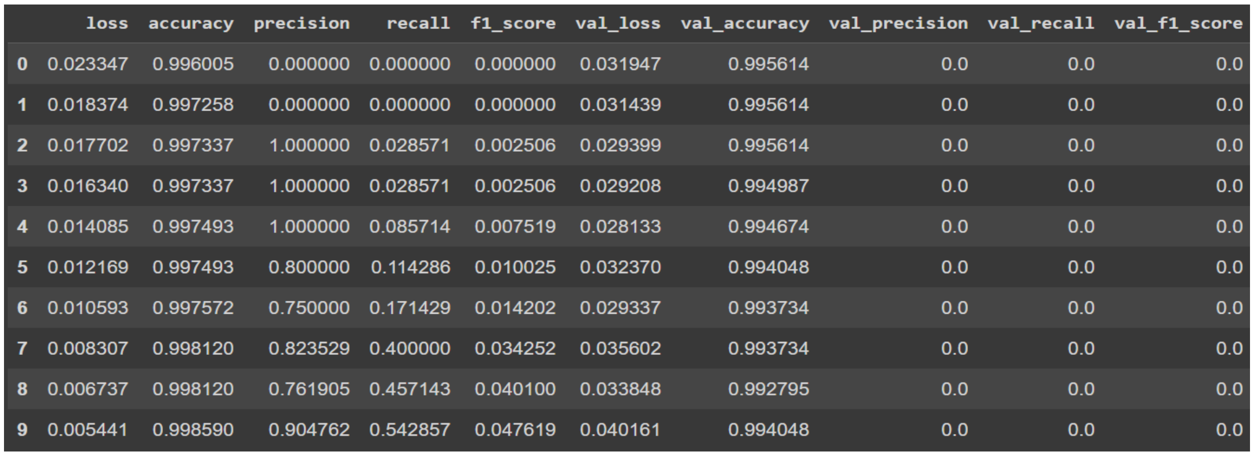
	loss	accuracy	precision	recall	f1_score	val_loss	val_accuracy	val_precision	val_recall	val_f1_score
0	0.170078	0.954018	0.789809	0.182891	0.161386	0.115114	0.968985	0.820755	0.520958	0.429881
1	0.105116	0.972975	0.855011	0.591445	0.529964	0.106745	0.972118	0.861111	0.556886	0.466548
2	0.089902	0.975795	0.878850	0.631268	0.544348	0.106652	0.971804	0.818182	0.592814	0.489595
3	0.079600	0.977362	0.876209	0.668142	0.560367	0.109866	0.971491	0.880000	0.526946	0.458310
4	0.070366	0.979163	0.891635	0.691740	0.586838	0.112486	0.972431	0.876190	0.550898	0.474619
5	0.057676	0.981827	0.893993	0.746313	0.633832	0.126142	0.968985	0.783333	0.562874	0.462286
6	0.048066	0.984960	0.913265	0.792035	0.661413	0.150549	0.964912	0.703704	0.568862	0.453381
7	0.040022	0.986918	0.916803	0.828909	0.691229	0.167864	0.964286	0.730435	0.502994	0.414619
8	0.033888	0.989190	0.932692	0.858407	0.705292	0.171708	0.966479	0.745902	0.544910	0.448619
9	0.032462	0.990052	0.935229	0.873156	0.724967	0.194501	0.966479	0.754237	0.532934	0.439952

For label “**insult**”:



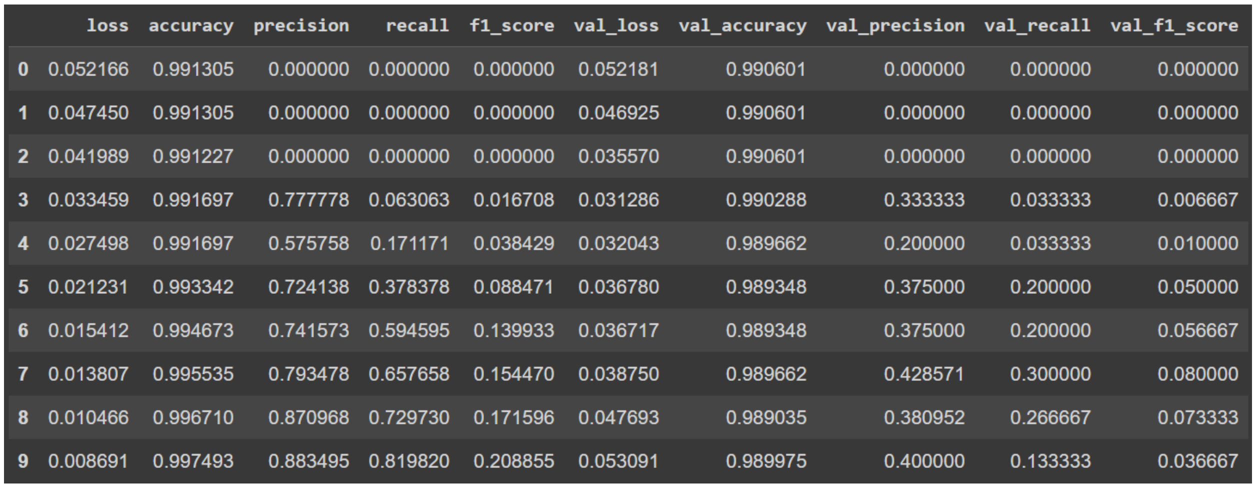
	loss	accuracy	precision	recall	f1_score	val_loss	val_accuracy	val_precision	val_recall	val_f1_score
0	0.196215	0.949553	0.259259	0.011094	0.008283	0.156969	0.952381	1.000000	0.031847	0.035000
1	0.130064	0.960520	0.727599	0.321712	0.275051	0.119168	0.963346	0.722222	0.414013	0.354460
2	0.107133	0.965768	0.720455	0.502377	0.404593	0.114356	0.963659	0.678261	0.496815	0.423333
3	0.094543	0.970938	0.771967	0.584786	0.487885	0.116329	0.961779	0.642276	0.503185	0.396841
4	0.086060	0.972505	0.789256	0.605388	0.506115	0.114257	0.963972	0.710000	0.452229	0.364032
5	0.077031	0.973288	0.782101	0.637084	0.507899	0.123300	0.963659	0.695238	0.464968	0.395127
6	0.067081	0.976500	0.816444	0.676704	0.538236	0.134515	0.960840	0.621212	0.522293	0.420365
7	0.060590	0.978145	0.810954	0.727417	0.564189	0.146423	0.963033	0.696970	0.439490	0.369698
8	0.054143	0.979555	0.824561	0.744849	0.587273	0.158528	0.958333	0.583333	0.535032	0.421159
9	0.048631	0.982453	0.847863	0.786054	0.622836	0.154217	0.963346	0.669492	0.503185	0.413730

For label “**threat**”:



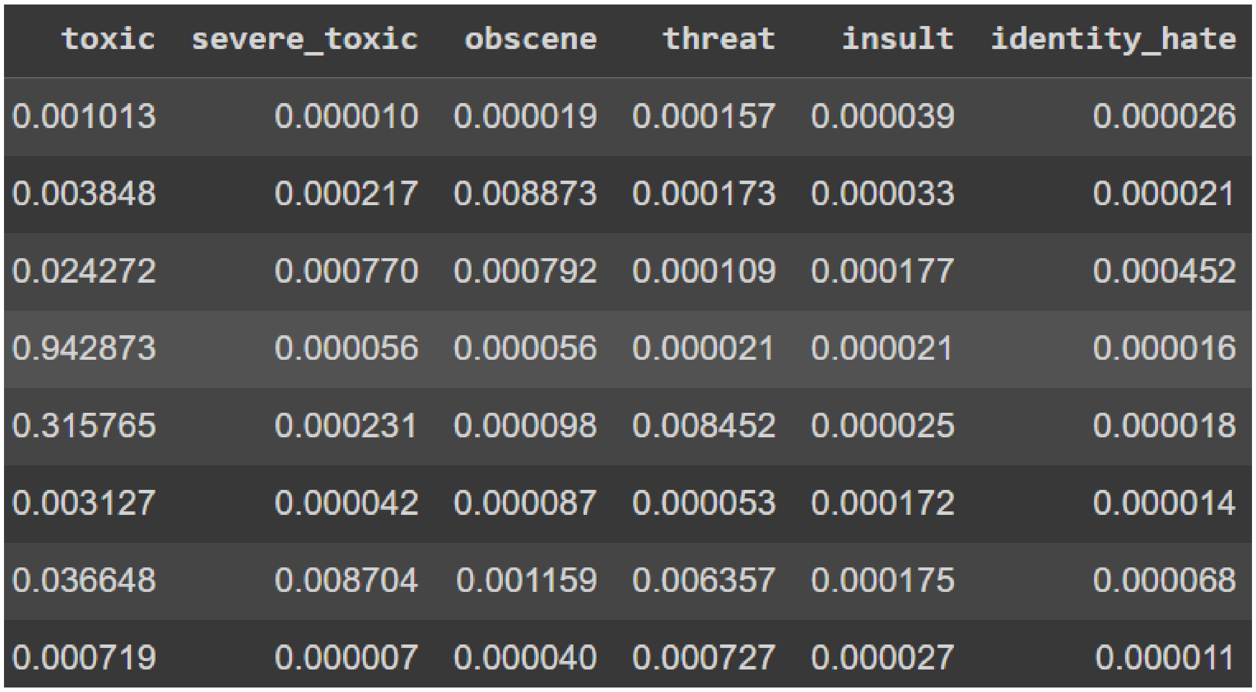
	loss	accuracy	precision	recall	f1_score	val_loss	val_accuracy	val_precision	val_recall	val_f1_score
0	0.023347	0.996005	0.000000	0.000000	0.000000	0.031947	0.995614	0.0	0.0	0.0
1	0.018374	0.997258	0.000000	0.000000	0.000000	0.031439	0.995614	0.0	0.0	0.0
2	0.017702	0.997337	1.000000	0.028571	0.002506	0.029399	0.995614	0.0	0.0	0.0
3	0.016340	0.997337	1.000000	0.028571	0.002506	0.029208	0.994987	0.0	0.0	0.0
4	0.014085	0.997493	1.000000	0.085714	0.007519	0.028133	0.994674	0.0	0.0	0.0
5	0.012169	0.997493	0.800000	0.114286	0.010025	0.032370	0.994048	0.0	0.0	0.0
6	0.010593	0.997572	0.750000	0.171429	0.014202	0.029337	0.993734	0.0	0.0	0.0
7	0.008307	0.998120	0.823529	0.400000	0.034252	0.035602	0.993734	0.0	0.0	0.0
8	0.006737	0.998120	0.761905	0.457143	0.040100	0.033848	0.992795	0.0	0.0	0.0
9	0.005441	0.998590	0.904762	0.542857	0.047619	0.040161	0.994048	0.0	0.0	0.0

For label “**identity-hate**”:



	loss	accuracy	precision	recall	f1_score	val_loss	val_accuracy	val_precision	val_recall	val_f1_score
0	0.052166	0.991305	0.000000	0.000000	0.000000	0.052181	0.990601	0.000000	0.000000	0.000000
1	0.047450	0.991305	0.000000	0.000000	0.000000	0.046925	0.990601	0.000000	0.000000	0.000000
2	0.041989	0.991227	0.000000	0.000000	0.000000	0.035570	0.990601	0.000000	0.000000	0.000000
3	0.033459	0.991697	0.777778	0.063063	0.016708	0.031286	0.990288	0.333333	0.033333	0.006667
4	0.027498	0.991697	0.575758	0.171171	0.038429	0.032043	0.989662	0.200000	0.033333	0.010000
5	0.021231	0.993342	0.724138	0.378378	0.088471	0.036780	0.989348	0.375000	0.200000	0.050000
6	0.015412	0.994673	0.741573	0.594595	0.139933	0.036717	0.989348	0.375000	0.200000	0.056667
7	0.013807	0.995535	0.793478	0.657658	0.154470	0.038750	0.989662	0.428571	0.300000	0.080000
8	0.010466	0.996710	0.870968	0.729730	0.171596	0.047693	0.989035	0.380952	0.266667	0.073333
9	0.008691	0.997493	0.883495	0.819820	0.208855	0.053091	0.989975	0.400000	0.133333	0.036667

After training the model, we have predicted the probabilities for test data, for each label. The probabilities of a random 8 comments are like this:



toxic	severe_toxic	obscene	threat	insult	identity_hate
0.001013	0.000010	0.000019	0.000157	0.000039	0.000026
0.003848	0.000217	0.008873	0.000173	0.000033	0.000021
0.024272	0.000770	0.000792	0.000109	0.000177	0.000452
0.942873	0.000056	0.000056	0.000021	0.000021	0.000016
0.315765	0.000231	0.000098	0.008452	0.000025	0.000018
0.003127	0.000042	0.000087	0.000053	0.000172	0.000014
0.036648	0.008704	0.001159	0.006357	0.000175	0.000068
0.000719	0.000007	0.000040	0.000727	0.000027	0.000011

Contributions:

Saksham Jain (B21EE059)

Visualization
Logistic Regression
Multinomial Naive Bayes

Atharva Ganesh Pade(B21EE014)

Trained Decision tree classifier,
Random Forest classifier
Adaboost, XGBoost, LightGBM.

Uppala Giridhar (B21EE072)

Data Preprocessing
EDA
Training Neural Networks