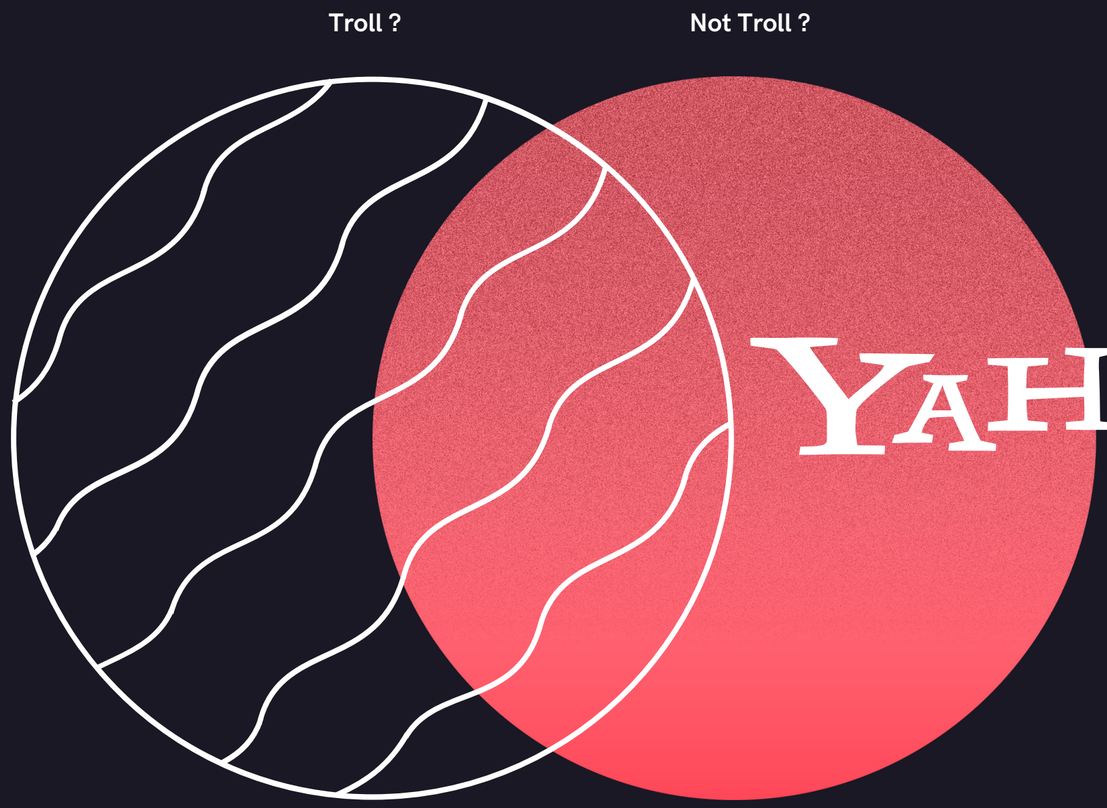


YAHOO!

Troll Detection Challenge

Team East India Company
Kaushik Mishra (IMT2020137)
Monjoy Narayan Choudhury(IMT2020502)

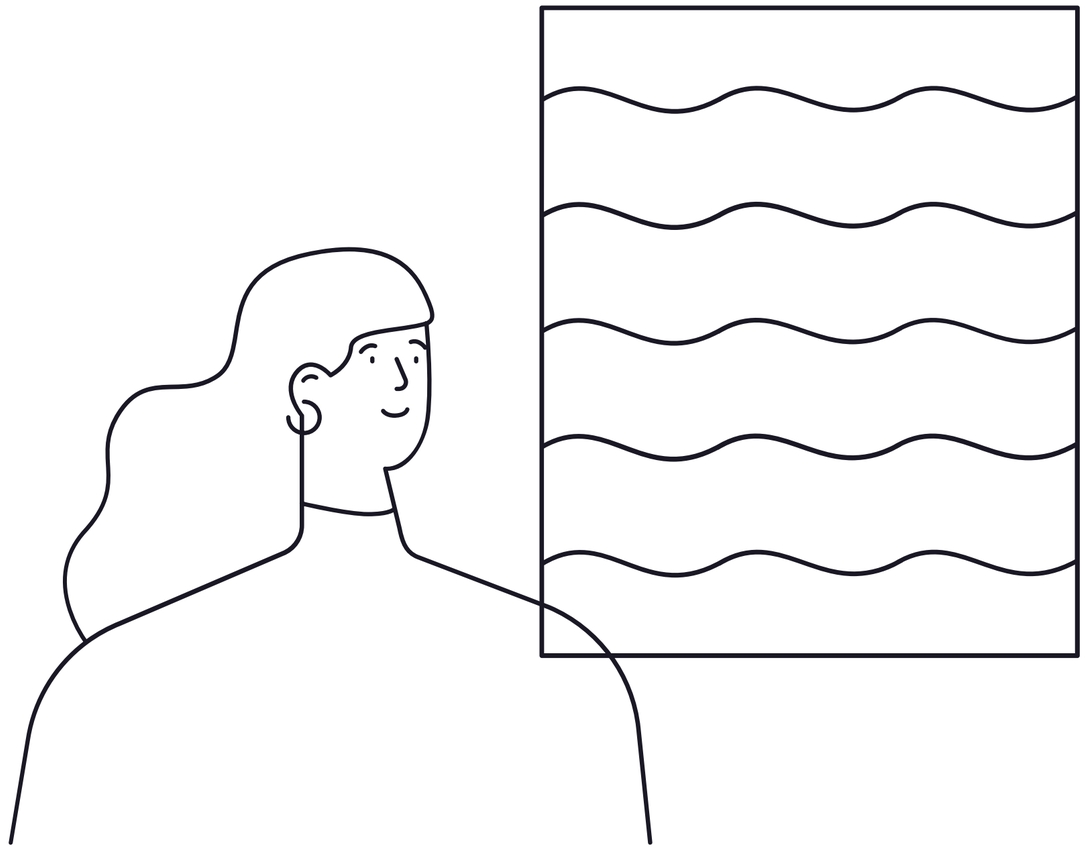
Introducing the dataset

Train dataset given to us consisted of:

- Question ID
- Question Text
- Troll/Not-Troll(Encoded 1 and 0 accordingly)

Some Precautionary checks were done

- Null value check -> Found none
- Duplicated rows -> Found none

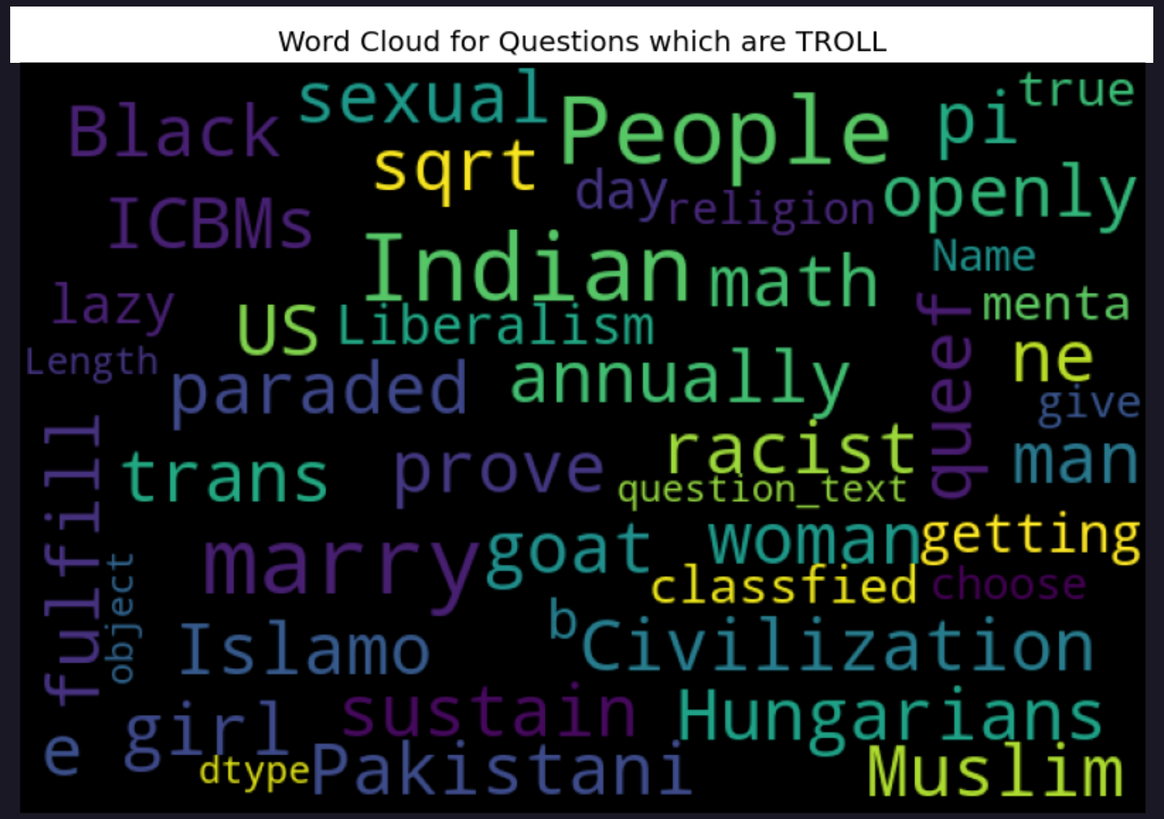


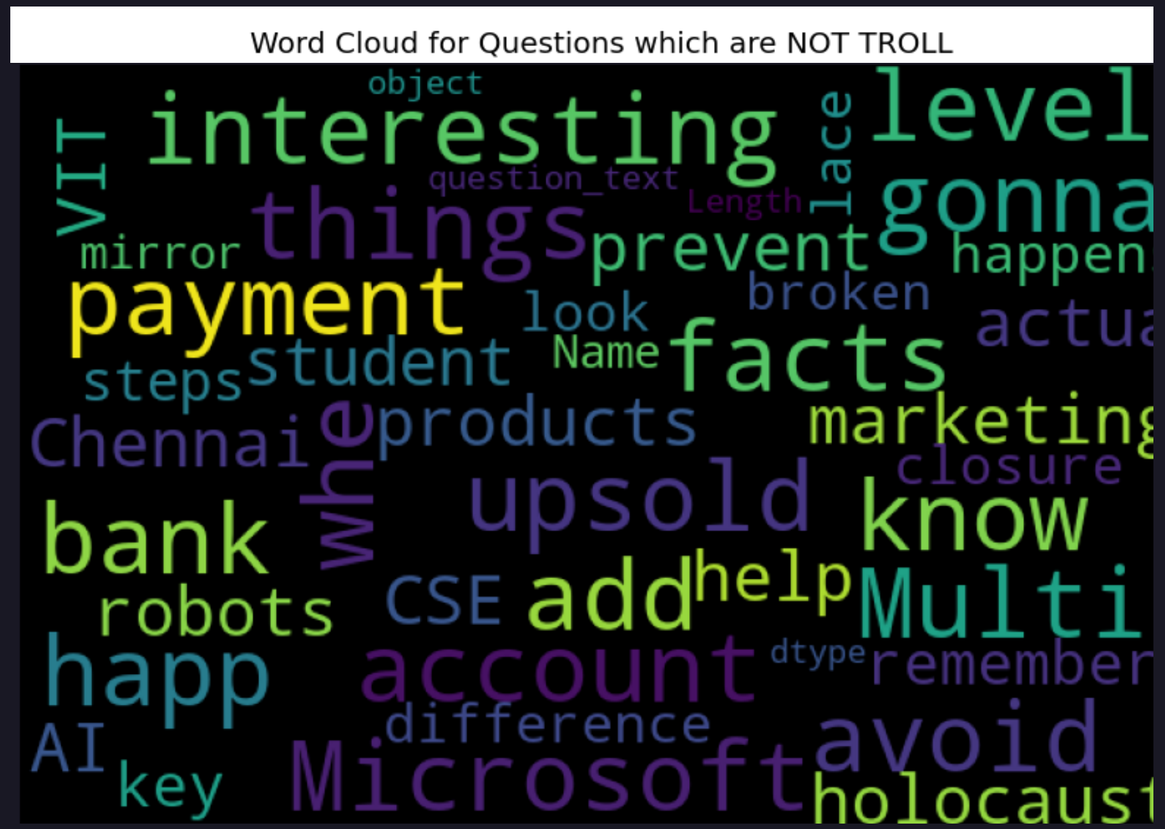
03

Visualisations

We went ahead with the following visualisation tool:

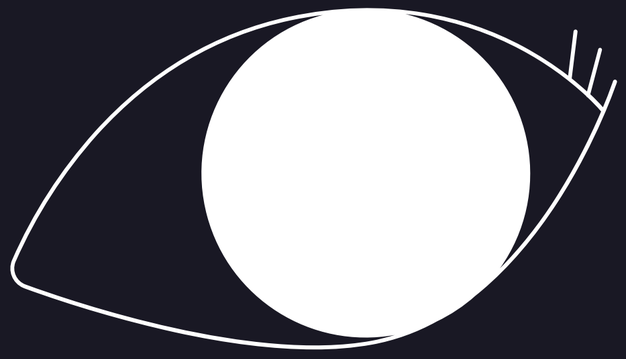
Word Cloud





PREPROCESSING
(Done so far)

- Lowercase conversion (A->a)
- Punctuation removal (, .)
- Stop word removal (is, shan't, will, shouldn't ...)
- White space removal
- Number removal
- URL removal
- Tokenization
- Stemming (Porter Stemmer)
- Bag of Words with Count Vectorizer (we haven't put a bound yet)
- Train Test Split: test_size=0.3, random_state=42, with stratify
- Oversampling



HERE ARE SOME PREDICTIONS

ML MODELS USED SO FAR

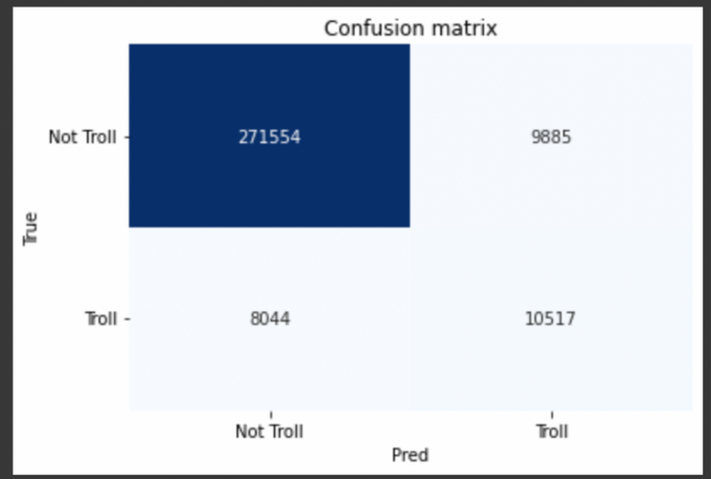
Multinomial Naive Bayes(Kaggle submission)

- ngram_range = None
- f1 score = 0.94
- f1_score(class = 0) = 0.97
- f1_score(class = 1) = 0.54

```
[[271554   9885]
 [  8044  10517]]
              precision    recall  f1-score   support

           0       0.97      0.96      0.97    281439
           1       0.52      0.57      0.54     18561

    accuracy                           0.94    300000
   macro avg       0.74      0.77      0.75    300000
weighted avg       0.94      0.94      0.94    300000
```



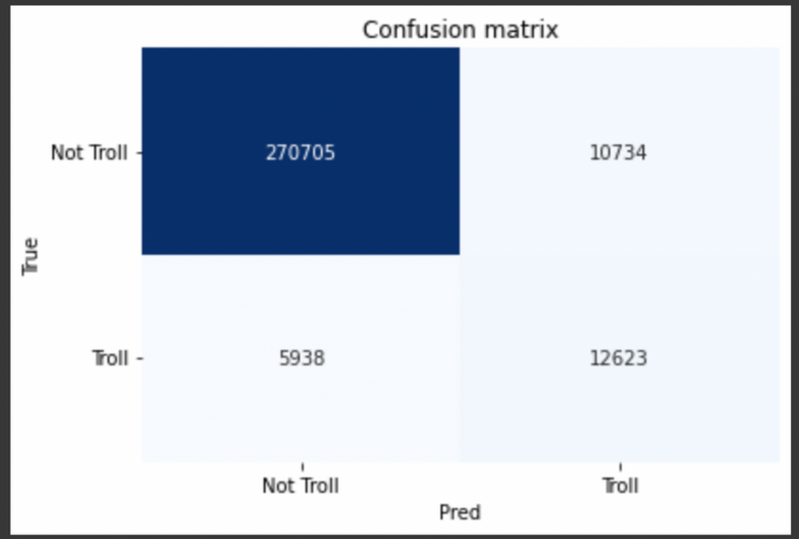
Logistic Regression
(Kaggle best submission so far)

- ngram_range = (1,2)
- Parameters:penalty='l2',class_weight='balanced', max_iter=1000
- f1 score = 0.92

```
[[270705  10734]
 [  5938  12623]]
              precision    recall  f1-score   support

           0       0.98      0.96      0.97    281439
           1       0.54      0.68      0.60     18561

    accuracy                           0.94    300000
   macro avg       0.76      0.82      0.79    300000
weighted avg       0.95      0.94      0.95    300000
```



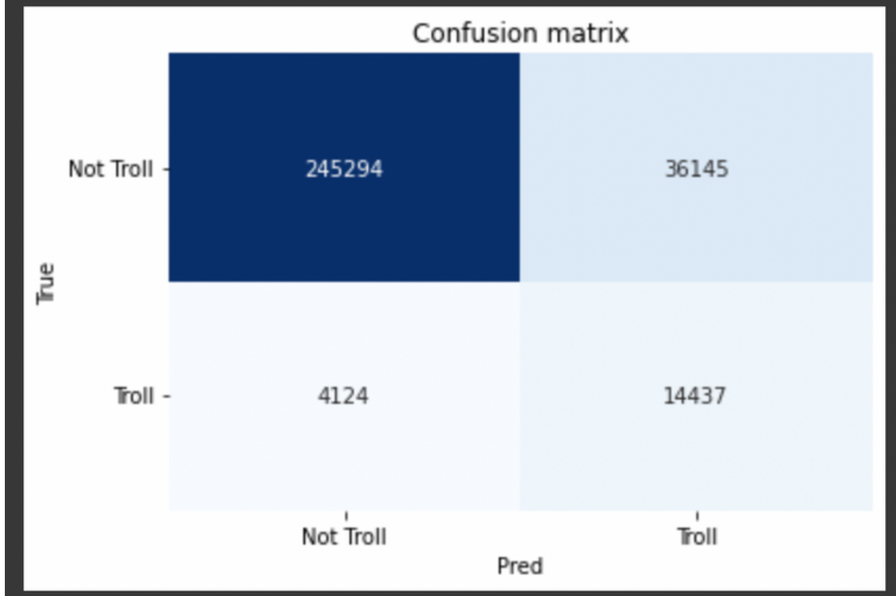
Perceptron

- Parameters:class_weight="balanced",random_state=42
- f1 score = 0.89

```
[[245294  36145]
 [  4124  14437]]
              precision    recall  f1-score   support

           0       0.98      0.87      0.92    281439
           1       0.29      0.78      0.42     18561

    accuracy                           0.87    300000
   macro avg       0.63      0.82      0.67    300000
weighted avg       0.94      0.87      0.89    300000
```



PLAN AHEAD

Trying more pre-processing techniques

- Using more stemmers as well as trying out TF-IDF
- Use of word embeddings like GlOVe or Word2Vec

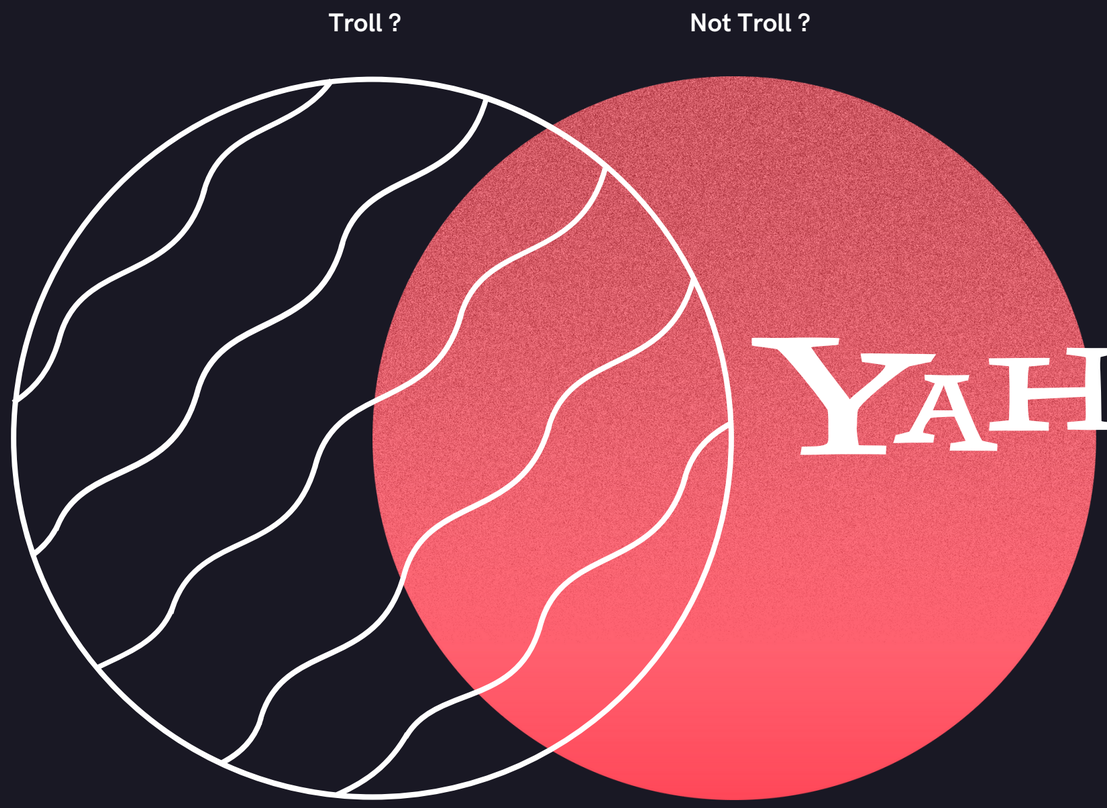
Handling imbalanced Dataset

- Exploring concepts and using:
  - Undersampling
  - Oversampling
  - SMOTE
  - Near Miss
  - Ensambles
  - TomekLinks

  and so on...

Trying More models.

- Decision Tree and Random Forest
  - XGBoost
  - AdaBoost
- Support Vector Classifiers
- Deep Learning Based Models
  - Neural Networks
- Ensamble models:
  - Combination of DT-RF and Neural Networks



YAHOO!

Troll Detection Challenge

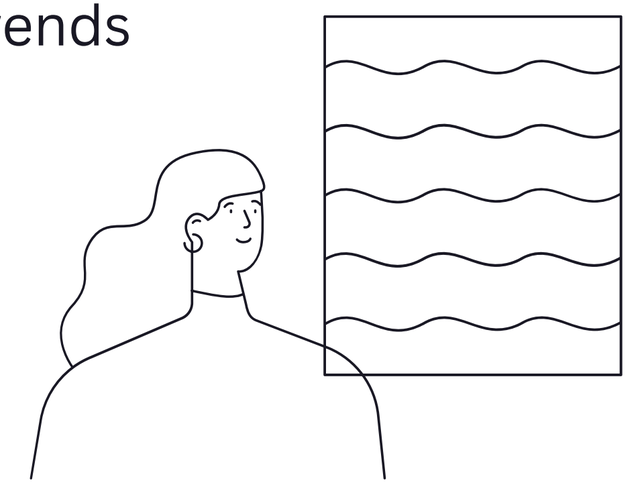
Evaluation 2

Team East India Company
Kaushik Mishra (IMT2020137)
Monjoy Narayan Choudhury(IMT2020502)

Some feature construction

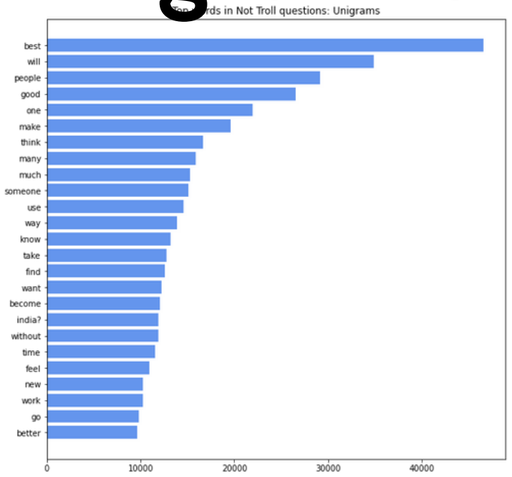
We constructed more features that might help to observe some trends

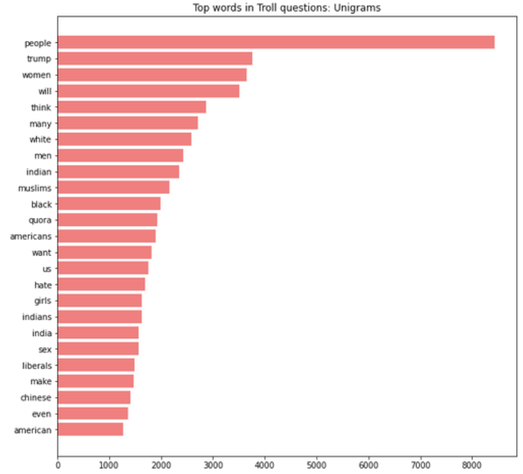
1. Number of Words
2. Number of Capital letters
3. Number of special characters.
4. Number of numerical values.
5. Number of characters.
6. Number of stopwords.

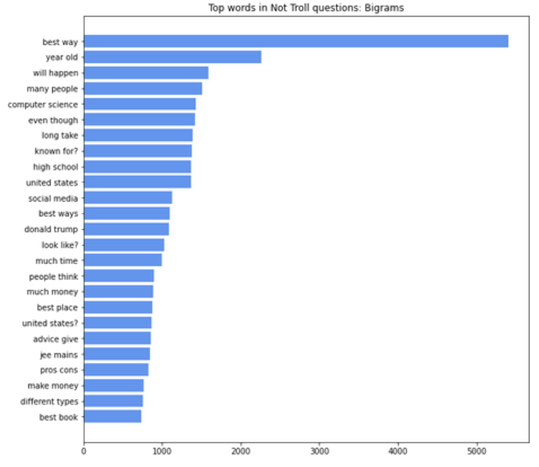


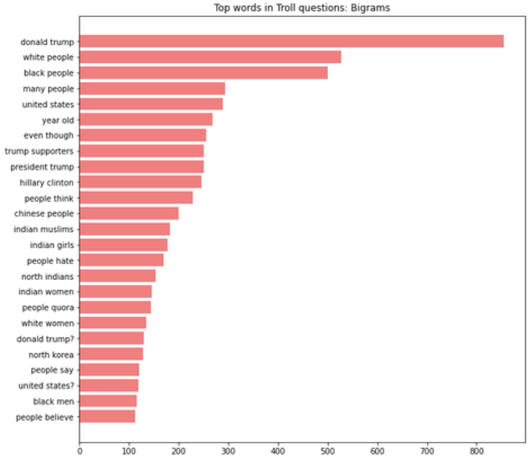
03

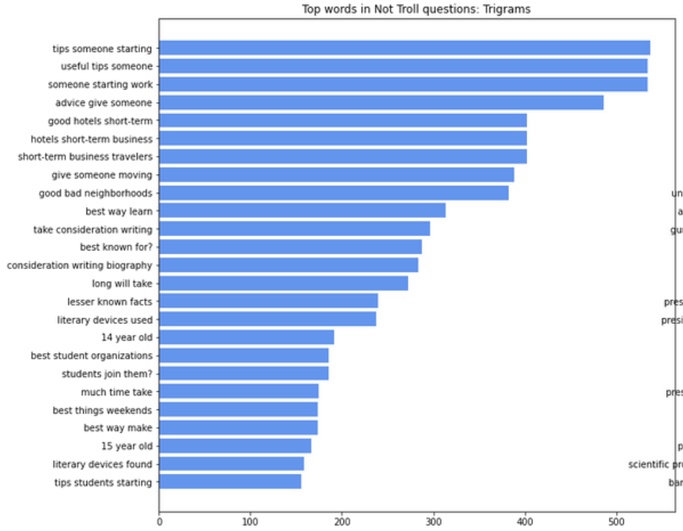
Some more visualizations ngram of words in both classes

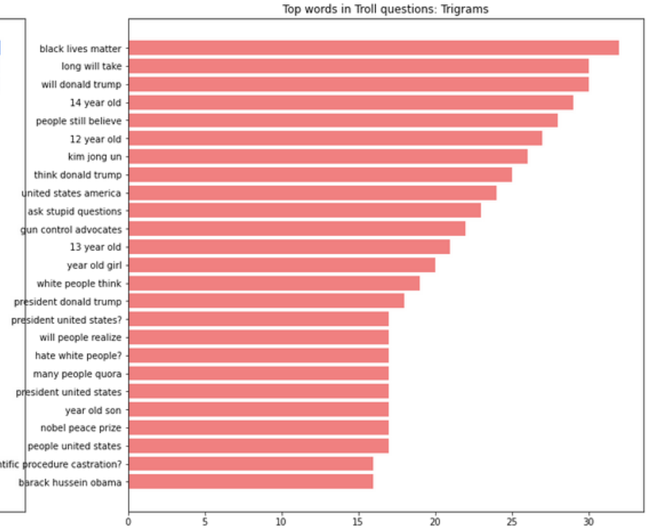




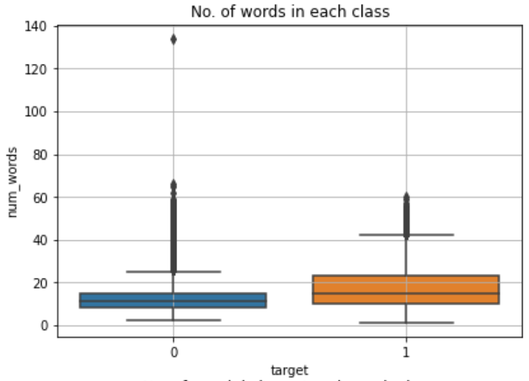


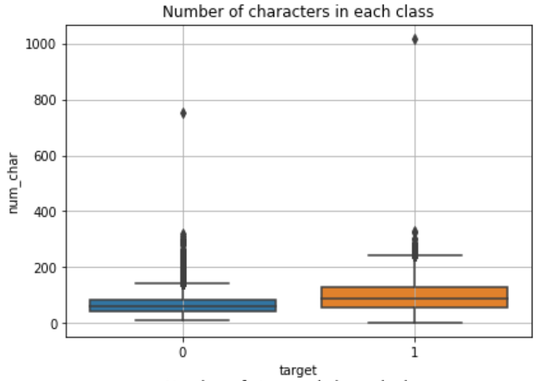


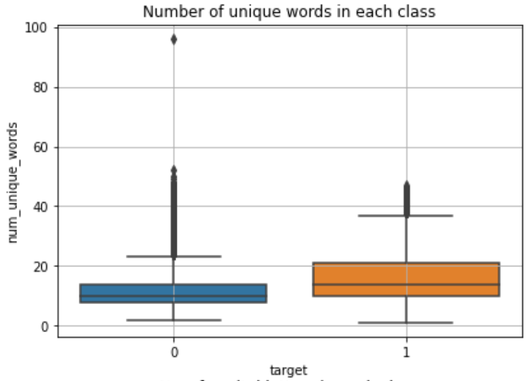


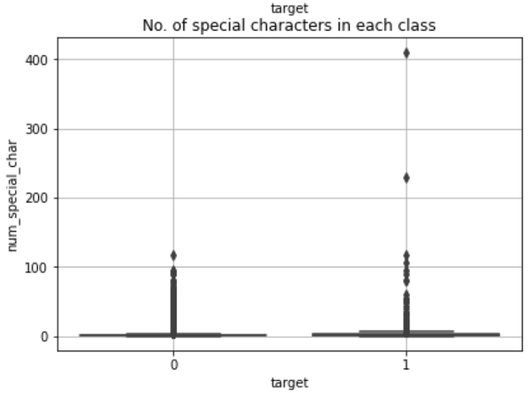


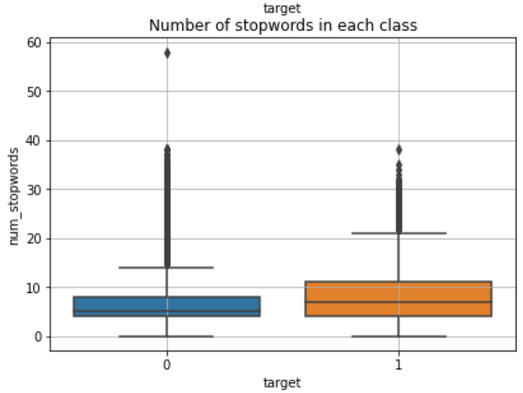
Boxplots on constructed features and some more observations:

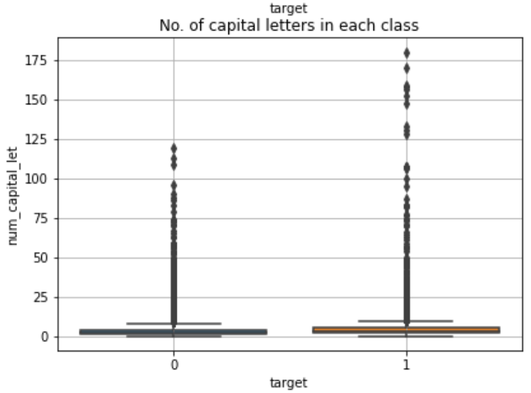












- In troll question the number of words is more
- In troll question the number of unique words is also more
- The same goes for the number of stopwords too
- Also when we queried for the list of all symbols we observed
  - A lot of mathematical symbols (most of which were called troll)
  - Emojis
  - Punctuations

SOME MORE PREPROCESSING

- Foreign Language Detection
  - language code : ['th', 'mr', 'ta','el', 'ja', 'vi', 'bn', 'ar', 'ru', 'bg', 'hi', 'unknown', 'zh-cn'] are properly isolated
  - There are some misclassification too done by the spacy_langdetect

- TF-IDF was used with ngram=(1,4)
- We got rid of punctuation removal

Some comments/Response to Questions

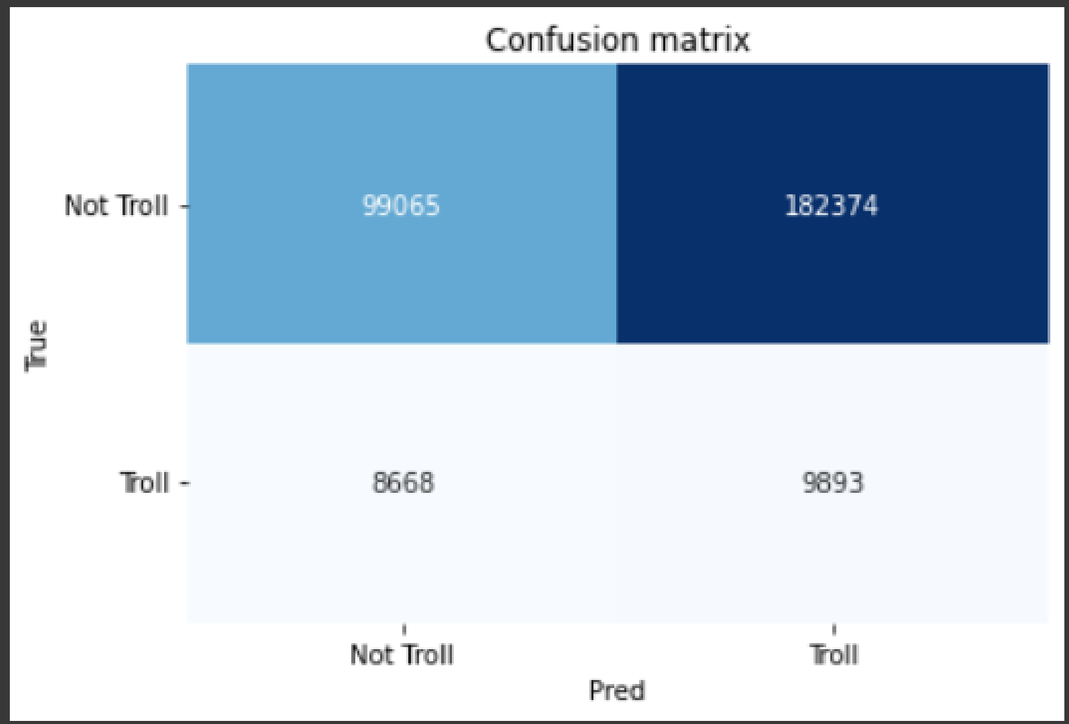
- We made an attempt for spellcheck and correction, However, the model ran on for so long that we had to stop it
- A similar story followed for translating the foreign language found. We tried google_trans 3rd party API which is unfortunately deprecated. The official Google Translate API is paid and offer only some.
- Till Now Logistic Regression is giving the best test accuracy in Kaggle as well as in training.
- No pre-processing of data has rather helped to increase the f1-score.
- TF-IDF didn't help much.
- Word2Vec was also tried but gave horrible results. This is due to the restriction on window size as well as the removal of features like punctuations which we showed are actually important.

Models Tried

Support Vector Classifier

- Parameters:class_weight="balanced", kernel='linear', cache_size=2000, max_iter=5000
- f1 score = 0.48

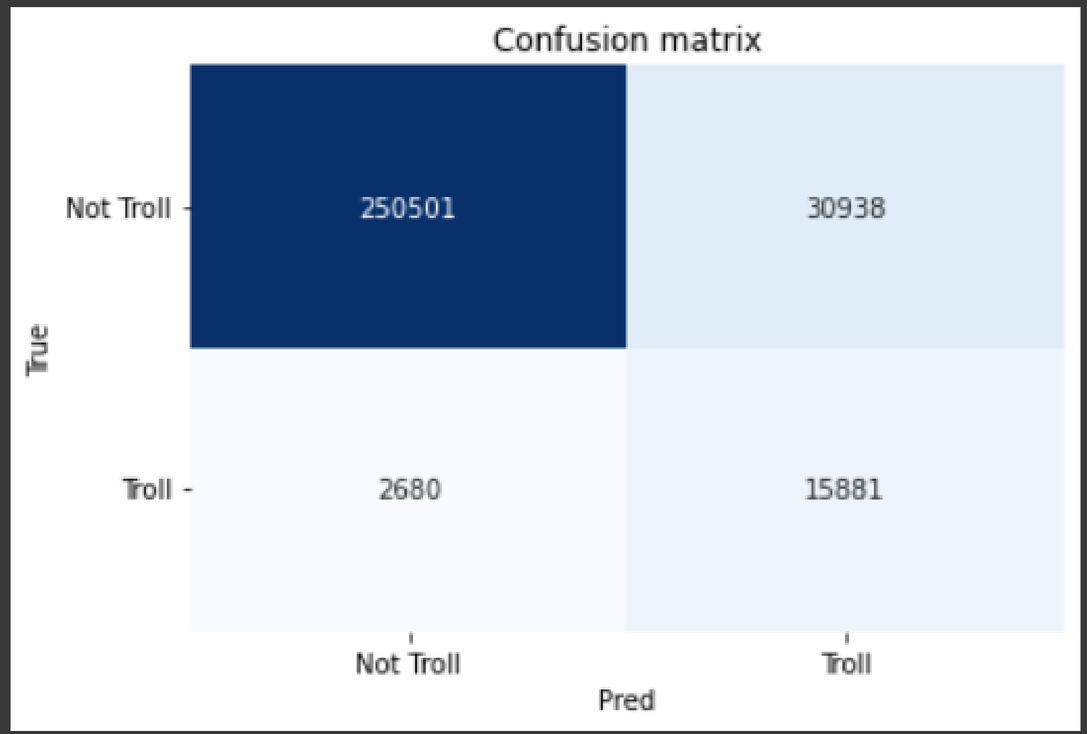
	precision	recall	f1-score	support
0	0.92	0.35	0.51	281439
1	0.05	0.53	0.09	18561
accuracy			0.36	300000
macro avg	0.49	0.44	0.30	300000
weighted avg	0.87	0.36	0.48	300000



Support Vector Classifier (Undersample)

- Parameters:class_weight="balanced", kernel='linear'
- f1 score = 0.91

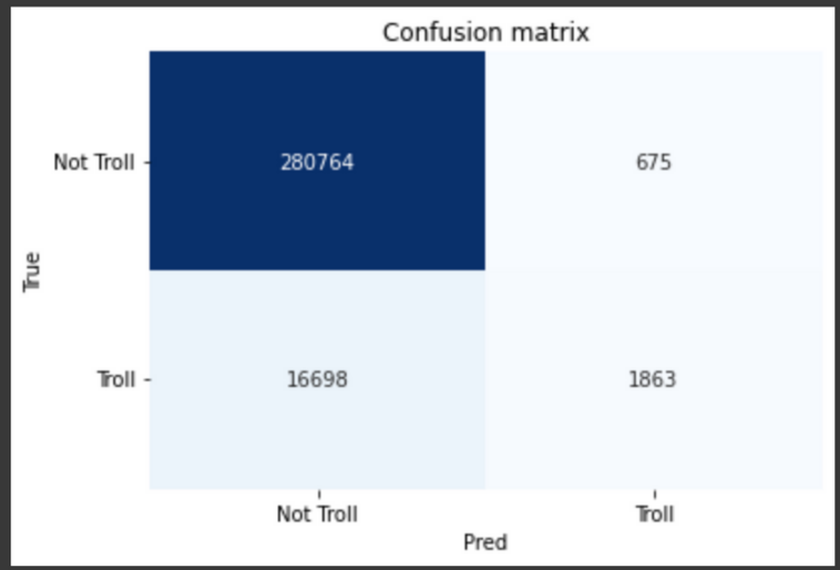
	precision	recall	f1-score	support
0	0.99	0.89	0.94	281439
1	0.34	0.86	0.49	18561
accuracy			0.89	300000
macro avg	0.66	0.87	0.71	300000
weighted avg	0.95	0.89	0.91	300000



Random Forest

- Took a lot of time to train (4-5hours)
- f1 score = 0.92 but poor classification for Troll

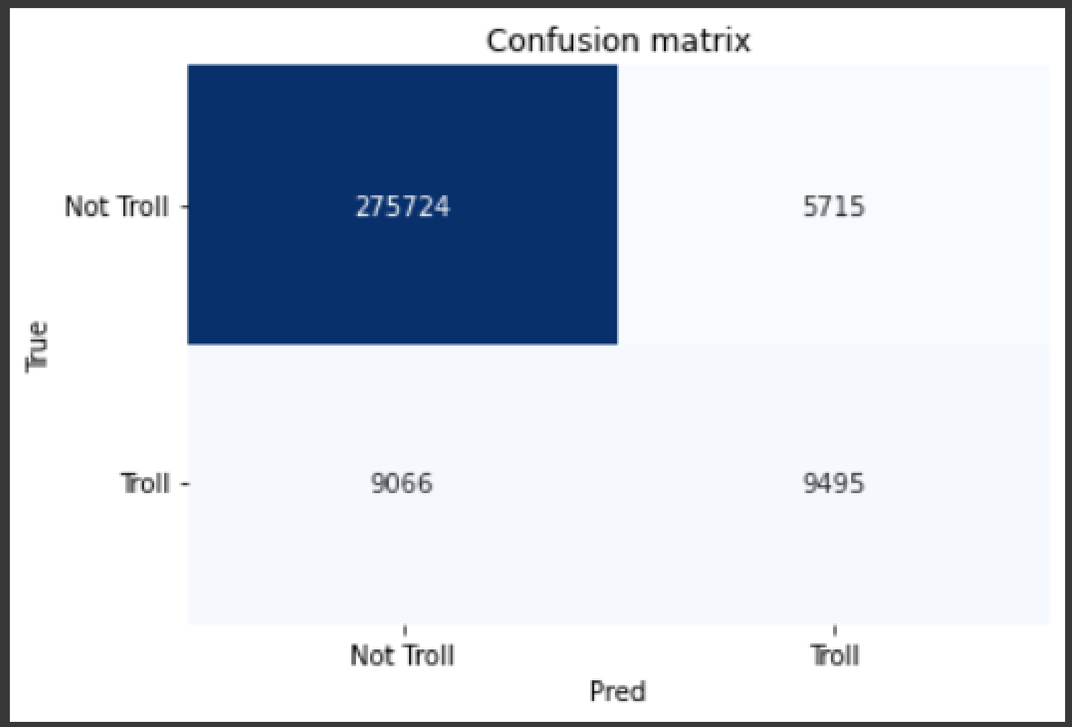
	precision	recall	f1-score	support
0	0.94	1.00	0.97	281439
1	0.73	0.10	0.18	18561
accuracy			0.94	300000
macro avg	0.84	0.55	0.57	300000
weighted avg	0.93	0.94	0.92	300000



Stochastic Gradient Descent Classifier

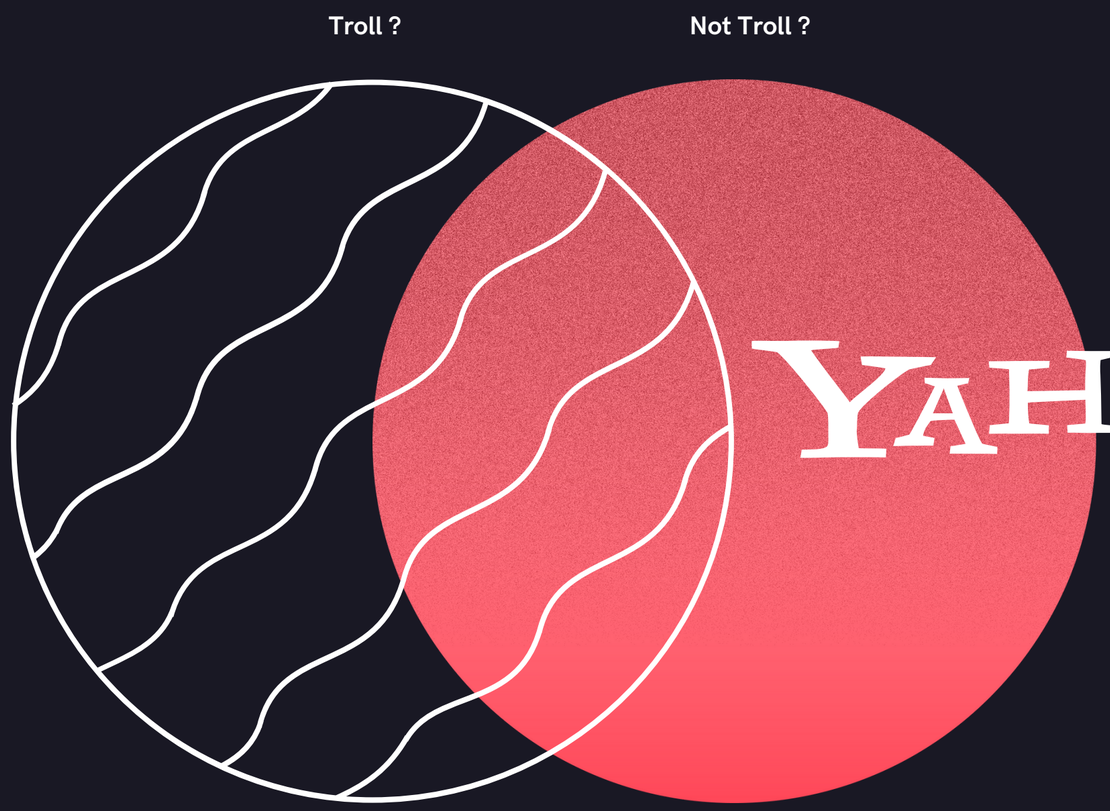
- Parameters:loss='hinge', penalty='l2', alpha=0.000001, max_iter=1000, tol=0.00001
- f1 score = 0.95

	precision	recall	f1-score	support
0	0.97	0.98	0.97	281439
1	0.62	0.51	0.56	18561
accuracy			0.95	300000
macro avg	0.80	0.75	0.77	300000
weighted avg	0.95	0.95	0.95	300000



Future Plans

- Try to tune Logistic Regression better: We still are getting better results with this on trying hyper-parameter tuning
- Try to fine-tune decision trees and random forests: if the train-time can be reduced with some tuning.



YAHOO!

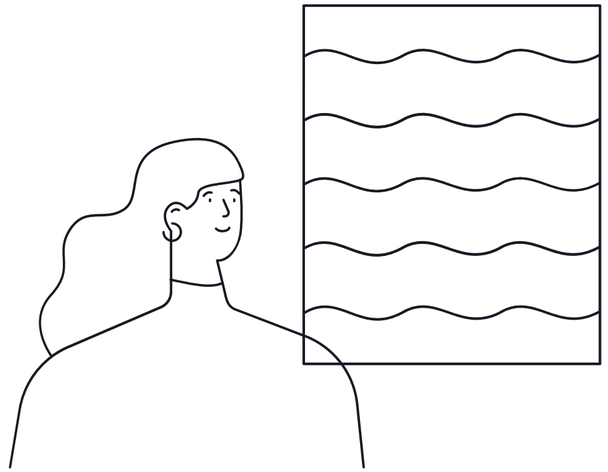
Troll Detection Challenge

Evaluation 3

Team East India Company
Kaushik Mishra (IMT2020137)
Monjoy Narayan Choudhury(IMT2020502)

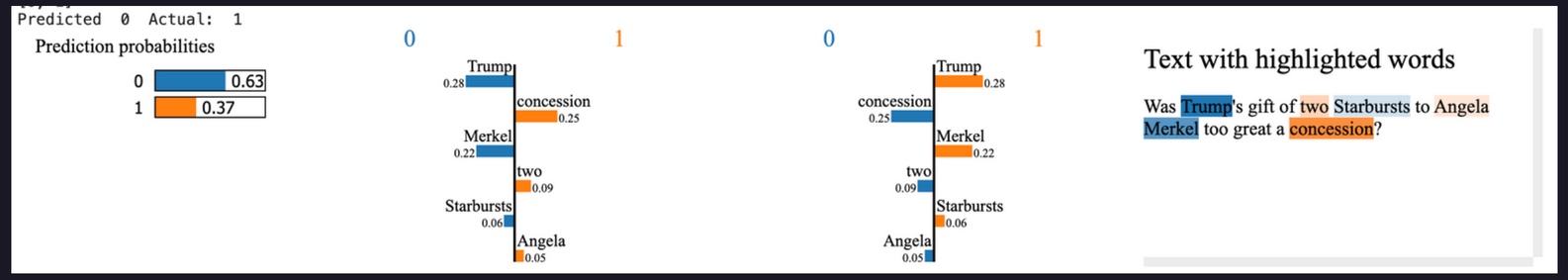
Final Week Struggles

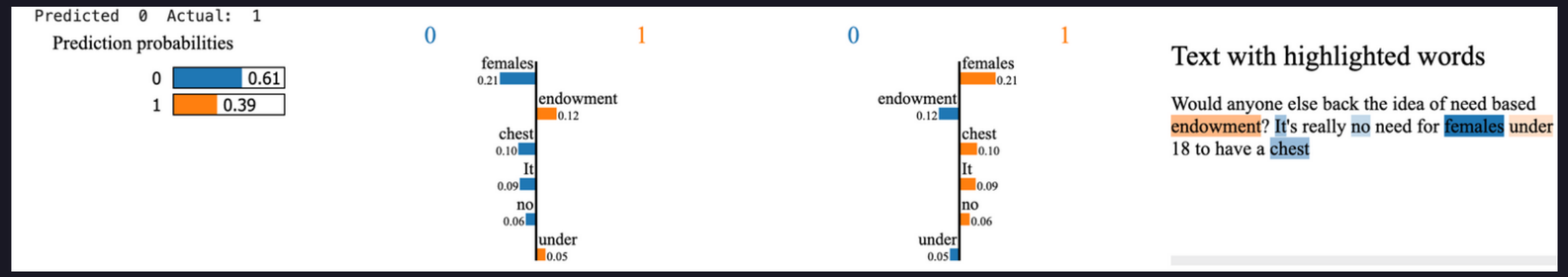
- In the last week, we came up with 3 approaches to boost our standings. We shifted from 14th to 2nd in the public leaderboard and our final position is 4th after the complete data test
- Our results differ from the 1st place result by 0.003.
- We tried to conduct a detailed analysis of where the model fails

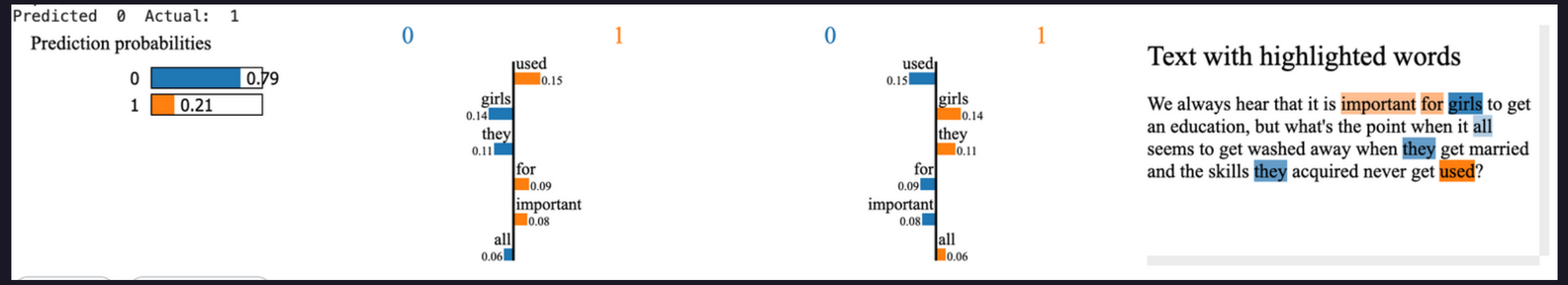


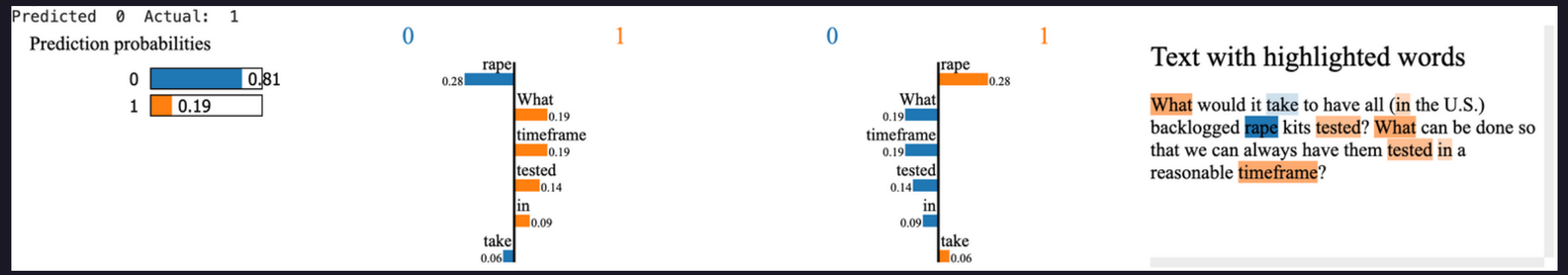
Digging out the misclassification

Prediction = Not Troll , Actual = Troll



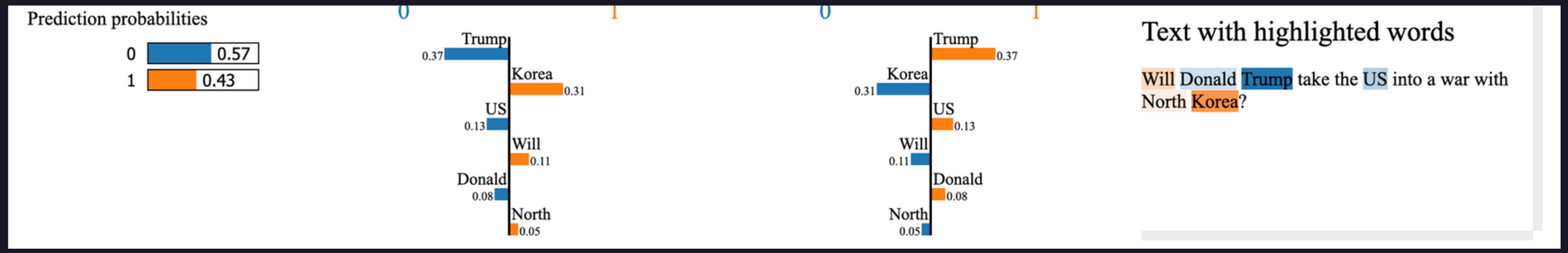


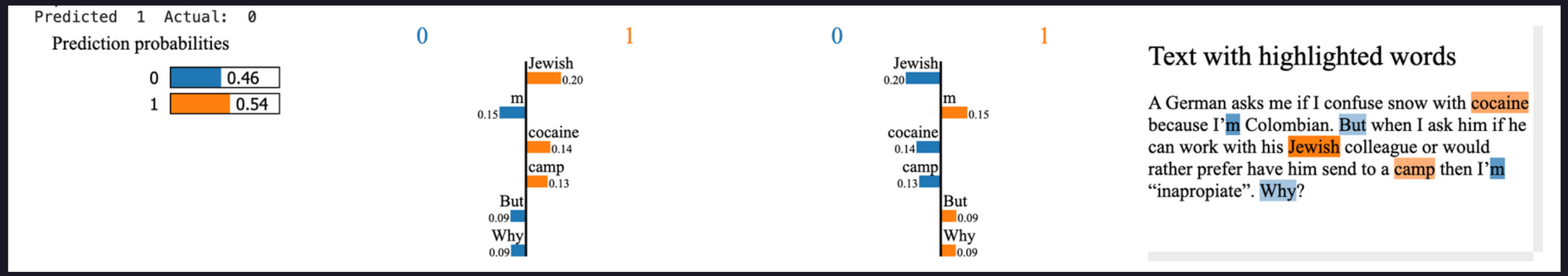


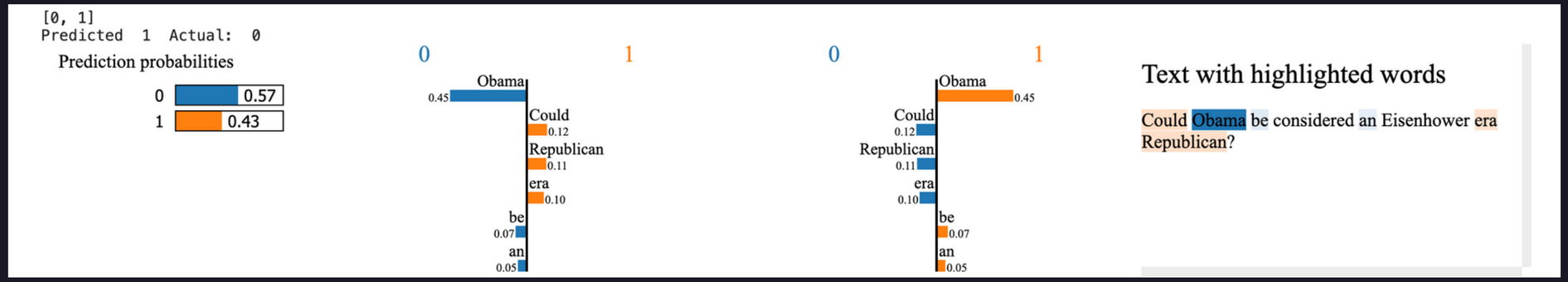


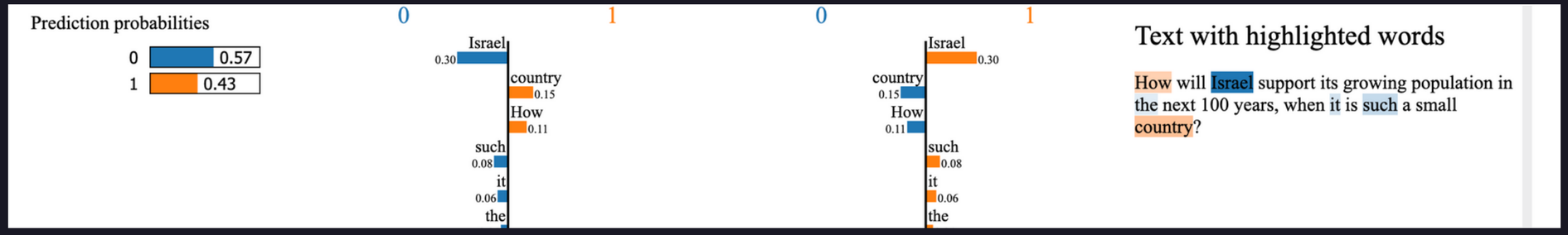
1. Proper nouns throw the model off balance
2. The model fails to take long statements where it is made of 2 statements which makes no sense.
3. Need to include domain knowledge to decide a good threshold
4. In the last example model clearly fails to take in the factor that there were more words citing it to be troll and considered an extreme word "rape" to be involved with something not-troll to be the decider

Prediction = Troll , Actual = Not Troll









Threshold Probability plays a role in misclassification mostly in this case

Random Forest tuning

We tried to tune random forest but we timed out for a single instance on Colab.

Also, we had put more resources towards tuning logistic regression and had stumbled upon a Naive Bayes Transformer solutions too and hence we didn't explore this much

THE 3 APPROACH TOWARDS IMPROVEMENT

- CountVectorizer fitted over both train and test data and transformed respectively and custom class_weight values were used.
- A one of a kind Naive Bayes Transformation of data on which log-reg was applied on.
- TF-IDF stack up with custom class_weights as well as probability threshold parameter **(THE BEST SUBMISSION!)**

Approach 1

- No Preprocessing
- CountVectorizer(ngram = (1,4))
- Fit over both train and test set merged into 1 dataframe
- Logistic Regression with parameters: class_weight= {0:0.4,1:0.4},max_iter=1500, solver='lbfgs', penalty = 'l2'
- Predicted probabilities with custom threshold = 0.2 (gave the best at that time) -> found using binary search.
- f1 on kaggle : 0.6474

Approach 2

We didnt go ahead with this much as we didn't understand the approach in its entirety which is an injustice to the elegant approach

- Extra spaces were removed
- TFIDF vectorizer with parameter : ngram_range=(1,4), tokenizer=tokenize,min_df=3,max_df=0.9,strip_accents='unicode',use_idf=True,smooth_idf=True,sublinear_tf=True
- Naive Bayes Transformer applied
- Log-reg applied with parameters : solver='lbfgs', dual=False, class_weight='balanced', C=0.5, max_iter=300
- Threshold for probability = 0.7
- f1 = 0.64087

Ref:
1.https://direct.mit.edu/coli/article/44/1/187/1585/Bayesian-Analysis-in-Natural-Language-Processing

2. https://www.kaggle.com/code/ryanzhang/tfidf-naivebayes-logreg-baseline/script

Approach 3 (Approach which got us in the place we are right now)

- No Preprocessing
- TFIDF Vectorizer stacked based on analyzer = word and char, ngram = (1,3), max_df = 0.8, max_features = 150000
- Logistic Regression with parameters: class_weight= {0:1,1:2},max_iter=300
- Almost a full data train as train_test split of 0.01 was used
- Predicted probabilities with custom threshold = 0.4 (gave the best at that time) -> found using binary search.
- f1 on kaggle : 0.65476 (the best we achieved)

Lessons Learnt

- Hyperparameter Tuning helps a lot
- Parameter tweaking can be a long process but does lead to interesting results
- Knowing/ Understanding some math behind the model always helps
- Achieving a perfect model is not possible (as we see how the threshold itself can correct a lot of examples and lead to wrong at the same time.
- Feature construction can help to analyse the nature of data and decide on pre-processing steps
- Pipelines and custom transformers can do miracles in NLP.