Why So Harsh?

AI-511 Machine Learning Mini-Project

Team Clones

Mayank Chadha (IMT2020045)

Shridhar Sharma (IMT2020065)

Table of contents

Preprocessing

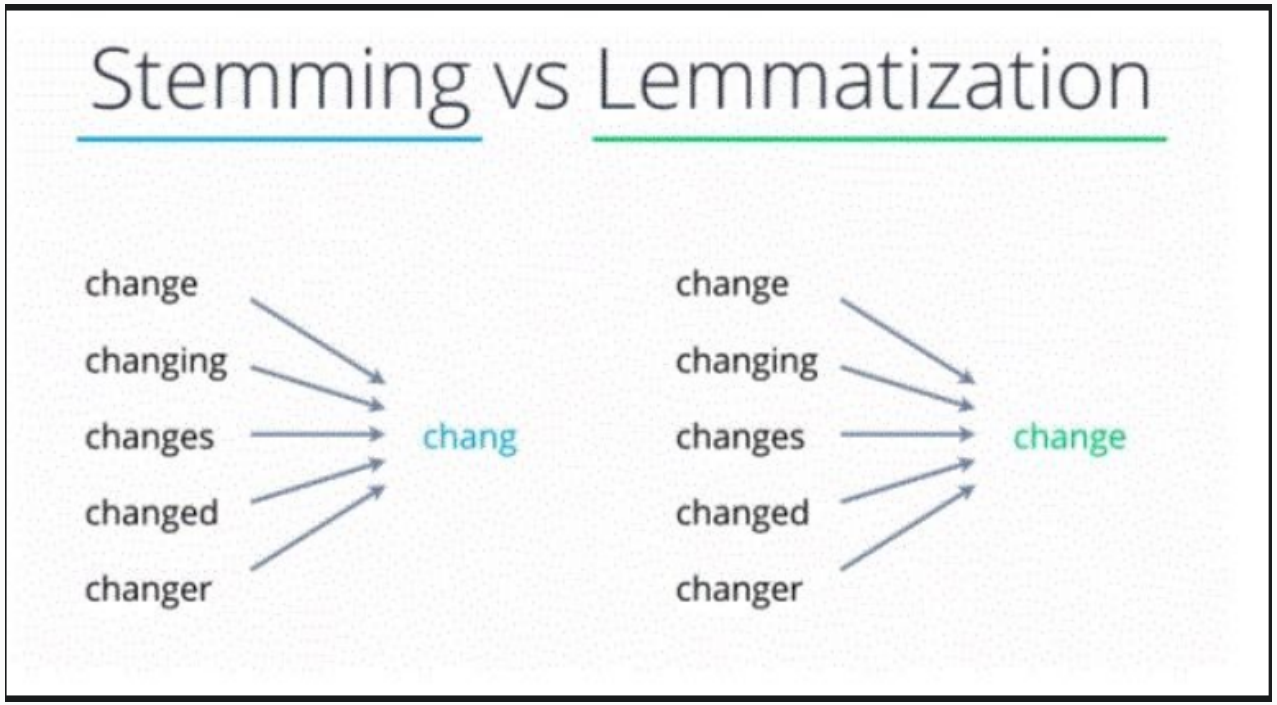
Preprocessing

- Remove URLs, HTML Tags.
- Remove Contractions.
- Remove Numbers and Punctuations.
- Words with three or more consecutive same letters optimized (zzz $\rightarrow$ z) or totally removed (EVAL 3).
- Convert to Lowercase.
- Tokenization (Implemented but not included).
- Lexical Normalization.
  1. Stemming.
  2. Lemmatization(Better Results).
- Remove Stop Words.
- Remove Words of length $<= 2$ (Now Disregarded).

Preprocessing : Lexical Normalization

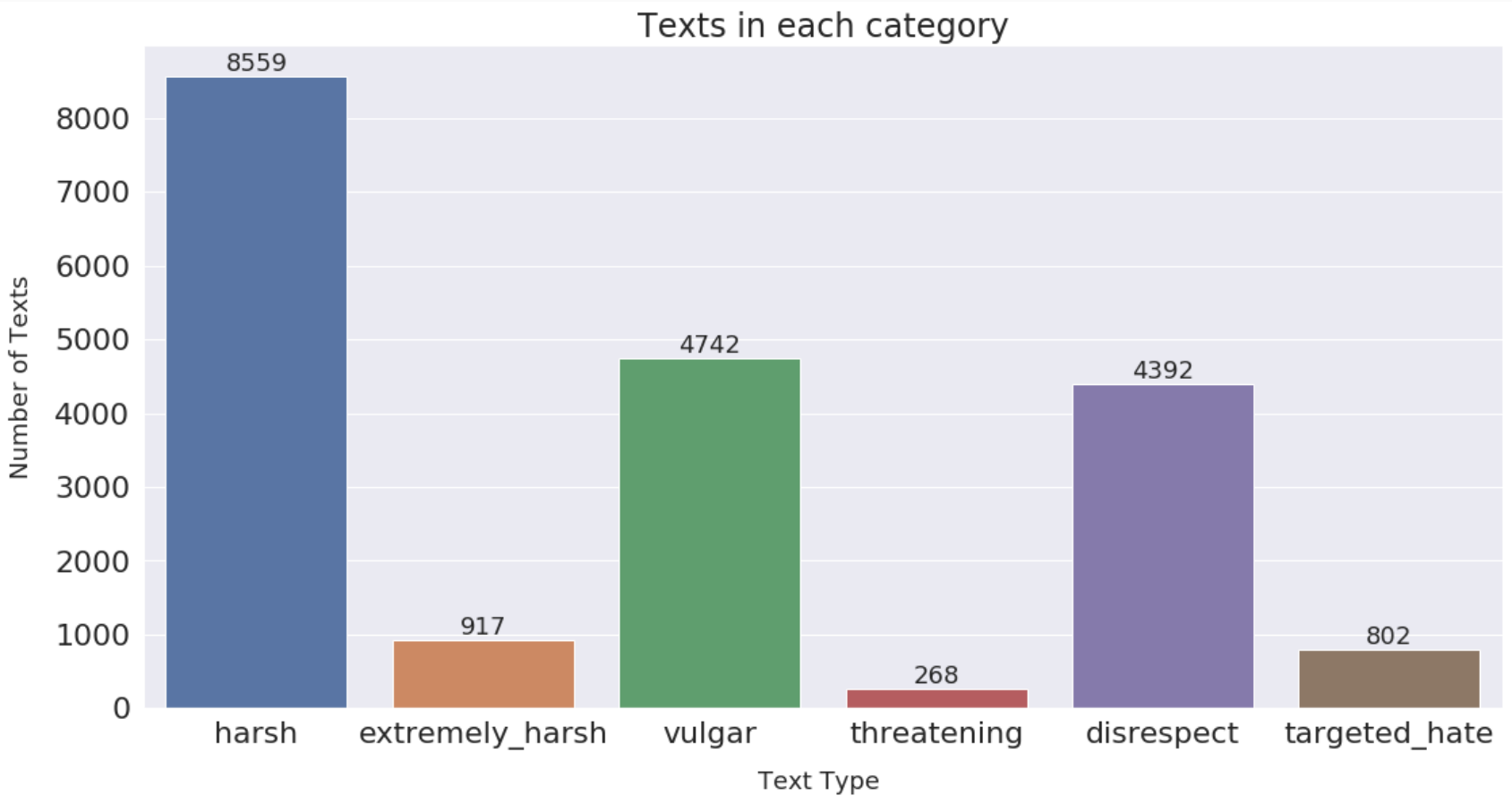
Stemming removes or stems the last few characters of a word, often leading to incorrect meanings and spelling.

Lemmatization considers the context and converts the word to its meaningful base form.



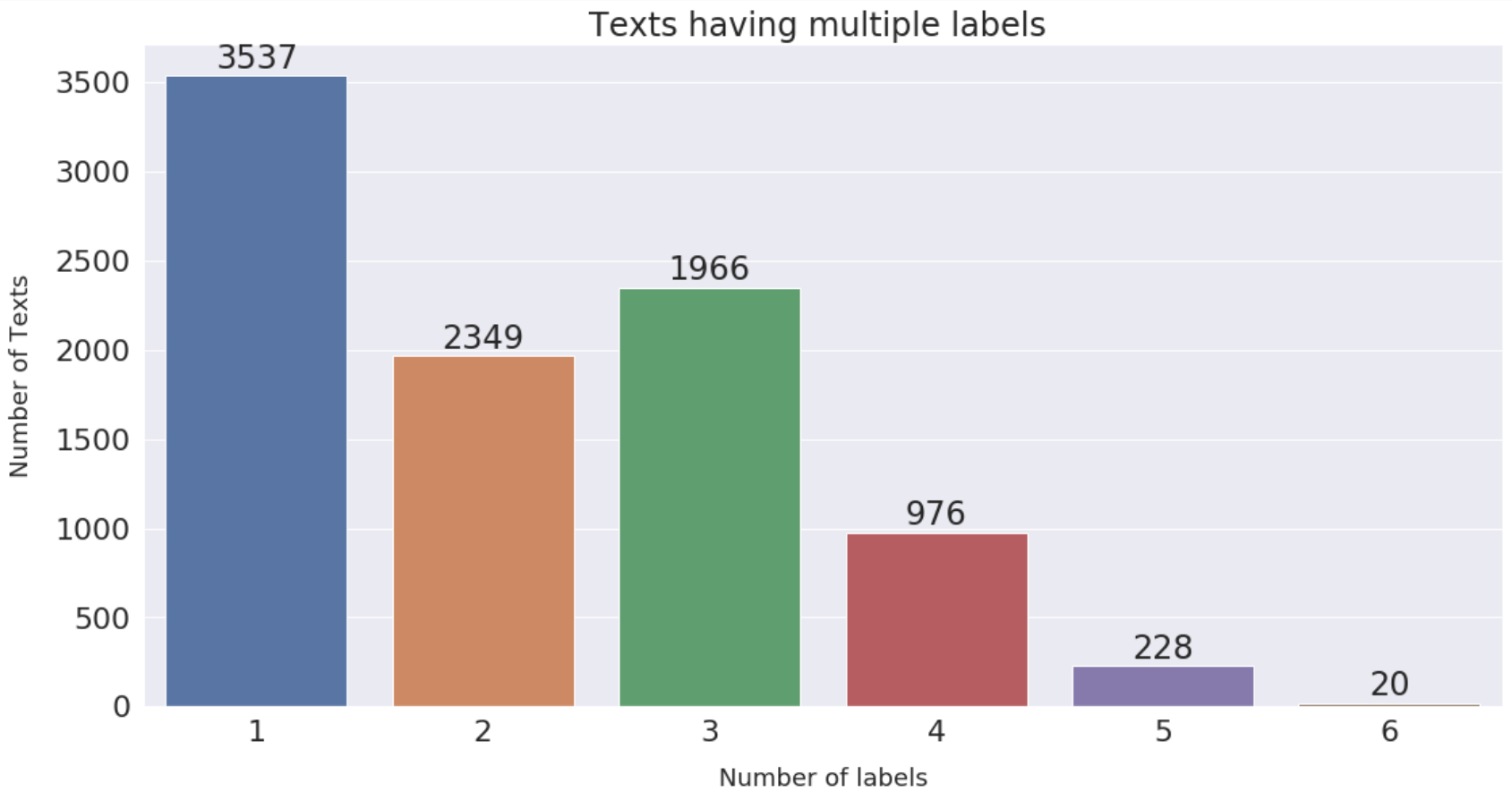
Exploratory Data Analysis

EDA



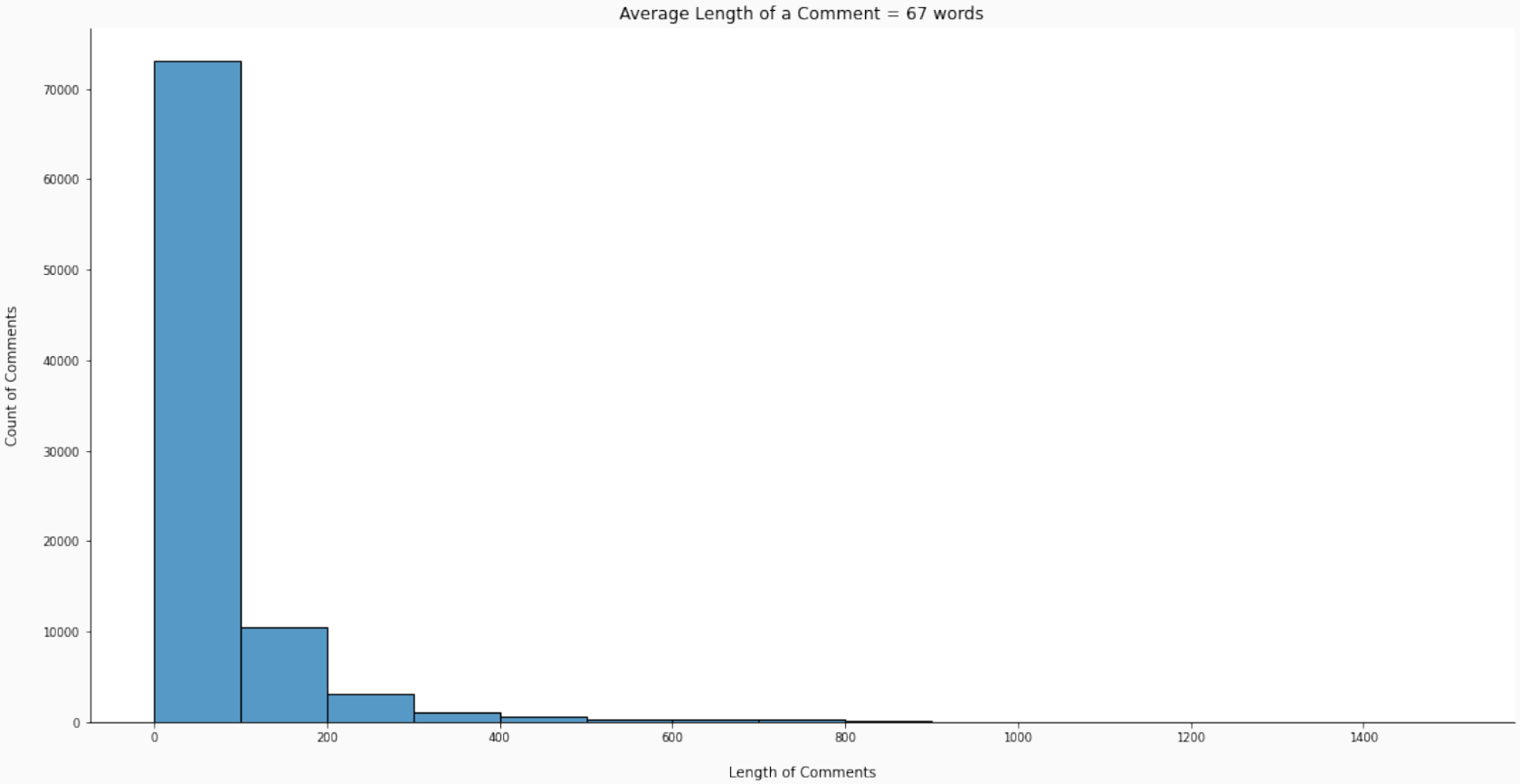
Observation - Our dataset has severe class imbalance. Most conventional classification algorithms perform badly on imbalanced datasets.

EDA



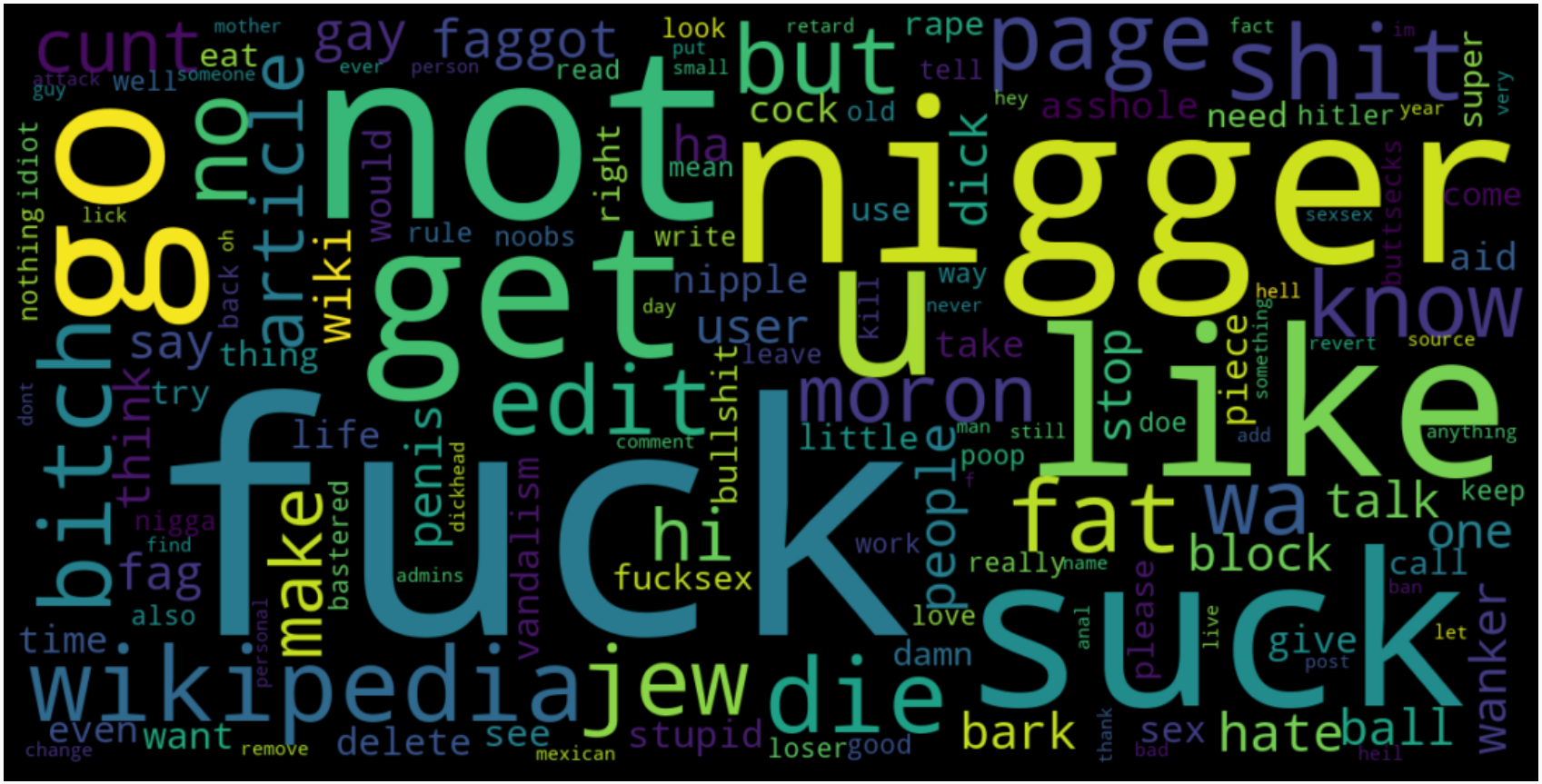
Observation - There are only 9076 'Toxic comments' out of 89,359 total comments. The majority of comments belong to less than 4 classes.

EDA



Observation - Most comments contain less than 100 words (without stopwords removal)

EDA



WordCloud for toxic comments

Text $\rightarrow$ Features

Bag Of Words

Each text is represented as a vector consisting of the frequency or occurrence (0,1) of vocabulary (list of unique words).

Disadvantages -

1. Vocabulary can become too large.
2. Each text vector will contain many 0's which will result in a sparse matrix.
3. Ordering of the words (grammar, meaning) is lost.
4. Words which occur frequently across all text vectors may introduce a bias.

TF-IDF (Term Frequency and Inverse Document Frequency)

TF-IDF kind of normalizes the text vectors. Frequent words in a text vector are "rewarded", but they also get "punished" if they are frequent in other text vectors too. Thus, higher weight is assigned to unique or rare terms considering all text vectors.

$$TF(t, d) = \frac{\text{Count of t in document d}}{\text{Total words in document d}}$$

$$IDF(t) = \log\left(\frac{\text{Total Documents}}{\text{Count of documents containing t}}\right)$$

Note

For now, we are using TF-IDF as it improves BOW by weights. However, even TF-IDF cannot capture the meaning of the text.

BOW vs TF-IDF

	this	movie	is	very	scary	and	long	not	slow	spooky	good
This movie is very scary and long	1	1	1	1	1	1	1	0	0	0	0
This movie is not scary and is slow	1	1	1	0	1	1	0	1	1	0	0
This movie is spooky and good	1	1	1	0	0	1	0	0	0	1	1

Bag of Words

	this	movie	is	very	scary	and	long	not	slow	spooky	good
tf	1/8	1/8	1/4	0	1/8	1/8	0	1/8	1/8	0	0
idf	log(3/3) = 0	log(3/3) = 0	log(3/3) = 0	log(3/1)=0.48	log(3/2)=0.18	log(3/3) = 0	log(3/1)=0.48	log(3/1)=0.48	log(3/1)=0.48	log(3/1)=0.48	log(3/1)=0.48
tf X idf	0	0	0	0	0.022	0	0	0.06	0.06	0	0

TF-IDF

Model Selection & Training

Models Tried

1. OneVsRest
   1.1 Naive Bayes Classifier
   1.2 Logistic Regression(Best Performance)
2. Classifier Chain
   2.1 Logistic Regression
3. Random Forest (Have to explore more)
4. XGBoost (Didn't tune hyperparameters yet)

Pipeline

- Pipelines in Sklearn enable us to sequentially apply a list of transformers (countVectorizer, Tfidftransformer, etc.) and a final classifier.
- Pipelines avoid leakage of test data into training.
- They make hyperparameter tuning easier.

OneVsRest

- Trains one binary classifier independently for each label.
- Assumes that the classes are independent of each other.

Results

Till now, OneVsRest with Logistic Regression (max_features = 30,000) gave us the best AUC score.

Classifier Chains

- A chain of classifiers is trained, in which each classifier takes the predictions of the previous classifier as input.
- Takes into account label correlations.

X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

Results

An ensemble of Classifier Chains with random label ordering gave better results than a single Classifier Chain. However, the AUC score was lesser than OneVsRest. This may indicate that there is no correlation between labels.

Random Forest

- Sklearn's implementation of RF supports multi-label classification, so we can use it as it is.

Results

Random Forest didn't give us better results. However, we haven't yet performed an exhaustive grid search for optimal hyperparameters.

Progress after First Evaluation

Progress after First Evaluation

- We have to figure out a way to deal with noisy words like 'zzzzzz', 'grrrreeeatttt', and wrong spellings. ✓
- Applied all kinds of sampling whether it is UnderSampling, OverSampling(Better Results) or SMOTE. ✓
- Experiments on TF-IDF and increasing the overall features gave us a significant push in AUC-ROC Score. ✓

Final Feature Selection - TF-IDF

1. Applied TF-IDF on both word and character levels.
2. TF-IDF on words -
    - binary = True
    - analyzer = 'words'
    - max features = 30,000
3. TF-IDF on characters -
    - binary = True
    - analyzer = 'char'
    - ngram range = (1, 4)
    - max features = 30,000
4. Combined both to get 60,000 features for training.

Handling Class Imbalance

- Used library Imbalanced-learn
- Tried various OverSampling techniques (RandomOverSampling, SMOTE, ADASYN) and Undersampling techniques (RandomUnderSampling, NearMiss)
- A mixture of RandomOverSampling and RandomUnderSampling worked best.

Best Hyperparameters

1. Label extremely_harsh -
   - RandomOverSampler(sampling_strategy = 0.15)
   - RandomUnderSampler(sampling_strategy = 1)
2. Label threatening -
   - RandomOverSampler(sampling_strategy = 0.18)
   - RandomUnderSampler(sampling_strategy = 1)

Cross Validation

- Used Stratified K-Fold Cross Validation (5 folds) on most models.
- In K-Fold Cross Validation we split the whole data into k sets and use k - 1 sets for training and the remaining set for testing.
- For some models which were taking too long to run, we used the holdout method to validate.
- Used iterative_train_test_split, which is used specifically to split multi-label dataset in a stratified manner.

Ridge Classifier

- Converts the target variable into -1 to 1
- Trains a Ridge Regression model with loss function MSE + L2 penalty
- Predicts positive class if the regression output value is > 0
- Doesn't give probability values
- So we had to wrap it in CalibratedClassifierCV, which uses cross-validation to estimate class probabilities.

Note

Best individual classifier

Voting Classifier

Uses majority vote or average predicted probabilities (soft vote) of different models to predict the class labels.

Models used -

1. LogisticRegression(C = param_C[i])
   param_C = [1.4, 1.3, 1.1, 1.3, 0.7, 1.1]
2. CalibratedClassifierCV(RidgeClassifier(alpha = param_alpha[i]))
   - param_alpha = [9, 14, 11, 15.6, 11.8, 20.1]
   - cv = 3
   - ensemble=True

Note

Best overall model and ensemble

Results

Some of the best models we tried -

Model	CV Score	Kaggle Score
Multinomial NB	0.956268	0.9604
Logistic Regression (LR)	0.984865	0.98594
LR with Sampling	0.987731	0.98649
Classifier Chains (10 chains)	-	0.98541
Ridge Classifier	0.988188	0.98691
Random Forest	0.977381	0.97629
Voting Classifier (Ensemble)	-	0.98727
XGBoost	0.97668	0.97732

Some other models like AdaBoost, Perceptron, EasyEnsemble, and Linear Discriminant Analysis gave very bad results so we didn't include them.

Challenges that we faced

- Applying GridSearchCV on RandomForestClassifier.
- Deciding the final preprocessing steps and its sequence.
- Choice of optimal parameters for sampling and type of sampling.
- Couldn't get SVM to run as running it with probabilities = True, requires too much RAM.
- Didn't get time to implement Neural Networks.