Toxic Comment Classification using Machine Learning

Aneri Dalwadi AU1940153, Kairavi Shah AU1940177,
Mananshi Vyas AU1940289, Nandini Bhatt AU1940283

***Abstract*—Considering to moderate countless inappropriate or toxic comments on social media manually is a serious impractical approach. Thus, in our project, we implement this using and comparing various ML models. The data is downloaded from Kaggle[1], containing 1.5 million comments from wikipedia's talk page and has to be implemented with multiclass-classification - toxic, obscene, identity hate, severe, etc. For the complete process till choosing the best model, we implement text vectorization, tf-idf calculations and basic use of linear algebra to get the reduced dimension of the vector form of out input. We then apply ML models like XGBoost, LGBM, CatBoost, etc. Models are tuned to give optimal test results. We infer that LGBM outperforms the other models. We would be discussing more about how we reached to this conclusion in later sections.**

.

***Index Terms*—dimension reduction, vectorization, tf-idf, eigen values, accuracy, multilabel classification, machine learning algorithms, toxic comments**

I. Introduction

SOcial media sites are one of the most popular websites on the internet today flooding with comments. It is vital to manage the user-generated offensive content on many of these sites that can make a user's online experience unpleasant, and we may want to filter the same (important for parents as to what their child reads online). Most researchers tried solving this problem using NLP using a Neural Network but we as a group avoid NLP and implement this using the crux of linear algebra (feature selection), statistics(model selection), ML models (classification), Data Science concepts (data collection, preprocessing) and mathematics (tf-idf value calculations).

Using different Machine learning classifiers we try to detect it into various types of toxicity categories.

Since ML models work with numbers, we need to convert our data into vector format after cleaning it. Starting briefly, we then split them into tokens: single, double and triple word tokens, 2, 3 and 5 gram, single character shingles, 2, 3 and 5 character shingles. We calculate the tf-idf value for all these combinations and analyze them using ggplot. We finally use the combinations of 2 character shingles to make a 3D matrix and fill in the previously calculated tf-idf values of the same in the matrix. We then fuse the concepts of linear algebra like eigenvalues and eigenvectors for dimensionality reduction on this square matrix. We store the top 13 eigenvalues and we use them as our features. We then use models for classification to train and test our model. The end goal of this project is to create a better, safer and appropriate environment online. In the long term, this would allow people to better connect with each other in this increasingly digital world.

II. Literature Survey

In past years the classification of toxic comments has been done by various researchers extensively. These researchers have applied different machine learning algorithms to classify the comments. Following are some the papers:

- In this paper [2], authors have used SVM with TF-IDF as the feature extraction and Chi Square as the feature selection. The best performance obtained using the SVM model with a linear kernel, without implementing Chi Square, and using stemming and stopwords removal with the F 1 Score equal to 76.57%.
- In another paper[3], authors have used a semi-supervised approach to detect profanity-related offensive content on Twitter. They achieved a 75.1 % TP rate with Logistic Regression and a 69.7 % TP rate with popular keyword matching baseline. The false-positive rate was identical for both at about 3.77%.

III. Implementations

Here we present our thinking and complete implementation along with explaining several concepts in brief. We begin with gathering the dataset and problem statement [1] from Kaggle.

We undertake data cleaning using Apache Spark (pySpark library) and regex and then we split the dataset into documents where each document is a comment. Each comment is split into tokens of single, double and triple word tokens, 2, 3 and 5 gram, single character shingles, 2, 3 and 5 character shingles using language R and its library (tidyverse). We focus on observing the patterns for the tf-idf values for these using ggplot. After understanding the importance of each token type, we choose 2 character shingles. Upon increasing the value of n-gram and increasing the number of character shingles our AUC score for all the models started decreasing till the extent that it reached 0.32 and hence we decided to move further with 2 charachter shingles.

For calculation purpose we had to create a 3D matrix using all the 2 character shingles. Following is the description of the dimensions(d) D1, d2 - characters from a-z D3 - above 2D matrix for each comment.

Now, since we have already created a matrix with the tf-idf values we use eigen values linear algebra for dimensionality reduction. Eigen values help us to determine the factor by

which my vector will change upon change in any input (2-character shingles). With motive of base check of our method, we begin with getting the 13 most significant eigen values and trained each gradient boosting model on it. We further add the number of eigen value as parameter which have tuned. We tried taking top 3, 5, 10, 13 and 15 eigen values and 13 proved out to be the optimum value. Further, we use these 13 values as from each comment as our integral input features. Now, we train our models using various Base Models such as Linear Discriminant Analysis, Logistic Regression, Qudratic Discriminant Analysis and then we trained Gradient Boosting models such as XGBoost, CatBoost and LGBM.

IV. Modelling

Base models:

- Logistic Regression
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)

We specifically train 3 major models apart from other 3 base models:

- Extreme Gradient Boosting (XGBoost)
- CatBoost
- Light Gradient Boosting Model (LGBM)

The first task was to establish a baseline model and after interpreting the results of the base models we train the aforementioned models. The justification for choosing these models is as given below -

Extreme Gradient Boosting (XGBoost)

- Reasoning:
  - This a highly popular choice used for both regression and classification models
  - It uses sequentially-built shallow decision trees to provide accurate results and a highly-scalable training method that **avoids overfitting**.

Light Gradient Boosting Model

- Reasoning:
  - LGBM is significantly **faster** than XGBoost.
  - It build trees in iterations, and each new tree is used to **correct the "errors"** of the previous trees.
  - Capable of handling **large-scale data** and lower memory usage.

CatBoost

- Reasoning:
  - It's boosting algorithm works and gives predictions in very less time thereby rendering fast model services.
  - Its boosting schemes helps to reduce over fitting and improves quality of the model.
  - CatBoost is about considerably faster than XGBoost.

V. Interpretation and Results

We look at the trends of how the AUC score changes with the number of features and varied tuning for each model

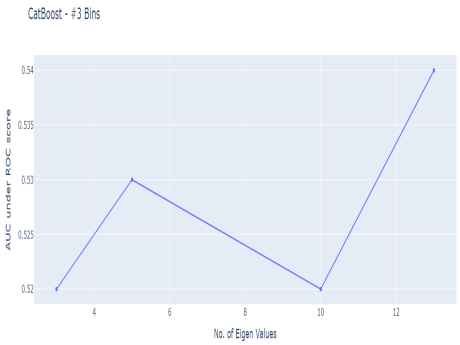

Figure 1
In CatBoost model for 3 bins we see that the highest AUC score achieved is around 0.54 for 13 eigenvalues.

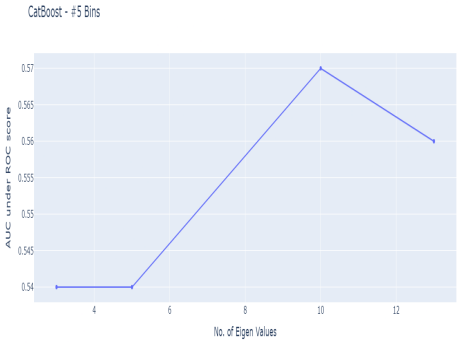

Figure 2
In CatBoost model for 5 bins we see that the highest AUC score achieved is around 0.57 for 10 eigenvalues. Also, the trend is not monotonically increasing but rather wiggly.

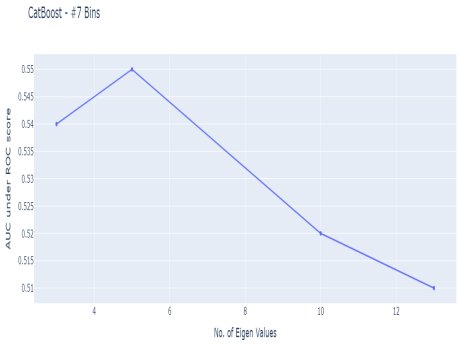

Figure 3
In CatBoost model for 7 bins we see that the highest AUC score achieved is around 0.55 for 5 eigenvalues. It is interesting to note that the value reaches its peak at 5 features and then decreases.

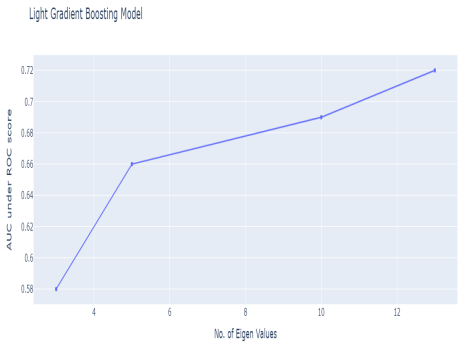

Figure 4
In LGBM model we see that the highest AUC score achieved is around 0.72 for 13 eigenvalues.

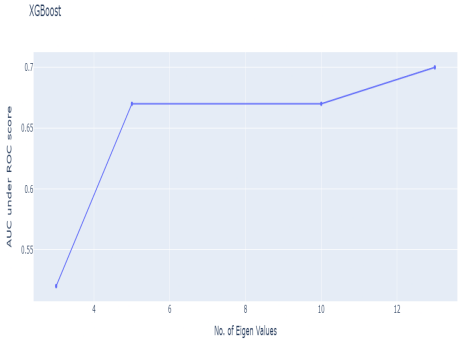

Figure 5
In XGBoost model we see that the highest AUC score achieved is around 0.7 for 13 eigenvalues.

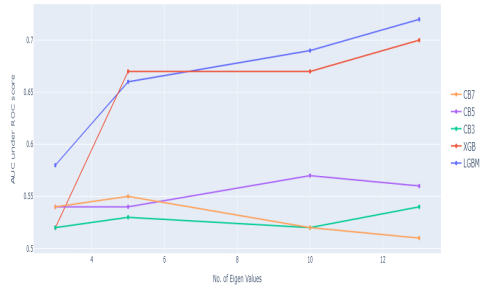

Figure 6
Here is the combined trend of how the AUC score changes upon changing the number of eigenvalues for all the aforementioned models.

Model Used (Time Taken in sec & hrs)	Number of eigen values 3	5	10	13
LGBM (AUC)	0.58 (12600s [3.5 hrs])	0.66 (16200s [4.5 hrs])	0.69 (21600s [6 hrs])	0.72 (23400s [6.5 hrs])
XGBoost	0.52 (18000s [5 hrs])	0.67 (21600s [6 hrs])	0.67 (28800s [8 hrs])	0.70 (34200s [9.5 hrs])
CatBoost (No. of bins): 3	0.52 (10800s [3 hrs])	0.53 (16200s [4.5 hrs])	0.52 (25200s [7 hrs])	0.54 (28800s [8 hrs])
CatBoost (No. of bins): 5	0.54 (12600s [3.5 hrs])	0.54 (16200s [4.5 hrs])	0.57 (26280s [7.3 hrs])	0.56 (28080s [7.8 hrs])
CatBoost (No. of bins): 7	0.54 (10800s [3 hrs])	0.55 (14400s [4 hrs])	0.52 (25200s [7 hrs])	0.57 (27000s [7.5 hrs])

Figure 7

Tuning of the models is done by changing setting the learning rate, tree depth, number of leaves(LGBM), number of bins(CatBoost), changing the character shingles and tokens and also as we are using eigenvalues to be our feature, we tune different models by varying the number of eigenvalues considered. It is quiet evident from the table the LGBM outperforms other models. The reason why we came to this conclusion is as discussed below -

- For CatBoost model, our hyperparameters are the depth of the tree, number of bins and number of eigenvalues. As we can observe the AUC score doesn't change very significantly upon changing the number of bins and the best score it gives is 0.57 which is not significant score. In addition to this poor AUC score, it took 7.5 hours for the model to get trained and hence we chose to discard the model.
- In XGBoost model, our hyperparameters were the depth of the tree, learning rate and number of eigenvalues. As we can observe the model gives us an AUC score of 0.7 with 13 eigenvalues but since it took 9.5 hours for the model to get trained we wait to see other model to choose the best one.
- Lastly, in the LGBM model, our hyper-parameters are the depth of tree and number of eigenvalues. It is evident that the model gives the highest AUC score of 0.72 for 13 eigenvalues. The AUC score for 10 and 13 eigenvalues are slightly significant but the time taken to train the model for 13 eigenvalues was just slightly more than for 10 eigenvalues. Hence, for us as this time was insignificant we chose the Model providing us the highest AUC score and getting trained in comparitively less amount of time.
- We note that the AUC score of XGBoost and LGBM for 13 features is significantly comparable with only difference of 0.02. However XGBoost takes nearly 3 hours more to run, which helps us to clearly choose LGBM as better model

VI. Conclusions

Our initial training dataset was very huge and as a result when we do some preprocessing qnd proceed for creating n-grams or character-shingles, we end up with a massive dataset that was impossible to handle on our local systems. So, to justify a proof of concept, we use a random sample of a size that provides us enough playground to experiment, implement and understand several models and feature selection. This tradeoff of the datasize helps us explore various modelling techniques along with smooth preprocessing as well. In our data, we initially saw that AUC under ROC curve of QDA(base model) is high but that of KNN is low which means that our data shows non-linearity between predictors and response but having some information about the posterior distribution might boost our results and so in such situations using models like Gradient Boosting Trees which handle interaction between predictors well can yield high results. Thus, we chose to use Gradient Boosting Models such as XGBoost, LGBM and CatBoost and from the

results we can observe that we achieved a significantly better AUC score with our LGBM model after tuning the model and selecting optimal number of features. Thus, we conclude that LGBM outperforms all other models giving us an AUC under ROC score of 7.2 within time span of 6.5 hours.

VII. Languages and libraries used

1) Apache Spark (pySpark) - Data pre-processing implementing Regex in SQL along with parallelizing the process in Spark
2) R (Tidyverse) - Data manipulation, tokenization, visualizations
3) ggplot - Used in R for visualizations
4) Python - Extracting features, model training, tuning and testing

References

[1] "Toxic comment classification challenge," Kaggle. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. [Accessed: 20-Mar-2022].

[2] K. M. Lhaksmana, D. T. Murdiansyah, and N. S. Azzahra, "Toxic Comment Classification on SocialMedia Using Support Vector Machine and Chi Square Feature Selection," View of toxic comment classification on social media using support Vector Machine and Chi Square feature selection. [Online]. Available: http://socj.telkomuniversity.ac.id/ojs/index.php/ijoict/article/view/552/316. [Accessed: 20-Mar-2022].

[3] G. Xiang, B. Fan, L. Wang, J. I. Hong, and C. P. Rose, Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus. [Online]. Available: https://www.cs.cmu.edu/ lingwang/papers/sp250-xiang.pdf. [Accessed: 20-Mar-2022].

[4] Van Rossum G, Drake Jr FL. Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands; 1995. Available: https://www.python.org/

[5] Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, 2016. Available: https://ggplot2.tidyverse.org

[6] R Core Team (2019). R: A language and environment for statistical¿ computing. R Foundation for Statistical Computing, Vienna, Austria.¿ Available: https://www.R-project.org/.