HIGH SCHOOL EDITION
Journal of Student Research
Volume 13 Issue 2 (2024)

# Twitter Sentiment Analysis Using Machine Learning

Srimayi Gupta[1] and Padmavathy Jawahar[#]

[1]Mountain House High School, USA
[#]Advisor

## ABSTRACT

In an age of social media, online forums, and chats, cyberbullying is a prevalent issue. On Twitter (now X), approximately 500 million tweets are shared per day (Antonakaki et.al., 2021). It is the job of the moderators to ensure these tweets follow standard community guidelines. However, the sheer number of tweets makes it difficult to sort manually and ensure they are following protocol. Sentiment analysis and machine learning algorithms can be used to classify these texts automatically as positive or negative. Normally, these machine learning models are much more efficient and may provide higher accuracy rates in identifying hate speech in Twitter. In this paper, we are exploring the use of five classical machine learning algorithms to classify Twitter hate speech as neutral, racist, or sexist. Model performance was compared after using raw tweet data versus pre-processed tweets through data cleanup. Furthermore, we highlight two methods to deal with imbalanced datasets to improve the prediction rates. Overall, we were able to achieve a 96% accuracy in correctly classifying tweets into the different labels.

## Introduction to Sentiment Analysis

Sentiment analysis is a technique that uses natural language processing (NLP) and machine learning algorithms to categorize text within a dataset as positive, negative, or neutral as per its content (Giachanou et.al., 2016). Sentiment analysis is typically used to sort through customer invoices, product reviews, and social media comments. By doing this analysis, we are able to determine the opinions of people and use these opinions for a variety of purposes. For example, businesses may look into the sentiment of product reviews to identify ways their product can improve. Since these data sets are very large, using a machine learning (ML) model to conduct a sentiment analysis guarantees a more efficient way compared to sorting through text-based data by hand. One question arises: what methods in sentiment analysis must be used to ensure the best prediction rates? When categorizing speech connotation as positive and negative, we desire the most precise results. Numerous factors impact these prediction rates such as the type of prediction model used or the amount of data sampled from a larger population.

For the purpose of this project, we are narrowing our focus to Twitter sentiment analysis where we use natural language processing and machine learning algorithms to categorize the emotional intent of tweets. Performing an analysis on these tweets can identify the prevalence of cyberbullying on social media platforms and creates a stepping stone for a targeted approach in preventing types of hate speech (sexism, racism, etc.). This additionally makes the platform's ability in reporting tweets easier. Since we are in an age of technology dependence, cyberbullying is inevitable but we can use sentiment analysis to minimize its impact as much as possible by censoring hate speech on these platforms.

## Introduction to Machine Learning

Machine learning assists a computer to learn through data driven algorithms. These algorithms can be trained with input data and its classification. Data values are separated into a train test split where a percentage of the data goes towards training the machine learning model and the remaining serves as an accuracy test to see how well the model learned the dataset.

Depending on the rationale, there are different types of machine learning models that can learn from input data and output generalized classifications. For Twitter sentiment analysis, we are using classical machine learning algorithms like Logistic Regression, KNeighbors, Naïve Bayes, and Random Forest for classification of data. As we are using labels to categorize the tweets and train the models, we are using supervised learning. Each model may vary in prediction accuracy, so it is up to the user to select the optimal model based on their criteria.

## Classification Algorithms Used for Twitter Sentiment Analysis

The following classical models were used to perform the Twitter Sentiment Analysis.

### *Logistic Regression*

Logistic regression is a linear algorithm for binary classification. However, there are techniques to extend logistic regression to classify more than two classes. The generated prediction values in logistic regression is between 0 and 1. In the twitter hate speech dataset, a logistic regression model will predict the likelihood of the tweet being neutral, sexist, or racist. Based on relationships between independent data sets, logistic regression will make well-versed predictions for categorical dependent variables (we are measuring the likelihood that a tweet is categorized into neutral, sexist, or racist). Logistic regression can be imported from sklearn library with the command below:

```
>>> from sklearn.linear_model import LogisticRegression
```

### *KNeighbors*

The Neighbors Algorithm is a model used for the classification and regression of data with labels. Given a set of training tweets that are predefined as neutral, sexist and racist, the KNeighbors algorithm compares new metrics to existing trained data to determine its category. For example, assume that the model draws a tweet called tweet #1. The Neighbors algorithm will measure tweet #1 relative to the nearest neighbors (categorization of tweets) to assign tweet #1 a score and determine its category. KNeighbors can be imported from sklearn library with the command below:

```
>>> from sklearn.neighbors import KNeighborsClassifier
```

### *Naïve Bayes*

Classification done through Naive Bayes models are probabilistic predictions that are useful in real-world applications. This machine learning algorithm is most commonly used for face recognition, article classification, and spam filtration. We use the concept of conditional probability to understand the function of Naive Bayes models. Based on the data sample used to train the model, the Naive Bayes algorithm calculates the probability of each label (neutral, sexist, or racist) and uses a test tweet to calculate the conditional probability of each tweet class. Complement Naïve Bayes is best used for imbalanced data sets. Balanced sets use Multinomial Naïve Bayes. Naïve Bayes algorithms can be imported from sklearn library with the command below:

```
>>> from sklearn.naive_bayes import MultinomialNB
>>> from sklearn.naive_bayes import ComplementNB
```

### *Random Forest*

Most preferred for imbalanced data sets (unequal tweet numbers in each category) is the Random Forest algorithm. Random Forest is an ensemble learning method that utilizes multiple decision trees. The algorithm makes a final decision based on the mode of all the individual decision trees. Random Forest can be imported from the sklearn library with the command below:

```
>>> from sklearn.ensemble import RandomForestClassifier
```

### Model Prediction Performance Metrics

In ML, algorithm performance is measured using precision, recall, F1 score, and overall accuracy. To measure the overall performance of a model, both precision and recall need to be analyzed equally.

#### *Precision*

Precision measures what fraction of positives were actually correct. False positives are instances that were classed as positive that were wrong classifications. A model with 0 false positives will have a precision score of 1.

$$\text{Precision} = \frac{\text{True Positive prediction counts}}{\text{True Positives+False Positives}}$$

#### *Recall*

Recall measures what proportion of actual positives was identified correctly. Both recall and precision depend on the classification threshold that was set for the model. Increasing the threshold might increase the precision, but will reduce the recall as many positive instances will drop below the threshold and will be classed as false negatives.

$$\text{Recall} = \frac{\text{True Positive prediction counts}}{\text{True Positives+False Negatives}}$$

#### *F1 Score*

A F1 score is the harmonic mean of precision and recall. A good F1 score indicates that the model is balanced between precision and recall. It suggests that the model's overall performance is satisfactory, considering both false positive and false negative classifications.

#### *Overall Accuracy*

Accuracy measures the overall correct predictions made across all classes. It is calculated as the ratio of the number of correctly classified instances to the total number of instances in the dataset.

## Details on Dataset

We retrieved Twitter hate speech data from Kaggle. The "Cyberbullying Dataset," refer to figure 1, is a collection of datasets from different social media platforms like Kaggle, Twitter, YouTube, and Wikipedia. For the purpose of this project, we used "twitter parsed data csv file" and converted it into a pandas data frame. This set contains 16848 tweets categorized into neutral, sexist, and racist comments. Originally, the type of tweet was annotated by explicitly stating if the tweet was neutral, sexist, or racist. For the machine learning model to easily analyze these tweets, the annotations must be converted into integers that will be listed under the column *label_num*. Neutral tweets will be denoted as '0', sexist as '1', and racist as '2.' Based on the dataset, the number of tweets that were classed as neutral was 11501, sexist was 3377, and racist was 1970. By looking at these numbers, we can conclude that this dataset is imbalanced as the number of neutral tweets are 3 to 5 times more

than the tweets classed as racist and sexist. The imbalanced nature of the sample data will impact the performance of the classification models in predicting a tweet as neutral, sexist, or racist.

| | index | id | Text | Annotation | oh_label | label_num |
|---|---|---|---|---|---|---|
| 0 | 5.74948705591165E+017 | 5.74948705591165E+017 | @halalflaws @biebervalue @greenlinerzjm I read... | none | 0.0 | 0.0 |
| 1 | 5.71917888690393E+017 | 5.71917888690393E+017 | @ShreyaBafna3 Now you idiots claim that people... | none | 0.0 | 0.0 |
| 2 | 3.90255841338601E+017 | 3.90255841338601E+017 | RT @Mooseoftorment Call me sexist, but when I ... | sexism | 1.0 | 1.0 |
| 3 | 5.68208850655916E+017 | 5.68208850655916E+017 | @g0ssipsquirrelx Wrong, ISIS follows the examp... | racism | 1.0 | 2.0 |
| 4 | 5.75596338802373E+017 | 5.75596338802373E+017 | #mkr No No No No No No | none | 0.0 | 0.0 |

**Figure 1.** Twitter Dataset from Kaggle. Source: https://www.kaggle.com/datasets/saurabhshahane/cyberbullying-dataset. The column 'Text' is the raw tweets and 'label_num' is the numerical conversion of the classes.

## Data Cleanup

While the Twitter data may be used in its raw form, it may not provide the best accuracy results in classification. Therefore, we can use the following data cleanup methods.

### Stopwords and Special Characters

Stopwords are filler words such as I, me, they, myself, themselves, etc. that do not add meaning to the context of the tweet (Nothman et.al., 2018). To reduce the size of the data set and to add relevance to the context of the tweet, it is recommended to filter out these words. Similarly, the other special characters (#, @) can be filtered out.

### Stemming

Stemming is a natural language processing technique that is used to reduce words to their root forms (Willett, 2006). In other words, it is reducing the word to its base form by removing the affixes (ing, es, ied). For example, waiting, waited, waits are reduced to the base word wait. The drawbacks of stemming are that it can reduce the words to base words which sometimes do not have a meaning or context in English. For instance, arguing to argu, where argu does not have a meaning. The word 'argu' does not exist but the stemming process removes 'ing' from the word 'arguing'. We used Porter Stemming (Khyani et.al., 2021) and the code snippet is as shown below.

```
>>>import nltk
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
word_stemmer.stem('Changing')
Result: Chang
```

### Lemmatization

Lemmatization uses the context of the words during reduction. Unlike stemming, lemmatization reduces to meaningful words that exist in the English dictionary. In the previous example where 'arguing' was reduced to

'argu' in stemming, if lemmatization was used, 'arguing' will reduce to 'argue' Figure 2 shows the effect of stemming vs lemmatization.

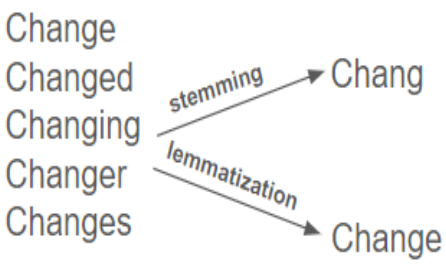


**Figure 2.** Stemming vs. Lemmatization

The following tweet shows the difference between the raw form, stemming, and lemmatization

Raw Tweet: RT @ahtweet: @freebsdgirl How dare you have feelings is a fantastic way to dehumanize someone

Stemming: rt ahtweet freebsdgirl dare feel fantast way dehuman someon

Lemmatization: rt ahtweet: freebsdgirl dare feelings fantastic way dehumanize someone

## TF-IDF Vectorization

Before going into detail of the model results, it is important to understand TF-IDF vectorization of text or string data to numerical matrices. TF-IDF stands for term frequency - inverse document frequency and is a technique used to convert a sentence or corpus into a numerical vector or matrix which can be used as the input feature to train the model. TF-IDF is used to determine how significant a word is to a sentence or a corpus. It is a weighing system calculation that assigns a weight to each word in a document based on its term frequency (TF) and the reciprocal document frequency (IDF). The words with higher weight scores are deemed to be more significant. TF-IDF can be imported using the following code:

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
```

Coding Tools and Methodology

Once the raw tweets were preprocessed with stopword filtering and stemming, we used 80% of the data for the training dataset and remaining 20% as a test data set. The test-train data was defined using the following module from sklearn:

```
>>> from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
   df.Text,
   df.label_num,
   test_size=0.2, # 20% samples will go to test dataset
   random_state=15,
   stratify=df.label_num
)
```

Where X_train and X_test are the sample tweets and y_train, y_test are the numerical classes assigned to label neutral, sexist, and racist classes. The ML models will be trained and tested with this data.

Sklearn (Buitinck et.al., 2013) offers various packages and ML tools that can help optimize the code into a simple flow. We used the pipeline function to process the train test data where TF-IDF vectorization and

any ML model can be passed as input variables. The code below shows a simple algorithm for KNeighbors classifier. The classification report function was used to derive the performance metrics.

```
>>> from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
clf = Pipeline([
   ('vectorizer_tfidf',TfidfVectorizer()),
   ('KNN', KNeighborsClassifier())
])
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

## Results and Discussion

The following outlines the results of the machine learning model classifications for the different techniques used.

### Results for ML Classification for Raw Data

Table 1 shows the results of the ML models for raw tweets classification. Overall accuracy for the best performing model, Logistic Regression, was around 84%. However, the other metrics, like recall, was around 60%, which implies that the proportion of the overall sexist or racist tweets that were identified correctly was low. In this case, many of the tweets from sexist or racist classes were wrongly classed as neutral. The overall performance of the raw data classification was not satisfactory.

**Table 1**. Prediction metrics for raw tweets data

| Model | Precision | | | Recall | | | F1-Score | | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | N[1] | S[2] | R[3] | N[1] | S[2] | R[3] | N[1] | S[2] | R[3] | |
| **KNeighborsClassifier** | 0.81 | 0.81 | 0.70 | 0.93 | **0.52** | **0.54** | 0.87 | 0.63 | 0.61 | 0.80 |
| **Multinomial N B** | 0.71 | 0.95 | 0.88 | 0.99 | **0.16** | **0.13** | 0.83 | 0.27 | 0.23 | 0.73 |
| **Complement N B** | 0.85 | 0.81 | 0.65 | 0.89 | **0.60** | **0.77** | 0.87 | 0.69 | 0.70 | 0.82 |
| **Random Forest** | 0.82 | 0.9 | 0.82 | 0.96 | **0.53** | **0.58** | 0.89 | 0.66 | 0.68 | 0.83 |
| **Logistic Regression** | 0.84 | 0.86 | 0.79 | 0.94 | **0.60** | **0.65** | 0.89 | 0.71 | 0.71 | 0.84 |

$N^1$ is neutral tweets, $S^2$ is Sexist tweets and $R^3$ is Racist tweets. Overall recall scores were low and shown in bold.

## Results for ML Classification after Data Cleanup

Table 2 shows the classification report after cleaning stop words in the data and using stemming. The accuracy of the classification improved marginally by 1% to 2% for most of the models. Again, the overall performance of the models was not at par due to a low recall score.

**Table 2**. Prediction metrics for cleaned data post stop words removal and stemming

| Model | Precision | | | Recall | | | F1-Score | | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | $N^1$ | $S^2$ | $R^3$ | $N^1$ | $S^2$ | $R^3$ | $N^1$ | $S^2$ | $R^3$ | |
| **KNeighborsClassifier** | 0.81 | 0.79 | 0.72 | 0.93 | **0.48** | **0.61** | 0.87 | 0.60 | 0.66 | 0.80 |
| **Multinomial N B** | 0.74 | 0.93 | 0.86 | 0.99 | **0.23** | **0.25** | 0.84 | 0.37 | 0.37 | 0.75 |
| **Complement N B** | 0.88 | 0.79 | 0.61 | 0.86 | **0.67** | **0.86** | 0.87 | 0.72 | 0.71 | 0.82 |
| **Random Forest** | 0.85 | 0.86 | 0.80 | 0.94 | **0.62** | **0.68** | 0.90 | 0.72 | 0.74 | 0.85 |
| **Logistic Regression** | 0.85 | 0.87 | 0.80 | 0.95 | **0.61** | **0.65** | 0.89 | 0.72 | 0.72 | 0.85 |

$N^1$ is neutral tweets, $S^2$ is Sexist tweets and $R^3$ is Racist tweets. Overall recall scores were low and shown in bold.

## Dealing with Unbalanced data set

The twitter data set was unbalanced as the total tweet counts for the 3 classes, neutral, sexist, and racist, were 11501, 3377 and 1970 respectively. The racist tweet count was about 6 times lower than the neutral tweet count. Typically, imbalanced data sets will give a poor accuracy rate for classification. To deal with this imbalanced data set (Vargas et al., 2022), we used two methods: downsampling and oversampling.

### *Downsampling*

In this method, we randomly down sampled the neutral tweets and the sexist tweets to match the number of racist tweets. The resulting downsampled data set had an equal number of tweets for each of the classes (1970). The train test data set was derived using the new set of down-sampled tweets. Table 3, shows the performance of the classification models with this new down sampled data set. Given the low number of sample tweets for training, the accuracy of the models decreased. Logistic regression, previously the best model, dropped from 85% to 81%. On a positive note, we can see the recall and F1 scores have improved from previous cases. This implies that a balanced data set might result in a better prediction across the classes.

**Table 3**. Prediction metrics for cleaned pre-processed and downsampled data

| Model | Precision | | | Recall | | | F1-Score | | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | N[1] | S[2] | R[3] | N[1] | S[2] | R[3] | N[1] | S[2] | R[3] | |
| **KNeighborsClassifier** | 0.65 | 0.79 | 0.85 | 0.70 | **0.71** | **0.87** | 0.68 | 0.75 | 0.86 | 0.76 |
| **Multinomial N B** | 0.77 | 0.79 | 0.83 | 0.62 | **0.82** | **0.96** | 0.69 | 0.80 | 0.89 | 0.80 |
| **Complement N B** | 0.84 | 0.75 | 0.81 | 0.53 | **0.88** | **0.97** | 0.65 | 0.81 | 0.89 | 0.79 |
| **Random Forest** | 0.70 | 0.83 | 0.90 | 0.78 | **0.74** | **0.90** | 0.74 | 0.78 | 0.90 | 0.81 |
| **Logistic Regression** | 0.71 | 0.85 | 0.90 | 0.79 | **0.76** | **0.89** | 0.75 | 0.80 | 0.89 | 0.81 |

$N^1$ is neutral tweets, $S^2$ is Sexist tweets and $R^3$ is Racist tweets. Overall recall scores have improved and shown in bold.

### *Oversampling*

In the case of oversampling, the sexist and racist tweets were duplicated to match the total number of neutral tweets. The end result is all classes have the same number of tweets (11501). Now we have a larger pool of training data set for the racist and sexist comments, same as the neutral tweet counts. The training and test data set is taken from this new data frame. The performance of the models for the oversampled data set is shown below in Table 4. In this scenario, all the models had a significant improvement in accuracy and all other metrics. The best performing classifiers were Random Forest and Logistic regression.

**Table 4**. Prediction metrics for cleaned pre-processed and oversampled data

| Model | Precision | | | Recall | | | F1-Score | | | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| | N[1] | S[2] | R[3] | N[1] | S[2] | R[3] | N[1] | S[2] | R[3] | |
| **KNeighborsClassifier** | 0.87 | 0.85 | 0.87 | 0.70 | **0.91** | **0.97** | 0.78 | 0.88 | 0.92 | 0.86 |
| **Multinomial N B** | 0.92 | 0.87 | 0.86 | 0.70 | **0.94** | **0.99** | 0.80 | 0.90 | 0.92 | 0.88 |
| **Complement N B** | 0.95 | 0.85 | 0.85 | 0.66 | **0.96** | **0.99** | 0.78 | 0.90 | 0.92 | 0.87 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Random Forest** | 0.98 | 0.95 | 0.94 | 0.89 | **0.99** | **1.0** | 0.94 | 0.97 | 0.97 | 0.96 |
| **Logistic Regression** | 0.92 | 0.92 | 0.93 | 0.85 | **0.93** | **0.99** | 0.88 | 0.93 | 0.96 | 0.92 |

$N^1$ is neutral tweets, $S^2$ is Sexist tweets and $R^3$ is Racist tweets. Overall, all metrics improved and recall scores were high, almost touching 100%

The recall improved significantly. For example, recall score of racist tweets in random forest classifier was 1. This means there was no false negative and 100% of the tweets were classed correctly. The recall was 99% for the sexist tweet class. The F1 scores were in the high 90's. The models performed significantly better than the cases we discussed previously. Figure 3 shows the confusion matrix for the best performing Random Forest classifier where the class numbers (0, 1, 2) are the neutral, sexist, and racist tweet labels. The figure shows that the classifier predicted majority of the tweet's classes correctly. The x-axis is the predicted class number of the test tweets and the y-axis is the true classification from the data set. We can see from the table that less than 5% of tweets were misclassed and majority of the tweets were predicted correctly.

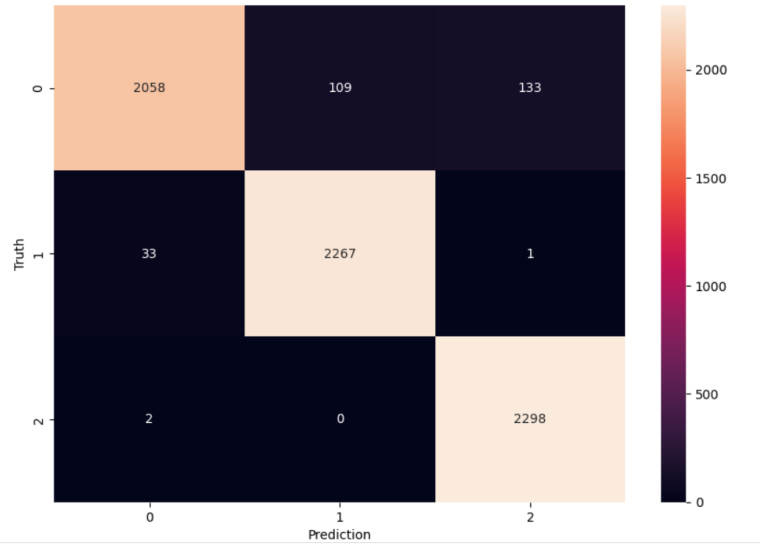


**Figure 3.** Random Forest Metrics with Balanced/Oversampled Data Set, Tweet Preprocessing, and Stemming

Figure 4 shows the precision trend for racist tweets for various ML classifiers that we tried. The precision trend shows with data preprocessing and balancing the data set with over sampling methods improved the performance across all ML models, except Multinomial NB. Figure 5 show the recall metric trend for racist tweet class and we observe how balancing the data between the classes strongly improved the recall score. Since the recall improved after balancing the data, the F1 score (Figure 6) and overall accuracy among all classes improved as shown in Figure 7.

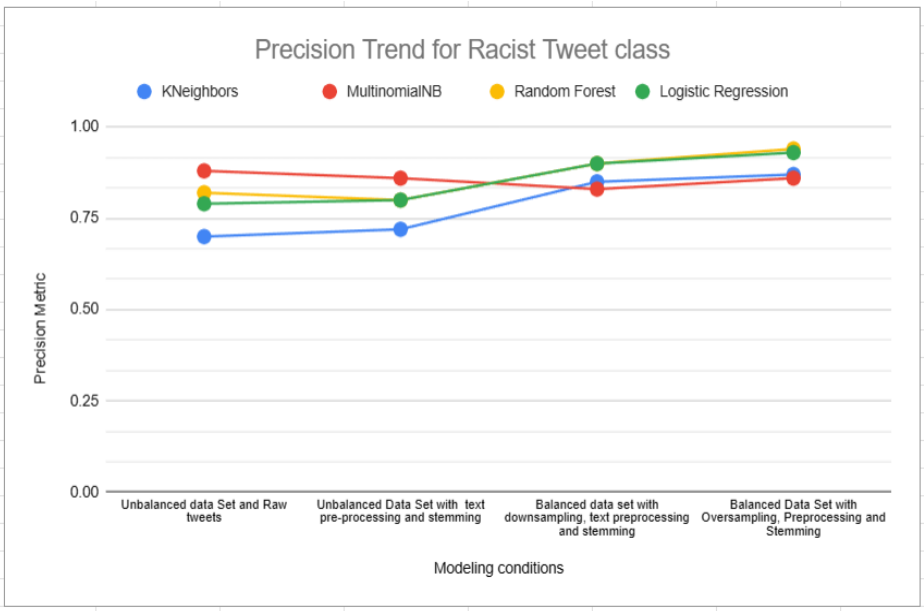


**Figure 4.** Precision trend plot for racist tweets across model classifiers and data sets. Balancing the data set improved the precision.

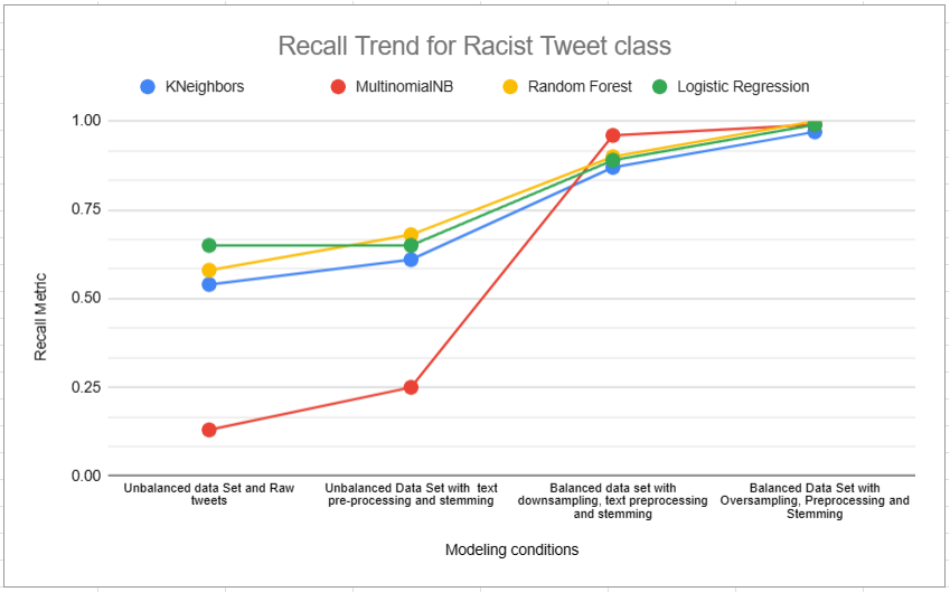


**Figure 5.** Recall trend plot for racist tweets across model classifiers and data sets. Balancing the data set improved the Recall dramatically.

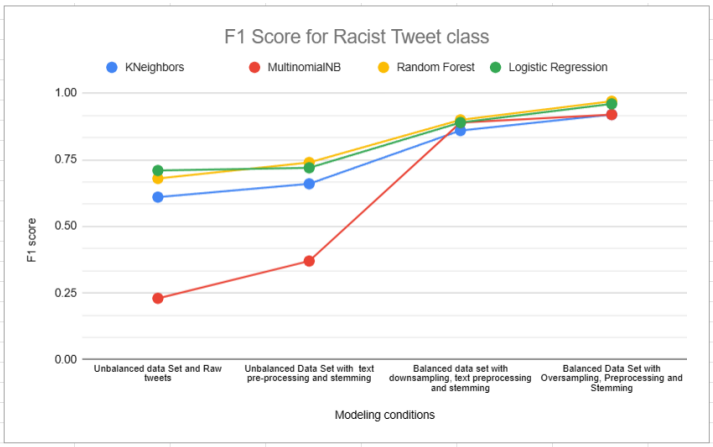


**Figure 6.** F1 score trend plot for racist tweets across model classifiers and data sets. Balancing the data set improved the precision considerably.

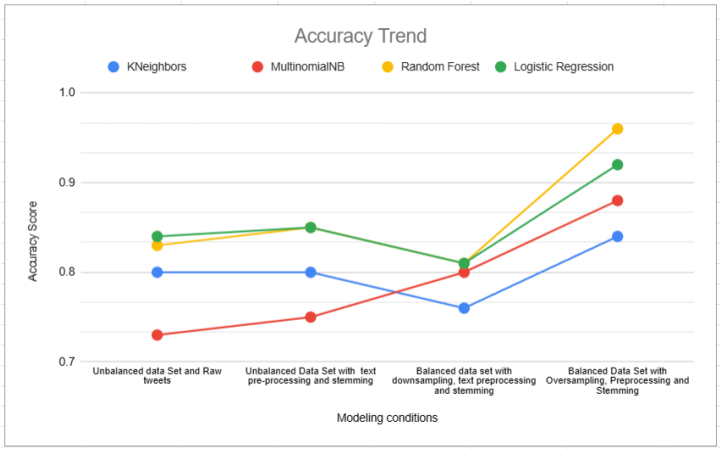


**Figure 7.** Accuracy trend plot across model classifiers and data sets.

## Conclusion

We used sentiment analysis and machine learning algorithms to properly classify Twitter hate speech as neutral, racist, or sexist. Different techniques such as data clean-up, downsampling, and oversampling were used to increase the accuracy of the machine learning model's metrics. An unbalanced dataset held the lowest predictions rates when labeling Twitter hate speech. When this data was balanced, precision scores improved especially when the data was oversampled. The highest accuracy rate was seen in the Random Forest Classifier with balanced, oversampled, and preprocessed data. Yielding a 96% accuracy rate, this model proves to be an efficient way to accurately label tweets as neutral, sexist, and racist. This can be applied to a larger picture when discussing effective methods to sort through tweets to ensure they follow platform guidelines.

In the future, we plan to expand this study further to improve the classification rates of imbalanced datasets without oversampling. This will be done using recent NLP techniques such as BERT and GPT.

## Acknowledgments

I would like to extend thanks to my advisor for guiding me through this project. I also thank JSR for giving me the opportunity to publish this paper.

I am also grateful to sklearn for machine learning models, Kaggle for providing the twitter dataset, and Google Cofab for providing a coding environment.

## References

Antonakaki, D., Fragopoulou, P., & Ioannidis, S. (2021). A survey of Twitter research: Data model, graph structure, sentiment analysis and attacks. *Expert Systems with Applications*, *164*, 114006. https://doi.org/10.1016/j.eswa.2020.114006

Giachanou, A., & Crestani, F. (2016). Like It or Not. *ACM Computing Surveys*, *49*(2), 1–41. https://doi.org/10.1145/2938640

*1.1. Linear Models — scikit-learn 0.22.2 documentation*. (n.d.). Scikit-Learn.org. https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

*1.6. Nearest Neighbors — scikit-learn 0.21.3 documentation*. (2019). Scikit-Learn.org. https://scikit-learn.org/stable/modules/neighbors.html

Scikit-learn. (2019). *1.9. Naive Bayes — scikit-learn 0.21.3 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/naive_bayes.html

*1.11. Ensemble methods*. (n.d.). Scikit-Learn. https://scikit-learn.org/stable/modules/ensemble.html#random-forests

Google Developers. (2019, March 5). *Classification: Precision and Recall - Google Developers*. https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

Nothman, J., Qin, H., & Yurchak, R. (2018). Stop Word Lists in Free Open-source Software Packages. *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. https://doi.org/10.18653/v1/w18-2502

Willett, P. (2006). The Porter stemming algorithm: then and now. *Program*, *40*(3), 219–223. https://doi.org/10.1108/00330330610681295

Khyani, Divya & B S, Siddhartha. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Shanghai Ligong Daxue Xuebao. Journal of University of Shanghai for Science and Technology. 22. 350-357.

*sklearn.feature_extraction.text.TfidfTransformer — scikit-learn 0.23.1 documentation*. (n.d.). Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

Vargas, V., Aranda, J., Costa, R., Pereira, P., & Luis, J. (2022). *Imbalanced data preprocessing techniques for machine learning: a systematic mapping study*. *65*(1), 31–57. https://doi.org/10.1007/s10115-022-01772-8

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Müller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). *API design for machine learning software: experiences from the scikit-learn project*.