SPRINGBOARD - DSC

CAPSTONE PROJECT # 2

COVID-19 NLP Text Classification and Sentiment Analysis

Introduction

The COVID-19 pandemic has had a significant impact on society, and the need for accurate and reliable information has never been greater. In this project, we propose to use natural language processing (NLP) techniques to classify text data related to COVID-19.

Goal

The objective of this project is to build a model that can accurately classify text data into different categories related to COVID-19, such as information about symptoms, treatments, and vaccines. The model will be trained on the dataset mentioned above and will be evaluated based on its ability to classify unseen text data.

Methodology:

The first step in this project will be to preprocess the text data, which will involve cleaning and normalizing the text, as well as removing any irrelevant information. Next, we will use various NLP techniques such as tokenization and lemmatization to prepare the text data for modeling. We will then use machine learning algorithms such as logistic regression, naive Bayes, Extreme Gradient Boosting, Random Forest, and SVM, etc to train the model.

Evaluation:

The performance of the model will be evaluated using metrics such as accuracy, precision, recall, and F1 score. The model will be trained and tested on a split of the dataset and we wil tune the hyperparameters to achieve the best performance.

Expected Outcomes:

The outcome of this project will be a model that can accurately classify text data related to COVID-19. This model can be used to automatically classify text data from various sources, such as social media posts or news articles, and can be used to quickly and easily identify relevant information about the pandemic.

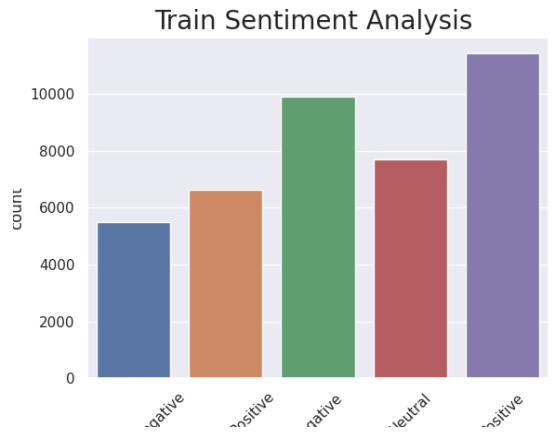
Data Specification

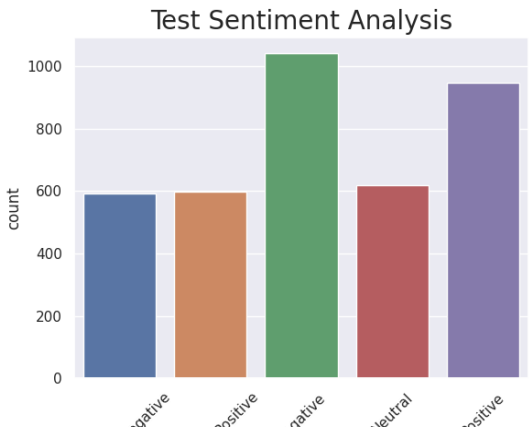
Specifically, we will use open-source dataset from this Kaggle link:

https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification

which includes thousands of text snippets from various sources such as news articles, social media posts, and government documents. The dataset of train contains 41,157 rows and, 6 columns while test dataset contains 3,798 rows and 6 columns, contains information about username, location, tweet timings, original tweets and sentiment types. Raw data and plot of the files looks like this,

```
UserName       int64
ScreenName     int64
Location       object
TweetAt        object
OriginalTweet  object
Sentiment      object
```

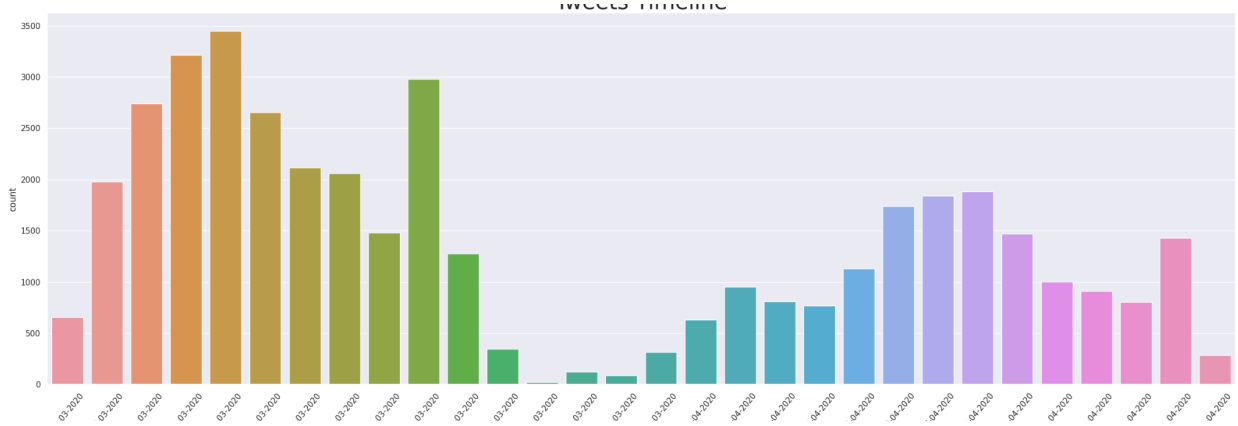




Exploratory Data Analysis - EDA

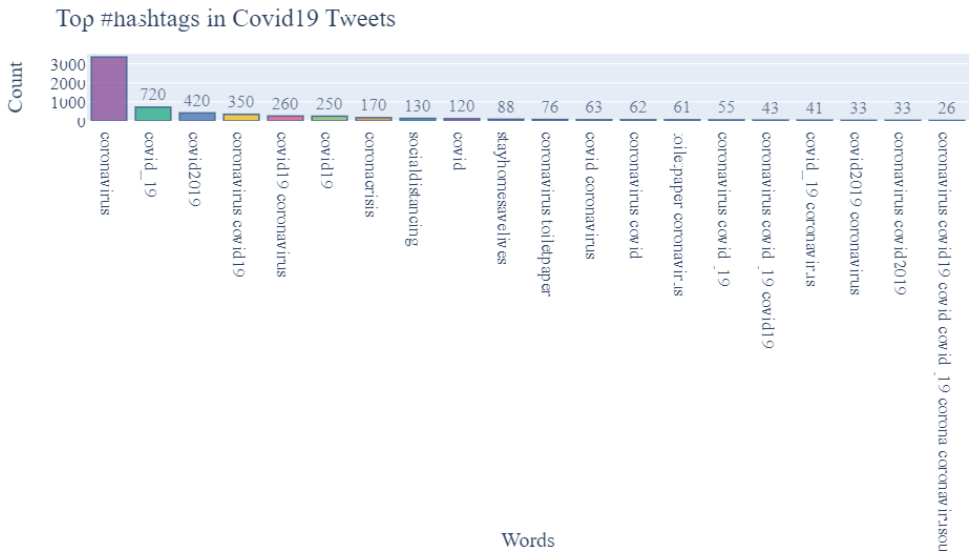
Tweets Timeline:

The timeline shows that most of the tweets originated in late March 2020, which was the peak season for Covid-19.

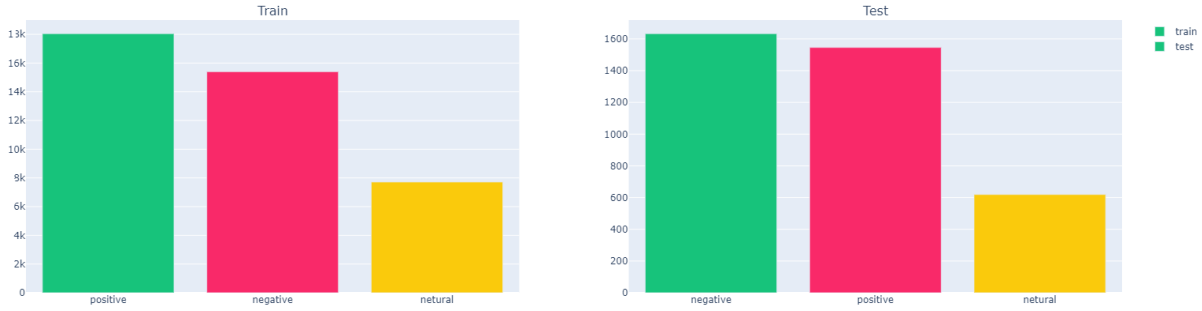


The plot below shows the top 10 tweets locations,

The plot below shows the most common hashtags with coronavirus and covid_19 being the most trending hashtags of the time.

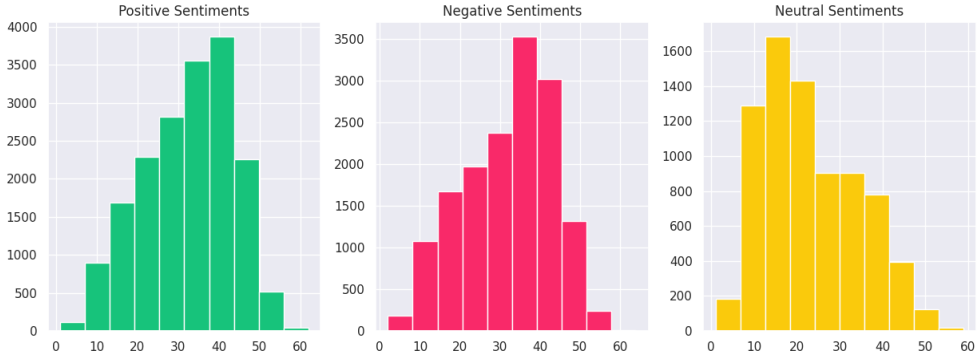


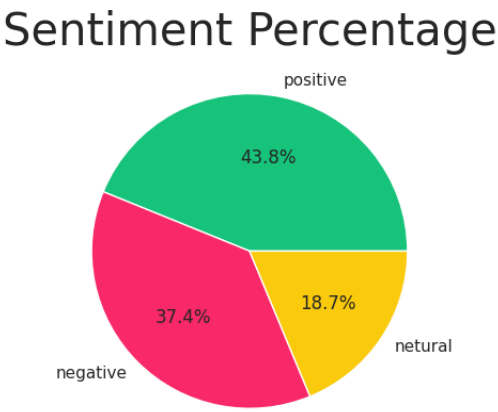
To perform EDA, we generalize five sentiment categories into three main categories which are positive, negative and neutral, as shown below,

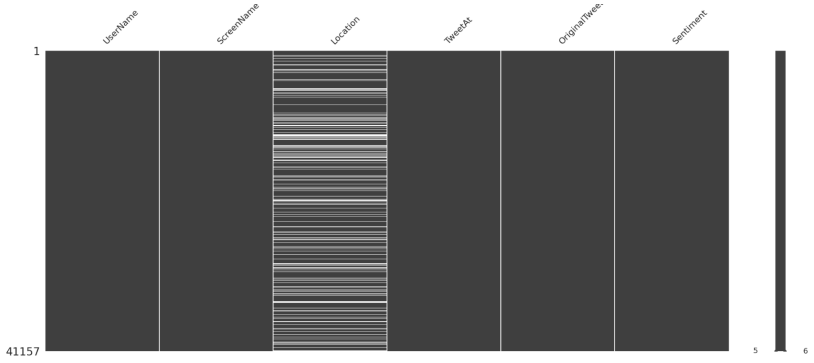


Most of the tweets were positive sentiments as people were wishing everyone to stay safe from Covid-19. The negative sentiment was due to a lack of awareness and carelessness by not wearing masks and taking other necessary precautions to stop the spread of this disease. A lot of people were confused and scared as they don't know what to expect in a pandemic.

The plot below shows the comparison of number of words used on average in every tweet based on their sentiment. Pie chart show the percentage of every sentiment.

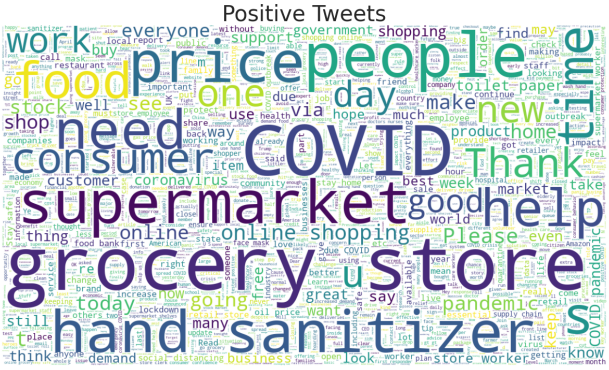






The msno matrix plot above shows that the data is very clean and only the Location column has some missing and null values. Word clouds of every sentiment shows the words that appear frequently in the tweets.

Positive Tweets



Negative Tweets

Neutral Tweets

After performing data wrangling and exploratory data analysis (EDA), the key takeaway is that the data set contains valuable information about the relationship between different variables. In addition, the analysis reveals important insights regarding the distribution of the data, the correlation between location, timeline, and sentiments, and any potential relationships between different variables. With this information, we can now begin to make informed decisions about the data and use it to make predictions and build models.

Data Modeling

Tokenization

Tokenization is the first step in NLP. It is the process of breaking strings into tokens, which in turn are small structures or units

Data cleaning and formatting

We did perfrom data cleaning once again to make sure all the data is in the same format such as making all text lower, removing text in square brackets, and removing links, punctuation, and words containing numbers.

Lemmatization

Lemmatization and stemming are two important steps in natural language processing (NLP). Lemmatization is the process of converting a word to its base form while stemming is a rule-based approach that removes the last few characters to try to derive the root form of a word. Although stemming is faster, it often leads to incorrect meanings and spelling errors. On the other hand, lemmatization has higher accuracy and is preferred for context analysis, while stemming is recommended when the context is not important. In our case, context is not important so we are going to use lemmatization.

TF-IDF

Term Frequency – Inverse Document Frequency (TF-IDF) is a widely used statistical method in natural language processing and information retrieval. It measures how important a term is within a document relative to a collection of documents.

CountVectorizer is also another term used to count the number of times a word appears in a document (using a bag-of-words approach), while TF-IDF Vectorizer takes into account not only how many times a word appears in a document but also how important that word is to the whole corpus.

Here we have to minimize the number of features to avoid system crashes by setting min_df to very low like 3. Also, there is no need to use toarray() for every system crash as it increases the data size.

Machine Learing Models

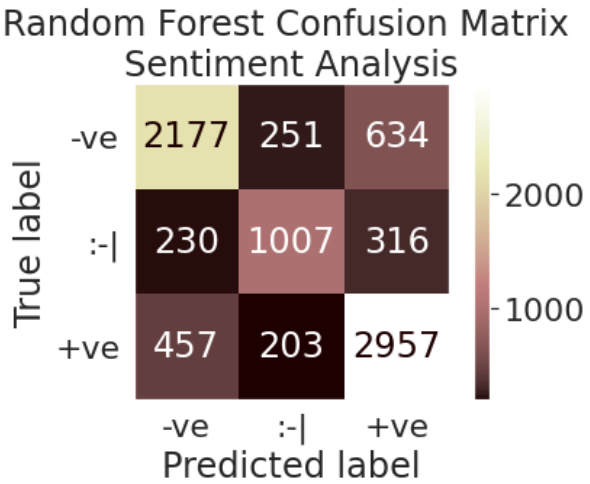
We will compare seven different supervised machine-learning models using the great Scikit-Learn library:

1. Random Forest Classifier

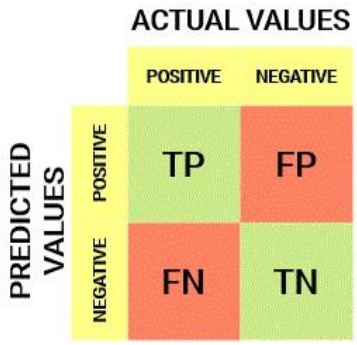
Random Forest is a supervised machine-learning algorithm made up of decision trees that are merged for a more accurate prediction. Random Forest is used for both classification (each tree gives a classification or a vote) and regression (the forest picks the average of the outputs of all trees) for example, classifying whether an email is “spam” or “not spam” Random Forest is used across many different industries, including banking, retail, and healthcare, to name just a few! Overall, Random Forest is accurate, efficient, and relatively quick to develop, making it an extremely handy tool for data professionals.

Here, Training Accuracy: 0.9998785117691723

Testing/Validation Accuracy: 0.7459912536443148



	precision	recall	f1-score	support
Negative	0.76	0.71	0.73	3062
Neutral	0.69	0.65	0.67	1553
Positive	0.76	0.82	0.79	3617
accuracy			0.75	8232
macro avg	0.74	0.73	0.73	8232
weighted avg	0.75	0.75	0.74	8232



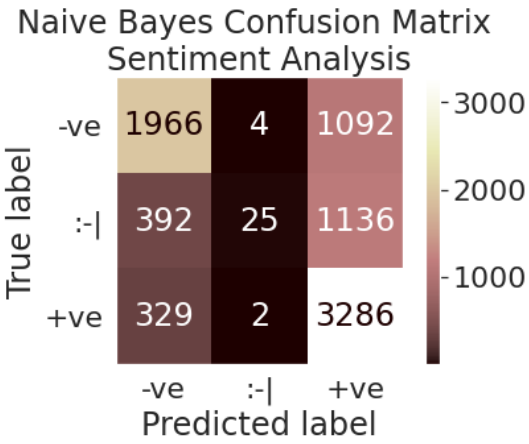
There are 4-potential outcomes here, True positive (TP), False positive (FP), False negative (FN), True. An increase in max_depth increases the training and testing accuracy.

2. Naive Bayes Classifier

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts based on the probability of an object.

Here, Training accuracy: 0.7351556567957479

Testing/Validation accuracy: 0.6410349854227405



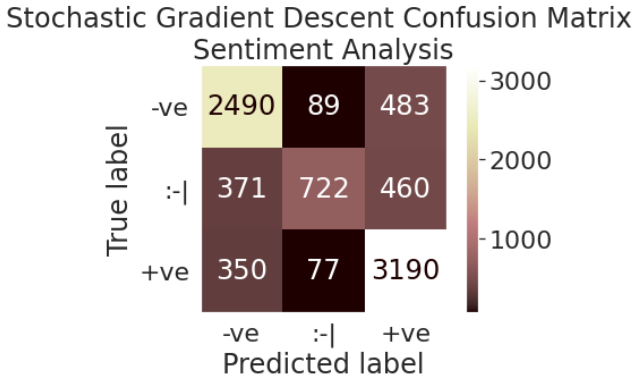
	precision	recall	f1-score	support
Negative	0.64	0.73	0.68	2687
Neutral	0.02	0.81	0.03	31
Positive	0.91	0.60	0.72	5514
accuracy			0.64	8232
macro avg	0.52	0.71	0.48	8232
weighted avg	0.82	0.64	0.71	8232

3. Stochastic Gradient Decent Classifier

A stochastic Gradient Descent Classifier is a type of machine learning algorithm that helps computers learn from data. It works by looking at a set of data points and using them to find patterns. It then uses those patterns to make predictions about future data points. It's like a teacher showing a student examples to help them understand a concept.

Here, Training accuracy: 0.8704935459377373

Test/Validation accuracy: 0.7776967930029155



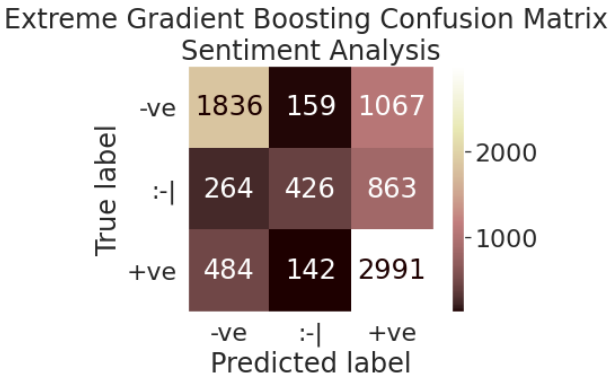
	precision	recall	f1-score	support
Negative	0.81	0.78	0.79	3211
Neutral	0.46	0.81	0.59	888
Positive	0.88	0.77	0.82	4133
accuracy			0.78	8232
macro avg	0.72	0.79	0.74	8232
weighted avg	0.81	0.78	0.79	8232

4. Extreme Gradient Boosting Classifier

Extreme Gradient Boosting - XGBoost Classifier is a type of XGBoost algorithm for binary classification problems, where the target variable can take only two values, such as "yes" or "no", "positive" or "negative". It uses decision trees as its base learners and builds a group of these decision trees to make predictions.

Here, Training accuracy: 0.6456795747911921

Test/Validation accuracy: 0.6381195335276968



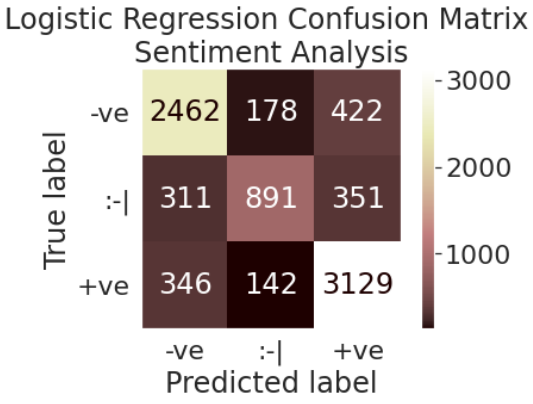
	precision	recall	f1-score	support
Negative	0.60	0.71	0.65	2584
Neutral	0.27	0.59	0.37	727
Positive	0.83	0.61	0.70	4921
accuracy			0.64	8232
macro avg	0.57	0.63	0.57	8232
weighted avg	0.71	0.64	0.66	8232

5. Logistic Regression Classifier

The logistic regression classifier model is also another commonly used machine learning model used for classifying data into two distinct categories such as "yes" or "no", or "positive" or "negative". This makes it well-suited for analyzing binary data and making binary predictions.

Here, Training accuracy: 0.9025360668185269

Test/Validation accuracy: 0.7874149659863946



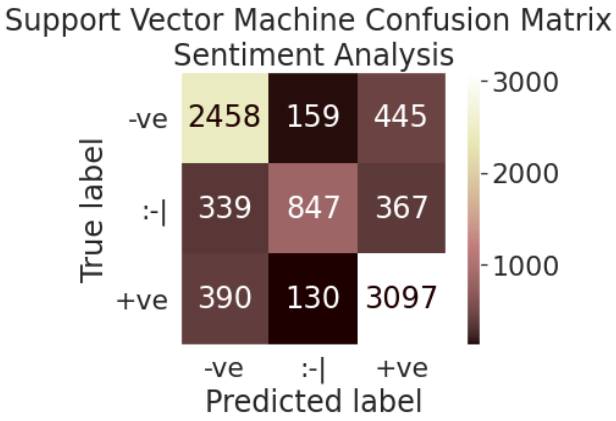
	precision	recall	f1-score	support
Negative	0.80	0.79	0.80	3119
Neutral	0.57	0.74	0.64	1211
Positive	0.87	0.80	0.83	3902
accuracy			0.79	8232
macro avg	0.75	0.78	0.76	8232
weighted avg	0.80	0.79	0.79	8232

6. Support Vector Machine Classifier

Support Vector Machine (SVM) is a powerful machine learning method for classification problems, which finds the optimal boundary to separate the data into different classes by maximizing the margin between the classes.

Here, Training accuracy: 0.9786788154897494

Test/Validation accuracy: 0.7776967930029155



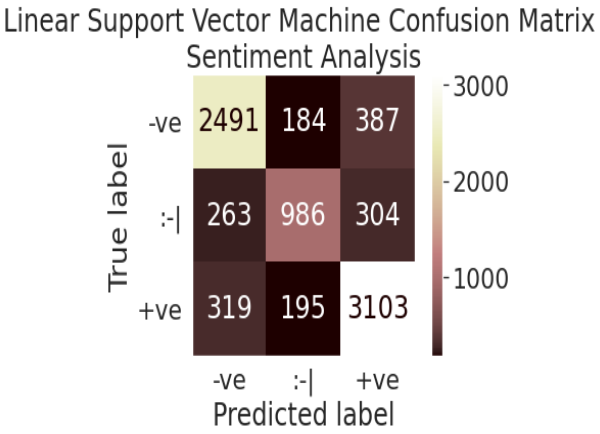
	precision	recall	f1-score	support
Negative	0.80	0.77	0.79	3187
Neutral	0.55	0.75	0.63	1136
Positive	0.86	0.79	0.82	3909
accuracy			0.78	8232
macro avg	0.73	0.77	0.75	8232
weighted avg	0.79	0.78	0.78	8232

7. Linear Support Vector Machine Classifier

Linear Support Vector Machine (SVM) Classifier is a type of SVM algorithm used for linear binary classification problems, where the data is separated into different classes using a linear boundary (or hyperplane) with the maximum margin.

Here, Training accuracy: 0.9705087319665907

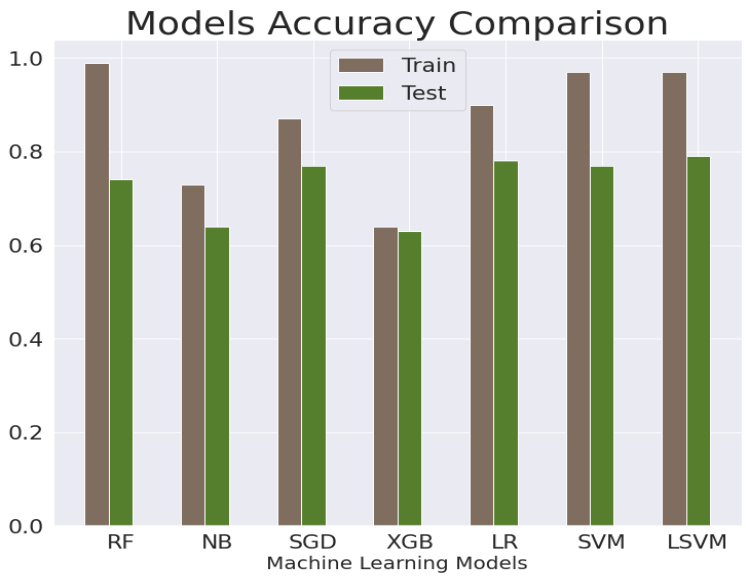
Test/Validation accuracy: 0.7993197278911565



	precision	recall	f1-score	support
Negative	0.81	0.81	0.81	3073
Neutral	0.63	0.72	0.68	1365
Positive	0.86	0.82	0.84	3794
accuracy			0.80	8232
macro avg	0.77	0.78	0.78	8232
weighted avg	0.80	0.80	0.80	8232

Model Accuracy Comparison

Now, let's compare and visualize the above models for more better understanding.



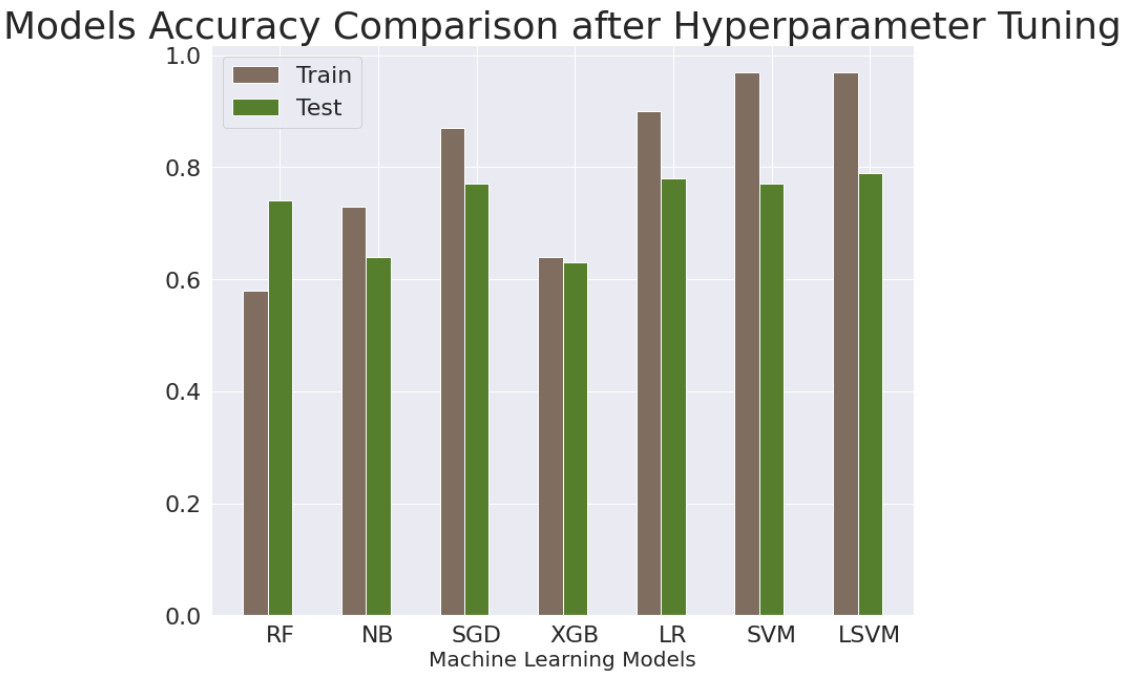
The comparison plot shows the XGB - Extreme gradient boosting classifier performs the best with an accuracy of 0.64 and 0.63 for training and testing data. Random forest classifier accuracy is very low.

Now let's tune the random forest classifier and see if the accuracy gets any better.

Hyperparameter Tunning

GridSearchCV and RandomizedSearchCV are two widely used techniques in Hyperparameter Tuning. Grid Search exhaustively searches through every combination of the hyperparameter values specified. In contrast to Grid Search, not all given parameter values are tried out in Randomized Search. Rather a fixed number of parameter settings is sampled from the specified distributions. In this project, we are using

RandomizedSearchCV only to tune the random forest classifier.



The best hyperparameters and the corresponding best accuracy score is printed after the search which are,

Best parameters: `{'n_estimators': 500, 'max_features': 'auto', 'max_depth': 25}`

Best accuracy score: 0.585725132877752

Here, the accuracy score is 0.58 but previously we got 0.99 accuracy on training, which means Random Forest Classifier may not be the best classifier in our case.

Feature Importances

Now check the most important features in our dataset. The table below shows the most import features in our data and their score respectively.

	feature	Gain_Score
0	Location	3.424412e-07
1	OriginalTweet	7.174077e-06
2	ScreenName	8.110908e-07
3	Sentiment	1.829644e-07
4	TweetAt	6.376677e-08
5	UserName	1.285435e-06
6	clean_text	3.638496e-06
7	final_text	3.639022e-06

Conclusion

In this study, we employed various Natural Language Processing techniques such as Tokenization, Lemmatization, and TF-IDF to structure our data, followed by cleaning, organization, and pre-processing. We evaluated multiple machine learning models to determine their accuracy and conducted hyperparameter tuning to optimize their performance.

All models achieved above-average results, except the Extreme Gradient Boosting Classifier, which had a training accuracy of 64% and a testing accuracy of 63%. This deviation could be attributed to the limited size of the data, as some models tend to perform better with larger datasets. Hyperparameter tuning revealed the optimal parameters for the Random Forest Classifier.

Our results indicate that the data was clean and unbiased, with no missing values. The models performed with minimal errors and losses, despite the small size of the dataset. We did not observe any overfitting or underfitting issues, as the data was not extensive enough to draw such conclusions.

Future Work and Recommendation

Due to the time constraint, we are unable to perform a more detailed analysis as a lot can be done with this data to learn more about it. Here are a few things we could do in the future,

1. Plot confusion matrix on test data for comparison with training data to perform a more detailed sentiment analysis on every model.

2. Overfitting and underfitting are two common problems in machine learning that can affect the accuracy and generalization of a model. In our case, the data is not huge but still there are some ways to check for these issues:

- Validation Curves: Plotting the performance of a model on both the training and validation datasets can help to identify overfitting and underfitting. If the model performs well on the training data but poorly on the validation data, then it is likely

to be overfitting. On the other hand, if the model performs poorly on both datasets, then it may be underfitting.

- Learning Curves: Plotting the performance of a model as a function of the number of training examples can also help to identify overfitting and underfitting. If the performance on the training set increases with more training examples but the performance on the validation set decreases, then the model is overfitting. If the performance on both datasets increases but not to a satisfactory level, then the model is underfitting.
- Cross-Validation: Cross-validation is a technique for evaluating the performance of a model by splitting the data into several folds and training the model on different combinations of these folds. If the model's performance varies greatly depending on the fold it was trained on, then it is overfitting.
- Regularization: Regularization is a technique for reducing overfitting by adding a penalty term to the loss function that discourages the model from learning noise in the data. If the performance of a model improves with regularization, then it is likely to be overfitting.

In general, finding the right balance between overfitting and underfitting requires experimentation and a thorough understanding of the problem and the data.