Sentiment Analysis on Covid Vaccination Tweets

Team Members

Aravinda Sagiraju – 002108862

Kimaya Khilare – 002958773

Smruti Jethwani – 002102116

Vini Wategaonkar - 002113660

ABSTRACT

The year 2020 has been a disastrous year for the entire world due to the breakout of the Covid-19 pandemic. The disease has spread across multiple nations and has infected over 100k people. The transmission of the disease is direct touch, droplet, and potential aerosol ways. The only way to suppress the same was to bring out an invention of the vaccine. Many research and development centers require a large sample size of infected subjects, to invent and test a vaccine on the same. The three major companies namely, Pfizer, Moderna and Johnson&Johnson have worked hard day and night to bring out a vaccine for the entire world. There are a couple of other companies too, but the top 3 were the ones widely used. We know that Twitter is a social media platform used to portray freedom of speech for all. In this project, we are going to use some of the machine learning models to check the sentiments of tweets based on categorizing them as positive, neutral, and negative by performing an analysis about the covid data.

INTRODUCTION

We will be using a dataset from Kaggle (https://www.kaggle.com/datasets/gpreda/all-covid19-vaccines-tweets). We have done a comparative study between the models- Logistic regression, Random Forest classifier, Decision Tree Model, KNN Classifier, Linear SVC Model, AdaBoost Classifier. Also, performed hyperparameter tuning on the models.

The dataset basically contains 228207 tweets. We have performed, EDA, data cleaning and preprocessing, feature engineering so that our models output high accuracy with a lot of efficiency. The data was preprocessed by removing the following:

a) Stop words

b) Usernames (words that start with @ on Twitter)

c) Punctuation

d) URLs

e) Emoji

We need to remove all the above since they do not contribute towards sentimental analysis. Furthermore, we perform tokenization of tweets (convert them to a list of independent words) separated by comma, so that we can perform lemmatization.

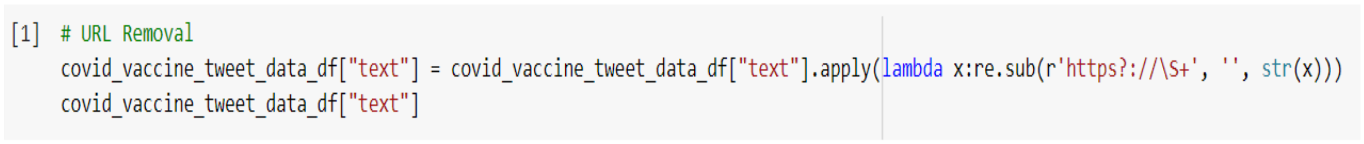
We then split the dataset into training (80%) and testing data (20%). We perform some data visualization and infer some insights as to how many are positive and/or negative and/or neutral tweets, getting the frequency of words.

After the dataset is split into training and testing data, we feed our dataset to the models which will run on our testing dataset to calculate the accuracy. Then, we get the model that outputs the best accuracy.

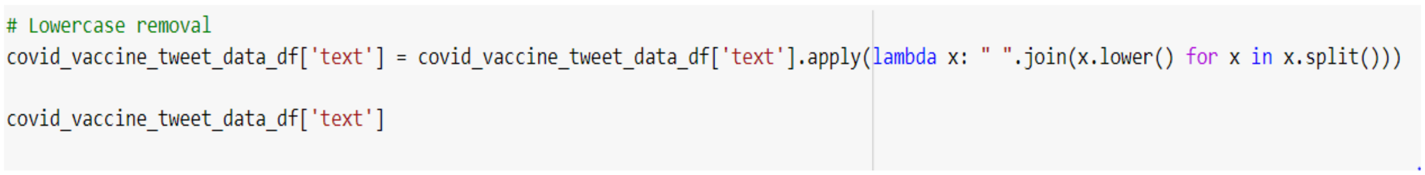
METHODS

1) Data Preprocessing:

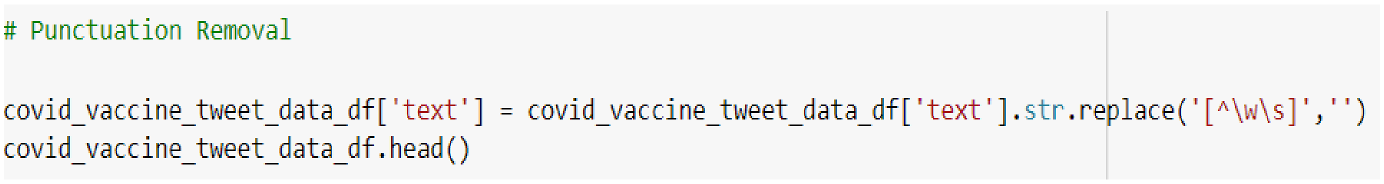
Below are some highlights of how we preprocessed the data. Once the data was cleaned, we moved on to the next steps of modelling. We used some Machine learning algorithms, performed hyperparameter tuning on them and did a comparative study between those algorithms



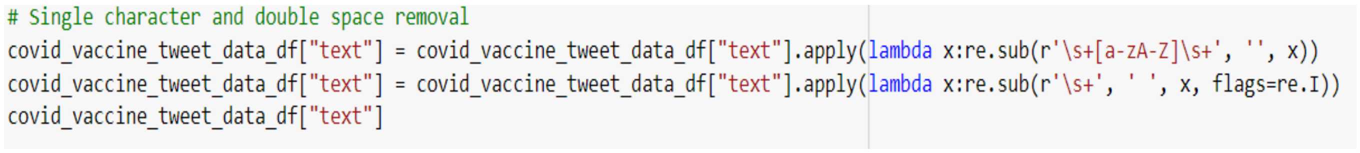
```
[1]  # URL Removal
     covid_vaccine_tweet_data_df["text"] = covid_vaccine_tweet_data_df["text"].apply(lambda x:re.sub(r'https?://\S+', '', str(x)))
     covid_vaccine_tweet_data_df["text"]
```



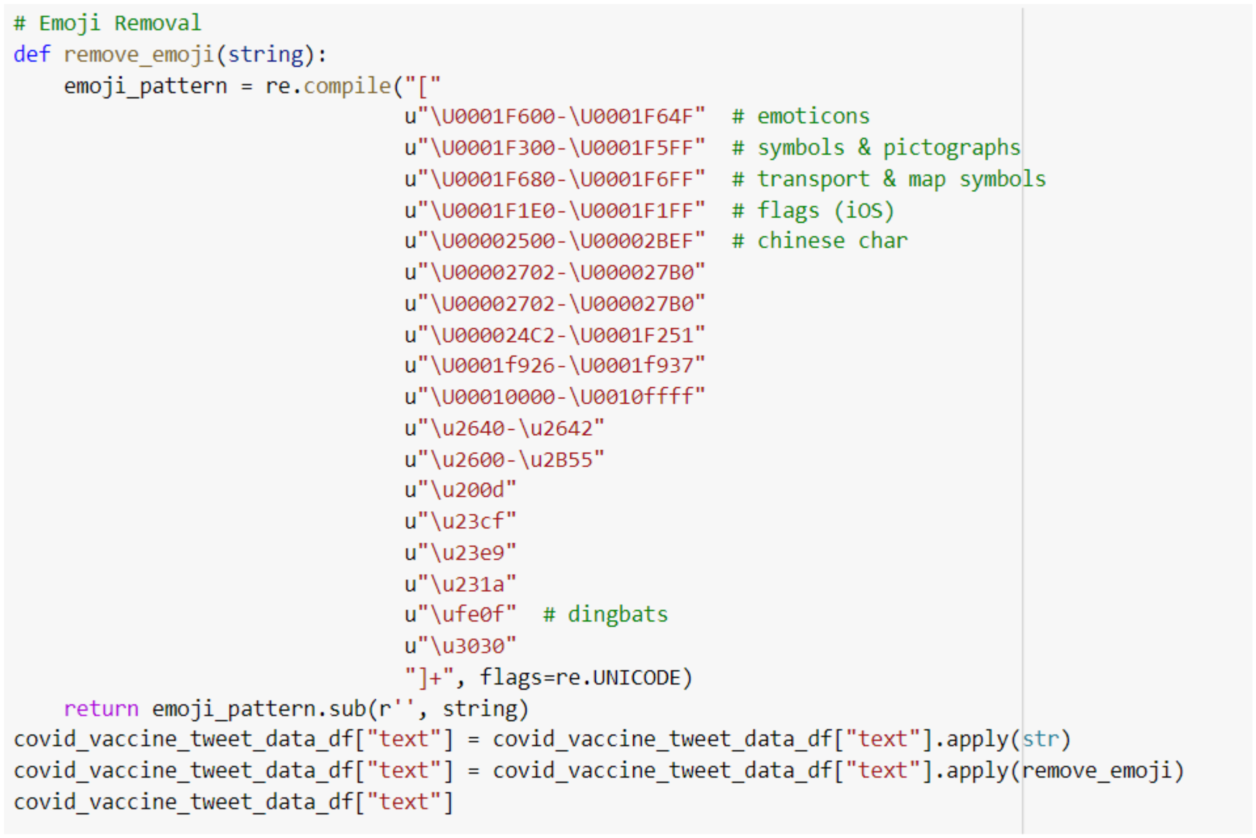
```
# Lowercase removal
covid_vaccine_tweet_data_df['text'] = covid_vaccine_tweet_data_df['text'].apply(lambda x: " ".join(x.lower() for x in x.split()))

covid_vaccine_tweet_data_df['text']
```

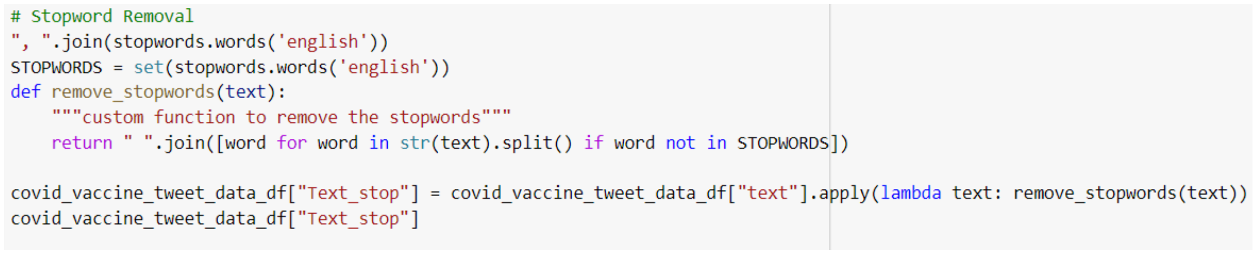


```
# Punctuation Removal

covid_vaccine_tweet_data_df['text'] = covid_vaccine_tweet_data_df['text'].str.replace('[^\w\s]','')
covid_vaccine_tweet_data_df.head()
```



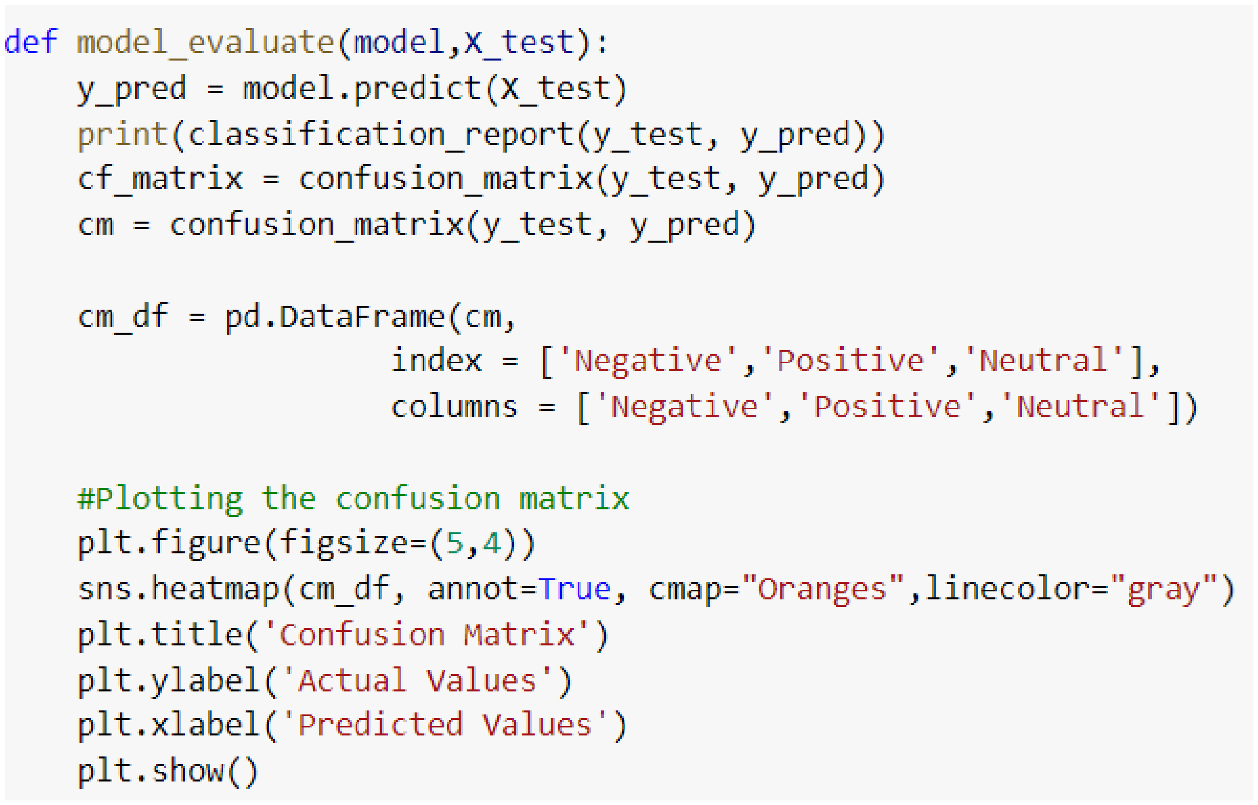
```
# Single character and double space removal
covid_vaccine_tweet_data_df["text"] = covid_vaccine_tweet_data_df["text"].apply(lambda x:re.sub(r'\s+[a-zA-Z]\s+', '', x))
covid_vaccine_tweet_data_df["text"] = covid_vaccine_tweet_data_df["text"].apply(lambda x:re.sub(r'\s+', ' ', x, flags=re.I))
covid_vaccine_tweet_data_df["text"]
```


```
# Emoji Removal
def remove_emoji(string):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002500-\U00002BEF"  # chinese char
                               u"\U00002702-\U000027B0"
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               u"\U0001f926-\U0001f937"
                               u"\U00010000-\U0010ffff"
                               u"\u2640-\u2642"
                               u"\u2600-\u2B55"
                               u"\u200d"
                               u"\u23cf"
                               u"\u23e9"
                               u"\u231a"
                               u"\ufe0f"  # dingbats
                               u"\u3030"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)
covid_vaccine_tweet_data_df["text"] = covid_vaccine_tweet_data_df["text"].apply(str)
covid_vaccine_tweet_data_df["text"] = covid_vaccine_tweet_data_df["text"].apply(remove_emoji)
covid_vaccine_tweet_data_df["text"]
```


```
# Stopword Removal
", ".join(stopwords.words('english'))
STOPWORDS = set(stopwords.words('english'))
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])

covid_vaccine_tweet_data_df["Text_stop"] = covid_vaccine_tweet_data_df["text"].apply(lambda text: remove_stopwords(text))
covid_vaccine_tweet_data_df["Text_stop"]
```

2) Model Evaluation:

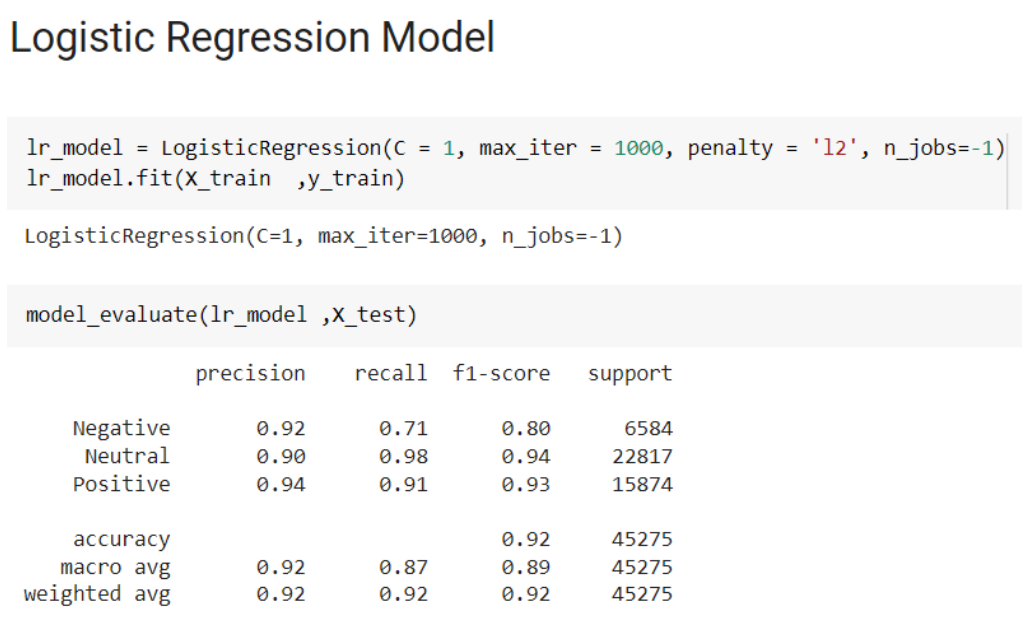
For model evaluation, we have written a common function that displays the confusion matrix along with precision, recall, f1-score and support. Below is the screenshot of the function that we will use and call and set the parameters as our model and the testing data.



```python
def model_evaluate(model,X_test):
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    cf_matrix = confusion_matrix(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

    cm_df = pd.DataFrame(cm,
                     index = ['Negative','Positive','Neutral'],
                     columns = ['Negative','Positive','Neutral'])

    #Plotting the confusion matrix
    plt.figure(figsize=(5,4))
    sns.heatmap(cm_df, annot=True, cmap="Oranges",linecolor="gray")
    plt.title('Confusion Matrix')
    plt.ylabel('Actual Values')
    plt.xlabel('Predicted Values')
    plt.show()
```

Now we will check the confusion matrix for each of the models we run.

LOGISTIC REGRESSION MODEL



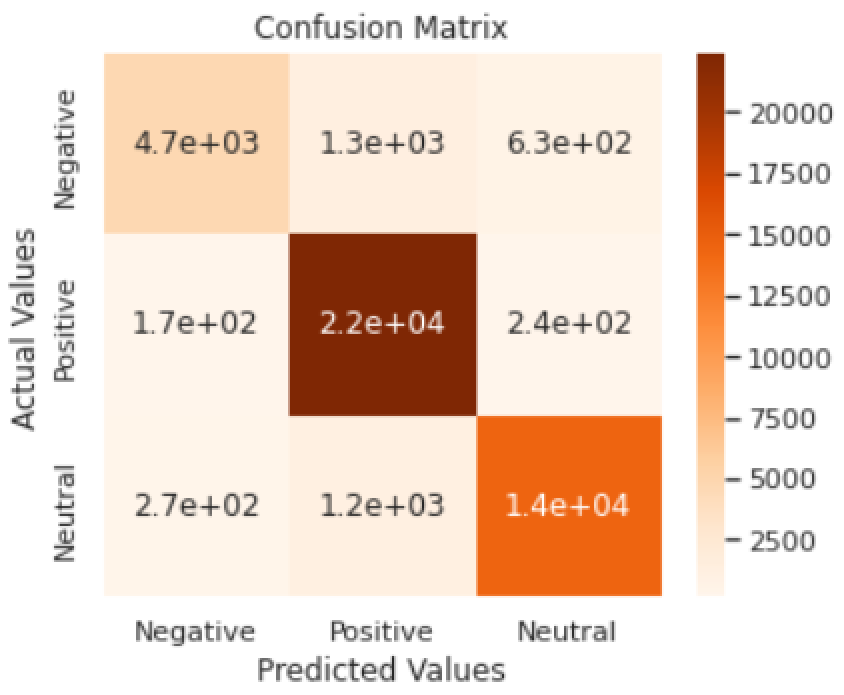
Logistic Regression Model

```
lr_model = LogisticRegression(C = 1, max_iter = 1000, penalty = 'l2', n_jobs=-1)
lr_model.fit(X_train  ,y_train)
```

```
LogisticRegression(C=1, max_iter=1000, n_jobs=-1)
```

```
model_evaluate(lr_model ,X_test)
```

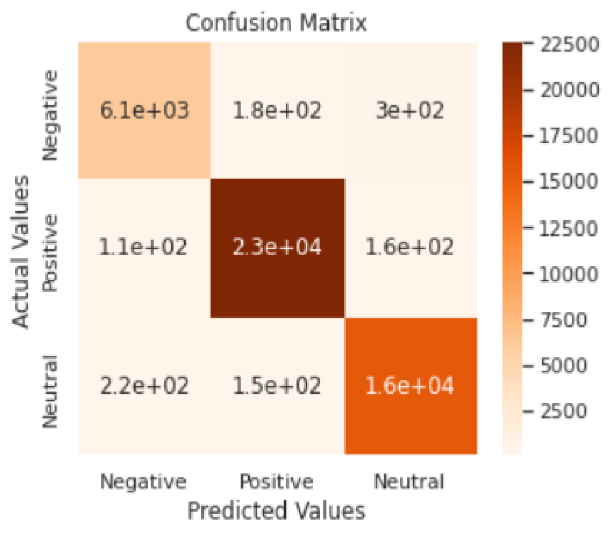
```
              precision    recall  f1-score   support

    Negative       0.92      0.71      0.80      6584
     Neutral       0.90      0.98      0.94     22817
    Positive       0.94      0.91      0.93     15874

    accuracy                           0.92     45275
   macro avg       0.92      0.87      0.89     45275
weighted avg       0.92      0.92      0.92     45275
```



Accuracy for logistic regression obtained: 92%

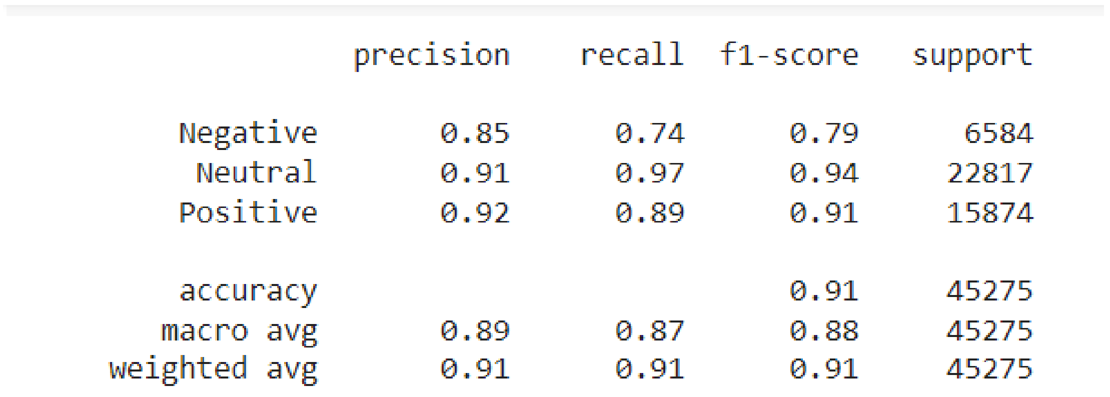
LOGISTIC REGRESSION MODEL AFTER HYPER TUNING

```
              precision    recall  f1-score   support

    Negative       0.95      0.93      0.94      6584
     Neutral       0.99      0.99      0.99     22817
    Positive       0.97      0.98      0.97     15874

    accuracy                           0.98     45275
   macro avg       0.97      0.96      0.97     45275
weighted avg       0.98      0.98      0.98     45275
```

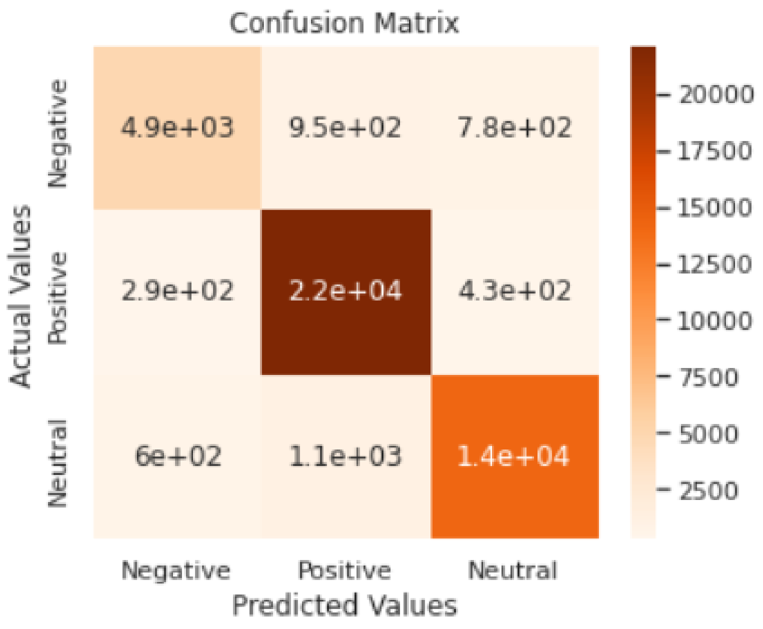


The accuracy of Logistic Regression Model has been increased to **98%** after using hyperparameters.

DECISION TREE MODEL



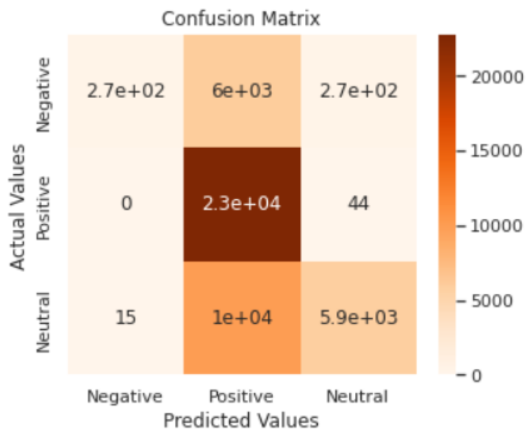
```
              precision    recall  f1-score   support

    Negative       0.85      0.74      0.79      6584
     Neutral       0.91      0.97      0.94     22817
    Positive       0.92      0.89      0.91     15874

    accuracy                           0.91     45275
   macro avg       0.89      0.87      0.88     45275
weighted avg       0.91      0.91      0.91     45275
```



Accuracy of Decision tree base model is **91%**

DECISION TREE MODEL AFTER HYPER TUNING

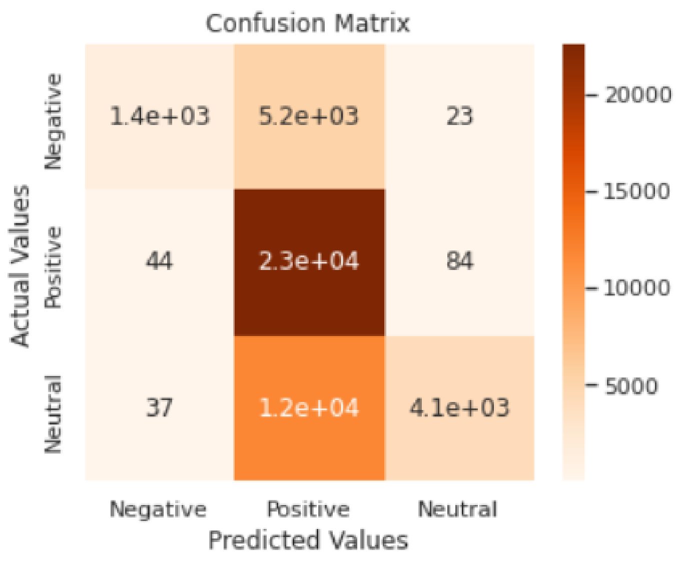
```
              precision    recall  f1-score   support

    Negative       0.95      0.04      0.08      6584
     Neutral       0.59      1.00      0.74     22817
    Positive       0.95      0.37      0.53     15874

    accuracy                           0.64     45275
   macro avg       0.83      0.47      0.45     45275
weighted avg       0.77      0.64      0.57     45275
```



Using hyperparameter tuning on Decision Tree did not have any positive effect as the decline in accuracy can be observed.

K-NEIGHBORS CLASSIFIER

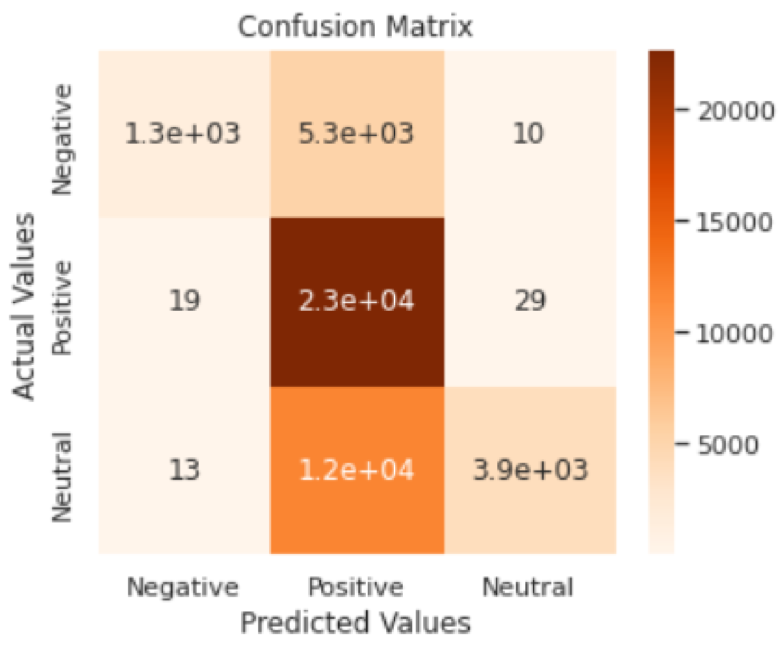
```
              precision    recall  f1-score   support

    Negative       0.95      0.21      0.35      6661
     Neutral       0.57      0.99      0.72     22696
    Positive       0.97      0.26      0.41     15918

    accuracy                           0.62     45275
   macro avg       0.83      0.49      0.49     45275
weighted avg       0.77      0.62      0.56     45275
```



The accuracy of the K-Neighbors Classifier model **62%**

K- NEIGHBORS CLASSIFIER AFTER HYPER TUNING

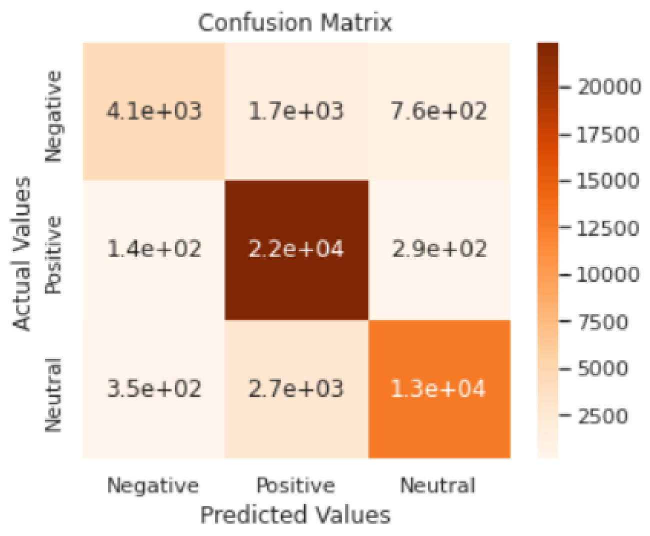
```
              precision    recall  f1-score   support

    Negative       0.98      0.20      0.33      6661
     Neutral       0.57      1.00      0.72     22696
    Positive       0.99      0.25      0.40     15918

    accuracy                           0.62     45275
   macro avg       0.84      0.48      0.48     45275
weighted avg       0.78      0.62      0.55     45275
```



There is no change in accuracy of the model after hyperparameter tuning

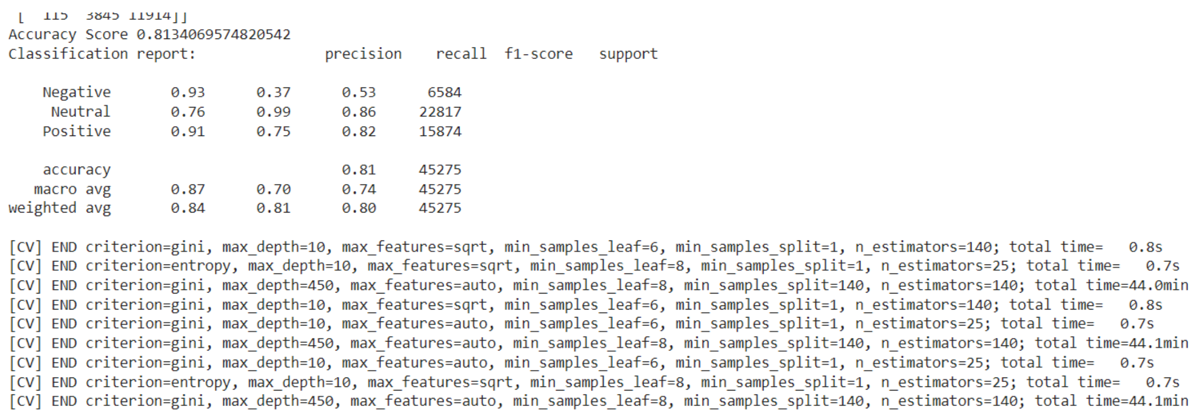
RANDOM FOREST CLASSIFIER

```
              precision    recall  f1-score   support

    Negative       0.89      0.63      0.74      6584
     Neutral       0.84      0.98      0.90     22817
    Positive       0.92      0.81      0.86     15874

    accuracy                           0.87     45275
   macro avg       0.88      0.81      0.83     45275
weighted avg       0.88      0.87      0.86     45275
```



The accuracy obtained for Random Forest Classifier is **87%**

RANDOM FOREST CLASSIFIER AFTER HYPER TUNING

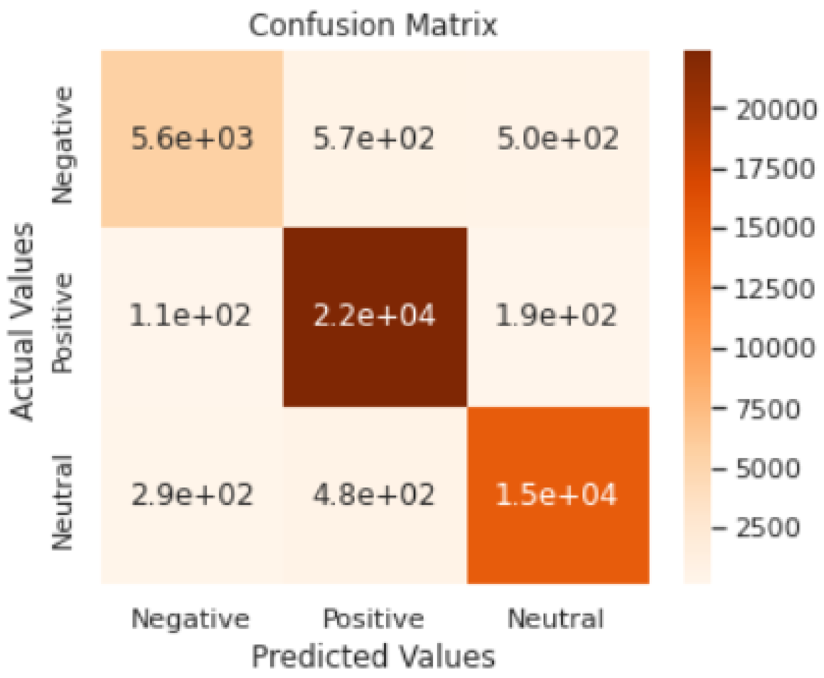


```
[  115  3845 11914]]
Accuracy Score 0.8134069574820542
Classification report:                precision    recall  f1-score   support

    Negative       0.93      0.37      0.53      6584
     Neutral       0.76      0.99      0.86     22817
    Positive       0.91      0.75      0.82     15874

    accuracy                           0.81     45275
   macro avg       0.87      0.70      0.74     45275
weighted avg       0.84      0.81      0.80     45275

[CV] END criterion=gini, max_depth=10, max_features=sqrt, min_samples_leaf=6, min_samples_split=1, n_estimators=140; total time=   0.8s
[CV] END criterion=entropy, max_depth=10, max_features=sqrt, min_samples_leaf=8, min_samples_split=1, n_estimators=25; total time=   0.7s
[CV] END criterion=gini, max_depth=450, max_features=auto, min_samples_leaf=8, min_samples_split=140, n_estimators=140; total time=44.0min
[CV] END criterion=gini, max_depth=10, max_features=sqrt, min_samples_leaf=6, min_samples_split=1, n_estimators=140; total time=   0.8s
[CV] END criterion=gini, max_depth=10, max_features=auto, min_samples_leaf=6, min_samples_split=1, n_estimators=25; total time=   0.7s
[CV] END criterion=gini, max_depth=450, max_features=auto, min_samples_leaf=8, min_samples_split=140, n_estimators=140; total time=44.1min
[CV] END criterion=gini, max_depth=10, max_features=auto, min_samples_leaf=6, min_samples_split=1, n_estimators=25; total time=   0.7s
[CV] END criterion=entropy, max_depth=10, max_features=sqrt, min_samples_leaf=8, min_samples_split=1, n_estimators=25; total time=   0.7s
[CV] END criterion=gini, max_depth=450, max_features=auto, min_samples_leaf=8, min_samples_split=140, n_estimators=140; total time=44.1min
```

The Accuracy for Random Forest Classifier after tuning is **81%**

LINEAR SVC MODEL

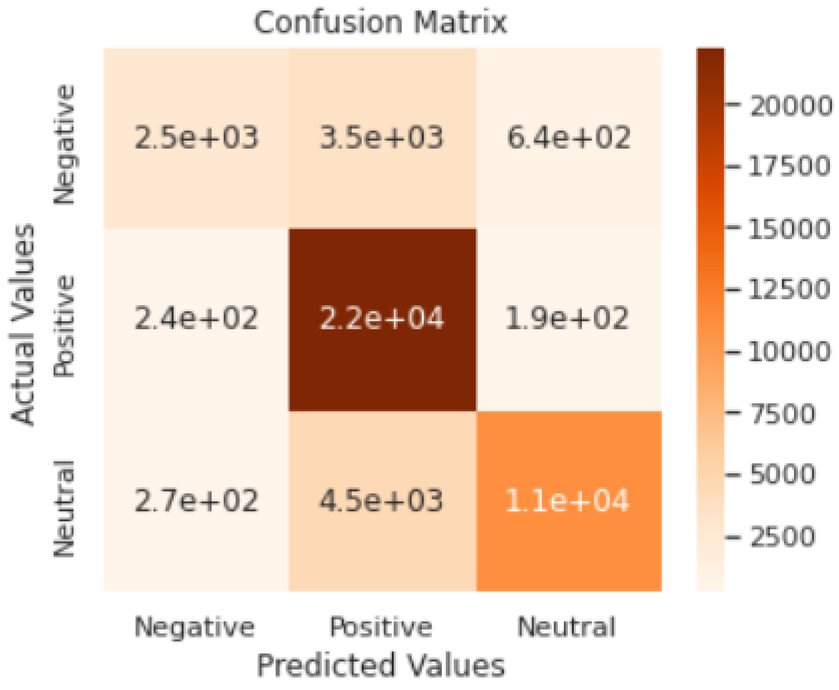
```
              precision    recall  f1-score   support

    Negative       0.93      0.84      0.88      6661
     Neutral       0.96      0.99      0.97     22696
    Positive       0.96      0.95      0.95     15918

    accuracy                           0.95     45275
   macro avg       0.95      0.93      0.94     45275
weighted avg       0.95      0.95      0.95     45275
```



Accuracy for LinearSVC model is **95%**

ADA BOOST CLASSIFIER

```
              precision    recall  f1-score   support

    Negative       0.83      0.38      0.52      6661
     Neutral       0.74      0.98      0.84     22696
    Positive       0.93      0.70      0.80     15918

    accuracy                           0.79     45275
   macro avg       0.83      0.69      0.72     45275
weighted avg       0.82      0.79      0.78     45275
```



Accuracy for AdaBoostCkassifier is **79%**

RESULTS AND CONCLUSION

Model	**Accuracy**	**Accuracy after Hyperparameter Tuning**
Logistic Regression	92%	98%
Decision Tree Model	91%	64% (grid search) 50% (random search)
K-Neighbors Classifier	64%	64%
Random Forest Classifier	87%	81%

Model	**Accuracy**
Linear SVC	95%
AdaBoost Classifier	79%

After a comparative study on the models above, we conclude that the classification model using the Logistic regression algorithm is the best model with an accuracy score of 98% after hyperparameter tuning.