```
!pip install nltk
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.9/dist-packages (3.8.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from nltk) (4.65.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.9/dist-packages (from nltk) (2022.10.31)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from nltk) (8.1.3)
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages (from nltk) (1.2.0)
```

Import Library

```
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt

from nltk.corpus import stopwords

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split
```

Load Dataset from Local Directory

```
from google.colab import files
uploaded = files.upload()
```

Choose Files twitter_training.csv

- twitter_training.csv(text/csv) - 10325088 bytes, last modified: 8/9/2021 - 100% done

```
Saving twitter_training.csv to twitter_training (1).csv
```

Importing Dataset

```
dataset = pd.read_csv('twitter_training.csv')
print(dataset.shape)
print(dataset.head(5))
```

```
(74681, 4)
   2401  Borderlands  Positive  \
0  2401  Borderlands  Positive
1  2401  Borderlands  Positive
2  2401  Borderlands  Positive
3  2401  Borderlands  Positive
4  2401  Borderlands  Positive

  im getting on borderlands and i will murder you all ,
0  I am coming to the borders and I will kill you...
1  im getting on borderlands and i will kill you ...
2  im coming on borderlands and i will murder you...
3  im getting on borderlands 2 and i will murder ...
4  im getting into borderlands and i can murder y...
```

Segregating Dataset into input & output

```
features = dataset.iloc[:, 3].values
labels = dataset.iloc[:, 2].values
print(labels)
```

```
['Positive' 'Positive' 'Positive' ... 'Positive' 'Positive' 'Positive']
```

Removing the Special Character

```
processed_features = []

for sentence in range(0, len(features)):
  #Remove all the special characters
  processed_feature = re.sub(r'\W', ' ', str(features[sentence]))

  # remove all single characters
  processed_feature = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

  # Remove single characters from the start
  processed_feature = re.sub(r'^[a-zA-Z]\s+', ' ', processed_feature)

  # Substituting multiple spaces with single space
  processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re. I)

  #Removing prefixed 'b'
  processed_feature = re.sub(r'^b\s+', '', processed_feature)

  # Converting to Lowercase
  processed_feature = processed_feature.lower()

  processed_features.append(processed_feature)
```

Feature Extraction from text

```
nltk.download('stopwords')
vectorizer = TfidfVectorizer()
processed_features = vectorizer.fit_transform(processed_features).toarray()
print(processed_features)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

Splitting Dataset into Train & Test

```
X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size = 0.2, random_state = 0)
```

Loading Random Forest Algorithm

```
text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)
```

Prediction the Test data with Trained Model

```
predictions = text_classifier.predict(X_test)
```

Score of the Model

```
print(accuracy_score(y_test, predictions))
```

Confusion Matrix

```
from sklearn import metrics
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
  plt.imshow(cm, interpolation='nearest', cmap=cmap)
  plt.title(title)
  plt.colorbar()
  tick_marks = np.arange(len(classes))
  plt.xticks(tick_marks, classes)
  plt.yticks(tick_marks, classes)

  thresh = cm.max() / 2.
  for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[1, 1]> thresh else "black")
  plt.tight_layout()
  plt.ylabel('True label')
  plt.xlabel('Predicted label')

cm = metrics.confusion_matrix(y_test, predictions, labels=['negative', 'neutral', 'positive'])
plot_confusion_matrix(cm, classes=['negative', 'neutral', 'positive'])
```