In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('twitter_training.csv')
```

In [3]:
```
df.head(2)
```

Out[3]:

	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
2401                                                  0
Borderlands                                           0
Positive                                              0
im getting on borderlands and i will murder you all ,    686
dtype: int64
```

In [5]:
```
df.columns = ['ID','Place','Remarks','Text']
```

In [6]:
```
df.head(2)
```

Out[6]:

	ID	Place	Remarks	Text
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...

In [7]:
```
df.drop(['ID','Place'], axis = 1, inplace = True)
```

In [8]:
```
df = df[['Text','Remarks']]
```

In [9]:
```
# For the sake of simplicity we will consider irrelvant remarks as neutral
```

In [10]:
```
df['Remarks'].value_counts()
```

Out[10]:
```
Negative      22542
Positive      20831
Neutral       18318
Irrelevant    12990
Name: Remarks, dtype: int64
```

In [11]:
```
df['Remarks'] = df['Remarks'].replace('Irrelevant','Neutral')
```

In [12]:
```
df['Remarks'].value_counts()
```

Out[12]:
```
Neutral     31308
Negative    22542
Positive    20831
Name: Remarks, dtype: int64
```

In [13]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Remarks'] = le.fit_transform(df['Remarks'])
```

In [14]:
```
df.head(4)
```

Out[14]:

	Text	Remarks
0	I am coming to the borders and I will kill you...	2
1	im getting on borderlands and i will kill you ...	2
2	im coming on borderlands and i will murder you...	2
3	im getting on borderlands 2 and i will murder ...	2

In [15]:
```
df['Remarks'].value_counts()
```

Out[15]:
```
1    31308
0    22542
2    20831
Name: Remarks, dtype: int64
```

Cleaning the Text

In [16]:
```
import re
from nltk.corpus import stopwords
```

In [17]:
```
def clean_text(text):
    if isinstance(text, str):  # Check if the value is a string (not NaN)
        text = text.lower()

        text = re.sub('\[.*?\]', '', text)

        text = re.sub('https?://\S+|www\.\S+', '', text)

        text = re.sub('[^a-zA-Z0-9\s]+', '', text)

        text = re.sub('\w*\d\w*', '', text)

        stop_words = set(stopwords.words('english'))
        words = text.split()

        filtered_words = [word for word in words if word not in stop_words]
        text = ' '.join(filtered_words)

        text = re.sub('\s+', ' ', text).strip()

    else:
        text = str(np.nan)  # Convert non-string values to NaN again

    return text
```

In [18]:
```
df['Text'] = df['Text'].apply(clean_text)
```

In [19]:
```
df.head(5)
```

Out[19]:

	Text	Remarks
0	coming borders kill	2
1	im getting borderlands kill	2
2	im coming borderlands murder	2
3	im getting borderlands murder	2
4	im getting borderlands murder	2

In [20]:
```
import nltk
nltk.download('stopwords')
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\shwet\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\shwet\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[20]: True

In [21]:
```
all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
```

In [22]:
```
from nltk.stem import SnowballStemmer

# initialize SnowballStemmer
stemmer = SnowballStemmer('english')

def stem_text(text):
    # Tokenize the input text into individual words
    tokens = nltk.word_tokenize(text)

    # Stem each token using the SnowballStemmer
    stemmed_tokens = [stemmer.stem(token) for token in tokens if not token in set(all_stopwor

    # Join the stemmed tokens back into a single string
    return ' '.join(stemmed_tokens)
```

In [23]:
```
df['Text'] = df['Text'].apply(stem_text)
```

In [24]:
```
df.head()
```

Out[24]:

	Text	Remarks
0	come border kill	2
1	im get borderland kill	2
2	im come borderland murder	2
3	im get borderland murder	2
4	im get borderland murder	2

Training Bag of Words model

In [25]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# Define TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Vectorize the text data
X = vectorizer.fit_transform(df['Text'])

# Define target variable
y = df['Remarks']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Training different classification Models

In [26]:
```
# SVM Model

from sklearn.svm import SVC
classifier = SVC(kernel = "linear", random_state = 0)
classifier.fit(X_train, y_train)
```

Out[26]:
```
SVC(kernel='linear', random_state=0)
```

In [28]:
```
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Out[28]:
```
[[3502  737  232]
 [ 493 5233  486]
 [ 314  807 3133]]
0.7945370556336614
```

In [29]:
```
# Random Forest

from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(criterion = 'entropy', n_estimators = 10, random_state =
classifier.fit(X_train, y_train)
```

Out[29]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

In [30]:
```
y_pred = classifier.predict(X_test)
```

In [31]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

Out[31]:
```
[[3870  477  124]
 [ 216 5776  220]
 [ 145  596 3513]]
0.8809667269197295
```

In [35]:
```
# Logistic Regression

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0 )
classifier.fit(X_train, y_train)
```

```
C:\Users\shwet\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[35]:
```
LogisticRegression(random_state=0)
```

In [36]:
```
y_pred = classifier.predict(X_test)
```

In [37]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[3324  868  279]
 [ 537 5158  517]
 [ 315  955 2984]]
0.7676240208877284
```

Out[37]:

In [38]:
```
# KNN

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski',p=2)
classifier.fit(X_train,y_train)
```

Out[38]:
```
KNeighborsClassifier()
```

In [39]:
```
y_pred = classifier.predict(X_test)
```

In [40]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[3986  278  207]
 [ 376 5505  331]
 [ 269  414 3571]]
0.8744727856999397
```

Out[40]:

In [41]:
```
# Decision Tree

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = "entropy", random_state = 0)
classifier.fit(X_train, y_train)
```

Out[41]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [42]:
```
y_pred = classifier.predict(X_test)
```

In [43]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[3536  702  233]
 [ 437 5282  493]
 [ 248  736 3270]]
0.8092655821115351
```

Out[43]:

In [44]:
```
# Kernel SVM

from sklearn.svm import SVC
classifier = SVC(kernel = "rbf", random_state = 0)
classifier.fit(X_train, y_train)
```

Out[44]:
```
SVC(random_state=0)
```

In [45]:
```
y_pred = classifier.predict(X_test)
```

In [46]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[3974  409   88]
 [ 105 5970  137]
 [  88  479 3687]]
0.9125661109995313
```

Out[46]:

In [ ]: