```
#Naive Bayes Approach
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report, accuracy_score
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

train_dataset = 'train.csv'
test_dataset = 'test.csv'

# Check if the path exists
print (os.path.exists(train_dataset))
print (os.path.exists(test_dataset))
```

```
True
True
```

```
train_df = pd.read_csv(train_dataset, encoding='ISO-8859-1')
test_df = pd.read_csv(test_dataset, encoding='ISO-8859-1')

train_df.head()
```

```
{"summary":"{\n  \"name\": \"train_df\",\n  \"rows\": 27481,\n
\"fields\": [\n    {\n      \"column\": \"textID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 27481,\n        \"samples\": [\n
\"a7f72a928a\",\n          \"ef42dee96c\",\n          \"07d17131b1\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"text\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
27480,\n        \"samples\": [\n          \" Enjoy! Family trumps
everything\",\n          \" --of them kinda turns me off of it all.
And then I buy more of them and dig a deeper hole, etc. ;;\",\n
\"Clive it`s my birthday pat me
http://apps.facebook.com/dogbook/profile/view/6386106\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"selected_text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 22430,\n        \"samples\": [\n
\"that is why I drive a (teeny tiny) honda civic\",\n
\"Sorry...but, I bet they aren`t that bad...\",\n          \"yummy\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"sentiment\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"neutral\",\n          \"negative\",\n          \"positive\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Time of Tweet\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"morning\",\n          \"noon\",\n          \"night\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age of User\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n          \"0-
20\",\n          \"21-30\",\n          \"70-100\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
195,\n        \"samples\": [\n          \"Philippines\",\n
\"Belgium\",\n          \"Sierra Leone\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Population -2020\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
150494590,\n        \"min\": 801,\n        \"max\": 1439323776,\n
\"num_unique_values\": 195,\n        \"samples\": [\n
109581078,\n          11589623,\n          7976983\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Land Area (Km\\u00b2)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1807424.6900064405,\n        \"min\": 0.0,\n        \"max\":
16376870.0,\n        \"num_unique_values\": 193,\n        \"samples\":
[\n          2267050.0,\n          1280000.0,\n          100250.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Density (P/Km\\u00b2)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2013,\n        \"min\": 2,\n        \"max\": 26337,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          400,\n
71,\n          331\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"train_df"}
```

```
train_df.tail()
```

```
{"summary":"{\n  \"name\": \"train_df\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"textID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"4f4c4fc327\",\n          \"6f7127d9d7\",\n          \"f67aae2310\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"text\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \" I`ve wondered about rake to.  The client
has made it clear .NET only, don`t force devs to learn a new lang
#agile #ccnet\",\n          \"   All this flirting going on - The ATG
smiles. Yay.  ((hugs))\",\n          \" Yay good for both of you.
Enjoy the break - you probably need it after such hectic weekend  Take
care hun xxxx\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"selected_text\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \", don`t force\",\n          \"All this flirting going
on - The ATG smiles. Yay.  ((hugs)\",\n          \"Yay good for both
of you.\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sentiment\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"negative\",\n          \"positive\",\n
\"neutral\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Time of Tweet\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"night\",\n          \"morning\",\n          \"noon\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age of User\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"46-
60\",\n          \"0-20\",\n          \"60-70\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Country\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"Greece\",\n          \"Guinea\",\n
\"Grenada\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Population -2020\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 11311650,\n        \"min\": 112523,\n
\"max\": 31072940,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          10423054,\n          13132795,\n
112523\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Land Area (Km\\u00b2)\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 99481.25712916981,\n
\"min\": 340.0,\n        \"max\": 246000.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n
128900.0,\n          246000.0,\n          340.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Density (P/Km\\u00b2)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
108,\n        \"min\": 53,\n        \"max\": 331,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          81,\n
53,\n          331\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   textID            27481 non-null  object 
 1   text              27480 non-null  object 
 2   selected_text     27480 non-null  object 
 3   sentiment         27481 non-null  object 
 4   Time of Tweet     27481 non-null  object 
 5   Age of User       27481 non-null  object 
 6   Country           27481 non-null  object 
 7   Population -2020  27481 non-null  int64  
 8   Land Area (Km²)   27481 non-null  float64
 9   Density (P/Km²)   27481 non-null  int64  
dtypes: float64(1), int64(2), object(7)
memory usage: 2.1+ MB
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4815 entries, 0 to 4814
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   textID            3534 non-null   object 
 1   text              3534 non-null   object 
 2   sentiment         3534 non-null   object 
 3   Time of Tweet     3534 non-null   object 
 4   Age of User       3534 non-null   object 
 5   Country           3534 non-null   object 
 6   Population -2020  3534 non-null   float64
 7   Land Area (Km²)   3534 non-null   float64
 8   Density (P/Km²)   3534 non-null   float64
dtypes: float64(3), object(6)
memory usage: 338.7+ KB
```

```
train_df.isnull().sum()
```

```
textID              0
text                1
selected_text       1
sentiment           0
Time of Tweet       0
Age of User         0
Country             0
Population -2020    0
Land Area (Km²)     0
Density (P/Km²)     0
dtype: int64
```

```
train_df = train_df.dropna()
train_df.isnull().sum()
```

```
textID              0
text                0
selected_text       0
sentiment           0
Time of Tweet       0
Age of User         0
Country             0
Population -2020    0
Land Area (Km²)     0
Density (P/Km²)     0
dtype: int64
```

```
test_df.isnull().sum()
```

```
textID              1281
text                1281
sentiment           1281
Time of Tweet       1281
Age of User         1281
Country             1281
Population -2020    1281
Land Area (Km²)     1281
Density (P/Km²)     1281
dtype: int64
```

```
test_df = test_df.dropna()
test_df.isnull().sum()
```

```
textID              0
text                0
sentiment           0
Time of Tweet       0
Age of User         0
Country             0
Population -2020    0
Land Area (Km²)     0
Density (P/Km²)     0
dtype: int64
```

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
True
```

```
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    text = " ".join([word for word in text.split() if word not in
stop_words])
    return text

train_df['processed_text'] = train_df['text'].apply(preprocess_text)
train_df.head()
```

```
{"summary":"{\n  \"name\": \"train_df\",\n  \"rows\": 27480,\n
\"fields\": [\n    {\n      \"column\": \"textID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 27480,\n        \"samples\": [\n
\"6c5505a37c\",\n          \"126b1e6a22\",\n          \"5bc4e623c4\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"text\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
27480,\n        \"samples\": [\n          \" Enjoy! Family trumps
everything\",\n          \" --of them kinda turns me off of it all.
And then I buy more of them and dig a deeper hole, etc. ;;\",\n
\"Clive it`s my birthday pat me
http://apps.facebook.com/dogbook/profile/view/6386106\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"selected_text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 22430,\n        \"samples\": [\n
\"that is why I drive a (teeny tiny) honda civic\",\n
\"Sorry...but, I bet they aren`t that bad...\",\n          \"yummy\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"sentiment\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"neutral\",\n          \"negative\",\n          \"positive\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Time of Tweet\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"morning\",\n          \"noon\",\n          \"night\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age of User\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n          \"0-
20\",\n          \"21-30\",\n          \"70-100\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
195,\n        \"samples\": [\n          \"Philippines\",\n
\"Belgium\",\n          \"Sierra Leone\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Population -2020\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
150497157,\n        \"min\": 801,\n        \"max\": 1439323776,\n
\"num_unique_values\": 195,\n        \"samples\": [\n
109581078,\n          11589623,\n          7976983\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Land Area (Km\\u00b2)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1807457.3166921895,\n        \"min\": 0.0,\n        \"max\":
16376870.0,\n        \"num_unique_values\": 193,\n        \"samples\":
[\n          2267050.0,\n          1280000.0,\n          100250.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Density (P/Km\\u00b2)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2013,\n        \"min\": 2,\n        \"max\": 26337,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          400,\n
71,\n          331\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"processed_text\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 27268,\n
\"samples\": [\n          \"thought like really hot. room hot
sleep\",\n          \"praying get better soon sweet one , sorry still
well\",\n          \"damm feel like song dead gone travis garland\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"train_df"}
```

```python
test_df['processed_text'] = test_df['text'].apply(preprocess_text)
test_df.head()
```

```
{"summary":"{\n  \"name\": \"test_df\",\n  \"rows\": 3534,\n
\"fields\": [\n    {\n      \"column\": \"textID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3534,\n        \"samples\": [\n
\"142108215\",\n          \"fb08563a7b\",\n          \"9a2c6ae21c\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"text\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
3534,\n        \"samples\": [\n          \" Thank you so much
phaoloo !!!!\",\n          \"Midnight ice-cream weather! So ****
bored\",\n          \"Ohh i forgot to tell you last night that when i
was a alton towers i touched a shark  it was amazing !!!! it was nt a
massive one tho\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sentiment\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"neutral\",\n          \"positive\",\n
\"negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Time of Tweet\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
```

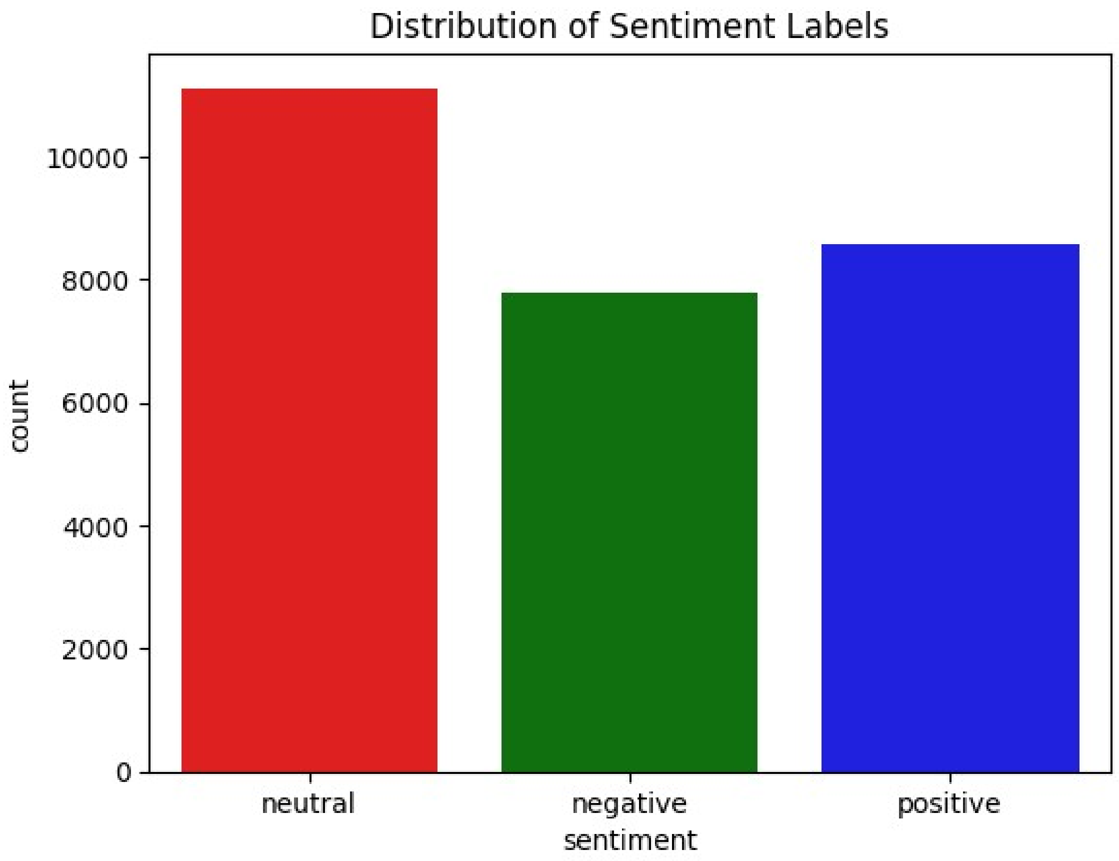
```
[\n          \"morning\",\n          \"noon\",\n          \"night\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age of User\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n          \"0-
20\",\n          \"21-30\",\n          \"70-100\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
195,\n        \"samples\": [\n          \"Philippines\",\n
\"Belgium\",\n          \"Sierra Leone\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Population -2020\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
146875664.36244348,\n        \"min\": 801.0,\n        \"max\":
1439323776.0,\n        \"num_unique_values\": 195,\n
\"samples\": [\n          109581078.0,\n          11589623.0,\n
7976983.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Land Area (Km\\u00b2)\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1839133.911427382,\n
\"min\": 0.0,\n        \"max\": 16376870.0,\n
\"num_unique_values\": 193,\n        \"samples\": [\n
2267050.0,\n          1280000.0,\n          100250.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Density (P/Km\\u00b2)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1967.012367010644,\n        \"min\": 2.0,\n        \"max\": 26337.0,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          400.0,\
n          71.0,\n          331.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"processed_text\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3527,\n        \"samples\": [\n
\"thank much phaoloo !!!!\",\n          \"i`m glad you`re little
prissy well. it`s obvious much love w/the treatment she`s getting\",\n
\"3d version sold regular version is!\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"test_df"}
```

```python
sns.countplot(x='sentiment', data=train_df, palette=['red', 'green',
'blue'])
plt.title("Distribution of Sentiment Labels")
plt.show()
```

```
<ipython-input-17-a42c8626a2cf>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
 sns.countplot(x='sentiment', data=train_df, palette=['red', 'green',
'blue'])
```



```
vectorizer = TfidfVectorizer(max_features=5000)
train_TFIDF = vectorizer.fit_transform(train_df['processed_text'])
test_TFIDF = vectorizer.transform(test_df['processed_text'])

# Assign X and y for the training data
X = train_TFIDF
y = train_df['sentiment']

NB_model = MultinomialNB()
NB_model.fit(X, y)

MultinomialNB()

predictions = NB_model.predict(test_TFIDF)

print(f"Accuracy: {accuracy_score(test_df['sentiment'],
predictions)}")
print(classification_report(test_df['sentiment'], predictions))
```

```
Accuracy: 0.6386530843237125
              precision    recall  f1-score   support

    negative       0.73      0.50      0.59      1001
     neutral       0.55      0.77      0.65      1430
    positive       0.76      0.60      0.67      1103

    accuracy                           0.64      3534
   macro avg       0.68      0.62      0.63      3534
weighted avg       0.67      0.64      0.64      3534
```

```
input_text = ["What a bad product!"]

input_TFIDF = vectorizer.transform(input_text)
predicted_sentiment = NB_model.predict(input_TFIDF)
print(f"Predicted sentiment: {predicted_sentiment[0]}")
```

```
Predicted sentiment: negative
```

```
input_text = ["Last session of the day"]

input_TFIDF = vectorizer.transform(input_text)
predicted_sentiment = NB_model.predict(input_TFIDF)
print(f"Predicted sentiment: {predicted_sentiment[0]}")
```

```
Predicted sentiment: neutral
```

```
input_text = ["I hate meetings!"]

input_TFIDF = vectorizer.transform(input_text)
predicted_sentiment = NB_model.predict(input_TFIDF)
print(f"Predicted sentiment: {predicted_sentiment[0]}")
```

```
Predicted sentiment: negative
```

```
input_text = ["I love cars!"]

input_TFIDF = vectorizer.transform(input_text)
predicted_sentiment = NB_model.predict(input_TFIDF)
print(f"Predicted sentiment: {predicted_sentiment[0]}")
```

```
Predicted sentiment: positive
```

```
input_text = ["Do you want help?"]

input_TFIDF = vectorizer.transform(input_text)
predicted_sentiment = NB_model.predict(input_TFIDF)
print(f"Predicted sentiment: {predicted_sentiment[0]}")
```

```
Predicted sentiment: neutral
```

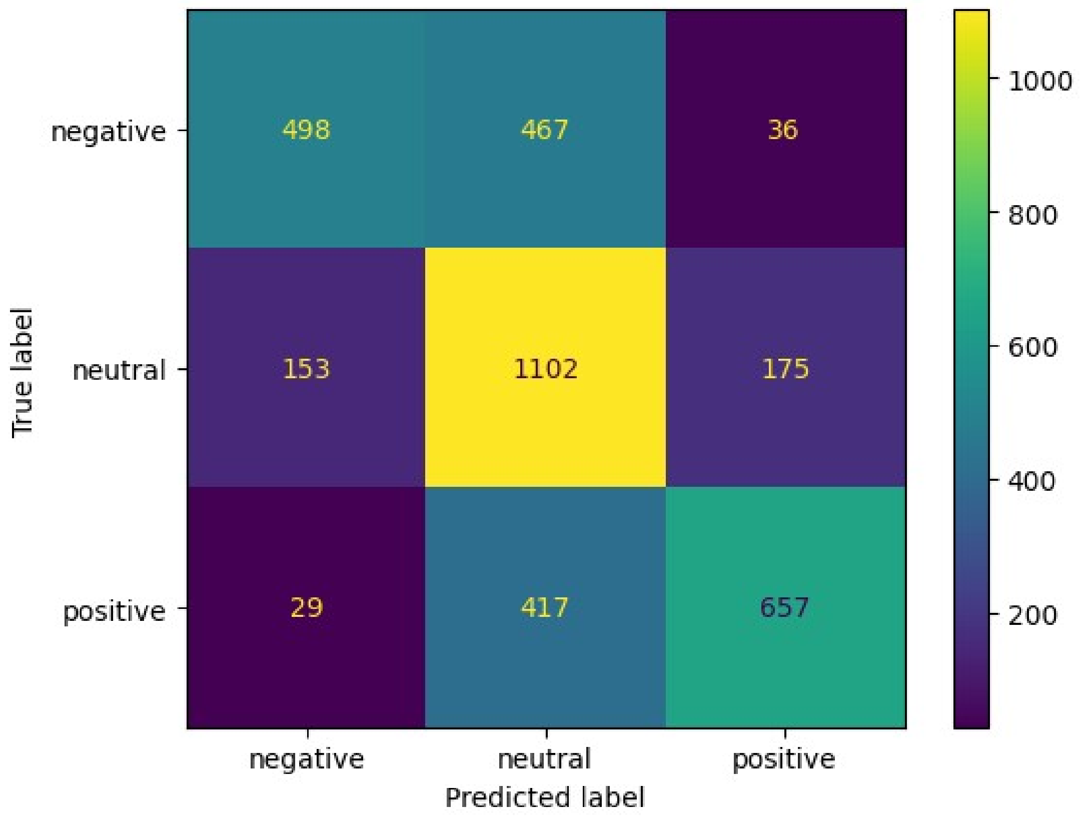
```
input_text = ["You are good girl"]

input_TFIDF = vectorizer.transform(input_text)
predicted_sentiment = NB_model.predict(input_TFIDF)
print(f"Predicted sentiment: {predicted_sentiment[0]}")
```

```
Predicted sentiment: positive
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(test_df['sentiment'], predictions)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=NB_model.classes_)
disp.plot()
plt.show()
```



```
import joblib

# Assuming you have already trained your model and vectorizer
joblib.dump(NB_model, 'NB_model.pkl')       # Save the Naive Bayes
model
joblib.dump(vectorizer, 'vectorizer.pkl')   # Save the TF-IDF
vectorizer
```

```
['vectorizer.pkl']
```

```python
from google.colab import files


files.download('NB_model.pkl')
files.download('vectorizer.pkl')
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
```