import libraries

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

import dataset

```
df=pd.read_csv("/content/twitter_training.csv")
```

```
df.rename(columns={'2401': 'id','Borderlands': 'place','Positive':'sentiment','im getting on borderlands and i will murder you all ,':'Review
```

```
df.head()
```

	id	place	sentiment	Review
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 ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

Next steps: Generate code with df View recommended plots

PreProcessing

```
df['Review'] = df['Review'].astype(str).apply(lambda x: x.lower())
df['Review'] = df['Review'].apply(lambda x: x.replace(r'[^\w\s]', ''))
```

Tokenize the text

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['Review'])
```

Split the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, df['sentiment'], test_size=1.0, random_state=42)
```

```
# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: 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(
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Evaluate the model
y_pred = model.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7705027783356765
```

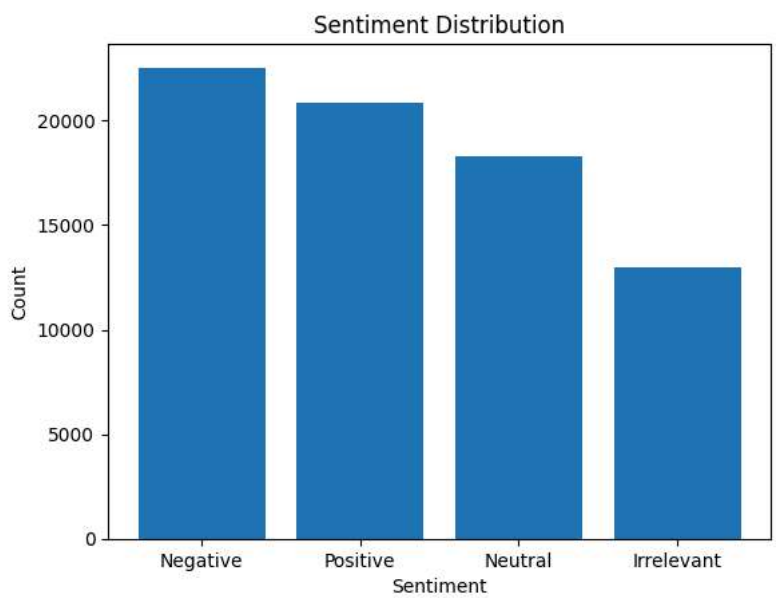
```
print('Classification Report:')
print(classification_report(y_test, y_pred))
```

```
Classification Report:
                   precision    recall  f1-score   support

       Irrelevant       0.80      0.64      0.71      2661
         Negative       0.79      0.84      0.81      4471
          Neutral       0.74      0.74      0.74      3551
         Positive       0.77      0.81      0.79      4254

         accuracy                           0.77     14937
        macro avg       0.77      0.76      0.76     14937
     weighted avg       0.77      0.77      0.77     14937
```

```
# Visualize sentiment distribution
sentiment_counts = df['sentiment'].value_counts()
plt.bar(sentiment_counts.index, sentiment_counts.values)
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Sentiment Distribution')
plt.show()
```



Test or Predict

```
sample_tweets = [
    "I love this product, it's amazing!",
    "The customer service was terrible, I'm so disappointed.",
    "This is a pretty good app, I'll keep using it.",
    "I hate this company, they're the worst!"
]
```

```
sample_X = vectorizer.transform(sample_tweets)
```

```
sample_y_pred = model.predict(sample_X)
```

```
for tweet, sentiment in zip(sample_tweets, sample_y_pred):
    print(f"Tweet: {tweet}")
    print(f"Predicted Sentiment: {sentiment}")
    print()
```

```
Tweet: I love this product, it's amazing!
Predicted Sentiment: Positive

Tweet: The customer service was terrible, I'm so disappointed.
```

```
Predicted Sentiment: Negative

Tweet: This is a pretty good app, I'll keep using it.
Predicted Sentiment: Positive

Tweet: I hate this company, they're the worst!
Predicted Sentiment: Negative
```