SABIN CHAULAGAIN 2358554

Importing libaries

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
import re
import string

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import nltk

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

Exercise

Load Dataset

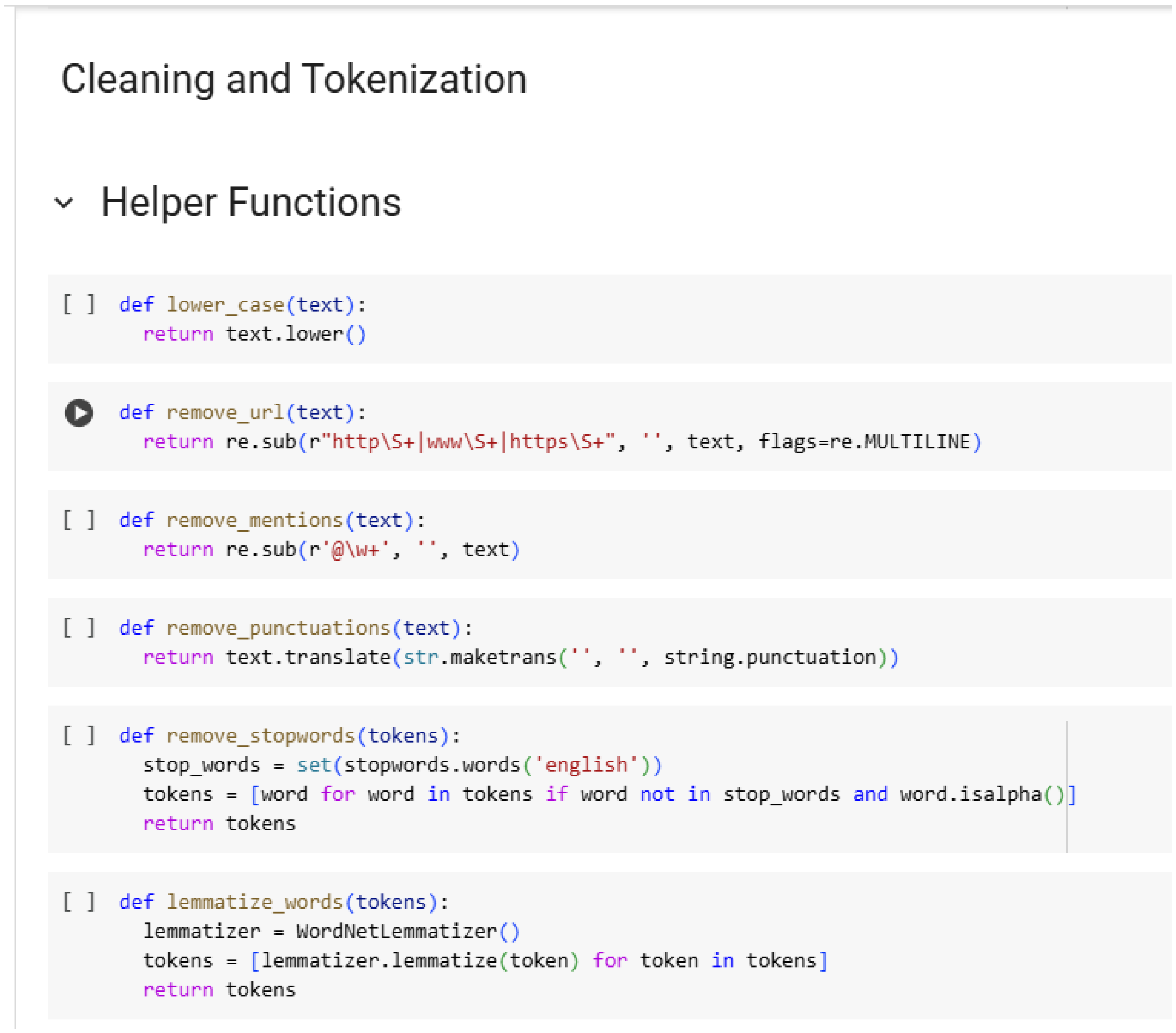
```python
df = pd.read_csv("/content/drive/MyDrive/Artificial intelligence and Machine learning/Week-8/Workshop/trum_tweet_sentiment_analysis.csv")
```

```python
df.columns
```

```
Index(['text', 'Sentiment'], dtype='object')
```

```python
assert 'text' in df.columns and 'Sentiment' in df.columns, "Dataset must contain 'text' and 'sentiment' columns."
```

Cleaning and Tokenization



Cleaning and Tokenization

Helper Functions

```python
def lower_case(text):
  return text.lower()
```

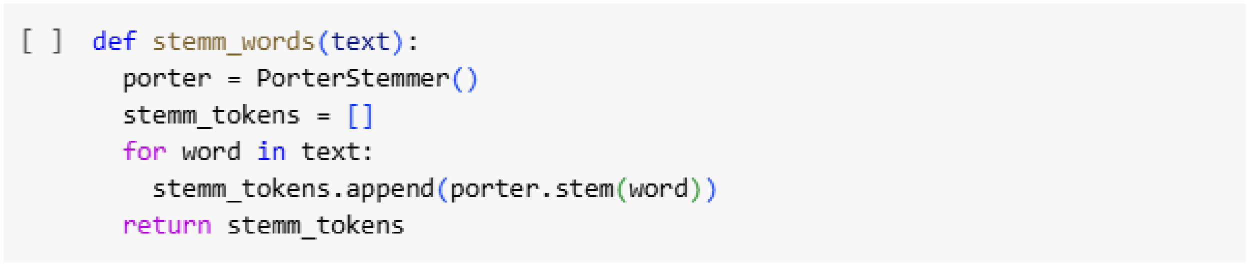
```python
def remove_url(text):
  return re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
```

```python
def remove_mentions(text):
  return re.sub(r'@\w+', '', text)
```

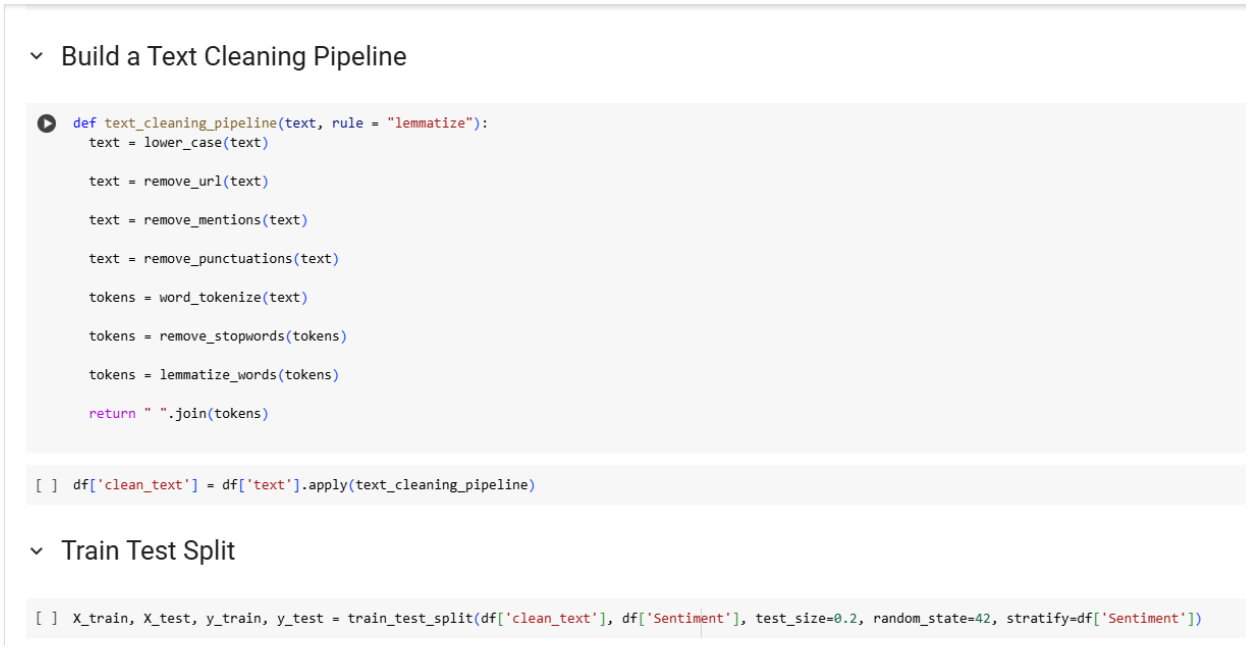
```python
def remove_punctuations(text):
  return text.translate(str.maketrans('', '', string.punctuation))
```

```python
def remove_stopwords(tokens):
  stop_words = set(stopwords.words('english'))
  tokens = [word for word in tokens if word not in stop_words and word.isalpha()]
  return tokens
```

```python
def lemmatize_words(tokens):
  lemmatizer = WordNetLemmatizer()
  tokens = [lemmatizer.lemmatize(token) for token in tokens]
  return tokens
```



```python
def stemm_words(text):
  porter = PorterStemmer()
  stemm_tokens = []
  for word in text:
    stemm_tokens.append(porter.stem(word))
  return stemm_tokens
```



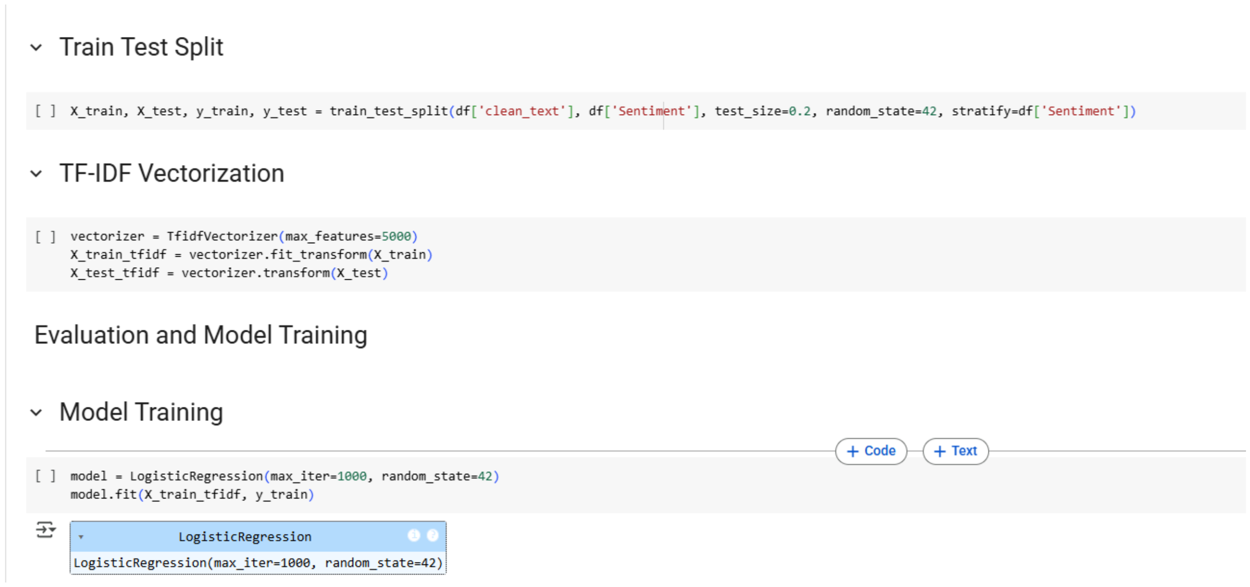
Build a Text Cleaning Pipeline

```
def text_cleaning_pipeline(text, rule = "lemmatize"):
    text = lower_case(text)

    text = remove_url(text)

    text = remove_mentions(text)

    text = remove_punctuations(text)

    tokens = word_tokenize(text)

    tokens = remove_stopwords(tokens)

    tokens = lemmatize_words(tokens)

    return " ".join(tokens)
```

```
[ ] df['clean_text'] = df['text'].apply(text_cleaning_pipeline)
```

Train Test Split

```
[ ] X_train, X_test, y_train, y_test = train_test_split(df['clean_text'], df['Sentiment'], test_size=0.2, random_state=42, stratify=df['Sentiment'])
```



Train Test Split

```
[ ] X_train, X_test, y_train, y_test = train_test_split(df['clean_text'], df['Sentiment'], test_size=0.2, random_state=42, stratify=df['Sentiment'])
```

TF-IDF Vectorization

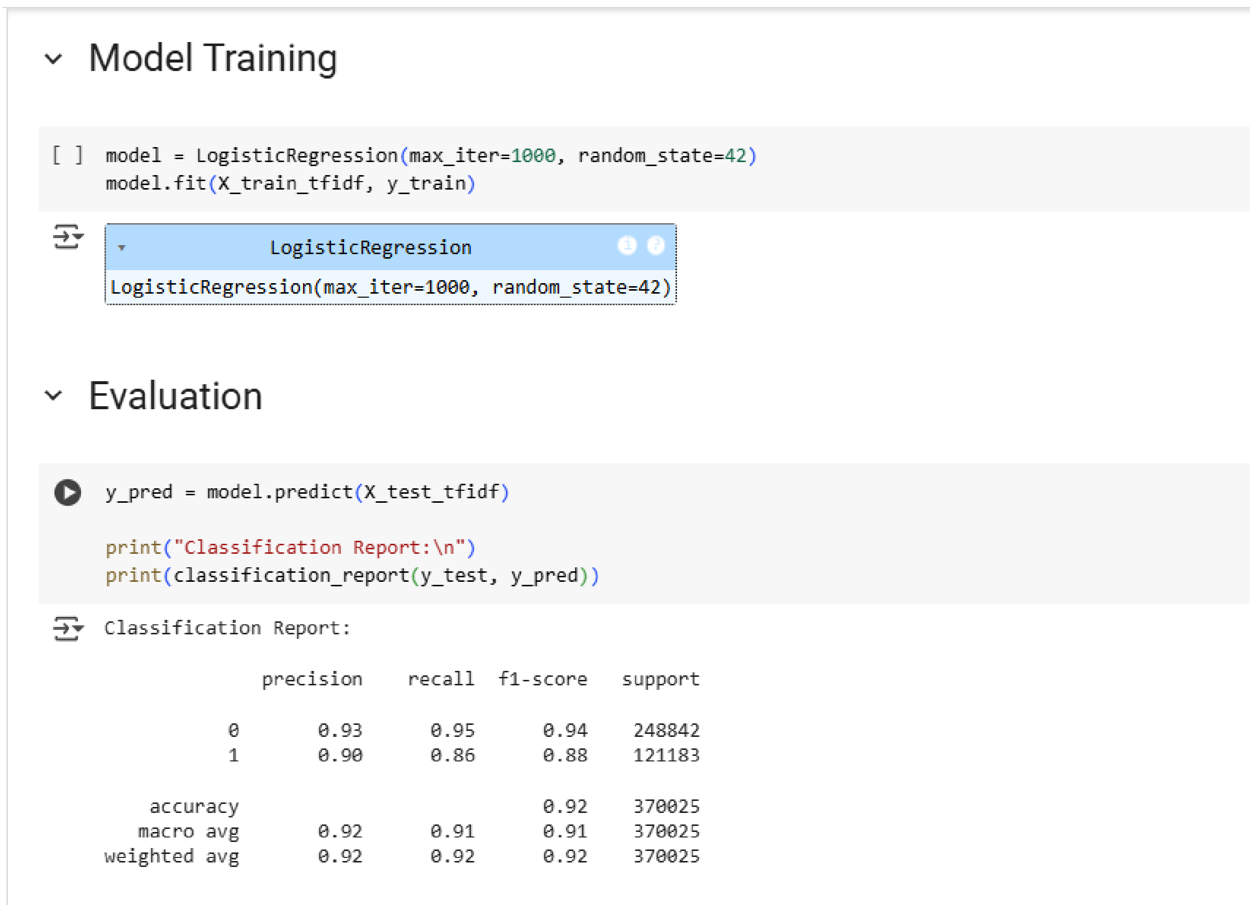
```
[ ] vectorizer = TfidfVectorizer(max_features=5000)
    X_train_tfidf = vectorizer.fit_transform(X_train)
    X_test_tfidf = vectorizer.transform(X_test)
```

Evaluation and Model Training

Model Training

```
[ ] model = LogisticRegression(max_iter=1000, random_state=42)
    model.fit(X_train_tfidf, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```



Model Training

```
[ ] model = LogisticRegression(max_iter=1000, random_state=42)
    model.fit(X_train_tfidf, y_train)
```

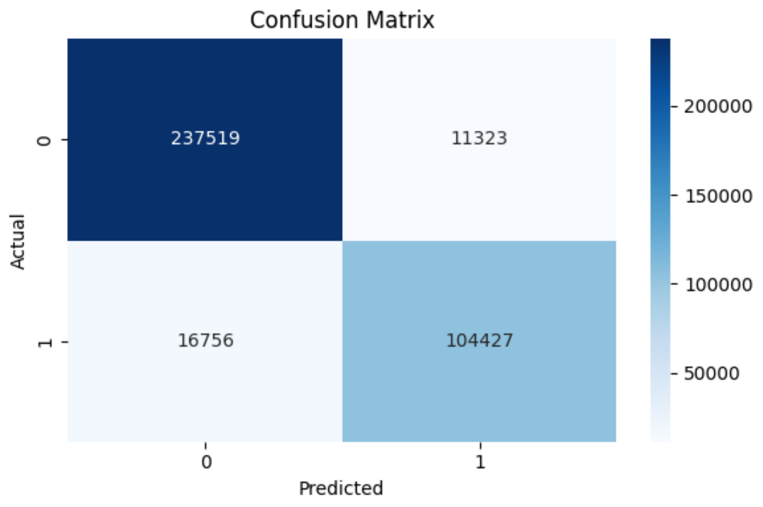
```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

Evaluation

```
y_pred = model.predict(X_test_tfidf)

print("Classification Report:\n")
print(classification_report(y_test, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.95      0.94    248842
           1       0.90      0.86      0.88    121183

    accuracy                           0.92    370025
   macro avg       0.92      0.91      0.91    370025
weighted avg       0.92      0.92      0.92    370025
```

```python
cm = confusion_matrix(y_test, y_pred, labels=model.classes_)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap="Blues", xticklabels=model.classes_, yticklabels=model.classes_)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```



```python
pred_df = pd.DataFrame({
    'Cleaned Text': X_test,
    'Actual Sentiment': y_test,
    'Predicted Sentiment': y_pred
})
pred_df.head()
```

	Cleaned Text	Actual Sentiment	Predicted Sentiment
1432084	rt maralago member pay trump hundred thousand ...	0	0
133054	rt seriously arkansas even trump know samesex ...	1	1
345307	rt bercow prefers north korea president trump ...	0	0
717727	rt breaking trump right look found raided mosq...	1	1
741002	rt edited robocop trump speech actually make s...	0	0