```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Helper Function for Text Cleaning:

Implement a Helper Function as per Text Preprocessing Notebook and Complete the following pipeline.

Build a Text Cleaning Pipeline

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer

nltk.download('stopwords')
nltk.download('wordnet')  # Required for WordNetLemmatizer

def text_cleaning_pipeline(dataset, rule="lemmatize"):
    # 1. Lowercasing the text
    data = dataset.lower()

    # 2. Removing URLs
    data = re.sub(r'http\S+|www\S+|https\S+', '', data, flags=re.MULTILINE)

    # 3. Removing emojis and punctuation
    data = re.sub(r'[^\w\s]', '', data)

    # 4. Create tokens
    tokens = data.split()

    # 5. Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [w for w in tokens if not w in stop_words]

    # 6. Lemmatization or Stemming
    if rule == "lemmatize":
        lemmatizer = WordNetLemmatizer()
        tokens = [lemmatizer.lemmatize(w) for w in tokens]
    elif rule == "stem":
        stemmer = PorterStemmer()
        tokens = [stemmer.stem(w) for w in tokens]
    else:
        print("Pick between lemmatize or stem")

    return " ".join(tokens)
```

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

Text Classification using Machine Learning Models

📝 Instructions: Trump Tweet Sentiment Classification

1. **Load the Dataset**
   Load the dataset named `"trump_tweet_sentiment_analysis.csv"` using `pandas`. Ensure the dataset contains at least two columns: `"text"` and `"label"`.
2. **Text Cleaning and Tokenization**
   Apply a text preprocessing pipeline to the `"text"` column. This should include:
   - Lowercasing the text
   - Removing URLs, mentions, punctuation, and special characters
   - Removing stopwords

- Tokenization (optional: stemming or lemmatization)
- "Complete the above function"

3. **Train-Test Split**
   Split the cleaned and tokenized dataset into **training** and **testing** sets using `train_test_split` from `sklearn.model_selection`.
4. **TF-IDF Vectorization**
   Import and use the `TfidfVectorizer` from `sklearn.feature_extraction.text` to transform the training and testing texts into numerical feature vectors.
5. **Model Training and Evaluation**
   Import **Logistic Regression** (or any machine learning model of your choice) from `sklearn.linear_model`. Train it on the TF-IDF-embedded training data, then evaluate it using the test set.
   - Print the **classification report** using `classification_report` from `sklearn.metrics`.

```
import pandas as pd

# Load the dataset
df = pd.read_csv('/content/drive/MyDrive/AI/trum_tweet_sentiment_analysis.csv')

# Check if 'text' and 'label' columns exist
if 'text' in df.columns and 'label' in df.columns:
  print("Dataset loaded successfully with 'text' and 'label' columns.")
else:
  print("Dataset does not contain the required columns.")

# Print the columns of the dataset
print(df.columns)
```

```
Dataset does not contain the required columns.
Index(['text', 'Sentiment'], dtype='object')
```

```
from sklearn.model_selection import train_test_split

# Assuming 'df' is your DataFrame containing the 'text' and 'label' columns
X = df['text']  # Get the text data
y = df['Sentiment'] # Get the corresponding labels

# Apply the cleaning pipeline to the text data
X_cleaned = X.apply(text_cleaning_pipeline)

# Now use the cleaned data for the split
X_train, X_test, y_train, y_test = train_test_split(X_cleaned, y, test_size=0.2, random_state=42)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# TF-IDF Vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # You can adjust max_features
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Model Training and Evaluation
model = LogisticRegression(solver='liblinear')  # You can choose a different solver
model.fit(X_train_tfidf, y_train)
y_pred = model.predict(X_test_tfidf)

# Print the classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94    248563
           1       0.90      0.86      0.88    121462

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