Name: Renjen Sherpa

Group: L6CG7

WORKSHEET 8

Necessary Imports

```
[1] import pandas as pd
    import re
    import string
    import nltk
    from nltk.corpus import stopwords
    from nltk.stem import PorterStemmer
    from nltk.stem import WordNetLemmatizer
    from nltk.tokenize import word_tokenize
    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

    # Download all required NLTK data
    nltk.download('punkt')
    nltk.download('stopwords')
    nltk.download('wordnet')
    nltk.download('omw-1.4')  # Required for WordNet lemmatization
```

```
    # Download all required NLTK data
    nltk.download('punkt')
    nltk.download('stopwords')
    nltk.download('wordnet')
    nltk.download('omw-1.4')  # Required for WordNet lemmatization
    nltk.download('punkt_tab')  # Specifically for the punkt tables

    # Initialize stemmer and lemmatizer
    stemmer = PorterStemmer()
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
[2] 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

```
def text_cleaning_pipeline(text, rule="lemmatize"):
    """
    This function performs text cleaning and preprocessing on input text.
    Steps include:
    1. Lowercasing
    2. Removing URLs
    3. Removing emojis
    4. Removing punctuation and special characters
    5. Tokenization
    6. Stopword removal
    7. Stemming or lemmatization

    Parameters:
    - text: Input text to clean
    - rule: Either "lemmatize" or "stem" for word normalization

    Returns:
    - Cleaned text as a single string
    """
    if not isinstance(text, str):
        return ""
```

```
    """
    if not isinstance(text, str):
        return ""

    # Convert the input to lowercase
    data = text.lower()

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

    # Remove emojis
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    data = emoji_pattern.sub(r'', data)

    # Remove all other unwanted characters
    data = re.sub(r'@\w+', '', data)  # Remove mentions
    data = re.sub(r'#', '', data)  # Remove hashtag symbol
    data = re.sub(r'[^\w\s]', '', data)  # Remove punctuation
    data = re.sub(r'\d+', '', data)  # Remove numbers

    # Create tokens
    tokens = word_tokenize(data)

    # Remove stopwords
```

```
    # Remove all other unwanted characters
    data = re.sub(r'@\w+', '', data)  # Remove mentions
    data = re.sub(r'#', '', data)  # Remove hashtag symbol
    data = re.sub(r'[^\w\s]', '', data)  # Remove punctuation
    data = re.sub(r'\d+', '', data)  # Remove numbers

    # Create tokens
    tokens = word_tokenize(data)

    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words and word not in string.punctuation]

    # Apply stemming or lemmatization
    if rule == "lemmatize":
        tokens = [lemmatizer.lemmatize(word) for word in tokens]
    elif rule == "stem":
        tokens = [stemmer.stem(word) for word in tokens]
    else:
        print("Pick between lemmatize or stem")

    return " ".join(tokens)
```

```
[3]
    return " ".join(tokens)
```

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"`.

```
# Load the dataset - handling potential empty rows
df = pd.read_csv("/content/drive/MyDrive/Artificial intelligence and machine learning/Worksheet8/trum_tweet_sentiment_analysis.csv", usecols=['text', 'Sentiment'])
df = df.dropna(subset=['text'])  # Drop rows where text is NaN
df = df[df['text'].str.strip().astype(bool)]  # Drop empty strings
```

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"

```
# Apply text cleaning pipeline
print("Cleaning text data...")
df['cleaned_text'] = df['text'].apply(lambda x: text_cleaning_pipeline(x, rule="lemmatize"))

# Check class distribution
print("\nSentiment distribution:")
print(df['Sentiment'].value_counts())
```

```
Cleaning text data...
```

Executing (5m 51s) <cell line: 0> › apply() › apply() › apply_standard() › _map_values() › map_array() › <lambda> › text_cleaning_pipeline() › word_tokenize() › <listcomp> › tokenize()

```
# Apply text cleaning pipeline
print("Cleaning text data...")
df['cleaned_text'] = df['text'].apply(lambda x: text_cleaning_pipeline(x, rule="lemmatize"))

# Check class distribution
print("\nSentiment distribution:")
print(df['Sentiment'].value_counts())
```

```
Cleaning text data...

Sentiment distribution:
Sentiment
0    1244211
1     605912
Name: count, dtype: int64
```

3. **Train-Test Split**
   Split the cleaned and tokenized dataset into **training** and **testing** sets using `train_test_split` from `sklearn.model_selection`.

3. **Train-Test Split**
   Split the cleaned and tokenized dataset into **training** and **testing** sets using `train_test_split` from `sklearn.model_selection`.

```
# Train-test split
X = df['cleaned_text']
y = df['Sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

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.

```python
# TF-IDF Vectorization
print("\nVectorizing text data...")
tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
Vectorizing text data...
```

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.

   embedded training data, then evaluate it using the test set.

   - Print the **classification report** using `classification_report` from `sklearn.metrics`.

```python
# Model Training and Evaluation
print("\nTraining model...")
model = LogisticRegression(max_iter=1000, class_weight='balanced')
model.fit(X_train_tfidf, y_train)
y_pred = model.predict(X_test_tfidf)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Negative (0)', 'Positive (1)']))

# Print some example predictions
print("\nSample predictions:")
sample = X_test.sample(5, random_state=42)
for text, pred in zip(sample, model.predict(tfidf_vectorizer.transform(sample))):
    print(f"\nOriginal: {df[df['cleaned_text'] == text]['text'].values[0]}")
    print(f"Cleaned: {text}")
    print(f"Predicted sentiment: {pred}")
```

```
Training model...

Classification Report:
              precision    recall  f1-score   support

Negative (0)       0.94      0.90      0.92    248842
Positive (1)       0.82      0.89      0.85    121183
```

```
Training model...

Classification Report:
              precision    recall  f1-score   support

Negative (0)       0.94      0.90      0.92    248842
Positive (1)       0.82      0.89      0.85    121183

    accuracy                           0.90    370025
   macro avg       0.88      0.90      0.89    370025
weighted avg       0.90      0.90      0.90    370025


Sample predictions:

Original: RT @Phil_Lewis_: Washington State Attorney General after Trump's "SEE YOU IN COURT" tweet:

?"We have seen him in court twice, and? https://t.co/R48rgQI8Tt
Cleaned: rt washington state attorney general trump see court tweet seen court twice
Predicted sentiment: 1

Original: RT @michaelianblack: Today:
1. Trump lost appeal.
2. NYT broke China won't take our call.
3. Wash P broke Flynn lied about Russia.
4. Conway broke the law.
TODAY
Cleaned: rt today trump lost appeal nyt broke china wont take call wash p broke flynn lied russia conway broke law today
Predicted sentiment: 0

Original: RT @EvilGalProds: omg if Rosie O'Donnell plays Steve Bannon on this weekend's #SNL, it might actually be enough to get Trump to fire him
Cleaned: rt omg rosie odonnell play steve bannon weekend snl might actually enough get trump fire
Predicted sentiment: 0

Original: RT @BreakingNewss3: Trump Lawyers To Court Reviewing Muslim Travel Ban: Stay Out Of It https://t.co/sCffxIlGK3 #breakingnews
Cleaned: rt trump lawyer court reviewing muslim travel ban stay breakingnews
```

```
1. Trump lost appeal.
2. NYT broke China won't take our call.
3. Wash P broke Flynn lied about Russia.
4. Conway broke the law.
TODAY
Cleaned: rt today trump lost appeal nyt broke china wont take call wash p broke flynn lied russia conway broke law today
Predicted sentiment: 0

Original: RT @EvilGalProds: omg if Rosie O'Donnell plays Steve Bannon on this weekend's #SNL, it might actually be enough to get Trump to fire him
Cleaned: rt omg rosie odonnell play steve bannon weekend snl might actually enough get trump fire
Predicted sentiment: 0

Original: RT @BreakingNewss3: Trump Lawyers To Court Reviewing Muslim Travel Ban: Stay Out Of It https://t.co/sCffxIlGK3 #breakingnews
Cleaned: rt trump lawyer court reviewing muslim travel ban stay breakingnews
Predicted sentiment: 0

Original: RT @now7grandkids: Flynn isn't the only character in the 45 foreign policy farce, remember roles played by Page, Manafort, and Trump himself. Investigate now.
Cleaned: rt flynn isnt character foreign policy farce remember role played page manafort trump investigate
Predicted sentiment: 1
```