Data Loading and Cleaning

```
import pandas as pd
```

```
# Loading the movie review dataset and reading the JSON to a dataframe
df_reviews = pd.read_json('/content/drive/MyDrive/IMDB_reviews.json', lines=True)

# Displaying the first few rows of the dataframe
df_reviews.head()
```

	review_date	movie_id	user_id	is_spoiler	review_text	rating	review_summ
0	10 February 2006	tt0111161	ur1898687	True	In its Oscar year, Shawshank Redemption (writt...	10	A classic pie unforgettable ma
1	6 September 2000	tt0111161	ur0842118	True	The Shawshank Redemption is without a	10	Simply ama The best fi the

```
# Displaying the shape of the dataframe
df_reviews.shape
```

```
(573913, 7)
```

```
#Loading the movie information dataset and reading the JSON to a dataframe
df_movies = pd.read_json('/content/drive/MyDrive/IMDB_movie_details.json',lines=Tru

# Displaying the first few rows of the data
df_movies.head()
```

	movie_id	plot_summary	duration	genre	rating	release_date	plot_synop
0	tt0105112	Former CIA analyst, Jack Ryan is in England wi...	1h 57min	[Action, Thriller]	6.9	1992-06-05	Jack Ryan (F is on a "wo vacation"
1	tt1204975	Billy (Michael Douglas), Paddy (Robert De Niro...	1h 45min	[Comedy]	6.6	2013-11-01	Four I around the of 10 are frie

```
# Displaying the shape of the data
df_movies.shape
```

```
(1572, 7)
```

```
#Merging the Movie and Review data on the common field 'movie_id'
merged_df = pd.merge(df_reviews, df_movies, on='movie_id', how='inner')

# Display the merged DataFrameset
```

```
(573906, 13)
```

```
#Exporting the Merged Data as a CSV
merged_df.to_csv('Movie_Review_Dataset_final.csv')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-91c5f0b7d709> in <cell line: 2>()
      1 #Exporting the Merged Data as a CSV
----> 2 merged_df.to_csv('Movie_Review_Dataset_final.csv')

NameError: name 'merged_df' is not defined
```

Next steps: Explain error

```
#Loading the merged dataset into a dataframe
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Movie_Review_Dataset_final.csv')
```

Text Cleaning

```
#Installing necessary packages for text cleaning and preprocessing
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

```
#Text Cleaning for the review text data

#Remove URLS
df['cleaned_review'] = df['review_text'].apply(lambda x: re.sub(r'http\S+|www\S+|

# Remove user mentions
df['cleaned_review'] = df['cleaned_review'].apply(lambda x: re.sub(r'@\w+', '', x

# Remove hashtags
df['cleaned_review'] = df['cleaned_review'].apply(lambda x: re.sub(r'#\w+', '', x

# Remove punctuation
df['cleaned_review'] = df['cleaned_review'].apply(lambda x: x.translate(str.maket

# Remove numbers
df['cleaned_review'] = df['cleaned_review'].apply(lambda x: re.sub(r'\d+', '', x)
# Displaying the cleaned review text column
df['cleaned_review']
```

```
0         In its Oscar year Shawshank Redemption written...
1         The Shawshank Redemption is without a doubt on...
2         I believe that this film is the best story eve...
3         Yes there are SPOILERS hereThis film has had s...
4         At the heart of this extraordinary movie is a ...
                                ...
573901    Go is wise fast and pure entertainment Assembl...
573902    Well what shall I say this one´s fun at any ra...
573903    Go is the best movie I have ever seen and Ive ...
573904    Call this  teenage version of Pulp Fiction wha...
573905    Why was this movie made No doubt to sucker in ...
Name: cleaned_review, Length: 573906, dtype: object
```

Text Preprocessing Techniques

```
#Lowercasing

#Convert review to lowercase
df['cleaned_review'] = df['cleaned_review'].str.lower()
```

```
#Stop word removal

#Download stopwords and wordnet
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')

#Remove stop words
stop_words = set(stopwords.words('english'))

def remove_stop_words(text):
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]
    return ' '.join(filtered_text)

df['cleaned_review'] = df['cleaned_review'].apply(lambda x: remove_stop_words(x))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
#Lemmatization
lemmatizer = WordNetLemmatizer()

def lemmatize_text(text):
    word_tokens = word_tokenize(text)
    lemmatized_text = [lemmatizer.lemmatize(word) for word in word_tokens]
    return ' '.join(lemmatized_text)

df['cleaned_review'] = df['cleaned_review'].apply(lambda x: lemmatize_text(x))
```

```
print(df['cleaned_review'])
```

```
0         oscar year shawshank redemption written direct...
1         shawshank redemption without doubt one brillia...
2         believe film best story ever told film im tell...
3         yes spoiler herethis film emotional impact fin...
4         heart extraordinary movie brilliant indelible ...
                                ...
573901    go wise fast pure entertainment assembling exc...
573902    well shall say one´s fun rate three plotlines ...
573903    go best movie ever seen ive seen lot movie rea...
573904    call teenage version pulp fiction whatever wan...
573905    movie made doubt sucker familyrebelling mtv fa...
Name: cleaned_review, Length: 573906, dtype: object
```

Feature Extraction

Model Creation

Logistic Regression

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, accuracy_score

# Split the data into features and target variable
X = df['cleaned_review']
y = df['is_spoiler']

# Split the data into training and testing sets (75% training, 25% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_

# Initialize the TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)

# Fit and transform the training data
X_train_tfidf = vectorizer.fit_transform(X_train)
```

```
# Transform the testing data
X_test_tfidf = vectorizer.transform(X_test)

# Initialize the RandomUnderSampler
rus = RandomUnderSampler(random_state=42)

# Apply undersampling to the training data
X_train_tfidf_resampled, y_train_resampled = rus.fit_resample(X_train_tfidf, y_tr

# Initialize the Logistic Regression model
model = LogisticRegression(max_iter=2000)

# Perform 5-fold cross-validation on the resampled training set
cv_scores = cross_val_score(model, X_train_tfidf_resampled, y_train_resampled, cv

# Print the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())

# Fit the model on the entire resampled training set
model.fit(X_train_tfidf_resampled, y_train_resampled)

# Make predictions on the test set
y_pred = model.predict(X_test_tfidf)

# Evaluate the model on the test set
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("Test Classification Report:\n", classification_report(y_test, y_pred))
```

```
Cross-validation scores: [0.69164456 0.68642794 0.68961096 0.68907359 0.68969
Mean cross-validation score: 0.6892899149695353
Test Accuracy: 0.7009416143353987
Test Classification Report:
               precision    recall  f1-score   support

       False       0.86      0.71      0.78    105652
        True       0.45      0.66      0.54     37825

    accuracy                           0.70    143477
   macro avg       0.65      0.69      0.66    143477
weighted avg       0.75      0.70      0.72    143477
```

SVM

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, accuracy_score
from imblearn.under_sampling import RandomUnderSampler

# Split the data into features and target variable
X = df['cleaned_review']
y = df['is_spoiler']

# Split the data into training and testing sets (75% training, 25% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_

# Initialize the TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)

# Fit and transform the training data
X_train_tfidf = vectorizer.fit_transform(X_train)

# Transform the testing data
X_test_tfidf = vectorizer.transform(X_test)

# Initialize the RandomUnderSampler
rus = RandomUnderSampler(random_state=42)

# Apply undersampling to the training data
X_train_tfidf_resampled, y_train_resampled = rus.fit_resample(X_train_tfidf, y_tr

# Initialize the SVM model
svm_model = SVC(kernel='linear', max_iter=1000)

# Perform 5-fold cross-validation on the resampled training set
cv_scores = cross_val_score(svm_model, X_train_tfidf_resampled, y_train_resampled

# Print the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())

# Fit the model on the entire resampled training set
svm_model.fit(X_train_tfidf_resampled, y_train_resampled)

# Make predictions on the test set
y_pred = svm_model.predict(X_test_tfidf)
```

```
# Evaluate the model on the test set
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("Test Classification Report:\n", classification_report(y_test, y_pred))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:299: Convergence
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:299: Convergence
  warnings.warn(
Cross-validation scores: [0.54214449 0.53926206]
Mean cross-validation score: 0.5407032776593957
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:299: Convergence
  warnings.warn(
Test Accuracy: 0.5267394774075287
Test Classification Report:
               precision    recall  f1-score   support

       False       0.77      0.51      0.61    105652
        True       0.30      0.58      0.39     37825

    accuracy                           0.53    143477
   macro avg       0.53      0.54      0.50    143477
weighted avg       0.65      0.53      0.55    143477
```

Naive Bayes

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, accuracy_score
from imblearn.under_sampling import RandomUnderSampler

# Split the data into features and target variable
X = df['cleaned_review']
y = df['is_spoiler']

# Split the data into training and testing sets (75% training, 25% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_

# Initialize the TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)
```

```
# Fit and transform the training data
X_train_tfidf = vectorizer.fit_transform(X_train)

# Transform the testing data
X_test_tfidf = vectorizer.transform(X_test)

# Initialize the RandomUnderSampler
rus = RandomUnderSampler(random_state=42)

# Apply undersampling to the training data
X_train_tfidf_resampled, y_train_resampled = rus.fit_resample(X_train_tfidf, y_tr

# Initialize the Naive Bayes model
nb_model = MultinomialNB()

# Perform 5-fold cross-validation on the resampled training set
cv_scores = cross_val_score(nb_model, X_train_tfidf_resampled, y_train_resampled,

# Print the cross-validation scores
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())

# Fit the model on the entire resampled training set
nb_model.fit(X_train_tfidf_resampled, y_train_resampled)

# Make predictions on the test set
y_pred = nb_model.predict(X_test_tfidf)

# Evaluate the model on the test set
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("Test Classification Report:\n", classification_report(y_test, y_pred))
```

```
Cross-validation scores: [0.66553935 0.66584881 0.66419098 0.66681403 0.66480
Mean cross-validation score: 0.6654391318148788
Test Accuracy: 0.6751325996501182
Test Classification Report:
               precision    recall  f1-score   support

       False       0.84      0.69      0.76    105652
        True       0.42      0.64      0.51     37825

    accuracy                           0.68    143477
   macro avg       0.63      0.66      0.63    143477
weighted avg       0.73      0.68      0.69    143477
```

Bi-LTSM

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional, Dropou
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report, accuracy_score

# Split the data into features and target variable
X = df['cleaned_review']
y = df['is_spoiler']

# Split the data into training and testing sets (75% training, 25% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_

# Initialize the CountVectorizer for Bag of Words
vectorizer = CountVectorizer(max_features=5000)

# Fit and transform the training data
X_train_bow = vectorizer.fit_transform(X_train)

# Transform the testing data
X_test_bow = vectorizer.transform(X_test)
```

```
# Convert the sparse matrix to a dense matrix
X_train_bow_dense = X_train_bow.toarray()
X_test_bow_dense = X_test_bow.toarray()

# Initialize the RandomUnderSampler
rus = RandomUnderSampler(random_state=42)

# Apply undersampling to the training data
X_train_resampled, y_train_resampled = rus.fit_resample(X_train_bow_dense, y_trai

# Tokenization and Padding
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X_train_resampled)

X_train_seq = tokenizer.texts_to_sequences(X_train_resampled)
X_test_seq = tokenizer.texts_to_sequences(X_test)

maxlen = 100  # Maximum length of the sequence
X_train_pad = pad_sequences(X_train_seq, padding='post', maxlen=maxlen)
X_test_pad = pad_sequences(X_test_seq, padding='post', maxlen=maxlen)

# Model configuration
embedding_dim = 100
num_classes = 2

# Build the Bi-LSTM model
model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=embedding_dim, input_length=maxlen
model.add(Bidirectional(LSTM(64, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(32)))
model.add(Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(learning_rat

# Train the model
history = model.fit(X_train_pad, y_train_resampled, epochs=5, batch_size=64, vali

# Evaluate the model
y_pred = np.argmax(model.predict(X_test_pad), axis=1)

# Print results
print("Bi-LSTM Accuracy:", accuracy_score(y_test, y_pred))
print("Bi-LSTM Classification Report:\n", classification_report(y_test, y_pred))
```