```python
import os
import shutil
import tarfile
import tensorflow as tf
from transformers import BertTokenizer, TFBertForSequenceClassification
import pandas as pd
from bs4 import BeautifulSoup
import re
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.offline as pyo
import plotly.graph_objects as go
from wordcloud import WordCloud, STOPWORDS
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```python
# Get the current working directory
current_folder = os.getcwd()

dataset = tf.keras.utils.get_file(
  fname ="aclImdb.tar.gz",
  origin ="http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz",
  cache_dir= current_folder,
  extract = True)
```

```
Downloading data from http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
84125825/84125825 [==============================] - 12s 0us/step
```

```python
dataset_path = os.path.dirname(dataset)
# Check the dataset
os.listdir(dataset_path)
```

```
['aclImdb.tar.gz', 'aclImdb']
```

```python
# Dataset directory
dataset_dir = os.path.join(dataset_path, 'aclImdb')

# Check the Dataset directory
os.listdir(dataset_dir)
```

```
['README', 'test', 'imdb.vocab', 'imdbEr.txt', 'train']
```

```python
train_dir = os.path.join(dataset_dir,'train')
os.listdir(train_dir)
```

```
['urls_pos.txt',
 'urls_neg.txt',
 'labeledBow.feat',
 'neg',
 'unsup',
 'unsupBow.feat',
 'urls_unsup.txt',
 'pos']
```

```python
for file in os.listdir(train_dir):
    file_path = os.path.join(train_dir, file)
    # Check if it's a file (not a directory)
    if os.path.isfile(file_path):
        with open(file_path, 'r', encoding='utf-8') as f:
            first_value = f.readline().strip()
            print(f"{file}: {first_value}")
    else:
        print(f"{file}: {file_path}")
```

```
urls_pos.txt: http://www.imdb.com/title/tt0453418/usercomments
urls_neg.txt: http://www.imdb.com/title/tt0064354/usercomments
labeledBow.feat: 9 0:9 1:1 2:4 3:4 4:6 5:4 6:2 7:2 8:4 10:4 12:2 26:1 27:1 28:1 29:2 32:1 41:1 45:1 47:1 50:1 54:2 57:1 59:1 63:2 6
neg: /content/datasets/aclImdb/train/neg
unsup: /content/datasets/aclImdb/train/unsup
unsupBow.feat: 0 0:8 1:6 3:5 4:2 5:1 7:1 8:5 9:2 10:1 11:2 13:3 16:1 17:1 18:1 19:1 22:3 24:1 26:3 28:1 30:1 31:1 35:2 36:1 39:2 40
urls_unsup.txt: http://www.imdb.com/title/tt0018515/usercomments
pos: /content/datasets/aclImdb/train/pos
```

```python
def load_dataset(directory):
    data = {"sentence": [], "sentiment": []}
    for file_name in os.listdir(directory):
        print(file_name)
        if file_name == 'pos':
```

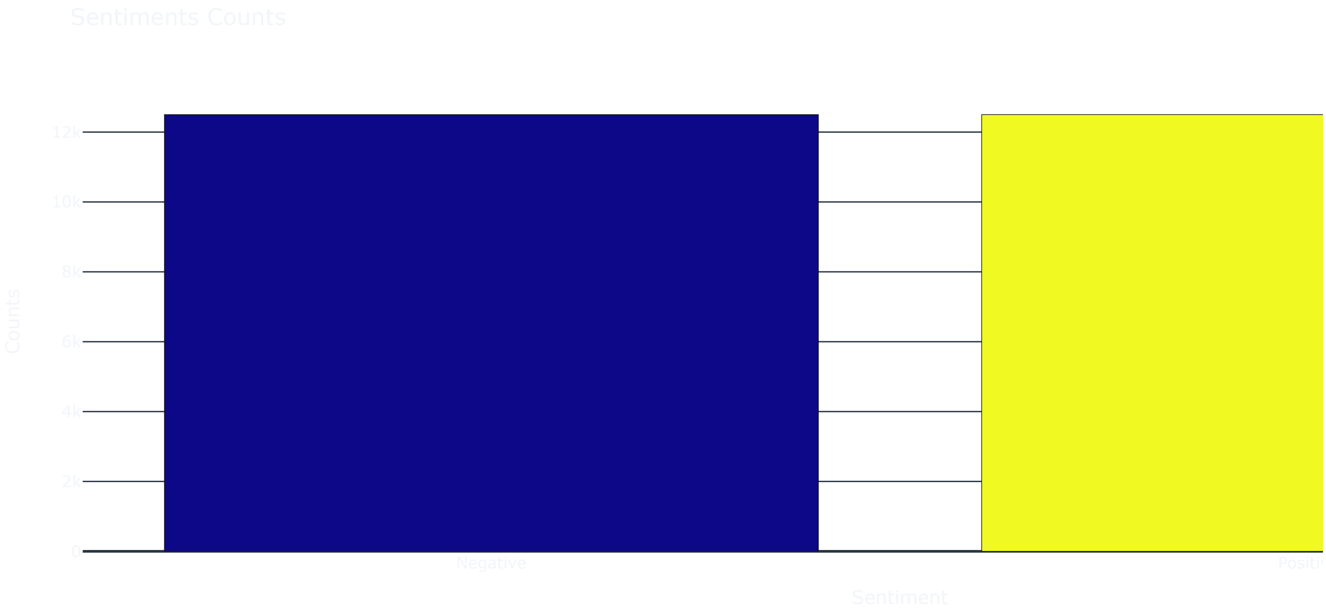
```
            positive_dir = os.path.join(directory, file_name)
            for text_file in os.listdir(positive_dir):
                text = os.path.join(positive_dir, text_file)
                with open(text, "r", encoding="utf-8") as f:
                    data["sentence"].append(f.read())
                    data["sentiment"].append(1)
        elif file_name == 'neg':
            negative_dir = os.path.join(directory, file_name)
            for text_file in os.listdir(negative_dir):
                text = os.path.join(negative_dir, text_file)
                with open(text, "r", encoding="utf-8") as f:
                    data["sentence"].append(f.read())
                    data["sentiment"].append(0)

    return pd.DataFrame.from_dict(data)


# Load the dataset from the train_dir
train_df = load_dataset(train_dir)
print(train_df.head())
```

```
urls_pos.txt
urls_neg.txt
labeledBow.feat
neg
unsup
unsupBow.feat
urls_unsup.txt
pos
                                            sentence  sentiment
0  When I rented this movie, I had very low expec...          0
1  'Major Payne' is a film about a major who make...          0
2  I'd been following this films progress for qui...          0
3  Although the beginning suggests All Quiet on t...          0
4  Cabin Fever is the first feature film directed...          0
```

```
test_dir = os.path.join(dataset_dir,'test')

# Load the dataset from the train_dir
test_df = load_dataset(test_dir)
print(test_df.head())
```

```
urls_pos.txt
urls_neg.txt
labeledBow.feat
neg
pos
                                            sentence  sentiment
0  The movie is nothing extraordinary. As a matte...          0
1  Rented the video with a lot of expectations, b...          0
2  The first time I saw a commercial for this sho...          0
3  We can conclude that there are 10 types of peo...          0
4  I seem to remember a lot of hype about this mo...          0
```

```
sentiment_counts = train_df['sentiment'].value_counts()

fig =px.bar(x= {0:'Negative',1:'Positive'},
            y= sentiment_counts.values,
            color=sentiment_counts.index,
            color_discrete_sequence = px.colors.qualitative.Dark24,
            title='<b>Sentiments Counts')

fig.update_layout(title='Sentiments Counts',
                xaxis_title='Sentiment',
                yaxis_title='Counts',
                template='plotly_dark')

# Show the bar chart
fig.show()
pyo.plot(fig, filename = 'Sentiments Counts.html', auto_open = True)
```

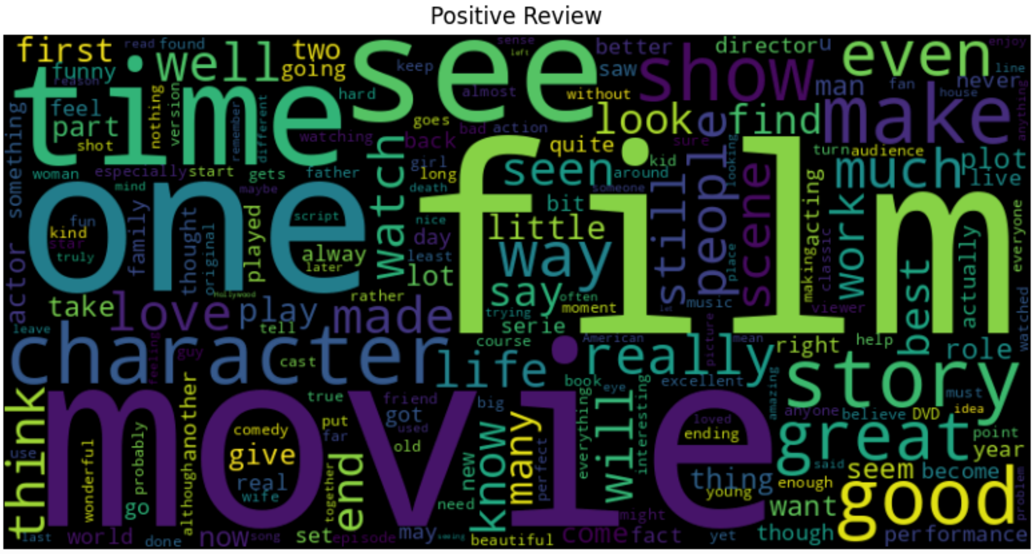


```
'Sentiments Counts.html'
```

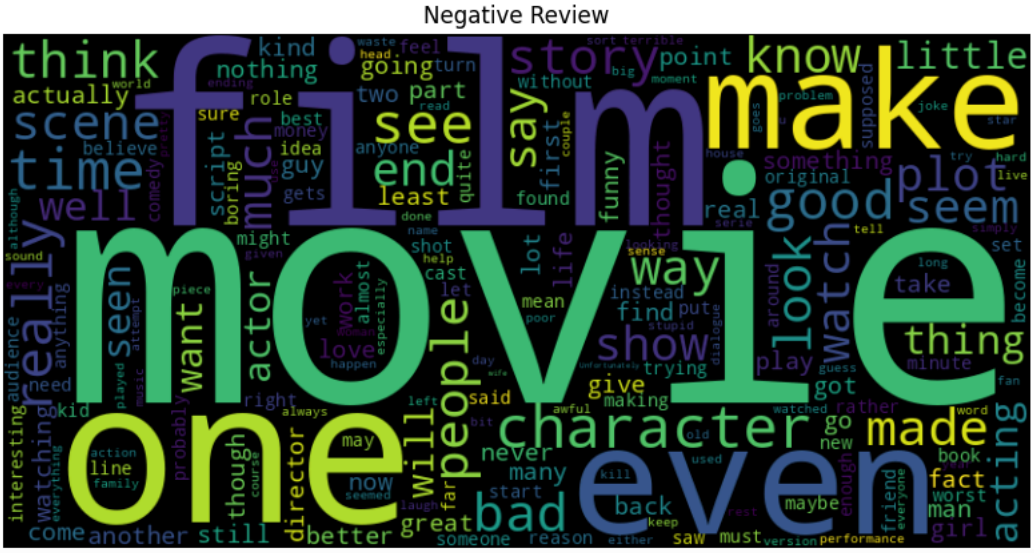
```
def text_cleaning(text):
    soup = BeautifulSoup(text, "html.parser")
    text = re.sub(r'\[[^]]*\]', '', soup.get_text())
    pattern = r"[^a-zA-Z0-9\s,']"
    text = re.sub(pattern, '', text)
    return text


# Train dataset
train_df['Cleaned_sentence'] = train_df['sentence'].apply(text_cleaning).tolist()
# Test dataset
test_df['Cleaned_sentence'] = test_df['sentence'].apply(text_cleaning)
```

```
<ipython-input-11-2374454d2258>:2: MarkupResemblesLocatorWarning:

The input looks more like a filename than markup. You may want to open this file and pass the filehandle into Beautiful Soup.

<ipython-input-11-2374454d2258>:2: MarkupResemblesLocatorWarning:

The input looks more like a filename than markup. You may want to open this file and pass the filehandle into Beautiful Soup.
```

```
# Function to generate word cloud
def generate_wordcloud(text,Title):
    all_text = " ".join(text)
    wordcloud = WordCloud(width=800,
                          height=400,
                          stopwords=set(STOPWORDS),
                          background_color='black').generate(all_text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.title(Title)
    plt.show()


positive = train_df[train_df['sentiment']==1]['Cleaned_sentence'].tolist()
generate_wordcloud(positive,'Positive Review')
```



```
negative = train_df[train_df['sentiment']==0]['Cleaned_sentence'].tolist()
generate_wordcloud(negative,'Negative Review')
```



```
# Training data
#Reviews = "[CLS] " +train_df['Cleaned_sentence'] + "[SEP]"
Reviews = train_df['Cleaned_sentence']
Target = train_df['sentiment']

# Test data
#test_reviews = "[CLS] " +test_df['Cleaned_sentence'] + "[SEP]"
test_reviews = test_df['Cleaned_sentence']
test_targets = test_df['sentiment']


x_val, x_test, y_val, y_test = train_test_split(test_reviews,
                                                test_targets,
                                                test_size=0.5,
                                                stratify = test_targets)


#Tokenize and encode the data using the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:


The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.

tokenizer_config.json: 100%                                              48.0/48.0 [00:00<00:00, 2.51kB/s]

vocab.txt: 100%                                                232k/232k [00:00<00:00, 1.41MB/s]

tokenizer.json: 100%                                                466k/466k [00:00<00:00, 5.70MB/s]

config.json: 100%                                                570/570 [00:00<00:00, 47.1kB/s]
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning:

`clean_up_tokenization_spaces` was not set. It will be set to `True` by default. This behavior will be depracted in transformers v4
```

```python
max_len= 128
# Tokenize and encode the sentences
X_train_encoded = tokenizer.batch_encode_plus(Reviews.tolist(),
                      padding=True,
                      truncation=True,
                      max_length = max_len,
                      return_tensors='tf')

X_val_encoded = tokenizer.batch_encode_plus(x_val.tolist(),
                      padding=True,
                      truncation=True,
                      max_length = max_len,
                      return_tensors='tf')

X_test_encoded = tokenizer.batch_encode_plus(x_test.tolist(),
                      padding=True,
                      truncation=True,
                      max_length = max_len,
                      return_tensors='tf')
```

```python
k = 0
print('Training Comments -->>',Reviews[k])
print('\nInput Ids -->>\n',X_train_encoded['input_ids'][k])
print('\nDecoded Ids -->>\n',tokenizer.decode(X_train_encoded['input_ids'][k]))
print('\nAttention Mask -->>\n',X_train_encoded['attention_mask'][k])
print('\nLabels -->>',Target[k])
```

```
Training Comments -->> When I rented this movie, I had very low expectationsbut when I saw it, I realized that the movie was less a

Input Ids -->>
 tf.Tensor(
[  101  2043  1045 12524  2023  3185  1010  1045  2018  2200  2659 10908
  8569  2102  2043  1045  2387  2009  1010  1045  3651  2008  1996  3185
  2001  2625  1037  2843  2625  2084  2054  1045  3517  1996  5889  2020
  2919  1996  3460  1005  1055  2564  2001  2028  1997  1996  5409  1010
  1996  2466  2001  2061  5236  4183  2071  2147  2005  1037  6373  3185
  3272  2005  1996  9916  1010  2021  2023  2028  2003  2025  1037  4038
  1010  2009  2003  1037  4756  3085 17743  1997 28072  1996  2516  2003
  2092  4217  3272  2005  2028  2518  2027  2071  5587  5236  3185  2044
  2757 19089  1045  2507  2009  1014  1998  1037  2431  2041  1997  1019
   102     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0], shape=(128,), dtype=int32)

Decoded Ids -->>
 [CLS] when i rented this movie, i had very low expectationsbut when i saw it, i realized that the movie was less a lot less than wh

Attention Mask -->>
 tf.Tensor(
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0], shape=(128,), dtype=int32)

Labels -->> 0
```

```python
# Intialize the model
model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
```

```
model.safetensors: 100%                                    440M/440M [00:07<00:00, 114MB/s]
All PyTorch model weights were used when initializing TFBertForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not initialized from the PyTorch model and are newl
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
# Compile the model with an appropriate optimizer, loss function, and metrics
optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
```

```
# Step 5: Train the model
history = model.fit(
    [X_train_encoded['input_ids'], X_train_encoded['token_type_ids'], X_train_encoded['attention_mask']],
    Target,
    validation_data=(
    [X_val_encoded['input_ids'], X_val_encoded['token_type_ids'], X_val_encoded['attention_mask']],y_val),
    batch_size=32,
    epochs=3
)
```

```
Epoch 1/3
782/782 [==============================] - 808s 980ms/step - loss: 0.3348 - accuracy: 0.8480 - val_loss: 0.2891 - val_accuracy: 0.87
Epoch 2/3
782/782 [==============================] - 765s 979ms/step - loss: 0.1963 - accuracy: 0.9238 - val_loss: 0.2984 - val_accuracy: 0.89
Epoch 3/3
782/782 [==============================] - 764s 978ms/step - loss: 0.1007 - accuracy: 0.9632 - val_loss: 0.3652 - val_accuracy: 0.88
```

```
#Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(
  [X_test_encoded['input_ids'], X_test_encoded['token_type_ids'], X_test_encoded['attention_mask']],
  y_test
)
print(f'Test loss: {test_loss}, Test accuracy: {test_accuracy}')
```

```
391/391 [==============================] - 106s 271ms/step - loss: 0.3560 - accuracy: 0.8798
Test loss: 0.3560144007205963, Test accuracy: 0.8797600269317627
```

```
path = '/content'
# Save tokenizer
tokenizer.save_pretrained(path +'/Tokenizer')

# Save model
model.save_pretrained(path +'/Model')
```

```
# Load tokenizer
bert_tokenizer = BertTokenizer.from_pretrained(path +'/Tokenizer')

# Load model
bert_model = TFBertForSequenceClassification.from_pretrained(path +'/Model')
```

```
Some layers from the model checkpoint at /content/Model were not used when initializing TFBertForSequenceClassification: ['dropout_
- This IS expected if you are initializing TFBertForSequenceClassification from the checkpoint of a model trained on another task o
- This IS NOT expected if you are initializing TFBertForSequenceClassification from the checkpoint of a model that you expect to be
All the layers of TFBertForSequenceClassification were initialized from the model checkpoint at /content/Model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertForSequenceClassification
```

```
pred = bert_model.predict(
    [X_test_encoded['input_ids'], X_test_encoded['token_type_ids'], X_test_encoded['attention_mask']])

# pred is of type TFSequenceClassifierOutput
logits = pred.logits

# Use argmax along the appropriate axis to get the predicted labels
pred_labels = tf.argmax(logits, axis=1)

# Convert the predicted labels to a NumPy array
pred_labels = pred_labels.numpy()

label = {
    1: 'positive',
    0: 'Negative'
}

# Map the predicted labels to their corresponding strings using the label dictionary
```

```
pred_labels = [label[i] for i in pred_labels]
Actual = [label[i] for i in y_test]

print('Predicted Label :', pred_labels[:10])
print('Actual Label :', Actual[:10])
```

```
391/391 [==============================] - 108s 270ms/step
Predicted Label : ['positive', 'positive', 'Negative', 'Negative', 'Negative', 'positive', 'Negative', 'positive', 'Negative', 'Nega
Actual Label : ['positive', 'Negative', 'Negative', 'Negative', 'Negative', 'positive', 'Negative', 'positive', 'Negative', 'Negativ
```

```
print("Classification Report: \n", classification_report(Actual, pred_labels))
```

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

    Negative       0.87      0.90      0.88      6250
    positive       0.90      0.86      0.88      6250

    accuracy                           0.88     12500
   macro avg       0.88      0.88      0.88     12500
weighted avg       0.88      0.88      0.88     12500
```

```
def Get_sentiment(Review, Tokenizer=bert_tokenizer, Model=bert_model):
    # Convert Review to a list if it's not already a list
    if not isinstance(Review, list):
        Review = [Review]

    Input_ids, Token_type_ids, Attention_mask = Tokenizer.batch_encode_plus(Review,
```