```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files IMDB Dataset.csv
• IMDB Dataset.csv(text/csv) - 66212309 bytes, last modified: 10/19/2019 - 100% done
Saving IMDB Dataset.csv to IMDB Dataset.csv
```

```
import pandas as pd
df = pd.read_csv("IMDB Dataset.csv")
```

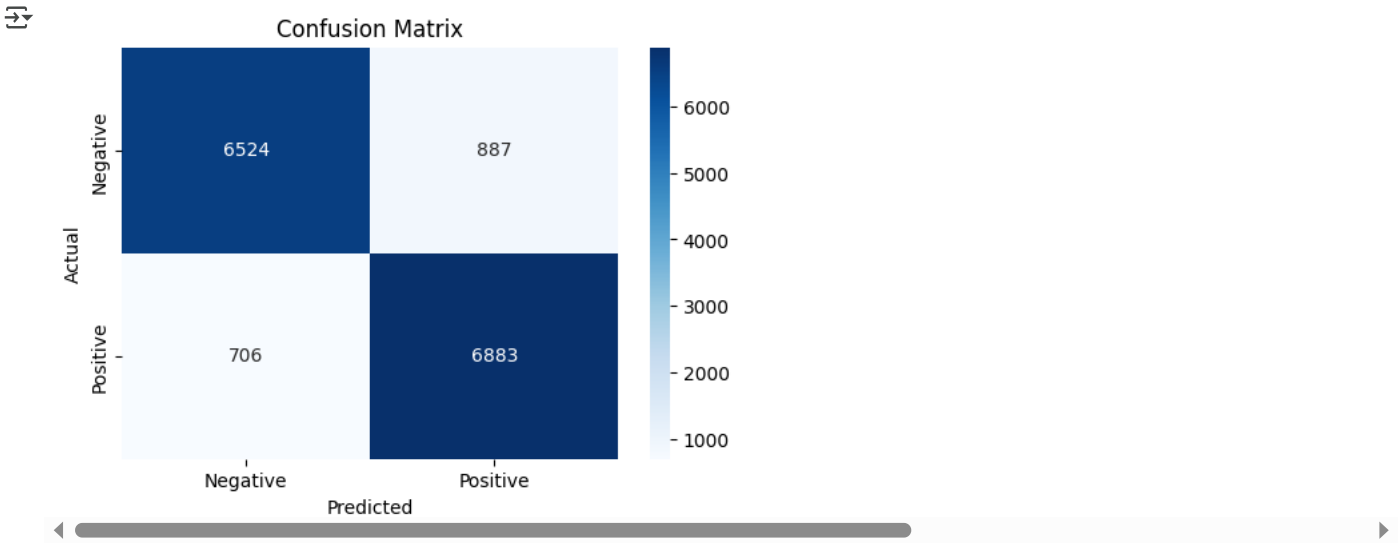
```
import os
print(os.getcwd())
```

```
/content
```

```
# Task 2: Sentiment Analysis with TF-IDF and Logistic Regression

import pandas as pd
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 accuracy_score, classification_report

# Step 1: Load the dataset
df = pd.read_csv("IMDB Dataset.csv")

# Step 2: Convert 'sentiment' to binary (0 = negative, 1 = positive)
df['sentiment'] = df['sentiment'].map({'positive': 1, 'negative': 0})

# Step 3: Split the dataset
X_train, X_test, y_train, y_test = train_test_split(df['review'], df['sentiment'], test_size=0.3, random_state=42)

# Step 4: TF-IDF Vectorization
vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Step 5: Train the Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train_tfidf, y_train)

# Step 6: Predict and Evaluate
y_pred = model.predict(X_test_tfidf)

print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Classification Report:\n", classification_report(y_test, y_pred))
```

```
✅ Accuracy: 0.8938
📊 Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      7411
           1       0.89      0.91      0.90      7589

    accuracy                           0.89     15000
   macro avg       0.89      0.89      0.89     15000
weighted avg       0.89      0.89      0.89     15000
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
Negative
Positive
Actual
6524
887
706
6883
Negative
Positive
Predicted
6000
5000
4000
3000
2000
1000