7	20201700871	مهاب محمد سيد عبدالغني
	2021170242	سندس وائل منصور عبد المنعم
	2021170391	فرح حسين محمد سالم
	2021170385	فاطمه الزهراء محمود حامد
	2021170145	جهاد السيد محمود عبدالمجيد
	2021170506	مريم سيد أحمد

📘 Sentiment Analysis Pipeline Documentation

🧹 [1] Preprocessing Steps

1- Objective :

This script is designed to preprocess raw text data from a dataset (specifically the review column in a DataFrame df) in preparation for Natural Language Processing tasks such as sentiment analysis or text classification.

2- Downloading NLTK Resources :

- Tokenizers (punkt)
- POS taggers
- Stopword lists
- WordNet for lemmatization

3- Slang and Abbreviation Dictionary :

A manually defined dictionary (slang_dict) that maps informal abbreviations and internet slang to their standard forms.

4- Detecting Unknown Words :

-Extracts all words from the input text.

-Uses SpellChecker to identify words not in the dictionary (possibly misspelled or slang).

-Returns a list of unknown words for each review.

5- Text Preprocessing Function:

-Remove Special Characters and Single Letters.

-Normalize Spaces.

-Expand Slang/Abbreviations.

-Tokenization.

-Remove Stopwords & Short Tokens.

-POS Tagging and Lemmatization.

-Return Cleaned Text.

📊 [2] Data Exploration

✅ Dataset Overview:

- **File Used**: prepared_reviews.csv
- **Number of Samples**:
- **Missing Values**:
  - Reviews: missing_values['review']
  - Labels: missing_values['label']
- **Duplicates**:

  df.duplicated().sum()
- **Review Length Analysis**:
  - Added review_length column
  - Range: min(df['review_length']) to max(df['review_length']) characters

✅ Sentiment Distribution:

df['label'].value_counts()

🛠️ [3] Preprocessing Functions

🔤 preprocess_text()

Lemmatization-based cleaning with POS tagging and slang replacement.

🔤 **preprocess_text_modified()**

Stemmer-based cleaning (Porter Stemmer), used for vectorization and model input.

Sample Columns Created:

- review_preprocessing_1 – Lemmatization version
- review_preprocessing_2 – Stemming version (used for training)

🔢 **[4] Label Encoding**

```
df['numerical_label'] = df['label'].map({'positive': 1, 'negative': 0})
```

✂️ **[5] Train-Test Split**

```
X_train, X_test, y_train, y_test = train_test_split(
    df['review_preprocessing_2'],
    df['numerical_label'],
    test_size=0.2,
    random_state=42,
    stratify=df['numerical_label']
)
```

🧠 **[6] Vectorization**

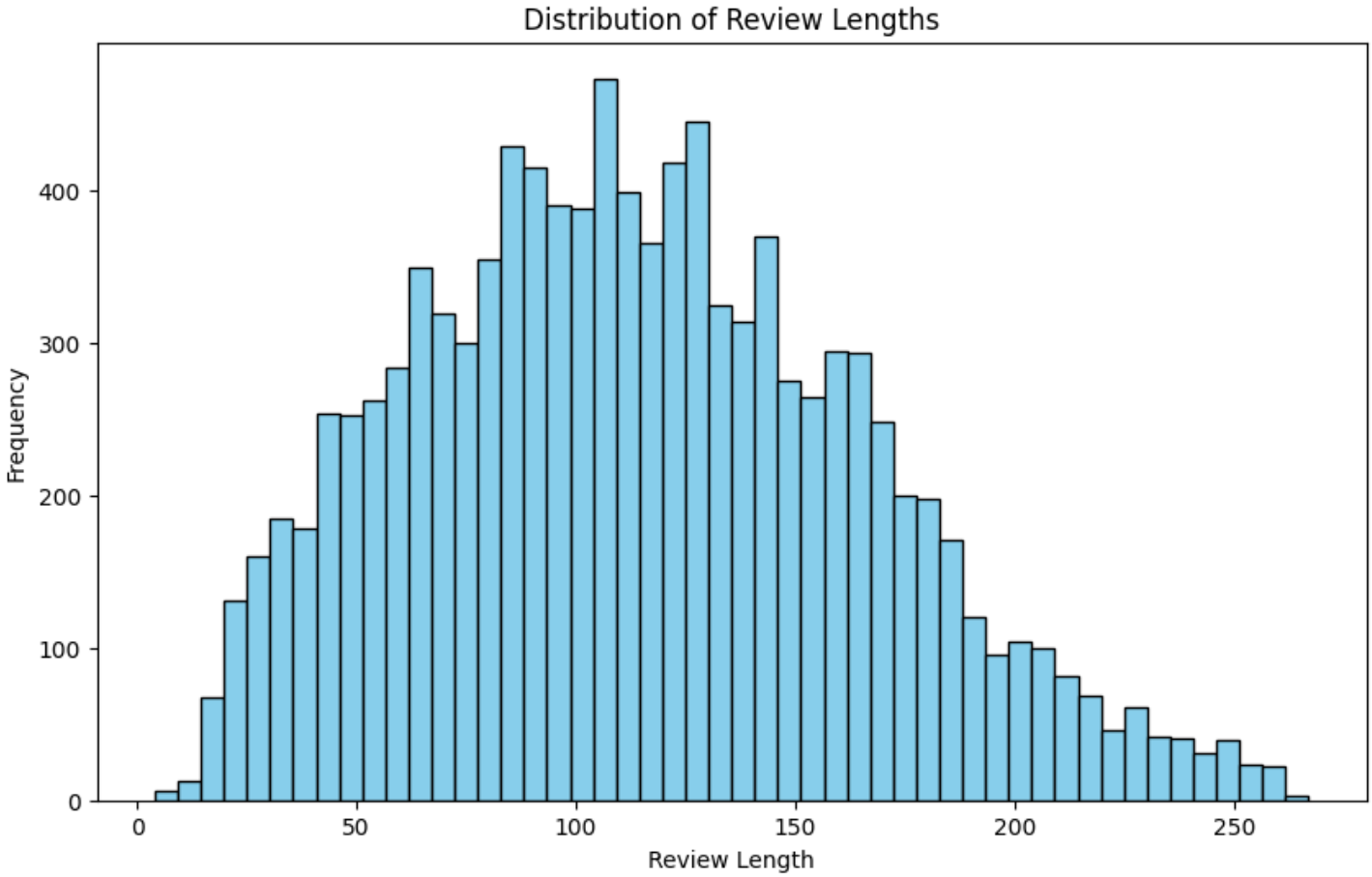
🟩 **Count Vectorizer**

```
count_vectorizer = CountVectorizer()
X_train_counts = count_vectorizer.fit_transform(X_train)
X_test_counts = count_vectorizer.transform(X_test)
joblib.dump(count_vectorizer, "count_vectorizer.pkl")
```

🟦 TF-IDF Vectorizer

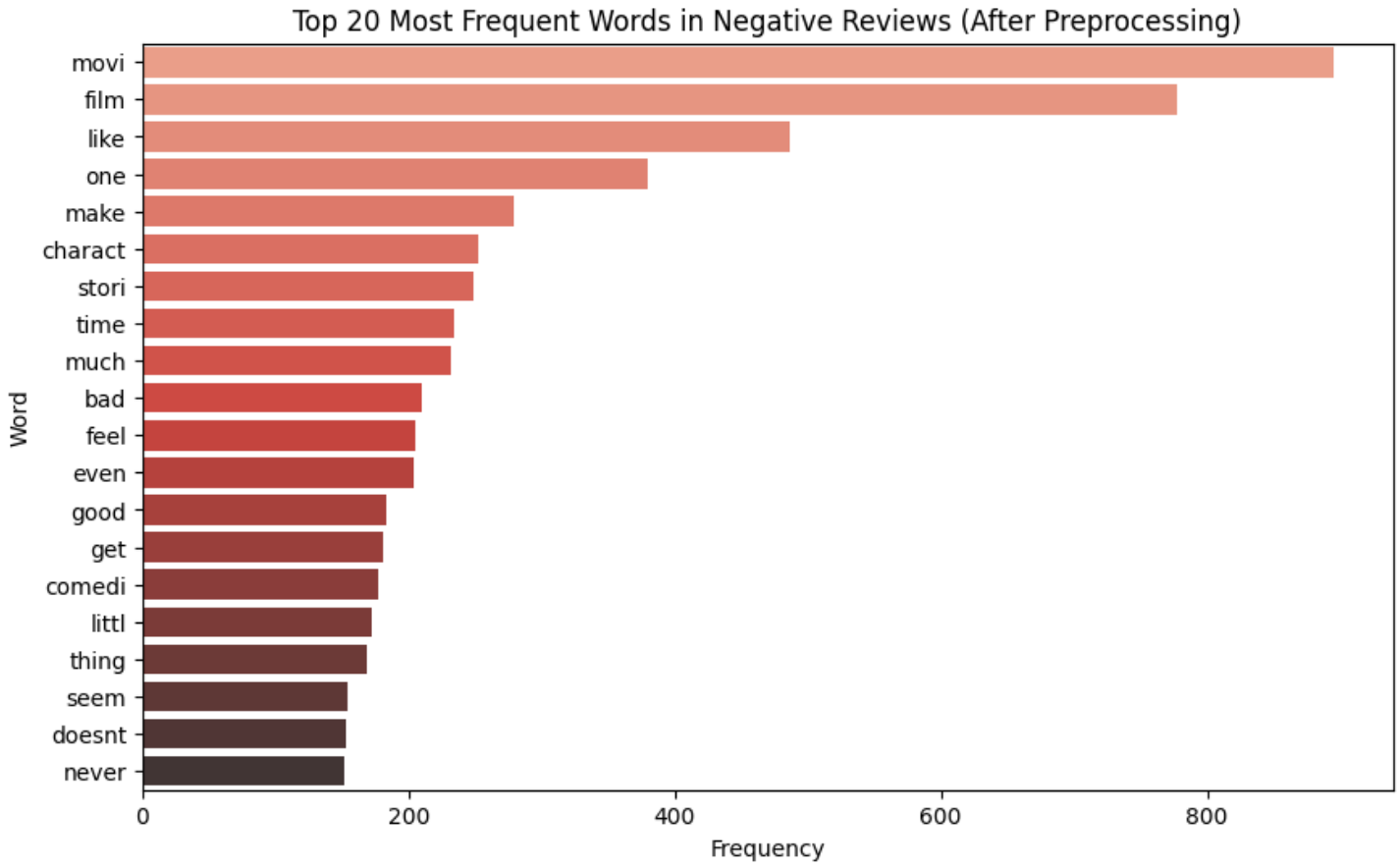
```
vectorizer = TfidfVectorizer(min_df=7, max_df=0.5, sublinear_tf=True, max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

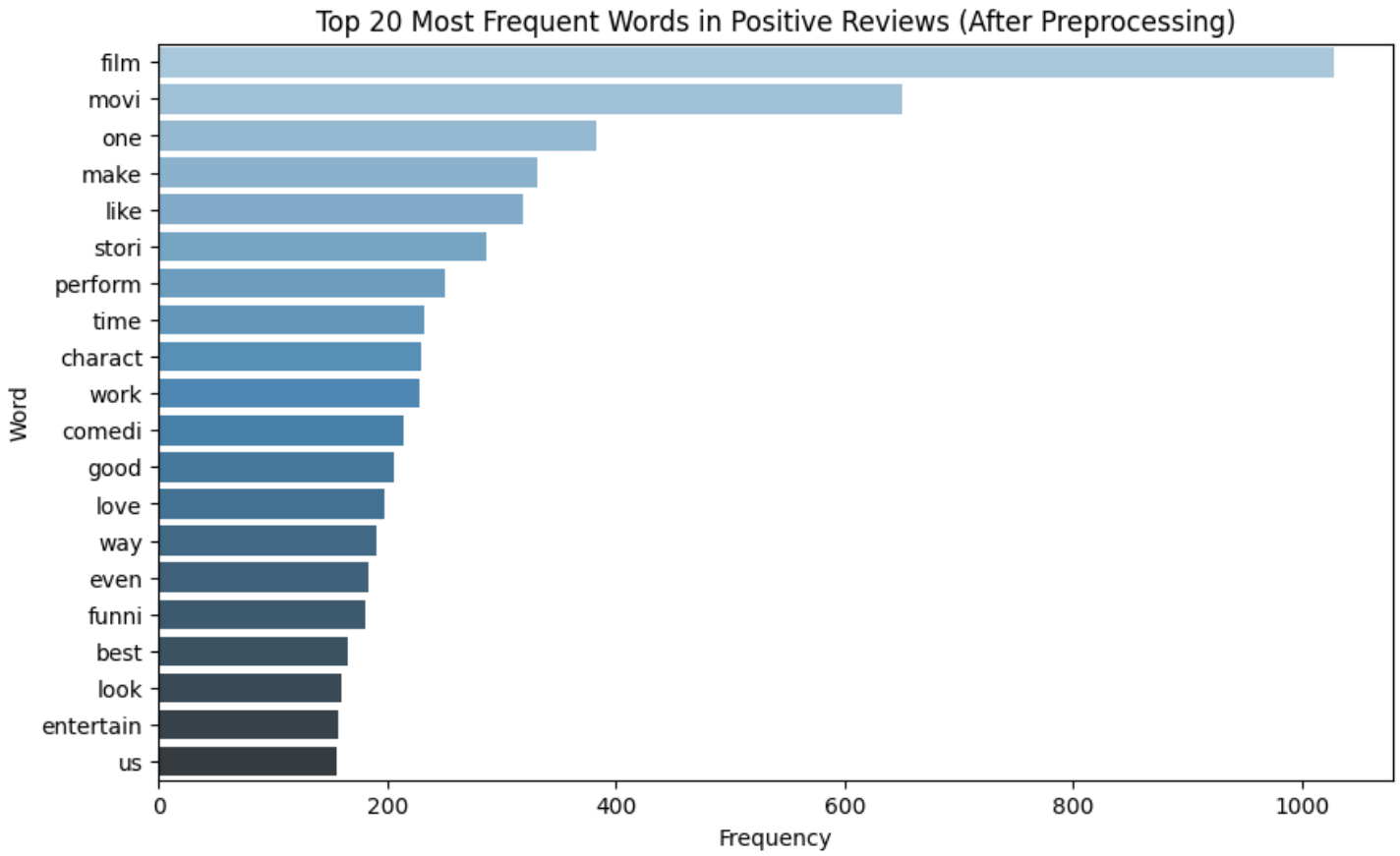
📈 [7] Visualization



📊 Review Length Distribution

Histogram of review lengths.



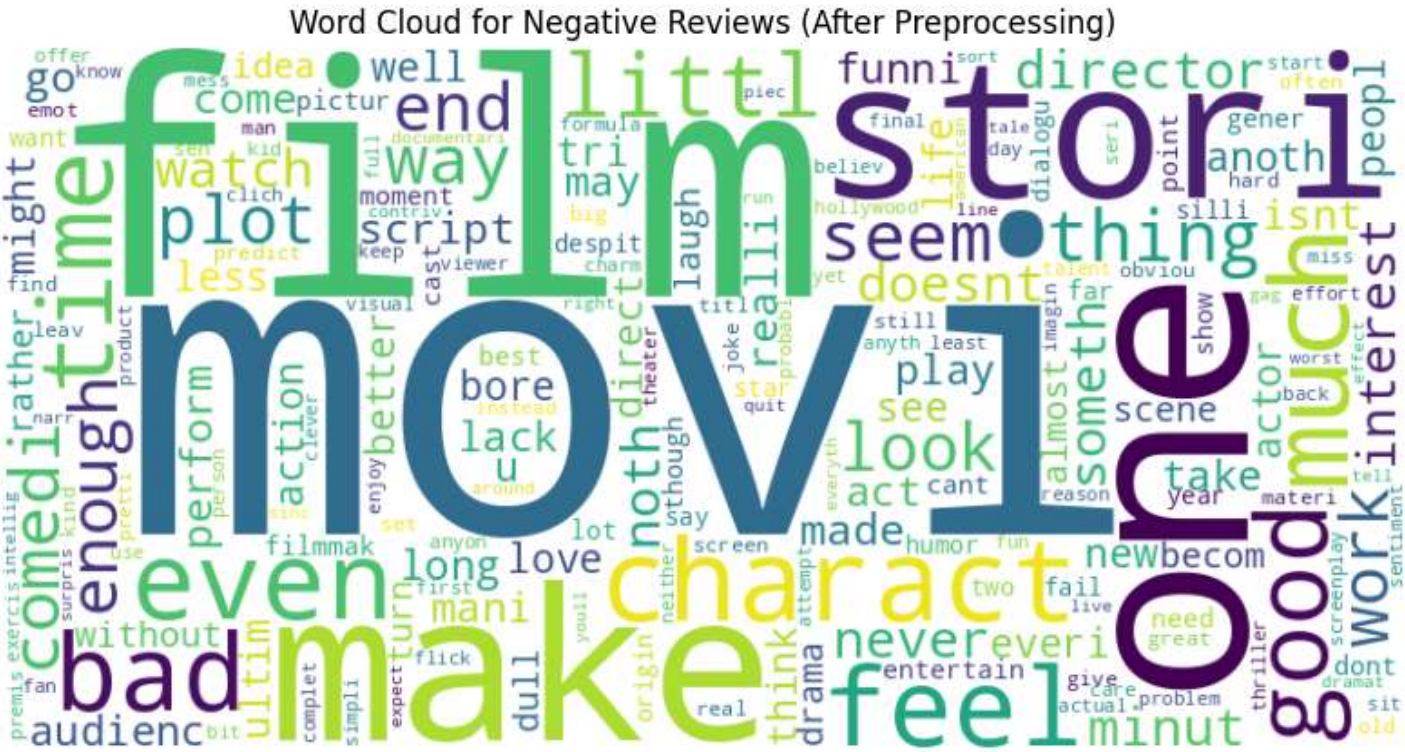


📌 Top Words (Bar Charts)

- Used frequency counts to show top 20 words in:
  - Positive Reviews
  - Negative Reviews

☁️ Word Cloud

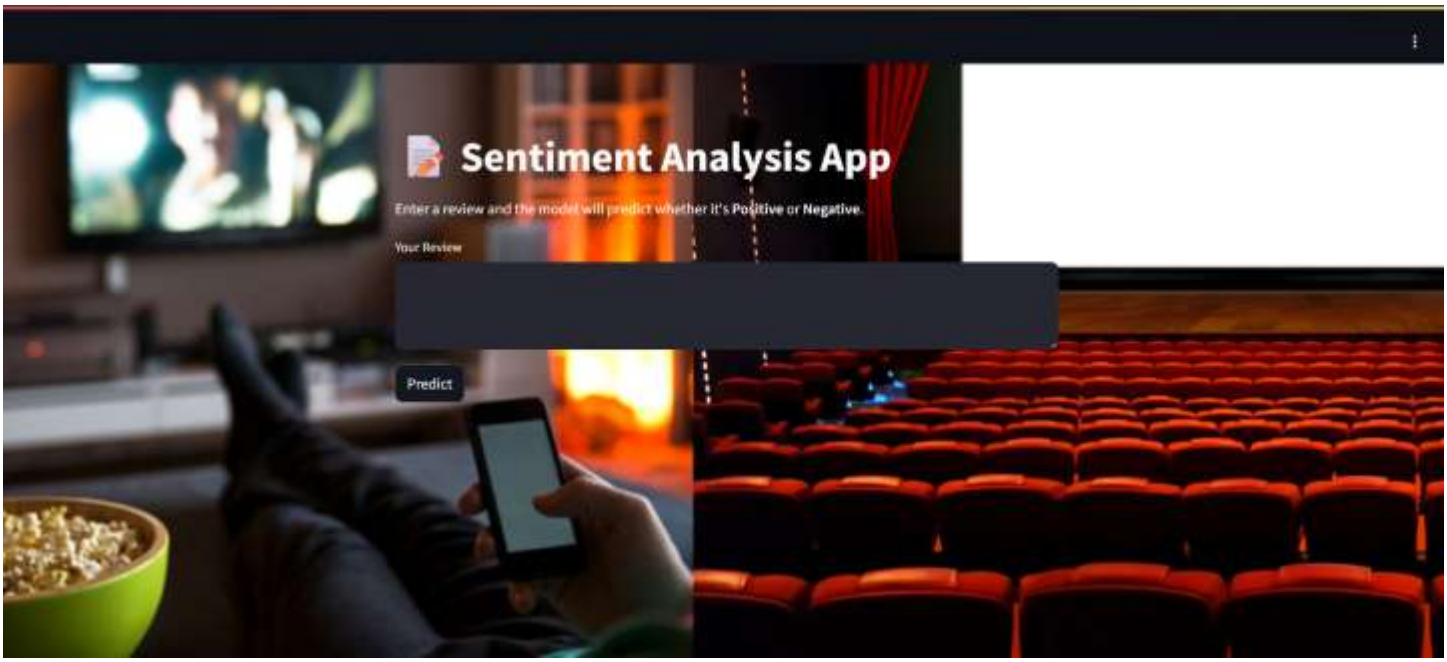
- Generated using WordCloud from:
  - All positive reviews
  - All negative reviews

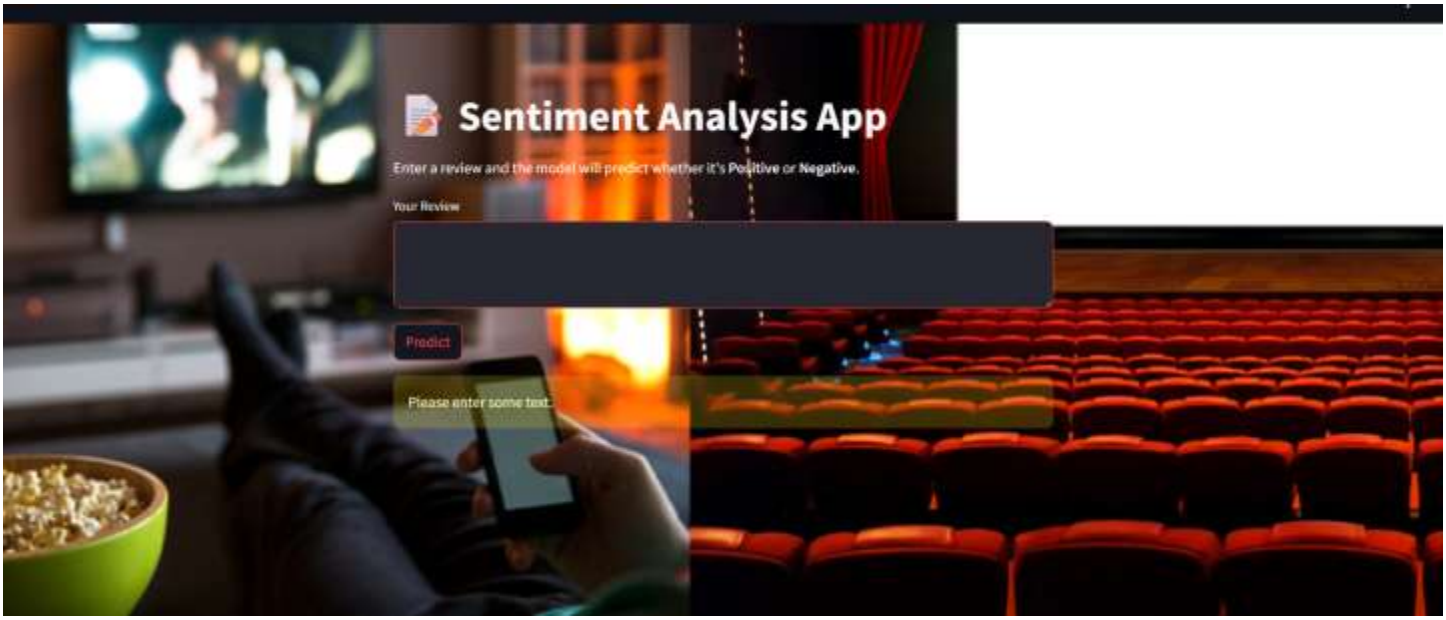


📦 [8] Tools & Libraries Used

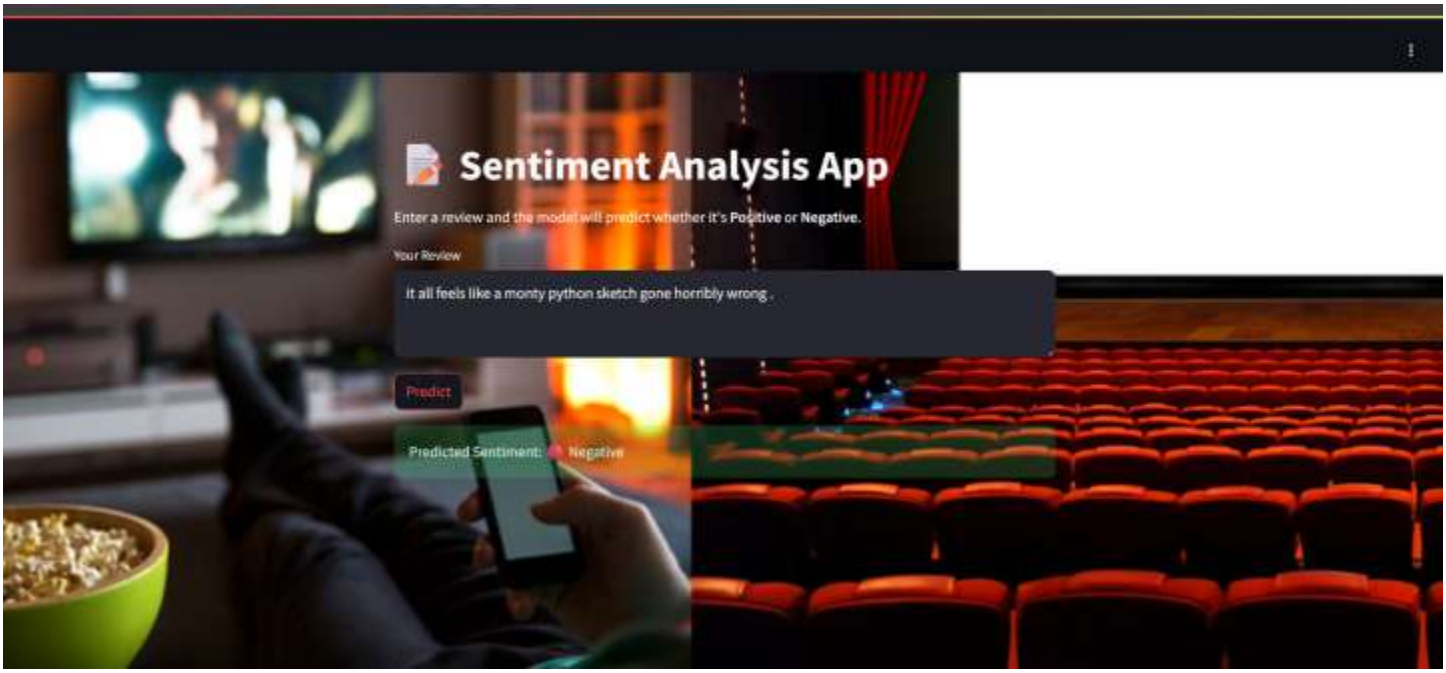
- pandas, matplotlib, seaborn
- nltk: tokenization, lemmatization, stemming, stopwords
- re, string, sklearn
- pyspellchecker for spell correction
- WordCloud for visualization
- joblib for saving vectorizers

Streamlit Deployment:





Negative Review



Positive Review