

7	20201700871	مهاب محمد سيد عبدالغنى
	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.

```

slang_dict = {
    "ibh": "to be honest", "imo": "in my opinion", "fyi": "for your information", "lol": "laugh out loud",
    "rofl": "rolling on the floor laughing", "btw": "by the way", "thx": "thanks", "pls": "please", "u": "you",
    "r": "are", "ur": "your", "b4": "before", "zday": "today", "gr8": "great", "bc": "because",
    "w/": "with", "w/o": "without", "def": "definitely", "omg": "oh my god", "idk": "i don't know", "wth": "what the heck",
    "wtf": "what the fuck", "asap": "as soon as possible", "tma": "tomorrow", "yday": "yesterday", "gonna": "going to",
    "wanna": "want to", "gotta": "get to", "kinda": "kind of", "sorta": "sort of", "dunno": "don't know", "gimme": "give me",
    "tryna": "trying to", "lema": "let me", "imho": "in my humble opinion", "sirc": "if i remember correctly",
    "brb": "be right back", "afak": "as far as i know", "aka": "also known as", "fb": "facebook", "ig": "instagram",
    "dm": "direct message", "fomo": "fear of missing out", "fwiw": "for what it's worth", "ftw": "for the win",
    "jsyk": "just so you know", "nvm": "never mind", "tba": "to be announced", "tbd": "to be determined",
    "tbt": "throwback thursday", "ttyl": "talk to you later", "yolo": "you only live once", "jk": "just kidding",
    "tfw": "that feeling when", "sm": "shaking my head", "tlr": "too long didn't read", "4ever": 'forever',
    "ain't": "is not", "gonna": "going to", "wanna": "want to", "gotta": "got to", "y all": "you all", "helluva": "hell of a",
    "outta": "out of", "kinda": "kind of", "sorta": "sort of", "coulda": "could have", "woulda": "would have",
    "shoulda": "should have", "migha": "might have", "lotta": "lot of", "dunno": "do not know", "ain't": "is not",
    "wassup": "what is up", "whatcha": "what are you", "betcha": "bet you", "cuz": "because", "coz": "because", "tho": "though",
    "till": "until", "imma": "i am going to", "gimme": "give me", "tryna": "trying to", "lm": "love", "bc": "because",
    "bdy": "birthday", "brb": "be right back", "btw": "by the way", "idk": "i do not know", "lir": "i know right",
    "imk": "let me know", "omg": "oh my god", "thx": "thanks", "ty": "thank you", "np": "no problem", "plz": "please",
    "rofl": "rolling on the floor laughing", "shh": "shaking my head", "tba": "to be honest", "tho": "though",
    "ttyl": "talk to you later", "wth": "what the heck", "ya": "you", "sup": "what is up", "ur": "your",
    "bcuz": "because", "ok": "okay", "bro": "brother", "sis": "sister",
    'finstagram': 'fake instagram night',
    'dyounno': 'do you know',
}

```

## 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.

```

def detect_unknown_words(text):
    words = re.findall(r"\b\w+\b", text.lower())
    unknown_words = spell.unknown(words) # Get unknown words
    return list(unknown_words) # Return as a list

```

## 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")
```

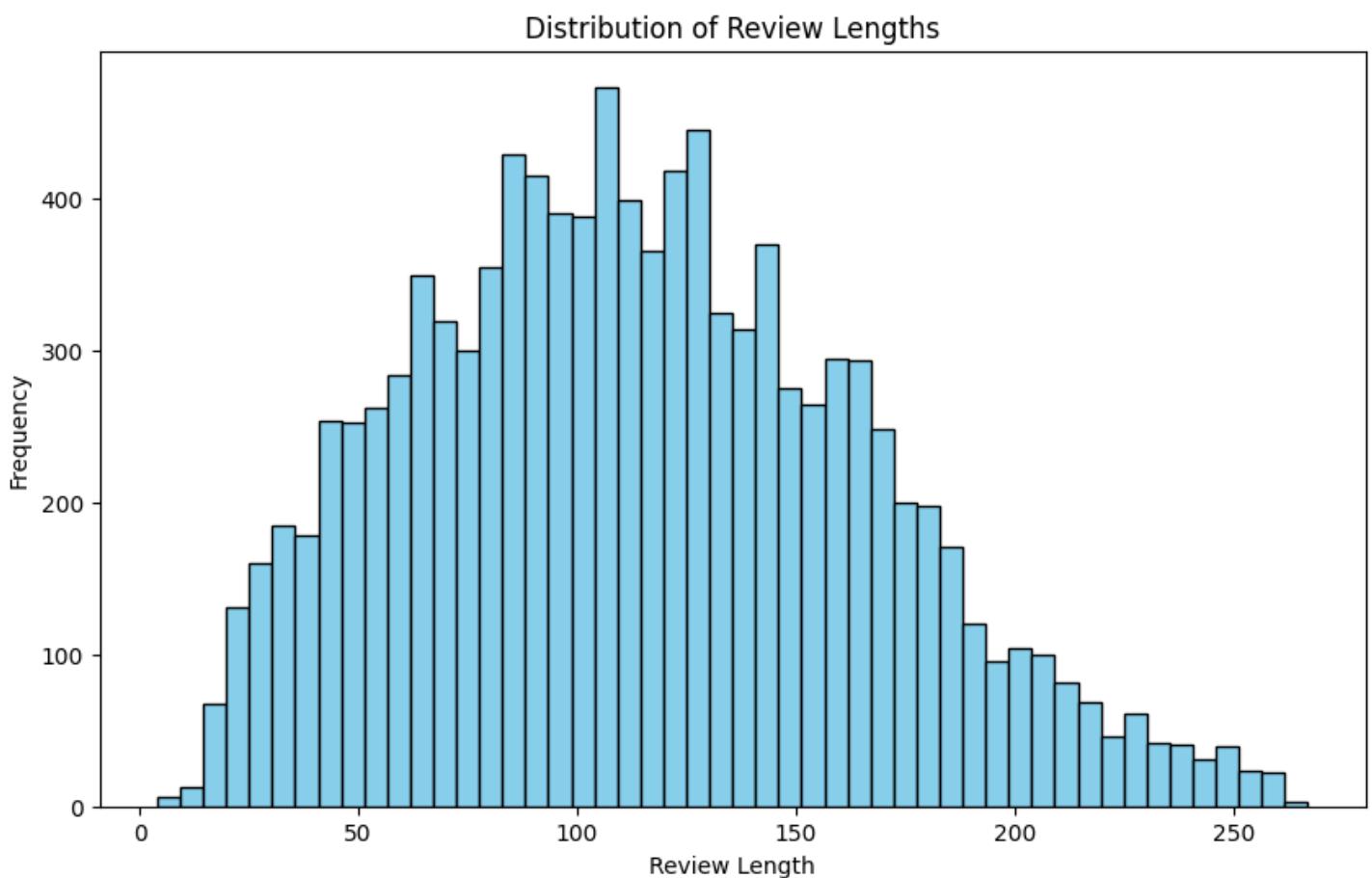
## ■ TF-IDF Vectorizer

```
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)
```

---

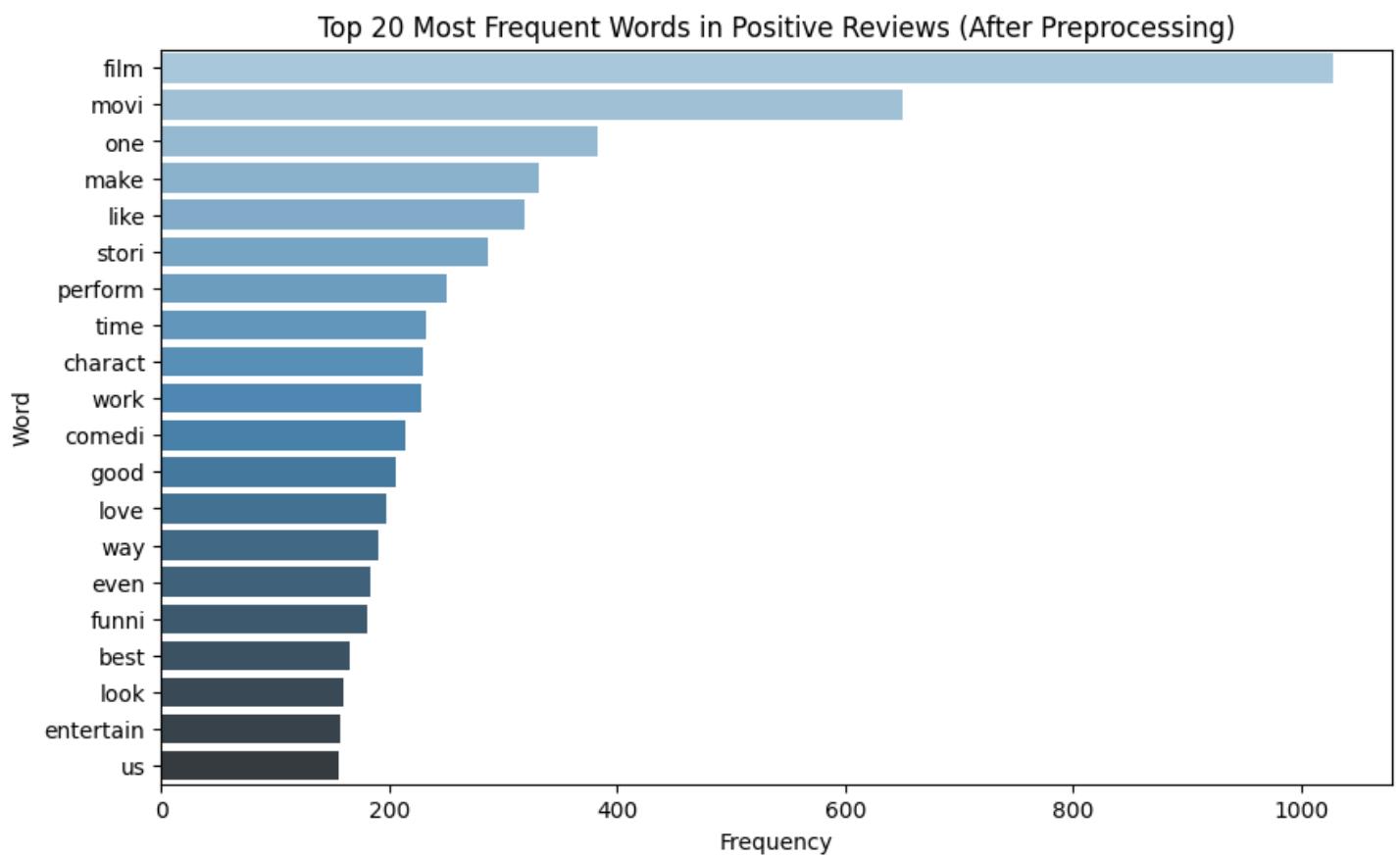
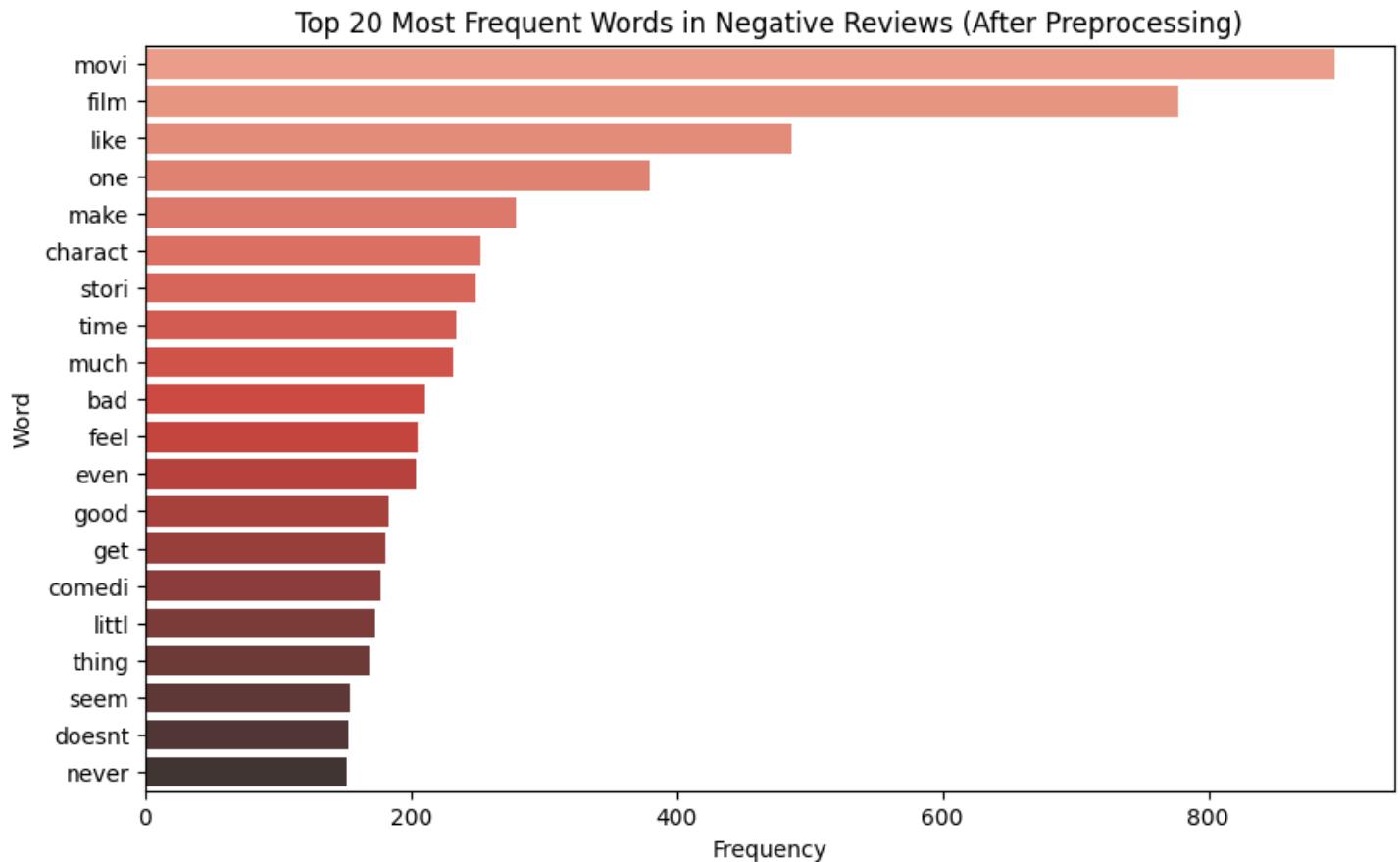


## [7] Visualization



## Review Length Distribution

# Histogram of review lengths.



# 📌 Top Words (Bar Charts)

- Used frequency counts to show top 20 words in:

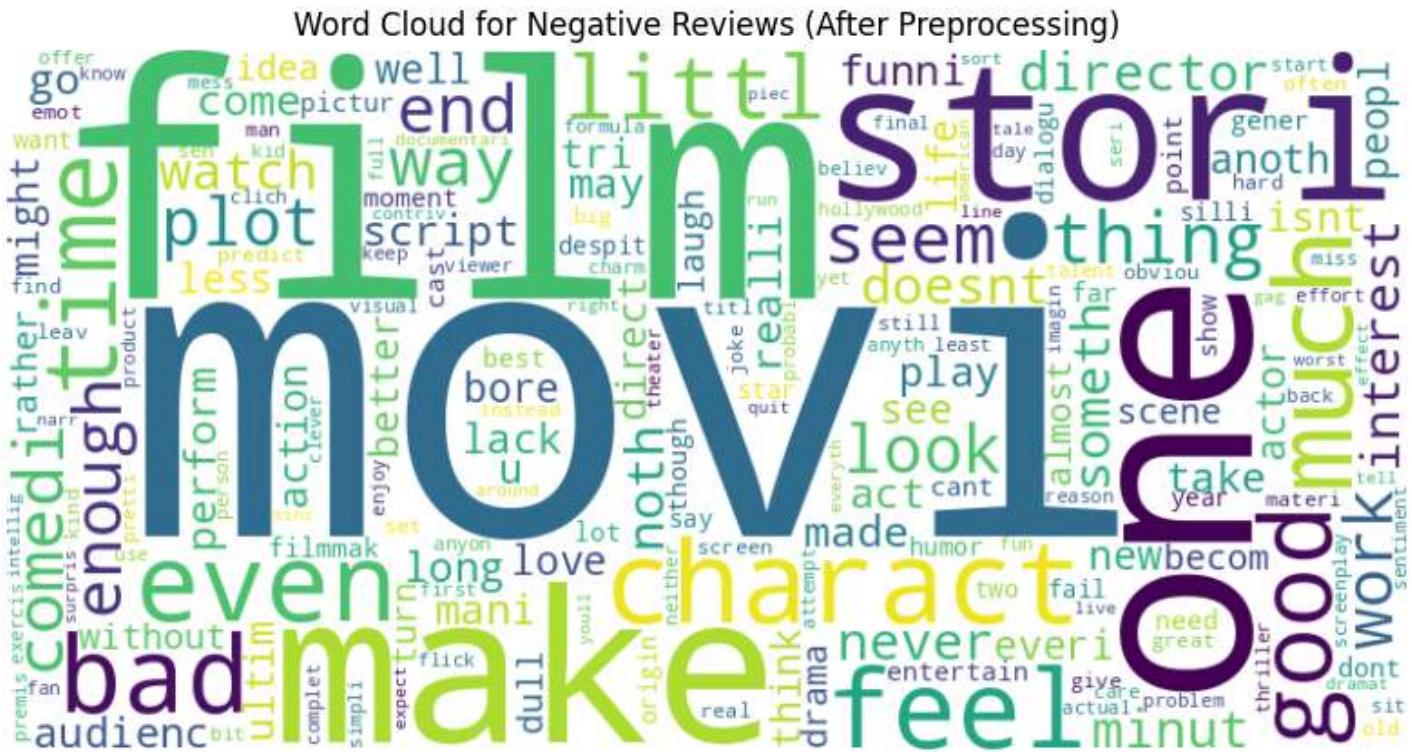
- Positive Reviews
  - Negative Review



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

## Word Cloud for Positive Reviews (After Preprocessing)

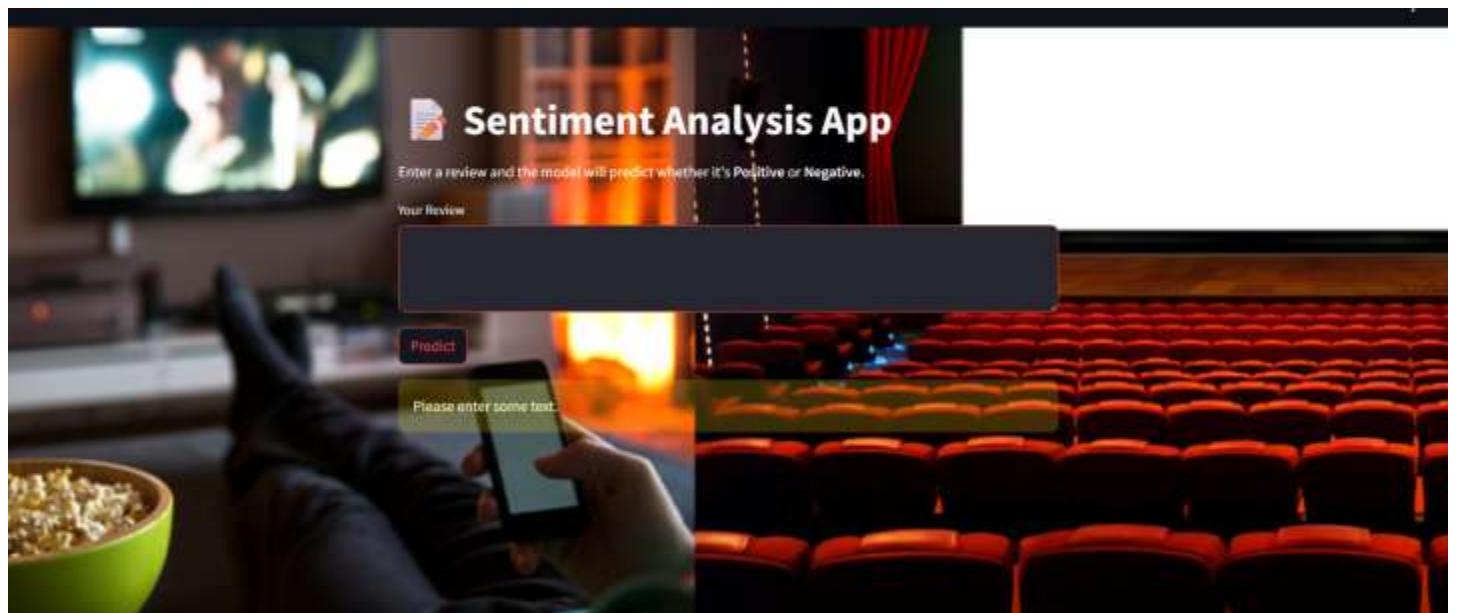
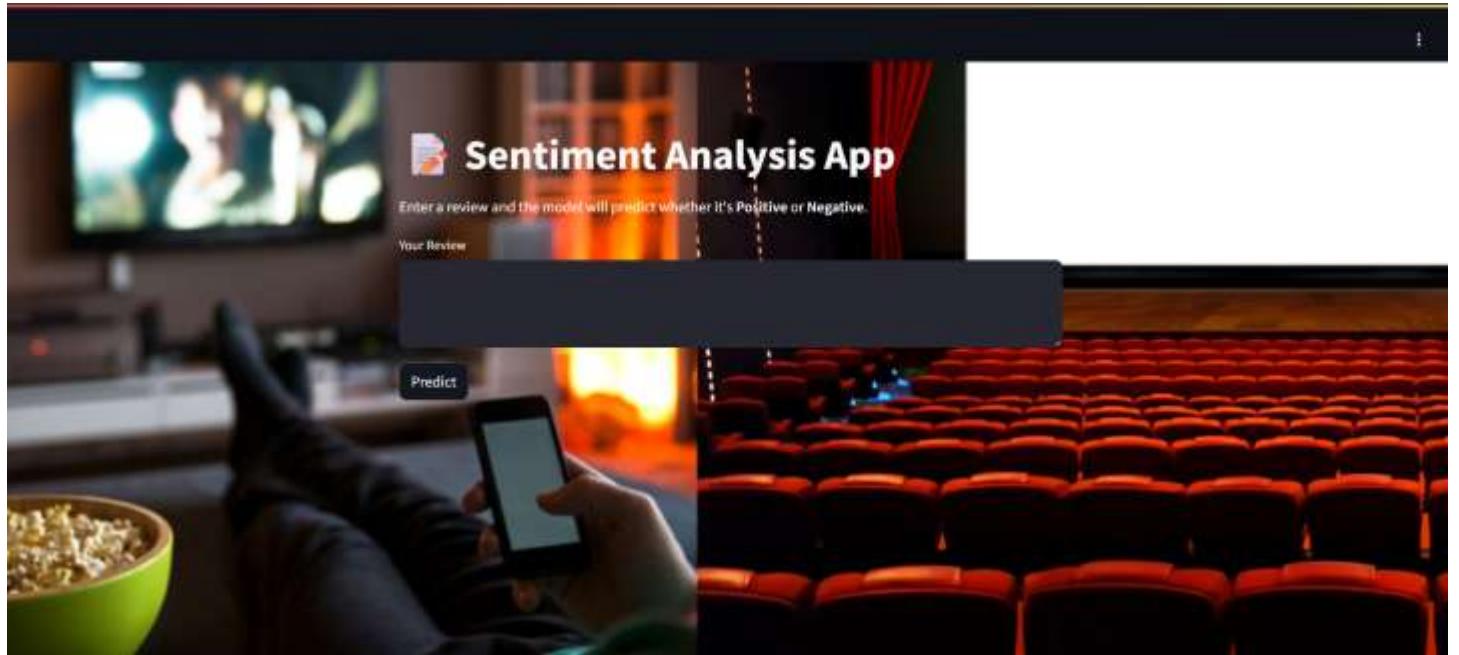




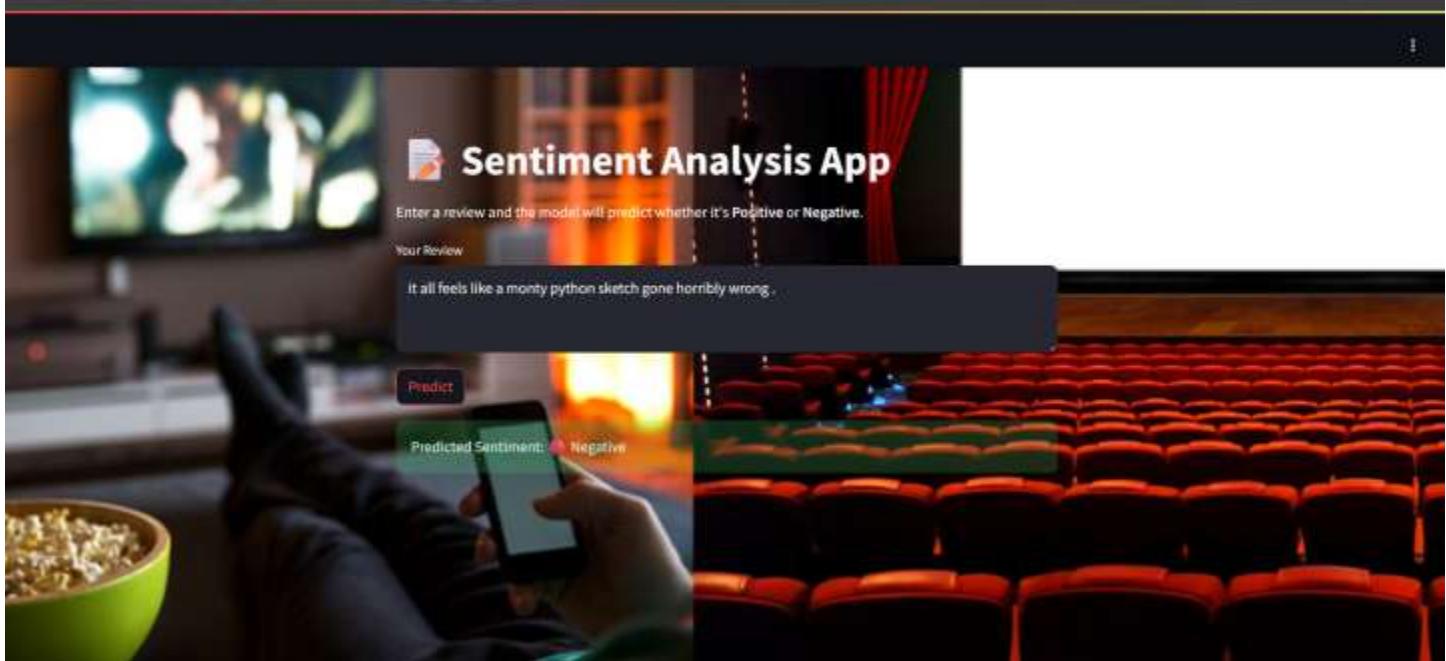
## [8] Tools & Libraries Used

- pandas, matplotlib, seaborn
- nltk: tokenization, lemmatization, stemming, stopwords
- re, string, sklearn
- pypinyin for Chinese text processing
- WordCloud for visualization
- joblib for saving vectorizers

## 📍 Streamlit Deployment:



## Negative Review



## Positive Review

