**MOVIE REVIEW SENTIMENT ANALYZER: PROJECT DOCUMENTATION**

**1. Project Content**

The **Movie Review Sentiment Analyzer** is an NLP-based machine learning system designed to classify movie reviews as either a **"Hit"** or **"Flop"** based on the sentiment expressed in the review. This project leverages core NLP techniques like **text preprocessing**, **vectorization**, and **supervised learning** to turn unstructured text into actionable insights.

**Objectives:**

* To understand public sentiment from text reviews.
* To categorize movie feedback into binary outcomes: Hit or Flop.
* To demonstrate how machine learning models can automate opinion analysis.
* To create an interactive, user-friendly interface using **Gradio** for real-time predictions.

**2. Project Code Structure**

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.datasets import imdb

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing.text import Tokenizer

import numpy as np

# Load IMDb dataset word index (to simulate tokenizer)

word\_index = imdb.get\_word\_index()

index\_word = {v+3: k for k, v in word\_index.items()}

word\_index = {k: (v+3) for k, v in word\_index.items()}

word\_index["<PAD>"] = 0

word\_index["<START>"] = 1

word\_index["<UNK>"] = 2

word\_index["<UNUSED>"] = 3

# Load pretrained model (or build one if needed)

# For demo, we'll build a quick one below. You can skip this block if you have a model already.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense

# Load dataset

(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=10000)

x\_train = pad\_sequences(x\_train, maxlen=256, padding='post')

x\_test = pad\_sequences(x\_test, maxlen=256, padding='post')

# Build and train a simple model (train once and save)

model = Sequential([

Embedding(10000, 16),

GlobalAveragePooling1D(),

Dense(16, activation='relu'),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.fit(x\_train, y\_train, epochs=2, batch\_size=512, validation\_split=0.2)

model.save("movie\_review\_model.h5")

# Load model (use this part once saved)

model = load\_model("movie\_review\_model.h5")

# Function to convert review text to sequence

def review\_to\_sequence(text):

tokens = text.lower().split()

seq = [word\_index.get(word, 2) for word in tokens]

return pad\_sequences([seq], maxlen=256, padding='post')

# Prediction function

def predict\_review(text):

sequence = review\_to\_sequence(text)

prediction = model.predict(sequence)[0][0]

return "Hit 🎬" if prediction > 0.5 else "Flop 💣"

# Main

if \_\_name\_\_ == "\_\_main\_\_":

review = input("Enter a movie review: ")

result = predict\_review(review)

print("Prediction:", result)

**3. Key Technologies Used**

| **Technology** | **Purpose** |
| --- | --- |
| Python | Programming language used for scripting and ML pipelines |
| Scikit-learn | ML library for vectorization, model training, and evaluation |
| Gradio | Simplifies building interactive web UIs for ML models |
| TF-IDF Vectorizer | Converts text into a weighted numerical format for classification |
| Pandas/NumPy | For efficient data manipulation and analysis |
| Pickle | For model and vectorizer serialization |

**4. Project Description**

**4.1 Problem Statement**

With the rise of digital reviews on websites and social platforms, analyzing public sentiment is crucial for film studios, marketers, and analysts. However, manually analyzing thousands of reviews is time-consuming. This project automates the process of classifying movie reviews as "Hit" or "Flop" based on sentiment.

**4.2 Methodology**

**Step-by-Step Workflow:**

1. **Input:**  
   Raw text reviews from users (e.g., “Absolutely thrilling movie!”)
2. **Preprocessing:**
   * Lowercasing
   * Removing punctuation and stop words
   * Tokenization
3. **Feature Extraction (Vectorization):**  
   TF-IDF is used to convert text into numeric vectors that reflect term importance across the dataset.
4. **Classification:**  
   Models like **Logistic Regression** or **Multinomial Naive Bayes** predict whether the review reflects a "Hit" or "Flop".
5. **Output:**  
   A label (Hit/Flop) along with a confidence score.

**5. Output**

**Example:**

* **Input:** “A well-directed masterpiece that keeps you on the edge!”
* **Output:** **Hit**
* **Confidence Score:** 89%

**Evaluation Metrics:**

* **Accuracy**: Measures overall correctness.
* **Precision**: Fraction of predicted hits that were actually hits.
* **Recall**: Fraction of actual hits that were predicted correctly.
* **F1 Score**: Harmonic mean of precision and recall.

**EXAMPLE**:

Enter a movie review: The plot was amazing and the acting was superb!

Prediction: Hit 🎬

Enter a movie review: Boring, dragged too long and made no sense.

Prediction: Flop 💣

**6. Further Research and Enhancements**

This model is a baseline sentiment analyzer. Future enhancements could include:

* **Transformer Models:**  
  Integrate pre-trained transformers like BERT, RoBERTa, or DistilBERT for improved contextual understanding.
* **Multilingual Support:**  
  Add support for movie reviews in different languages using language detection and multilingual embeddings.
* **Emotion Classification:**  
  Classify reviews into a broader set of emotions (joy, anger, disappointment, etc.).
* **Aspect-Based Sentiment Analysis:**  
  Break reviews into aspects (e.g., direction, acting, music) and rate each aspect individually.
* **Visualization Dashboard:**  
  Create an interactive dashboard with graphs showing sentiment trends by genre, actor, director, etc.