Project Introduction

Nowadays, online platforms receive thousands of movie reviews, and manually understanding whether these reviews are positive or negative is difficult and time-consuming. Sentiment analysis helps automatically identify emotions in text and understand public opinion more efficiently.

Problem Statement

The IMDB dataset contains 50,000 movie reviews that are highly unstructured, noisy, and text-based. Manually determining whether each review expresses a positive or negative sentiment is not feasible. Therefore, an automated system is required to classify sentiments accurately using NLP and machine learning techniques.

Objective

The main objective of this project is to build an intelligent sentiment analysis model that can analyze IMDB movie reviews and classify them as positive or negative. This involves applying text preprocessing, feature extraction methods, machine learning models, and transformer-based models like BERT to achieve high accuracy and reliable predictions.

Expected Outcomes

The system will accurately classify IMDB movie reviews into positive or negative sentiments. The project will generate meaningful visualizations, such as word clouds and confusion matrices, to show sentiment patterns and model performance.

The BERT-based model is expected to achieve the highest accuracy due to its advanced language understanding capabilities.

The project will produce an efficient, automated sentiment classifier that can be used for real-world applications like review analysis, customer feedback monitoring, and opinion mining.

Step 1:

Installing basic data-handling and NLP libraries like pandas, numpy, nltk, spacy, gensim, and wordcloud.

Installing machine learning tools like scikit-learn.

Installing deep learning frameworks (TensorFlow & Keras) for LSTM models.

Installing Hugging Face transformers for BERT-based sentiment analysis.

Installing evaluation and training utilities such as tqdm, datasets, and accelerate.

```
pip install pandas numpy scikit-learn matplotlib seaborn nltk spacy gensim wordcloud
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages (3.8.8)
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.4.0)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-packages (1.9.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.13)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.10)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (8.3.8)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.4.2)
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.20.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.32.4)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.11.10)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.5.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (2.33.2)
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.15.0)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.4.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2025.10.5)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1->gensim) (2.0.1)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (1.3.0)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy) (0.1.5)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2->spacy) (0.23.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->spacy) (3.0.3)
```

```
pip install tensorflow keras --upgrade
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.20.0)
Requirement already satisfied: keras in /usr/local/lib/python3.12/dist-packages (3.12.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.9.23)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google_pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt_einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.4.0)
```

```
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.0)
Requirement already satisfied: protobuf>=5.28.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (5.29.5)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.32.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.2.0)
Requirement already satisfied: typing_extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (4.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.76.0)
Requirement already satisfied: tensorboard~=2.20.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.20.0)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.15.1)
Requirement already satisfied: ml_dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.5.3)
Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras) (0.17.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2025.10.5)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.20.0->tensorflow) (3.10)
Requirement already satisfied: pillow in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.20.0->tensorflow) (11.3.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.20.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.20.0->tensorflow) (3.1.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras) (4.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras) (2.19.2)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->tensorboard~=2.20.0->tensorflow) (3.0.3)
```

```
pip install transformers datasets evaluate accelerate
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.57.1)
Requirement already satisfied: datasets in /usr/local/lib/python3.12/dist-packages (4.4.1)
Requirement already satisfied: evaluate in /usr/local/lib/python3.12/dist-packages (0.4.6)
Requirement already satisfied: accelerate in /usr/local/lib/python3.12/dist-packages (1.11.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.20.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.36.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.22.1)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: pyarrow>=21.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (22.0.0)
Requirement already satisfied: dill<0.4.1,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: httpx<1.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.28.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.12/dist-packages (from datasets) (3.6.0)
Requirement already satisfied: multiprocess<0.70.19 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2025.10.0,>=2023.1.0 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025.10.0,>=2023.1.0->datasets) (20
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from accelerate) (2.8.0+cu126)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025.10.0,>=2023.1.0->datasets) (3.13
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (4.11.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (2025.10.5)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (1.0.9)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (3.11)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1.0.0->datasets) (0.16.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,>=0.34.0->transformers) (4.15.0
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,>=0.34.0->transformers) (1.2.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.4.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2.5.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.4.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<=2025.10.0,>=
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<=2025.10.0,>=2023.1.
```

```
pip install tqdm
```

```
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (4.67.1)
```

```
pip install gensim
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.4.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gensim) (2.0.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.16.3)
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.5.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1->gensim) (2.0.1)
```

Step 2

In this step, I load the IMDB 50k movie reviews dataset into a pandas DataFrame so that I can preprocess and analyze the text for sentiment classification.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/IMDB Dataset.csv")
df = df[['review', 'sentiment']]
```

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

print("Rows:", len(df))
df.head()
```

```
Rows: 50000
```

	review	sentiment	label
0	One of the other reviewers has mentioned that ...	positive	1
1	A wonderful little production. The...	positive	1
2	I thought this was a wonderful way to spend ti...	positive	1
3	Basically there's a family where a little boy ...	negative	0
4	Petter Mattei's "Love in the Time of Money" is...	positive	1

```
print("Rows:", len(df))
```

```
Rows: 50000
```

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

	review	sentiment	label
0	One of the other reviewers has mentioned that ...	positive	1
1	A wonderful little production. The...	positive	1
2	I thought this was a wonderful way to spend ti...	positive	1
3	Basically there's a family where a little boy ...	negative	0
4	Petter Mattei's "Love in the Time of Money" is...	positive	1

```
df.shape
df.info()
df['review'].head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
 2   label      50000 non-null  int64
dtypes: int64(1), object(2)
memory usage: 1.1+ MB
```

	review
0	One of the other reviewers has mentioned that ...
1	A wonderful little production. The...
2	I thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...
4	Petter Mattei's "Love in the Time of Money" is...

dtype: object

Step 3:

To clean IMDB reviews by removing unwanted characters, converting text to lowercase, removing stopwords, and applying stemming + lemmatization so the text becomes suitable for machine learning models.

```
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer

stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
# html tag removal
def remove_html(text):
    return re.sub(r'<.*?>', '', text)
df['no_html'] = df['review'].apply(remove_html)
df[['review', 'no_html']].head()
```

	review	no_html
0	One of the other reviewers has mentioned that ...	One of the other reviewers has mentioned that ...
1	A wonderful little production. The...	A wonderful little production. The filming tec...
2	I thought this was a wonderful way to spend ti...	I thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...	Basically there's a family where a little boy ...
4	Petter Mattei's "Love in the Time of Money" is...	Petter Mattei's "Love in the Time of Money" is...

```
#Lowercase conversion
def convert_lowercase(text):
    return text.lower()
df['lowercase'] = df['no_html'].apply(convert_lowercase)
df[['no_html', 'lowercase']].head()
```

	no_html	lowercase
0	One of the other reviewers has mentioned that ...	one of the other reviewers has mentioned that ...
1	A wonderful little production. The filming tec...	a wonderful little production. the filming tec...
2	I thought this was a wonderful way to spend ti...	i thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...	basically there's a family where a little boy ...
4	Petter Mattei's "Love in the Time of Money" is...	petter mattei's "love in the time of money" is...

```
#Special Character Removal
def remove_special_characters(text):
    return re.sub(r'[^a-zA-Z\s]', ' ', text)
df['no_special'] = df['lowercase'].apply(remove_special_characters)
df[['lowercase', 'no_special']].head()
```

	lowercase	no_special
0	one of the other reviewers has mentioned that ...	one of the other reviewers has mentioned that ...
1	a wonderful little production. the filming tec...	a wonderful little production the filming tec...
2	i thought this was a wonderful way to spend ti...	i thought this was a wonderful way to spend ti...
3	basically there's a family where a little boy ...	basically there s a family where a little boy ...
4	petter mattei's "love in the time of money" is...	petter mattei s love in the time of money is...

```
#Tokenization
def tokenize_text(text):
    return word_tokenize(text)
df['tokens'] = df['no_special'].apply(tokenize_text)
df[['no_special', 'tokens']].head()
```

	no_special	tokens
0	one of the other reviewers has mentioned that ...	[one, of, the, other, reviewers, has, mentione...
1	a wonderful little production the filming tec...	[a, wonderful, little, production, the, filmin...
2	i thought this was a wonderful way to spend ti...	[i, thought, this, was, a, wonderful, way, to,...
3	basically there s a family where a little boy ...	[basically, there, s, a, family, where, a, lit...
4	petter mattei s love in the time of money is...	[petter, mattei, s, love, in, the, time, of, m...

```
#Stopwords Removal
def remove_stopwords(tokens):
    return [t for t in tokens if t not in stop_words]
df['no_stopwords'] = df['tokens'].apply(remove_stopwords)
df[['tokens', 'no_stopwords']].head()
```

	tokens	no_stopwords
0	[one, of, the, other, reviewers, has, mentione...	[one, reviewers, mentioned, watching, oz, epis...
1	[a, wonderful, little, production, the, filmin...	[wonderful, little, production, filming, techn...
2	[i, thought, this, was, a, wonderful, way, to,...	[thought, wonderful, way, spend, time, hot, su...
3	[basically, there, s, a, family, where, a, lit...	[basically, family, little, boy, jake, thinks,...
4	[petter, mattei, s, love, in, the, time, of, m...	[petter, mattei, love, time, money, visually, ...

```
#Stemming
def apply_stemming(tokens):
    return [stemmer.stem(t) for t in tokens]
df['stemmed'] = df['no_stopwords'].apply(apply_stemming)
df[['no_stopwords', 'stemmed']].head()
```

	no_stopwords	stemmed
0	[one, reviewers, mentioned, watching, oz, epis...	[one, review, mention, watch, oz, episod, hook...
1	[wonderful, little, production, filming, techn...	[wonder, littl, product, film, techniqu, unass...
2	[thought, wonderful, way, spend, time, hot, su...	[thought, wonder, way, spend, time, hot, summe...
3	[basically, family, little, boy, jake, thinks,...	[basic, famili, littl, boy, jake, think, zombi...
4	[petter, mattei, love, time, money, visually, ...	[petter, mattei, love, time, money, visual, st...

```
#Lemmatization
def apply_lemmatization(tokens):
    return [lemmatizer.lemmatize(t) for t in tokens]
df['lemmatized'] = df['no_stopwords'].apply(apply_lemmatization)
df[['no_stopwords', 'lemmatized']].head()
```

	no_stopwords	lemmatized
0	[one, reviewers, mentioned, watching, oz, epis...	[one, reviewer, mentioned, watching, oz, episo...
1	[wonderful, little, production, filming, techn...	[wonderful, little, production, filming, techn...
2	[thought, wonderful, way, spend, time, hot, su...	[thought, wonderful, way, spend, time, hot, su...
3	[basically, family, little, boy, jake, thinks,...	[basically, family, little, boy, jake, think, ...
4	[petter, mattei, love, time, money, visually, ...	[petter, mattei, love, time, money, visually, ...

```
df['lemmatized'] = df['lemmatized']  # should be list of tokens
df['length'] = df['lemmatized'].apply(len)
print(df['length'].describe())
```

```
count    50000.000000
mean       118.119980
std         89.383593
min          3.000000
25%         63.000000
50%         88.000000
75%        144.000000
max       1416.000000
Name: length, dtype: float64
```

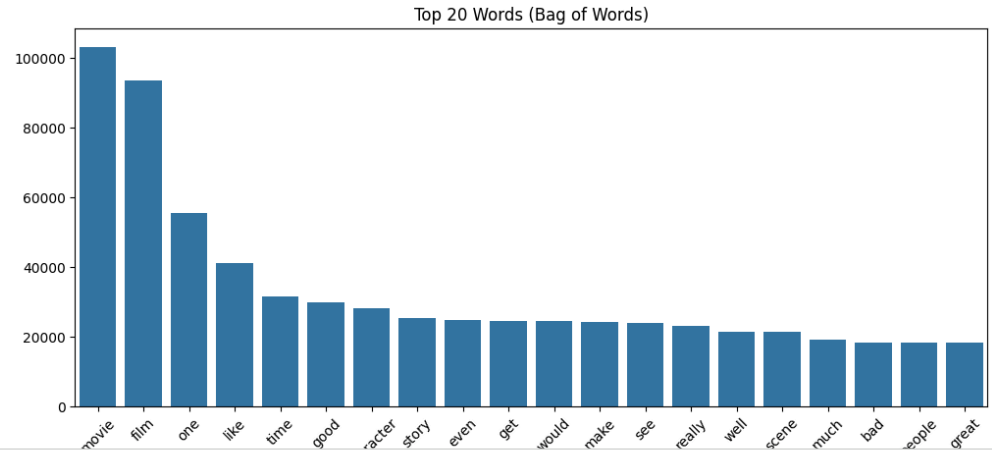
```
df['lemmatized_text'] = df['lemmatized'].apply(lambda x: " ".join(x) if isinstance(x, list) else str(x))
```

Step 3:

we transformed the cleaned text into numerical features using methods like BoW, TF-IDF, Word2Vec, GloVe, and N-grams to capture both statistical and semantic information from reviews. We also visualized the top contributing words and embedding patterns through graphs, and evaluated each feature representation using accuracy, precision, recall, and F1-score.

```
#Bag of Words (BoW)
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
import seaborn as sns
# Using lemmatized text
text_data = df['lemmatized_text']
bow = CountVectorizer(max_features=5000)
X_bow = bow.fit_transform(text_data)
print("BoW Shape:", X_bow.shape)
# Get top 20 words
word_counts = X_bow.sum(axis=0).A1
words = bow.get_feature_names_out()

# Sort words by frequency
sorted_idx = word_counts.argsort()[::-1][:20]
top_words = words[sorted_idx]
top_counts = word_counts[sorted_idx]
plt.figure(figsize=(12,5))
sns.barplot(x=top_words, y=top_counts)
plt.xticks(rotation=45)
plt.title("Top 20 Words (Bag of Words)")
plt.show()
```

```
BoW Shape: (50000, 5000)
```




```
text_data = df['lemmatized_text']
X_bow = bow.fit_transform(text_data)
from sklearn.model_selection import train_test_split

X_train_bow, X_test_bow, y_train_bow, y_test_bow = train_test_split(
    X_bow, df['label'], test_size=0.2, random_state=42
)
```

```
from sklearn.linear_model import LogisticRegression
clf_bow = LogisticRegression(max_iter=300)
clf_bow.fit(X_train_bow, y_train_bow)
```

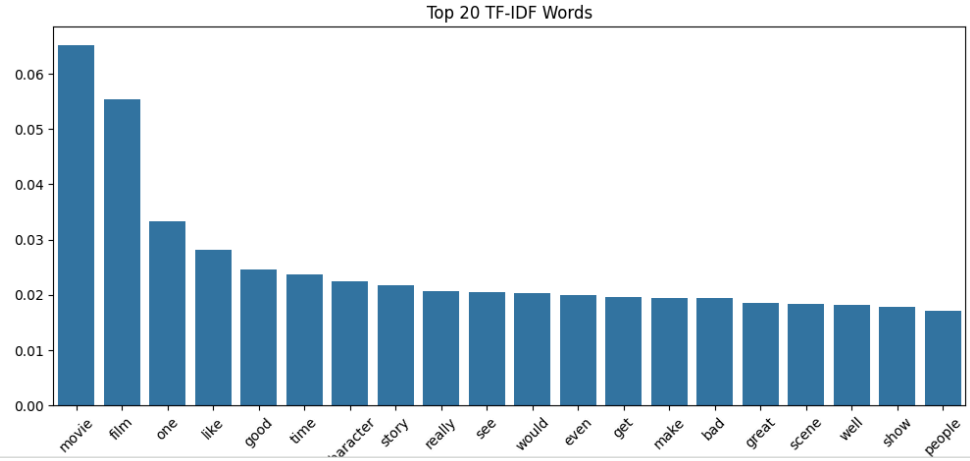
```
▾  LogisticRegression
LogisticRegression(max_iter=300)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

pred_bow = clf_bow.predict(X_test_bow)

print("Bag of Words Evaluation Results")
print("----------------------------------------")
print("Accuracy :", accuracy_score(y_test_bow, pred_bow))
print("Precision:", precision_score(y_test_bow, pred_bow))
print("Recall   :", recall_score(y_test_bow, pred_bow))
print("F1 Score :", f1_score(y_test_bow, pred_bow))
```

```
Bag of Words Evaluation Results
----------------------------------------
Accuracy : 0.8738
Precision: 0.8697865674564323
Recall   : 0.8815241119269697
F1 Score : 0.8756160063079046
```

```
#TF-IDF VECTORIZATION
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf.fit_transform(text_data)
print("TF-IDF Shape:", X_tfidf.shape)
# Top words by TF-IDF
import numpy as np
tfidf_scores = np.asarray(X_tfidf.mean(axis=0)).ravel()
sorted_idx = tfidf_scores.argsort()[::-1][:20]
top_words_tfidf = tfidf.get_feature_names_out()[sorted_idx]
top_scores = tfidf_scores[sorted_idx]

# Plot
plt.figure(figsize=(12,5))
sns.barplot(x=top_words_tfidf, y=top_scores)
plt.xticks(rotation=45)
plt.title("Top 20 TF-IDF Words")
plt.show()
```

```
TF-IDF Shape: (50000, 5000)
```




```
tfidf = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf.fit_transform(text_data)
from sklearn.model_selection import train_test_split

X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf = train_test_split(
    X_tfidf, df['label'], test_size=0.2, random_state=42
)
```

```
from sklearn.linear_model import LogisticRegression
clf_tfidf = LogisticRegression(max_iter=300)
clf_tfidf.fit(X_train_tfidf, y_train_tfidf)
```

```
LogisticRegression
LogisticRegression(max_iter=300)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
pred_tfidf = clf_tfidf.predict(X_test_tfidf)
print("TF-IDF Evaluation Results")
print("----------------------------------------")
print("Accuracy :", accuracy_score(y_test_tfidf, pred_tfidf))
print("Precision:", precision_score(y_test_tfidf, pred_tfidf))
print("Recall   :", recall_score(y_test_tfidf, pred_tfidf))
print("F1 Score :", f1_score(y_test_tfidf, pred_tfidf))
```

```
TF-IDF Evaluation Results
----------------------------------------
Accuracy : 0.8891
Precision: 0.8791972211501351
Recall   : 0.9041476483429252
F1 Score : 0.8914978964876236
```

```
#N-GRAMS (BIGRAM, TRIGRAM)
#Bi-gram
bigram = CountVectorizer(ngram_range=(2,2), max_features=5000)
X_bigram = bigram.fit_transform(text_data)
print("Bigram Shape:", X_bigram.shape)

#Tri-gram
trigram = CountVectorizer(ngram_range=(3,3), max_features=5000)
X_trigram = trigram.fit_transform(text_data)
print("Trigram Shape:", X_trigram.shape)
```

```
Bigram Shape: (50000, 5000)
Trigram Shape: (50000, 5000)
```

```
df = df[df['lemmatized'].apply(lambda x: len(x) > 0)]
df = df.reset_index(drop=True)
print("Remaining reviews:", len(df))
```

```
Remaining reviews: 50000
```

```
#Train Word2Vec model
from gensim.models import Word2Vec

sentences = df['lemmatized'].tolist()

w2v_model = Word2Vec(
```

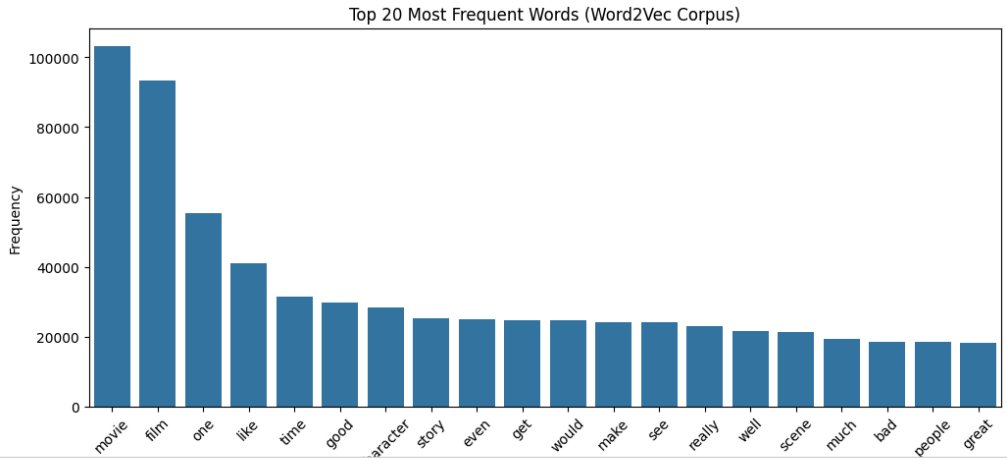
```
    sentences=sentences,
    vector_size=100,
    window=5,
    min_count=2,
    workers=4,
    sg=0
)

print("Vocabulary size:", len(w2v_model.wv.index_to_key))
```

```
Vocabulary size: 55224
```

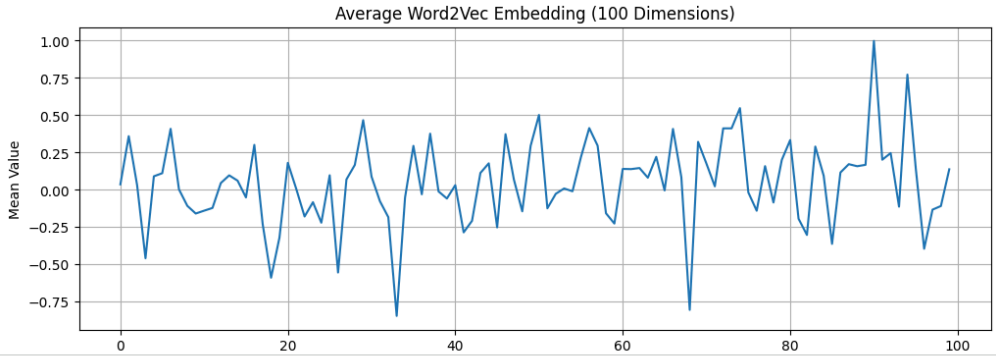
```
#Convert each review → 100-dim vector
import numpy as np
def get_w2v_vector(tokens):
    tokens = [word for word in tokens if word in w2v_model.wv]

    if len(tokens) == 0:
        return np.zeros(100)
    else:
        return np.mean(w2v_model.wv[tokens], axis=0)
X_w2v = np.vstack(df['lemmatized'].apply(get_w2v_vector))
print("Word2Vec Shape:", X_w2v.shape)
```

```
Word2Vec Shape: (50000, 100)
```

```
#Plot Top 20 Most Frequent Words
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns
all_tokens = [token for tokens in df['lemmatized'] for token in tokens]
top20 = Counter(all_tokens).most_common(20)
words = [w for w,c in top20]
counts = [c for w,c in top20]
plt.figure(figsize=(12,5))
sns.barplot(x=words, y=counts)
plt.xticks(rotation=45)
plt.title("Top 20 Most Frequent Words (Word2Vec Corpus)")
plt.xlabel("Word")
plt.ylabel("Frequency")
plt.show()
```




```
avg_vec = X_w2v.mean(axis=0)
plt.figure(figsize=(12,4))
plt.plot(avg_vec)
plt.title("Average Word2Vec Embedding (100 Dimensions)")
plt.xlabel("Dimension Index")
plt.ylabel("Mean Value")
plt.grid(True)
plt.show()
```




```
from sklearn.model_selection import train_test_split
X_train_w2v, X_test_w2v, y_train_w2v, y_test_w2v = train_test_split(
    X_w2v, df['label'], test_size=0.2, random_state=42
)
```

```
from sklearn.linear_model import LogisticRegression
clf_w2v = LogisticRegression(max_iter=300)
clf_w2v.fit(X_train_w2v, y_train_w2v)
```

```
LogisticRegression(max_iter=300)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
pred_w2v = clf_w2v.predict(X_test_w2v)
print("Word2Vec Evaluation Results")
print("----------------------------------------")
print("Accuracy :", accuracy_score(y_test_w2v, pred_w2v))
print("Precision:", precision_score(y_test_w2v, pred_w2v))
print("Recall   :", recall_score(y_test_w2v, pred_w2v))
print("F1 Score :", f1_score(y_test_w2v, pred_w2v))
```

```
Word2Vec Evaluation Results
----------------------------------------
Accuracy : 0.8602
Precision: 0.8567509308250049
Recall   : 0.8676324667592776
F1 Score : 0.8621573654111615
```

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove.6B.zip
```

```
--2025-11-15 12:56:51--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2025-11-15 12:56:52--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2025-11-15 12:56:52--  https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip.1’

glove.6B.zip.1      100%[===================>] 822.24M  5.10MB/s    in 2m 42s

2025-11-15 12:59:35 (5.09 MB/s) - ‘glove.6B.zip.1’ saved [862182613/862182613]

Archive:  glove.6B.zip
replace glove.6B.50d.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: glove.6B.50d.txt
replace glove.6B.100d.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: glove.6B.100d.txt       y

replace glove.6B.200d.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename:   inflating: glove.6B.200d.txt       y
y

replace glove.6B.300d.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename:   inflating: glove.6B.300d.txt       y
y
```

```
#Load GloVe Embeddings
import numpy as np
glove_path = "glove.6B.100d.txt"
glove_embeddings = {}
with open(glove_path, 'r', encoding="utf8") as f:
    for line in f:
        values = line.split()
        word = values[0]
        vector = np.asarray(values[1:], dtype='float32')
        glove_embeddings[word] = vector

print("Loaded GloVe vectors:", len(glove_embeddings))
```

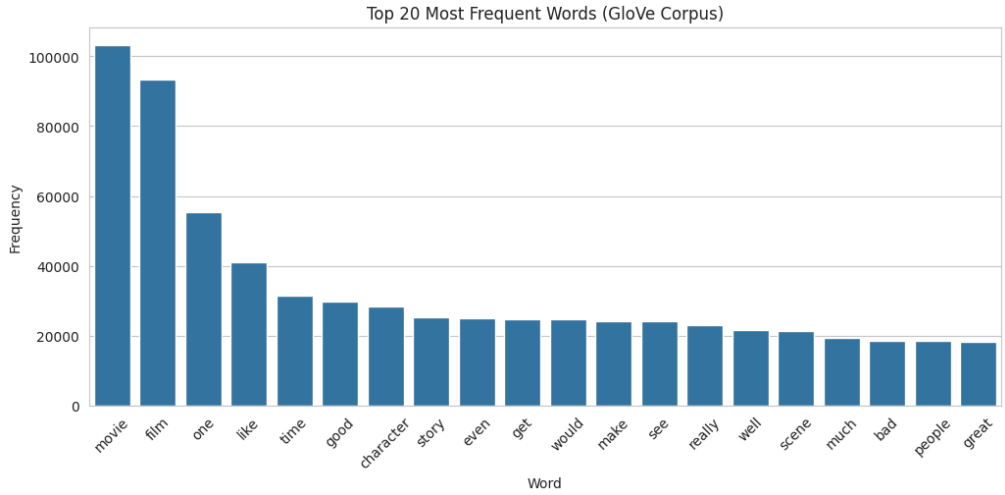
```
Loaded GloVe vectors: 400000
```

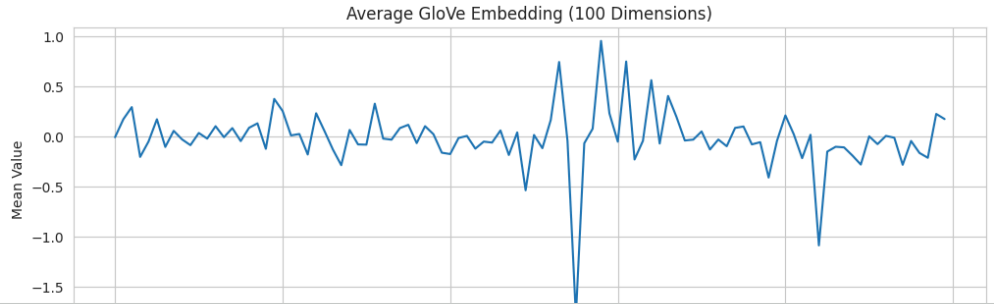
```
#Convert Each Review to a GloVe Vector
EMBEDDING_DIM = 100
def get_glove_vector(tokens):
    vectors = [glove_embeddings[w] for w in tokens if w in glove_embeddings]
    if len(vectors) == 0:
        return np.zeros(EMBEDDING_DIM)
    return np.mean(vectors, axis=0)
X_glove = np.vstack(df['lemmatized'].apply(get_glove_vector))
print("GloVe Shape:", X_glove.shape)
```

```
GloVe Shape: (50000, 100)
```

```
#Graph: Top 20 Most Frequent Words
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")
all_tokens = [t for tokens in df['lemmatized'] for t in tokens]
top20 = Counter(all_tokens).most_common(20)
words = [w for w,c in top20]
counts = [c for w,c in top20]
plt.figure(figsize=(12,5))
sns.barplot(x=words, y=counts)
plt.xticks(rotation=45)
plt.title("Top 20 Most Frequent Words (GloVe Corpus)")
plt.xlabel("Word")
plt.ylabel("Frequency")
plt.show()
avg_vec = X_glove.mean(axis=0)
plt.figure(figsize=(12,4))
plt.plot(avg_vec)
plt.title("Average GloVe Embedding (100 Dimensions)")
plt.xlabel("Dimension Index")
```

```
plt.ylabel("Mean Value")
plt.grid(True)
plt.show()
```






```
from sklearn.model_selection import train_test_split
X_train_glove, X_test_glove, y_train_glove, y_test_glove = train_test_split(
    X_glove, df['label'], test_size=0.2, random_state=42
)
```

```
from sklearn.linear_model import LogisticRegression
clf_glove = LogisticRegression(max_iter=300)
clf_glove.fit(X_train_glove, y_train_glove)
```

```
LogisticRegression
LogisticRegression(max_iter=300)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

pred_glove = clf_glove.predict(X_test_glove)

print("GloVe Evaluation Results")
print("----------------------------------------")
print("Accuracy :", accuracy_score(y_test_glove, pred_glove))
print("Precision:", precision_score(y_test_glove, pred_glove))
print("Recall   :", recall_score(y_test_glove, pred_glove))
print("F1 Score :", f1_score(y_test_glove, pred_glove))
```

```
GloVe Evaluation Results
----------------------------------------
Accuracy : 0.7947
Precision: 0.7972919155714855
Recall   : 0.7946021035919826
F1 Score : 0.7959447371036676
```

Step 4: Fine-tuned the BERT transformer model on the IMDB dataset to learn deep contextual representations of text. After training, we evaluated the model using accuracy, precision, recall, and F1-score, achieving significantly better performance compared to traditional machine learning and embedding-based methods.

```
!pip install --upgrade transformers accelerate datasets evaluate
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.57.1)
Requirement already satisfied: accelerate in /usr/local/lib/python3.12/dist-packages (1.11.0)
Requirement already satisfied: datasets in /usr/local/lib/python3.12/dist-packages (4.4.1)
Requirement already satisfied: evaluate in /usr/local/lib/python3.12/dist-packages (0.4.6)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.20.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.36.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.22.1)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from accelerate) (2.8.0+cu126)
Requirement already satisfied: pyarrow>=21.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (22.0.0)
Requirement already satisfied: dill<0.4.1,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: httpx<1.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.28.1)
```

```
Requirement already satisfied: xxhash in /usr/local/lib/python3.12/dist-packages (from datasets) (3.6.0)
Requirement already satisfied: multiprocess<0.70.19 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2025.10.0,>=2023.1.0 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025.10.0,>=2023.1.0->datasets) (20
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025.10.0,>=2023.1.0->datasets) (3.13
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (4.11.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (2025.10.5)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (1.0.9)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->datasets) (3.11)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1.0.0->datasets) (0.16.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,>=0.34.0->transformers) (4.15.0
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,>=0.34.0->transformers) (1.2.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.4.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2.5.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.4.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<=2025.10.0,>=
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]<=2025.10.0,>=2023.1
```

```
import torch
from transformers import BertTokenizerFast, BertForSequenceClassification, Trainer, TrainingArguments
from datasets import Dataset
import numpy as np
import evaluate
```

```
dataset = Dataset.from_pandas(df[['review', 'label']])

dataset = dataset.train_test_split(test_size=0.2, seed=42)
```

```
tokenizer = BertTokenizerFast.from_pretrained("bert-base-uncased")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: 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 secret in your Google Colab
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.
  warnings.warn(
```

```
#Tokenize
def tokenize_function(example):
    return tokenizer(
        example["review"],
        truncation=True,
        padding="max_length",
        max_length=256
    )

tokenized_dataset = dataset.map(tokenize_function, batched=True)
tokenized_dataset = tokenized_dataset.remove_columns(["review"])
tokenized_dataset.set_format("torch")
```

```
Map: 100%  40000/40000 [01:19<00:00, 634.72 examples/s]

Map: 100%  10000/10000 [00:15<00:00, 644.23 examples/s]
```

```
#Load model
model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels=2
)
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['classifier.bias'
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
#Metrics
accuracy = evaluate.load("accuracy")
f1 = evaluate.load("f1")

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=-1)
    return {
        "accuracy": accuracy.compute(predictions=preds, references=labels)["accuracy"],
        "f1": f1.compute(predictions=preds, references=labels)["f1"]
    }
```

```
#Training Arguments
training_args = TrainingArguments(
    output_dir="./bert_output",
    num_train_epochs=2,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    learning_rate=2e-5,
    weight_decay=0.01,
    logging_steps=100,
```

```
    do_train=True,
    do_eval=True,
    report_to="none"
)
```

```
#Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset["test"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
```

```
/tmp/ipython-input-2596076320.py:2: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class
  trainer = Trainer(
```

```
trainer.train()
```

[10000/10000 1:20:50, Epoch 2/2]

Step	Training Loss
100	0.493200
200	0.361900
300	0.360800
400	0.399900
500	0.360300
600	0.335400
700	0.351400
800	0.275600
900	0.347300
1000	0.324400
1100	0.295500
1200	0.296100
1300	0.316300
1400	0.339300
1500	0.267400
1600	0.335700
1700	0.357100
1800	0.280600
1900	0.370100
2000	0.287800
2100	0.266600
2200	0.329900
2300	0.293000
2400	0.303700
2500	0.313800
2600	0.300300
2700	0.250100
2800	0.290700
2900	0.309500
3000	0.291700
3100	0.279300
3200	0.243000
3300	0.254500
3400	0.267300
3500	0.281900
3600	0.264500
3700	0.207300

```
trainer.evaluate()
```

[1250/1250 02:28]

```
{'eval_loss': 0.2954573929309845,
 'eval_accuracy': 0.9306,
 'eval_f1': 0.9306206113237961,
 'eval_runtime': 148.7113,
 'eval_samples_per_second': 67.244,
 'eval_steps_per_second': 8.406,
 'epoch': 2.0}
```

```
import torch

def predict_sentiment(text):
    try:
        bert_model = trainer.model
    except:
        bert_model = model

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    bert_model.to(device)
    bert_model.eval()
    tokens = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=256)
    tokens = {k: v.to(device) for k, v in tokens.items()}
    with torch.no_grad():
        outputs = bert_model(**tokens)
        preds = torch.argmax(outputs.logits, dim=1).item()

    return "Positive" if preds == 1 else "Negative"
print(predict_sentiment("This movie was awesome!"))
print(predict_sentiment("Terrible film, waste of time."))
```

5200	0.132900

```
Positive
Negative
```

5300	0.212800
5400	0.191200

```
#Model Evaluation (Accuracy, Precision, Recall, F1 Score)
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import numpy as np
```

```
pred_output = trainer.predict(tokenized_dataset["test"])
logits = pred_output.predictions
predictions = np.argmax(logits, axis=1)
true_labels = pred_output.label_ids
```

6100	0.206300
6200	0.182300

```
accuracy = accuracy_score(true_labels, predictions)
precision = precision_score(true_labels, predictions)
recall = recall_score(true_labels, predictions)
f1 = f1_score(true_labels, predictions)

print("BERT Evaluation Results")
print("----------------------------------------")
print("Accuracy: ", accuracy)
print("Precision:", precision)
print("Recall:   ", recall)
print("F1 Score: ", f1)
```

7000	0.140500

```
BERT Evaluation Results
----------------------------------------
Accuracy:  0.9306
Precision: 0.9248647565618112
Recall:    0.9353596757852077
F1 Score:  0.9300826113237961
```

7100	0.196900
7200	0.209100
7300	0.166000
7400	0.177900
7500	0.144300
7600	0.178600
7700	0.149600
7800	0.132000

Step 5 :

Compared the performance of all models—BoW, TF-IDF, Word2Vec, GloVe, and BERT—using a combined results table and visual comparison graphs. This helped clearly identify which model achieved the highest accuracy, precision, recall, and F1-score, with BERT outperforming all traditional and embedding-based approaches.

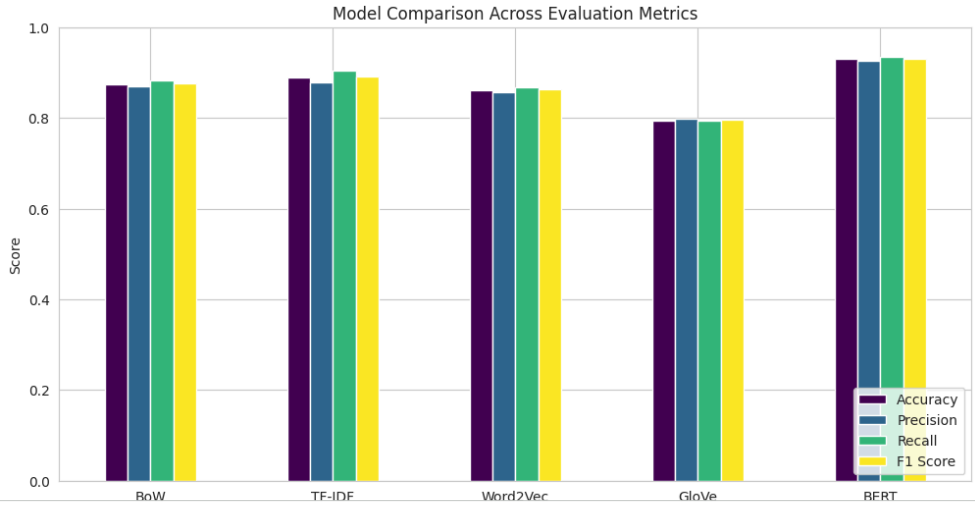
```
import pandas as pd

results = {
    "Model": ["BoW", "TF-IDF", "Word2Vec", "GloVe", "BERT"],

    "Accuracy": [
        accuracy_score(y_test_bow, pred_bow),
        accuracy_score(y_test_tfidf, pred_tfidf),
        accuracy_score(y_test_w2v, pred_w2v),
        accuracy_score(y_test_glove, pred_glove),
        accuracy
    ],

    "Precision": [
        precision_score(y_test_bow, pred_bow),
        precision_score(y_test_tfidf, pred_tfidf),
        precision_score(y_test_w2v, pred_w2v),
        precision_score(y_test_glove, pred_glove),
        precision
    ],

    "Recall": [
        recall_score(y_test_bow, pred_bow),
        recall_score(y_test_tfidf, pred_tfidf),
        recall_score(y_test_w2v, pred_w2v),
        recall_score(y_test_glove, pred_glove),
        recall
    ],

    "F1 Score": [
        f1_score(y_test_bow, pred_bow),
        f1_score(y_test_tfidf, pred_tfidf),
        f1_score(y_test_w2v, pred_w2v),
        f1_score(y_test_glove, pred_glove),
        f1
    ]
}

comparison_df = pd.DataFrame(results)
comparison_df
```

	Model	Accuracy	Precision	Recall	F1 Score
0	BoW	0.8738	0.869787	0.881524	0.875616
1	TF-IDF	0.8891	0.879197	0.904148	0.891498
2	Word2Vec	0.8602	0.856751	0.867632	0.862157
3	GloVe	0.7947	0.797292	0.794602	0.795945
4	BERT	0.9306	0.924865	0.935360	0.930083

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")
plt.figure(figsize=(12,6))

comparison_df.set_index("Model")[["Accuracy", "Precision", "Recall", "F1 Score"]].plot(
    kind="bar",
    figsize=(12,6),
    colormap="viridis"
)
plt.title("Model Comparison Across Evaluation Metrics")
plt.ylabel("Score")
plt.xticks(rotation=0)
plt.ylim(0, 1)
```

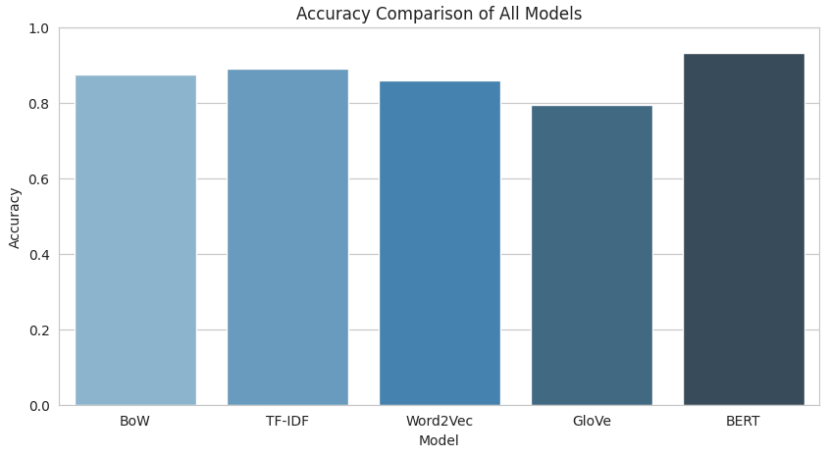
```
plt.legend(loc="lower right")
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```




```
plt.figure(figsize=(10,5))
sns.barplot(x=comparison_df["Model"], y=comparison_df["Accuracy"], palette="Blues_d")
plt.title("Accuracy Comparison of All Models")
plt.ylabel("Accuracy")
plt.ylim(0,1)
plt.show()
```

```
/tmp/ipython-input-1941131041.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same ef

  sns.barplot(x=comparison_df["Model"], y=comparison_df["Accuracy"], palette="Blues_d")
```



Conclusion:

This project analyzes sentiment by converting movie reviews into numerical representations using NLP techniques such as BoW, TF-IDF, Word2Vec, GloVe, and BERT. These features are then fed into machine learning or transformer-based models that learn patterns indicating positive or negative opinions. Finally, the trained models classify new reviews by predicting whether the expressed sentiment is positive or negative based on learned linguistic and semantic cues.