Machine Learning Internship Task: Sentiment Analysis on Movie Reviews

Task Description :
Your task for the machine learning internship application is to build a sentiment analysis model that can predict whether a movie review is positive or negative. You will be using the IMDB Movie Reviews dataset for this task. The goal is to create a machine learning model that can accurately classify the sentiment of a given review.

Steps on building the model:

1. **Downloading the dataset the transforming into csv format:**
   The initial dataset was organised into "train" and "test" folders, each containing "pos" and "neg" subfolders. These subfolders held text files named 'ID_rating.txt', where 'ID' serves as a unique identifier and 'rating' indicates the sentiment polarity. To facilitate data processing, the script 'merge.py' was used in both the "train" and "test" folders to combine the positive and negative reviews and generate 'movie_reviews.csv' and 'movie_reviews_test.csv' CSV files.

2. **Understanding the data:**

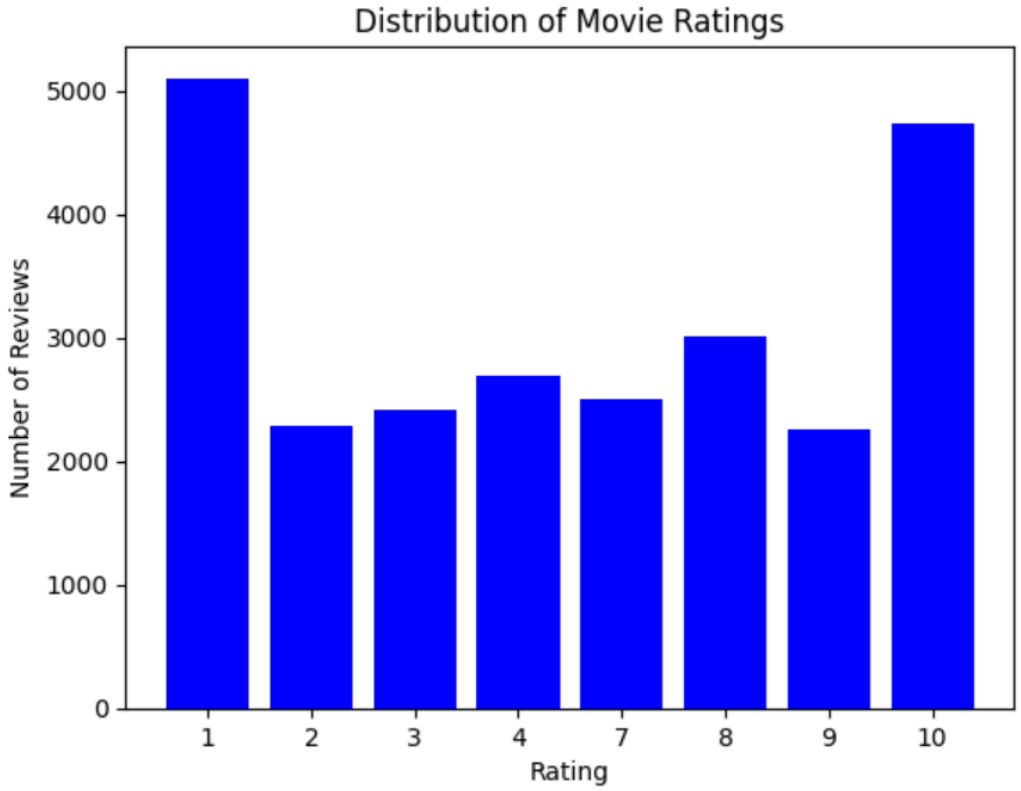

Figure 1: Distribution of Movie Ratings

The provided visual highlights the distribution of Movie Ratings within the training dataset. Among the 25,000 records present, ratings span the spectrum from 1 to 10. Curiously, there's an absence of records for ratings 5 and 6. Notably, the dataset

showcases a significant concentration of records in ratings 1 and 10. This observed imbalance prompts consideration of potential biases that could arise during classification tasks. The prevalence of records in ratings 1 and 10 might inadvertently steer machine learning models towards favoring these specific labels in classification. This underscores the importance of devising strategies to address class imbalance, ensuring fair and accurate model performance.

3. **Preprocessing the data:**
   For effective text preprocessing applied to the 'Comment' column of a dataset, a combination of the WordNet Lemmatizer and NLTK's stopwords is employed. This dynamic process involves cleaning, tokenization, punctuation removal, and lemmatization. The `preprocess_text(text)` function acts as the core of this procedure, wherein input text is initially transformed into lowercase, while punctuation is expunged and tokens are generated. Handling '
' tags is also integrated seamlessly. Subsequently, token lemmatization and stopword elimination are executed, followed by reassembling the refined tokens into coherent text. The result is a thoroughly cleaned text output. To operationalize this, the `apply` method effectively applies the `preprocess_text` function to the 'Comment' column, culminating in comprehensive text preprocessing.

4. **Vectorization:**
   The process of converting text data into TF-IDF (Term Frequency-Inverse Document Frequency) features utilizes the `TfidfVectorizer` from scikit-learn, the code efficiently transforms the input text data into a matrix of TF-IDF values. The `max_features` parameter allows for limiting the number of features created, enhancing efficiency and relevance. Once executed, `X_train_tfidf` holds the transformed features derived from the text data, ready for further analysis on model training. This segment seamlessly integrates text data into a format suitable for various machine learning tasks, such as classification or clustering.

5. **Preprocessing the test data:**
   Apply the above 3 and 4 step on the test dataset also to make it ready for testing after model building.

6. **Model Selection:**
   I have trained the above pre-processed dataset on 7 different models. The below summary shows a overview of observations on the training.

   Summary of Classification Reports:

   - Random Forest Classifier:
     - Best Performance: Rating 10 (Precision: 0.24, Recall: 0.84, F1-score: 0.37)
     - Overall Accuracy: 0.28

   - Naive Bayes:
     - Best Performance: Rating 10 (Precision: 0.29, Recall: 0.16, F1-score: 0.21)

- Overall Accuracy: 0.17

- SVM - linear:
  - Best Performance: Rating 10 (Precision: 0.29, Recall: 0.26, F1-score: 0.27)
  - Overall Accuracy: 0.21

- SVM - rbf:
  - Best Performance: Rating 10 (Precision: 0.26, Recall: 0.51, F1-score: 0.35)
  - Overall Accuracy: 0.25

- Gradient Boosting Classifier:
  - Best Performance: Rating 10 (Precision: 0.24, Recall: 0.72, F1-score: 0.36)
  - Overall Accuracy: 0.26

- Neural Networks:
  - Best Performance: Rating 10 Accuracy: 0.7883
  - Overall Accuracy: 0.1890 (Test accuracy)

- Logistic Regression:
  - Best Performance: Rating 10 (Precision: 0.27, Recall: 0.36, F1-score: 0.31)
  - Overall Accuracy: 0.23

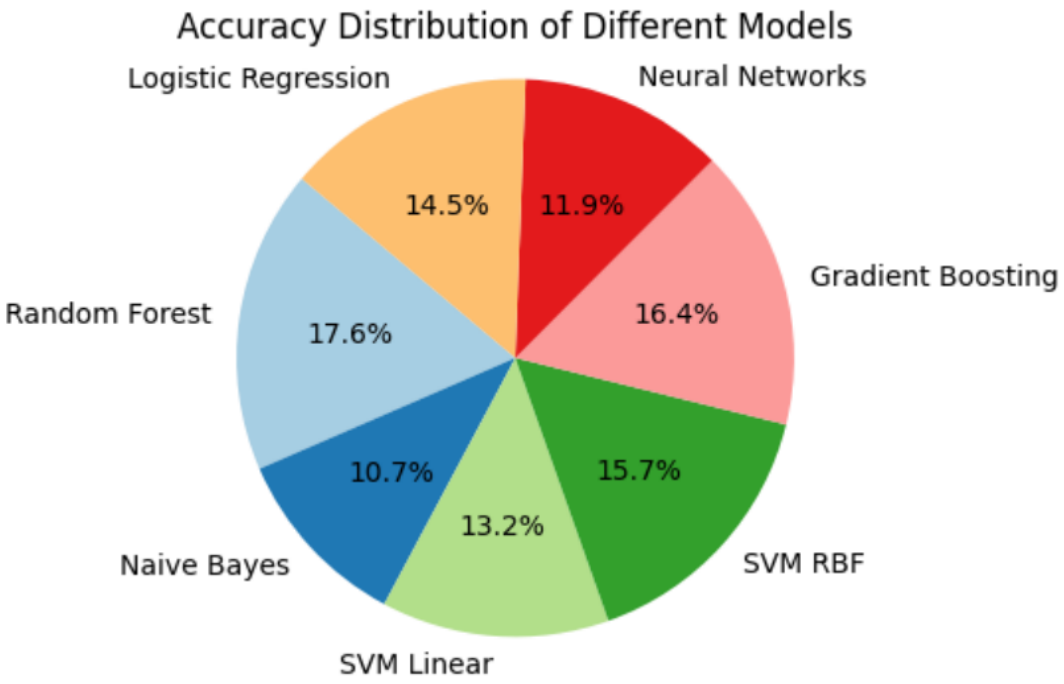

Figure 2: Accuracy Distribution of Different Models

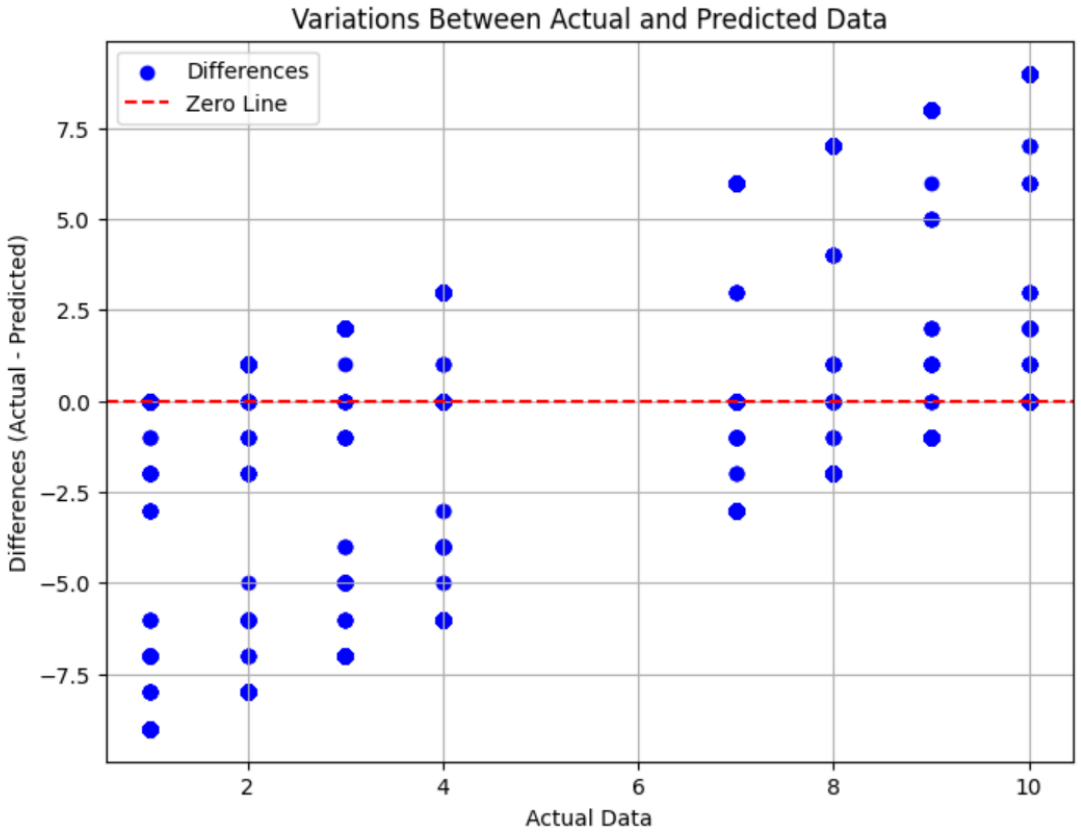

Figure 3: Variation between Actual Data and Predicted Data(Random Foresr Classifier)