Computational Intelligence Project

Dr. Jalal Oldin Nasiri

Sina Mollazadeh

Winter 1403

This code performs data preprocessing and machine learning tasks for sentiment analysis on the IMDB dataset of 50,000 movie reviews. It includes data cleaning, feature engineering, and the application of multiple machine learning models, such as Naive Bayes, Logistic Regression, and Neural Networks. The models' performance is evaluated using metrics such as F1-score, precision, recall, accuracy, and confusion matrices.

Importing and using the Dataset

1. **Setup and File Upload**
   The code starts by uploading the kaggle.json file, which contains the user's Kaggle API credentials, to authenticate the download of datasets from Kaggle.
2. **Download and Unzip Dataset**
   Using the Kaggle API, the IMDB dataset is downloaded and unzipped.
3. **Load and Explore Dataset**
   The dataset (CSV file) is loaded into a Pandas DataFrame, and initial exploration is performed using .head() and .info() to check the data.
4. **Visualize Sentiment Distribution**
   The code counts the number of positive and negative reviews and visualizes the sentiment distribution using a bar chart (Figure1).
5. **Word Count in Sentiments**
   The function CountWords counts the total number of words in positive and negative sentiment reviews.(Figure 2)
6. **Data Preprocessing: Remove Stop Words and Symbols**
   The code loads a list of stop words from a file and removes them from the reviews. Additionally, non-alphanumeric characters are removed using a regular expression.
7. **Word Frequency Analysis**
   The frequency of words in both positive and negative sentiment reviews is calculated and stored in a DataFrame.
8. **Rarity Grouping**
   Words are grouped into "high," "mid," and "low" rarity categories based on their frequency.
9. **Mean Positive/Negative Ratio Calculation**
   A dictionary is created for the positive-to-negative ratio of words, and this ratio is calculated for each review.

10. **Sampling of Reviews**
    Reviews are sampled based on quartiles of word counts in positive and negative sentiment reviews to create balanced datasets to further use in our models. (Figure 3)

11. **Feature Engineering**
    Various features such as word counts in positive and negative sentiment, rarity counts, and quartile information are extracted and added to the data.

12. **Correlation and Feature Scaling**
    A correlation matrix is generated for the features, and numerical features are scaled using MinMaxScaler for the models.

    According to (Figure 4) The Highest Correlation Between the Features and Sentiment is within the “mean_p_n_ratio” which is 0.61 and provides a good correlation between the model and the data.

13. Models

1. **The Naive Bayes model** is a probabilistic classifier that is based on Bayes' theorem, assuming that the features are independent. It is typically used for text classification tasks like sentiment analysis. The most common version used here is the Multinomial Naive Bayes, which works well for word counts in text data.

   Key Parameters:

   1. X_train: The training data, which consists of the features (word counts, word frequency ratios, etc.) of the reviews.
   2. y_train: The target labels for the training data, which are the sentiment labels (positive/negative).
   3. X_val: The validation data, which consists of the features for evaluating model performance during training.
   4. y_val: The target labels for the validation set.

2. **Logistic Regression** Model is a linear model used for binary classification. It predicts the probability of a class using a logistic function.

Key Parameters:

1. X_train: The training data (features).
2. y_train: The training labels (sentiment).
3. X_val: The validation set (features).
4. y_val: The validation set labels (sentiment).
5. Logistic Regression Hyperparameters:
    1. C: Regularization strength. Smaller values specify stronger regularization. A very small C can lead to underfitting, while large values can lead to overfitting.
    2. solver: The algorithm to use for optimization. Some options include:
    3. 'lbfgs': an optimizer in the quasi-Newton family.
    4. 'liblinear': a solver for small datasets.
    5. 'saga': an optimizer that can handle large datasets efficiently.
    6. max_iter: The maximum number of iterations for the solver to converge.
    7. penalty: The norm used in the penalization. Options include: 'l2': L2 regularization (default),
        'l1': L1 regularization.

3. **Neural Network Model** (Multilayer Perceptron), specifically a Multilayer Perceptron (MLP), is a type of artificial neural network used for complex classification tasks. It contains one or more layers of neurons (or perceptrons), where each layer is fully connected to the next.

Key Parameters:

1. X_train: The training data (features).
2. y_train: The training labels (sentiment).
3. X_val: The validation set (features).
4. y_val: The validation labels (sentiment).
5. Neural Network Hyperparameters:
    1. hidden_layer_sizes: The number of neurons in each hidden layer. A common example is (100,) for one hidden layer with 100 neurons, or (100, 100) for two hidden layers, each with 100 neurons.
    2. activation: The activation function for the hidden layers. Common options are:
    3. 'relu': Rectified Linear Unit, typically works well for most problems.
    4. 'tanh': Hyperbolic tangent function.
    5. 'logistic': Sigmoid function, useful for binary classification.
    6. solver: The optimization algorithm used to minimize the loss function. Options include:
    7. 'adam': Stochastic gradient-based optimization.
    8. 'sgd': Stochastic gradient descent.
    9. 'lbfgs': A quasi-Newton method.
    10. learning_rate: The learning rate for weight updates. Options include:
    11. 'constant': A constant learning rate.
    12. 'adaptive': The learning rate is adjusted based on the training process.
    13. max_iter: The maximum number of iterations for training.

Naive Bayes works by calculating probabilities for each class (positive/negative) based on the features (word frequencies). It assumes that the features are independent of each other.

Logistic Regression is a linear classifier that learns a decision boundary to separate classes based on input features.

Neural Networks (MLP) are a more complex type of model that uses multiple layers of neurons to learn intricate patterns in data, suitable for handling complex tasks like sentiment analysis. The Neural Networks Model Improvement is Shown Below.

(Figure 5, 6)

14. **Confusion Matrix**
    The confusion matrix is calculated and displayed for each model.

15. **Final Comparison of Models**
    The performance metrics of the three models are compiled into a DataFrame and printed for comparison.

Figure 1:

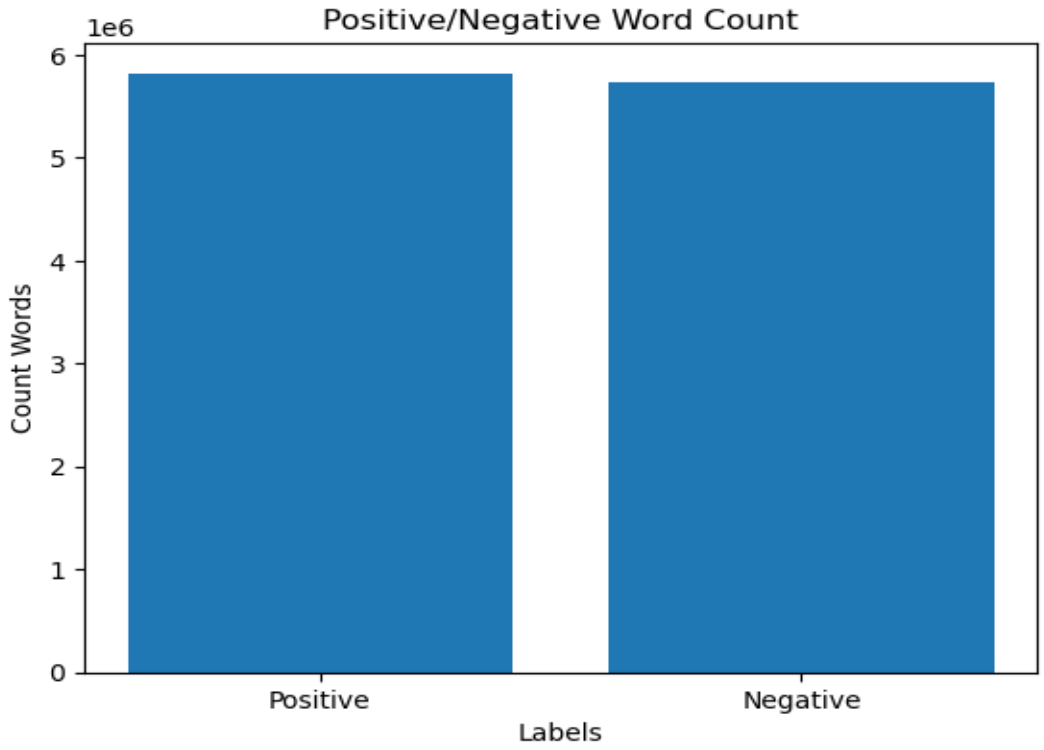


Figure 2

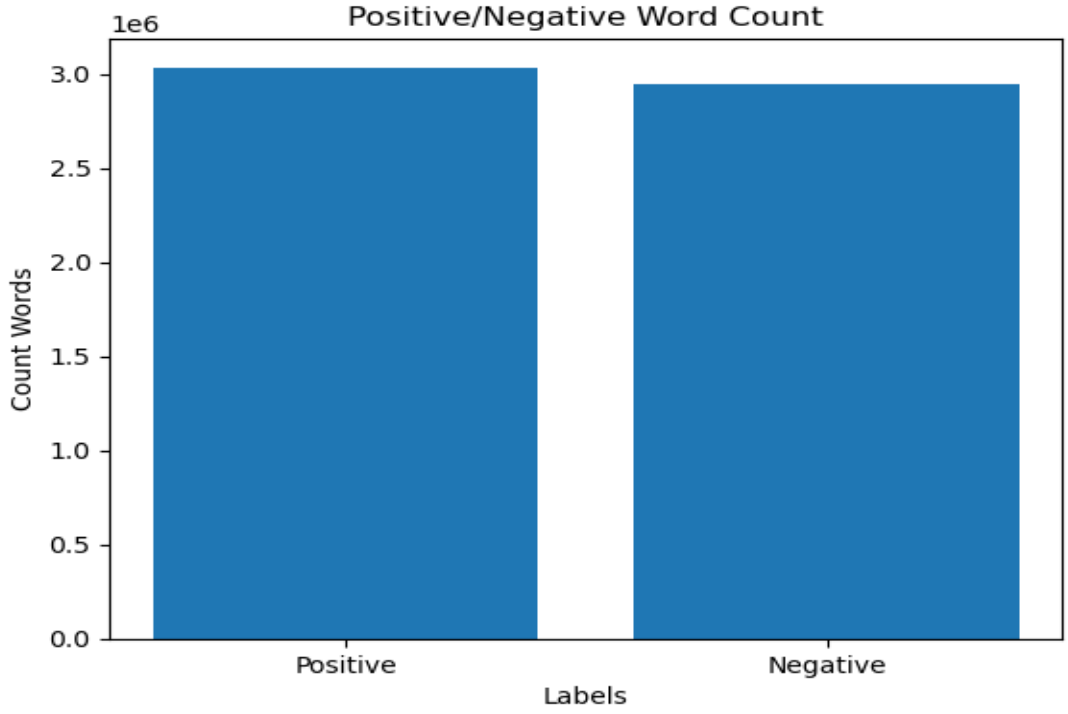


Figure 3

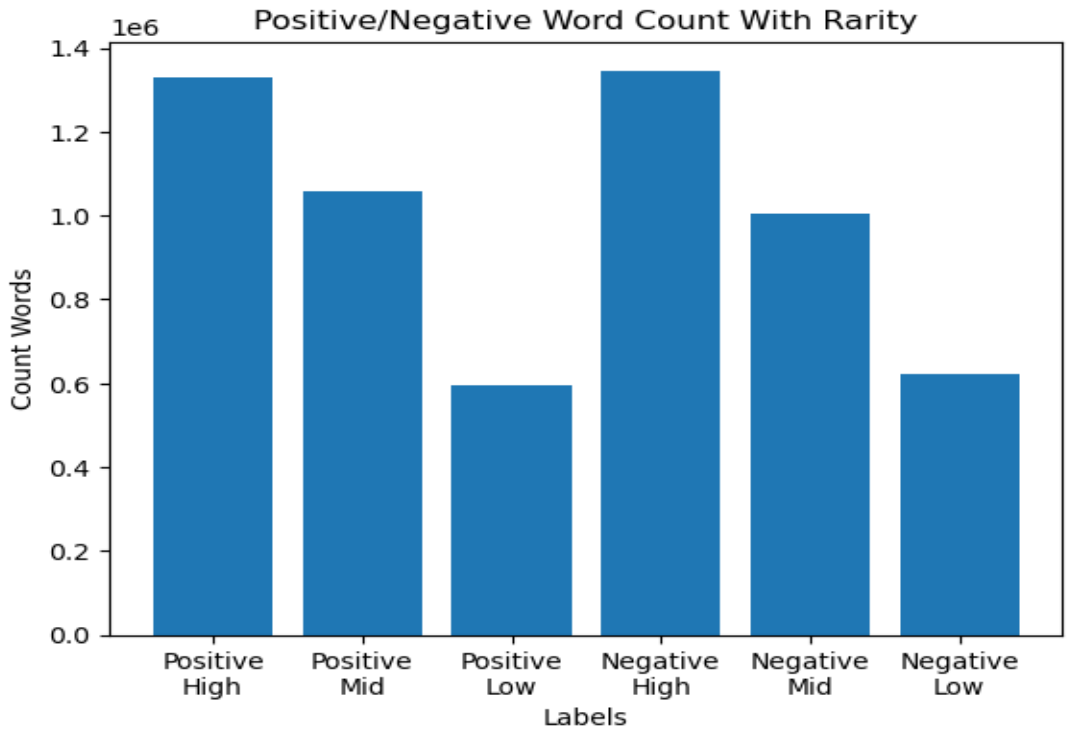


Figure 4

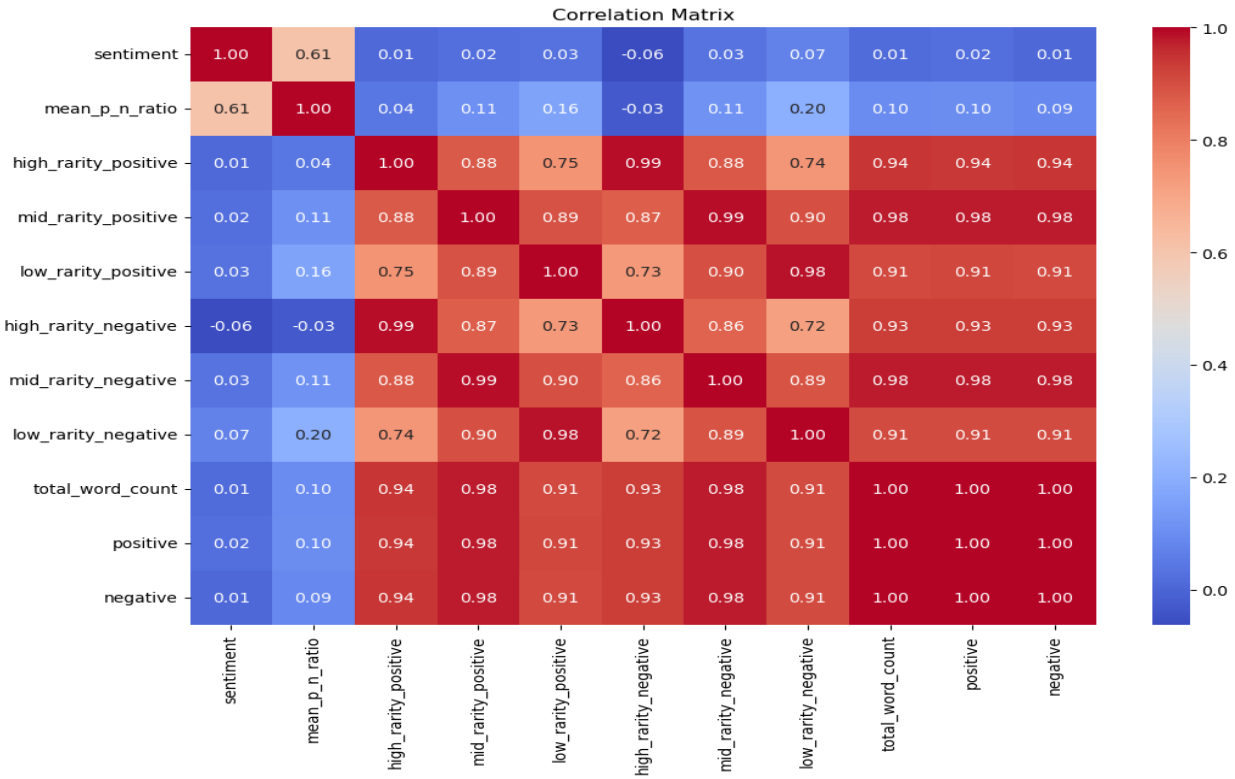


Figure 5

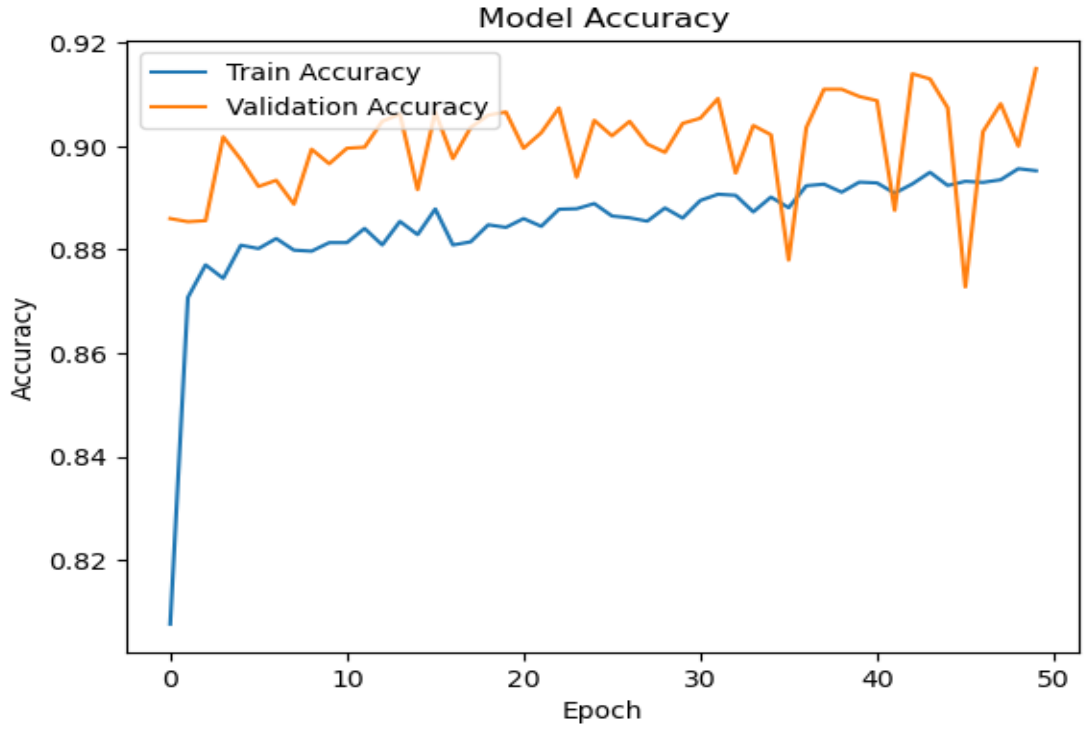
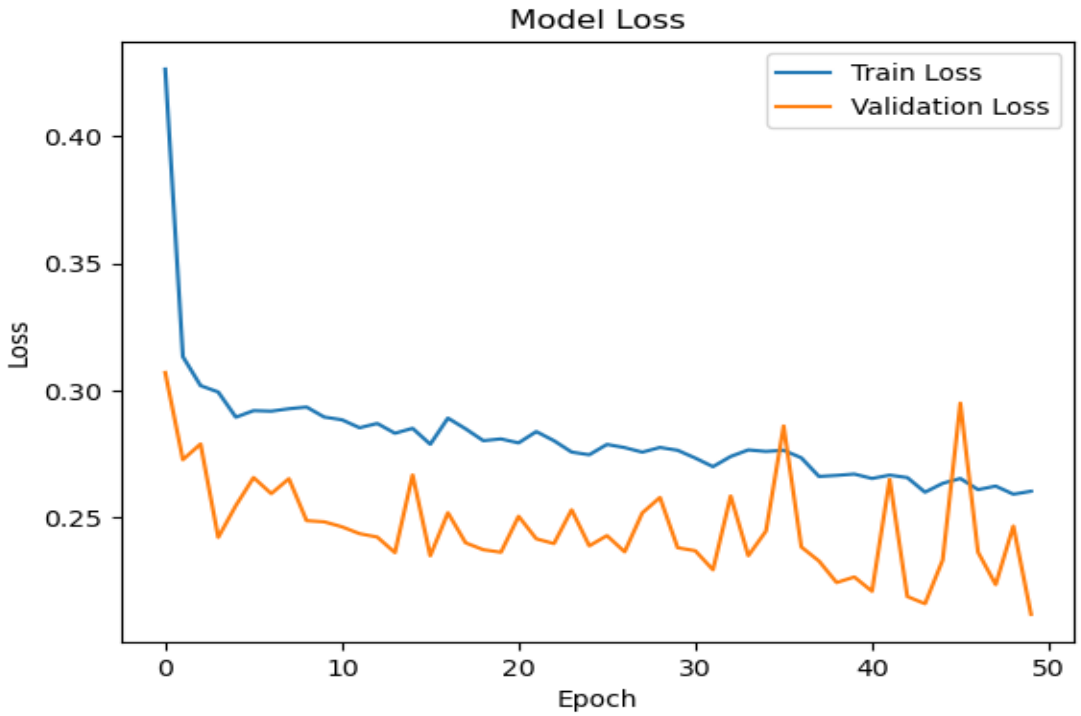


Figure 6


Model Loss
Train Loss
Validation Loss
0.40
0.35
0.30
0.25
Loss
0
10
20
30
40
50
Epoch