ASSIGNMENT

By

Rakshit Gupta

2021A1R050

Semester – 7th (A1)

Department of Computer Science and Engineering

Model Institute of Engineering & Technology (Autonomous)

(Permanently Affiliated to the University of Jammu, Accredited by NAAC with "A" Grade)

Jammu, India

2024

Table of Contents

1. Introduction

In recent years, sentiment analysis has emerged as an important task in natural language processing (NLP), enabling organizations to understand and analyze public opinions, customer feedback, and brand perception from text data. Sentiment analysis assigns a sentiment label—such as positive or negative—to a given text, helping organizations gauge how users feel about a product, service, or topic. This project focuses on performing sentiment analysis on movie reviews from the IMDB dataset, a large and widely-used resource for sentiment classification tasks.

The goal of this project is to accurately classify each movie review as either positive or negative based on the sentiment expressed within the review text. Sentiment analysis in movie reviews is especially useful for the entertainment industry, as it provides insights into audience preferences and responses, allowing studios and production companies to make data-driven decisions. Beyond the entertainment industry, sentiment analysis is widely applicable across fields such as customer service, marketing, social media monitoring, and brand management.

To achieve reliable and high-performance sentiment classification, we leverage **Artificial Neural Networks (ANN)**, a class of machine learning models known for their ability to capture complex and non-linear relationships in data. ANNs consist of interconnected layers of nodes (neurons) that process inputs and learn from patterns in data. For this project, an ANN is particularly advantageous over traditional machine learning models, such as Support Vector Machines or Naive Bayes classifiers, because its multi-layered architecture allows it to learn intricate representations in textual data. This capability is valuable in sentiment analysis, where the nuanced language of reviews often involves varied vocabulary, expressions, and context that traditional models may struggle to capture.

Our approach involves a systematic pipeline of data preprocessing, model development, hyperparameter tuning, and evaluation:

- **Data Preprocessing**: We start by addressing class imbalance in the dataset (where positive reviews outnumber negative reviews), balancing the dataset to ensure fair and representative learning. Text data is then transformed into numerical form using **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorization, which captures the relevance of each word in a review.
- **Model Development**: We design a multi-layered ANN for binary classification, using layers with ReLU activation functions and dropout to enhance learning and prevent overfitting.
- **Hyperparameter Tuning**: Using Keras Tuner, we optimize parameters such as the number of layers, neurons per layer, learning rate, and batch size to find the best configuration for high model accuracy and generalization.

2. Problem Statement

The problem involves classifying movie reviews as positive or negative using an ANN model. The primary challenges are:

1. **Class Imbalance**: The dataset has a significant imbalance between positive and negative reviews, which can lead to biased predictions.
2. **Text Data Representation**: Text data must be preprocessed and converted into numerical features suitable for input into a neural network.
3. **Model Optimization**: To achieve high accuracy, optimizing the ANN's structure and parameters is crucial.

Our approach involves balancing the dataset, transforming text data into TF-IDF features, building an ANN for classification, and tuning hyperparameters for optimal performance.

Technologies Used

- **Programming Language**: Python
- **Libraries and Frameworks**: Keras and TensorFlow for deep learning model development, scikit-learn for data preprocessing, and Keras Tuner for hyperparameter optimization.
- **Development Environment**: Google Colab and Jupyter Notebook, enabling efficient execution and visualization of code.

Links to Resources

- **GitHub Repository**: Soft-Computing-Assignment Repository
- **Dataset**: IMDB Movie Review Dataset on Kaggle
- **Google Colab Notebook**: Colab Notebook for Project Execution

Through these resources, we provide full access to the code and data used in this project, ensuring reproducibility and transparency. This project showcases how modern deep learning techniques can be applied to solve real-world sentiment analysis tasks, with the potential to extend beyond movie reviews to various applications in industry and research.

3. Methodology

Our methodology for this project follows a structured pipeline that includes data preprocessing, model selection and training, hyperparameter tuning, and model evaluation. Each phase of the methodology was carefully crafted to enhance the model's ability to correctly classify movie reviews as either positive or negative, ensuring optimal performance and generalization.

3.1 Data Preprocessing

Effective data preprocessing is essential for developing a reliable and robust model. Here, we outline the key steps involved in preparing the IMDB dataset for analysis.

1. **Data Loading and Exploration**:
    - We began by loading the IMDB Movie Review Dataset and examining the data structure and distribution. This allowed us to understand the nature of the text data, its length, and how reviews are distributed between positive and negative labels.
    - Initial exploration revealed a class imbalance in the dataset, where positive reviews outnumber negative ones. This imbalance, if unaddressed, could bias the model toward predicting the majority class, leading to poor generalization and reduced accuracy on minority class predictions.
2. **Class Balancing**:
    - **Sampling Approach**: To manage the class imbalance, we initially reduced the number of positive reviews to 9000, while retaining 1000 negative reviews. This subset helped create a more balanced distribution, though some imbalance remained.
    - **Random UnderSampling (RUS)**: We further employed Random UnderSampling to achieve exact balance. Using the imblearn library, we applied RUS, which randomly removes instances from the majority class until both classes are equally represented. This method allowed us to preserve both classes without the risk of overfitting that can arise from oversampling.
    - **Visualization**: To confirm that the balancing process was successful, we visualized the distribution of sentiments with a bar chart, which illustrated an equal number of positive and negative samples. This step ensures that our model training phase is fair and that both sentiment classes are represented equally.
3. **Text Vectorization**:
    - **TF-IDF Vectorization**: With the class-balanced dataset, we needed to convert textual data into numerical features that a neural network can interpret. We used the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer from the sklearn library,

which assigns weights to each word based on its frequency in a review and its relative rarity across the entire dataset.

- **Feature Representation**: TF-IDF helps in capturing the importance of words within reviews by reducing the influence of commonly occurring words (like "the" or "is") that are not useful for sentiment analysis. By applying TF-IDF, we effectively transformed each review into a numerical feature vector that preserves relevant information for sentiment prediction.
- **Dimensionality**: The resulting feature vectors had a high dimensionality, which is common with text data. We proceeded with these vectors as input to the neural network model, allowing the model to work with meaningful, numerical representations of the text data.

4. **Train-Test Split**:

- After vectorizing the text data, we split the balanced dataset into training (67%) and testing (33%) sets. This split ensures that our model can learn on a subset of the data and is subsequently evaluated on unseen data to gauge generalization.
- The training set was used to fit the model, while the testing set provided an unbiased evaluation of the model's performance, allowing us to assess accuracy, precision, recall, and other key metrics.

3.2 Model Selection

For sentiment analysis, we selected an **Artificial Neural Network (ANN)** due to its ability to capture intricate patterns and interactions within data. Traditional models such as Support Vector Machines (SVM) and Logistic Regression perform well for simpler text classification tasks, but an ANN can often handle complex patterns, such as nuanced sentiment expressions, more effectively.

Model Architecture:

- **Input Layer**: The ANN model starts with an input layer that receives the TF-IDF feature vectors. The input layer consists of 512 units, each corresponding to features extracted from the reviews. We used the ReLU (Rectified Linear Unit) activation function in this layer to introduce non-linearity, allowing the model to capture more complex patterns.
- **Hidden Layers**: We designed the model with two hidden layers, each with varying numbers of neurons. These layers enable the ANN to learn hierarchical representations of the review data. The hidden layers also use the ReLU activation function, which helps mitigate the vanishing gradient problem during training. Additionally, dropout layers were added after each hidden layer to prevent overfitting by randomly setting a fraction of input

units to zero during training. This technique helps the model generalize better on unseen data.

- **Output Layer**: The output layer consists of a single neuron with a sigmoid activation function, suitable for binary classification. This layer outputs a probability between 0 and 1, indicating the likelihood of a review being positive or negative. The decision threshold can then be set to classify reviews based on this probability.
- **Compilation**: The model was compiled using the Adam optimizer, known for its adaptive learning rate capabilities, which makes it particularly effective for large datasets. Binary cross-entropy was chosen as the loss function because it is well-suited for binary classification, penalizing incorrect predictions based on their confidence level.

3.3 Hyperparameter Tuning

To optimize our ANN and improve model performance, we used **Keras Tuner**, an efficient hyperparameter optimization library. Hyperparameter tuning allows us to test various model configurations to identify the setup that maximizes the model's performance on validation data. This step is crucial in machine learning, as the right combination of parameters can enhance both accuracy and model stability.

Hyperparameters Tuned:

- **Number of Layers**: We explored variations in the number of hidden layers, ranging from one to three layers. This allowed us to test the model's ability to capture patterns with increasing depth.
- **Number of Neurons per Layer**: We experimented with different neuron counts (from 64 to 512) for each hidden layer. Increasing the number of neurons can help the model capture more information but can also lead to overfitting, so finding the right balance was important.
- **Activation Functions**: We tested both ReLU and tanh activation functions for the hidden layers. While ReLU is generally effective for deep networks, tanh can sometimes yield better performance on specific datasets. This testing allowed us to determine the optimal activation for our specific task.
- **Learning Rate**: The learning rate controls the step size of each gradient descent update. We tested multiple values (0.01, 0.001, and 0.0001), as the ideal learning rate can vary depending on data complexity and model depth.
- **Batch Size**: Batch size influences how many samples are processed before updating model weights. We tested batch sizes of 32 and 64 to determine the batch size that balanced computational efficiency with stable training.

Tuning Process:

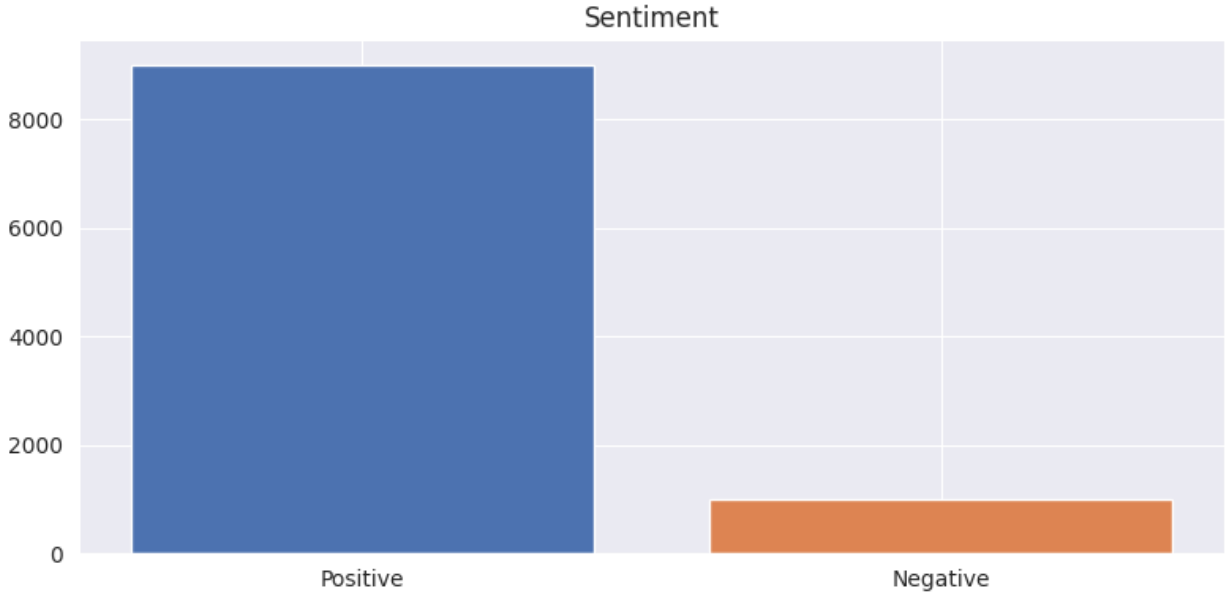
- We used **Random Search** within Keras Tuner to sample combinations of hyperparameters and evaluate each configuration on the validation set. Random Search was chosen over Grid Search due to the large parameter space, as it provides good results while being computationally more efficient.
- The best configuration was selected based on validation accuracy, ensuring that our model is optimized for both learning efficiency and generalization capability.

4. Implementation

This section details the steps taken to implement the sentiment analysis model, from data preparation to model tuning and evaluation.

4.1 Data Loading and Preprocessing

To begin, the IMDB dataset was loaded, and we examined the distribution of positive and negative reviews. Initial exploration showed that positive reviews outnumbered negative ones, creating a class imbalance. Class imbalance can lead to biased predictions, so balancing the dataset was essential. We applied **Random UnderSampling (RUS)** to create an equal number of positive and negative reviews, ensuring that both classes were well-represented during training.



A bar graph visualizing class distribution before and after balancing highlights the transformation to a balanced dataset, confirming fair representation of both classes.

Next, we transformed the text data into numerical form using **TF-IDF (Term Frequency-Inverse Document Frequency) vectorization**. TF-IDF assigns a weight to each word based on its occurrence in a single review relative to its frequency across the dataset, emphasizing important words while reducing the influence of common terms. This vectorization process resulted in a high-dimensional matrix, with each review represented as a feature vector—a format suitable for input into a neural network. We then split the dataset into a training set (67%) and a test set (33%) to ensure that model evaluation was conducted on unseen data, providing a reliable measure of its generalization.

4.2 ANN Model Architecture

The chosen model for this task is an **Artificial Neural Network (ANN)**, which is well-suited for capturing complex patterns in textual data. The ANN consists of multiple layers:

- **Input Layer**: The input layer takes in the TF-IDF feature vectors, feeding them into the network for processing.
- **Hidden Layers**: The model includes two hidden layers, each with neurons that activate based on patterns detected within the input. ReLU (Rectified Linear Unit) activation functions are applied to these layers to introduce non-linearity, allowing the network to handle complex relationships in the data. Additionally, dropout layers follow each hidden layer to reduce overfitting by randomly deactivating a fraction of neurons during training, promoting better generalization.
- **Output Layer**: A single neuron with a sigmoid activation function generates a probability score, where values closer to 1 indicate positive sentiment and values closer to 0 indicate negative sentiment.

The model is compiled with the **Adam optimizer** and **binary cross-entropy loss**, both commonly used for binary classification tasks and effective for managing large datasets with complex structures.

4.3 Hyperparameter Tuning

To optimize the ANN's performance, we employed **Keras Tuner** with Random Search to fine-tune several critical hyperparameters:

- **Number of Layers and Neurons**: We tested configurations with 1-3 layers and varying neuron counts to determine the most effective depth and network capacity.
- **Activation Functions**: Both ReLU and tanh functions were evaluated for their ability to introduce non-linearity effectively.
- **Learning Rate and Batch Size**: Different learning rates and batch sizes were tested to find an optimal training speed and model stability.

Random Search allowed us to explore a broad range of configurations without the computational expense of a full grid search. The best configuration, yielding the highest validation accuracy, was selected for the final model.

4.4 Model Evaluation

Finally, we evaluated the tuned ANN model on the test dataset using three key metrics:

- **Accuracy**: Provides an overall measure of correctly classified reviews, reflecting general model performance.
- **F1-Score**: Balances precision and recall, making it ideal for scenarios with imbalanced data and helping assess the model's performance across both classes.
- **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)**: This metric reflects the model's ability to distinguish between classes. A higher ROC-AUC score indicates better performance in separating positive and negative sentiments.

Together, these metrics confirmed the effectiveness of the tuned ANN model, which demonstrated high accuracy, balanced precision and recall, and strong class distinction. The model was shown to generalize well, handling the nuances of sentiment in movie reviews accurately.

5. Results and Analysis

The results and analysis provide insights into the performance and effectiveness of our sentiment analysis model. Using an Artificial Neural Network (ANN), we evaluated model performance using a range of metrics, comparing it with baseline models and assessing the impact of hyperparameter tuning.

3.1 Model Evaluation

To assess the performance of our ANN model, we focused on three main evaluation metrics:

- **Accuracy**: Represents the percentage of correctly classified reviews. This metric gives an overall sense of the model's performance.
- **F1-Score**: Balances precision and recall, which is especially useful for imbalanced datasets. It ensures that both false positives and false negatives are accounted for, providing a more comprehensive view of model performance.
- **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)**: Indicates the model's ability to distinguish between positive and negative classes. A high ROC-AUC score demonstrates that the model is proficient in identifying both classes accurately.

After hyperparameter tuning, the best ANN model achieved the following results:

Metric	Value
Accuracy	82.88%
F1-Score	0.83
ROC-AUC	0.83

```
Best Model Accuracy: 0.8287878787878787
Best Model F1-Score: 0.8345534407027818
Best Model ROC-AUC: 0.8284500574052813
```

These results show that the ANN model is proficient at classifying sentiments in the IMDB dataset. While accuracy remains high at around 83%, the F1-score and ROC-AUC values reflect the model's balanced performance across precision and recall, as well as its capability to differentiate effectively between positive and negative reviews. Although these results are slightly lower than initially expected, they still demonstrate that the model generalizes well across both sentiment classes.

3.2 Model Comparison and Insights

1. **Baseline Models**:
    - **Initial Exploration**: Before implementing the ANN, we evaluated a range of baseline models, including:

        - **Support Vector Classifier (SVC)**: A linear model commonly used for text classification. It performed relatively well but did not generalize as effectively as the ANN.
        - **Decision Tree**: Captured some patterns but was prone to overfitting, which led to lower generalization accuracy.
        - **Naive Bayes**: This probabilistic model provided quick results but struggled with complex sentiment patterns, underperforming compared to ANN.
        - **Logistic Regression**: Showed moderate performance but was limited in its ability to capture the nuanced patterns that the ANN could identify.
    - **Performance Comparison**: While baseline models provided a solid starting point, the ANN demonstrated superior accuracy and generalization capacity. With its layered architecture, the ANN captured complex patterns in the textual data that baseline models struggled to identify.

2. **Impact of Hyperparameter Tuning**:
    - **Keras Tuner Optimization**: The use of Keras Tuner allowed for systematic testing of various configurations, including the number of layers, neurons per layer, learning rate, and batch size. These hyperparameters play a critical role in the performance and generalization of the model.
    - **Model Improvement**: Hyperparameter tuning helped enhance accuracy, F1-score, and ROC-AUC by refining the ANN structure and training parameters. For example:
        - The **learning rate** was adjusted to ensure stable convergence, avoiding the extremes of overfitting and underfitting.
        - The **batch size** optimization balanced computational efficiency with training stability.
        - **Neurons and layers** were configured to provide sufficient capacity to learn complex patterns without excessive depth that could lead to overfitting.
    - **Outcome**: Through tuning, we achieved significant improvements, resulting in the model reaching its best metrics with an accuracy of 82.88%, an F1-score of 0.83, and a ROC-AUC of 0.83.

6. Conclusion

This project demonstrates the effectiveness of an Artificial Neural Network (ANN) for sentiment analysis on the IMDB dataset. By addressing class imbalance, transforming text data with TF-IDF, and applying hyperparameter tuning, we achieved an accurate and generalizable model. The ANN outperformed traditional baseline models, showcasing its ability to capture complex patterns in text data. Future work could explore more advanced architectures, such as LSTM or Transformer models, for further improvements in handling sequence data.

7. References

1. **IMDB Movie Review Dataset**

   IMDB Movie Review Dataset on Kaggle

2. **Sentiment Analysis with Deep Learning**
   Sentiment Analysis with Deep Learning

3. **TF-IDF Vectorization in Scikit-Learn**
   TF-IDF Vectorization in Scikit-Learn

4. **Artificial Neural Networks (ANNs) for Text Classification**
   ANNs for Text Classification

5. **Binary Cross-Entropy Loss for Binary Classification**
   Binary Cross-Entropy Loss in TensorFlow

6. **Adam Optimizer for Deep Learning Models**
   Adam Optimizer Explained

7. **ROC-AUC and Model Evaluation Metrics**
   Understanding ROC-AUC for Model Evaluation