

Sentiment Analysis on IMDB Movie Review Dataset Using Artificial Neural Networks

ASSIGNMENT – SOFT COMPUTING

SUBMITTED BY

Rakshit Gupta
2021a1r050



SUBMITTED TO

Dr. Surbhi Gupta

Assistant Professor

Department of Computer Science & Engineering

Model Institute of Engineering and Technology

Table of Contents

1. Introduction	1
Problem Statement.....	1
Dataset Overview	1
2. Methodology	2
2.1 Data Preprocessing	2
2.2 Model Selection	3
2.3 Hyperparameter Tuning	4
3. Results and Analysis.....	5
3.1 Model Evaluation	5
3.2 Model Comparison and Insights	6
4. Solutions Implemented	7
4.1 Data Preprocessing Solutions	7
4.2 Model Selection and Development Solutions	7
4.4 Model Evaluation and Comparison	9

1. Introduction

Problem Statement

Sentiment analysis is a significant task in natural language processing (NLP) that involves classifying textual content based on its underlying sentiment—often as "positive," "negative," or "neutral." In this project, we tackle sentiment analysis on a set of movie reviews from IMDB, where each review is labeled either as positive or negative. The goal is to use an Artificial Neural Network (ANN) to automatically classify each review based on the sentiment expressed within the text.

The insights generated from sentiment analysis are widely applicable across various fields. For instance, in marketing, companies leverage sentiment analysis to understand customer feedback on products. On social media platforms, sentiment analysis tools help monitor public opinion, allowing companies and organizations to track brand reputation in real-time. Similarly, sentiment analysis assists in customer service by identifying the tone of customer communications, enabling more personalized responses.

In this project, we aim to apply an ANN model to classify the IMDB reviews with high accuracy and reliability. The selection of ANN, as opposed to traditional machine learning models, allows us to capture complex patterns in text data and improve classification performance. Our approach involves pre-processing the text data, balancing the dataset, and using advanced techniques like hyperparameter tuning to optimize the model.

Dataset Overview

The IMDB Movie Review Dataset, used in this project, consists of thousands of movie reviews collected from IMDB, each labeled with a "positive" or "negative" sentiment. A common challenge with real-world datasets is class imbalance, where one class significantly outnumbers the other. This dataset is no exception, as it generally contains more positive than negative reviews. This imbalance can potentially bias the model, leading it to favor the majority class in its predictions. To address this, we performed class balancing using under sampling techniques, creating a more representative and fair dataset for training.

The dataset provides a broad variety of linguistic structures, expressions, and sentiments, offering a rich set of challenges for a sentiment classification model. Reviews in the dataset contain subtle variations in language and sentiment expressions, which necessitates robust text representation and a well-optimized model for accurate classification.

The project was conducted on Kaggle, where the complete code and notebook are available at the following link: [Sentiment Analysis on IMDB Movie Reviews Notebook](#).

2. 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.

2.1 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.

2.2 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.

2.3 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.

3. 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.

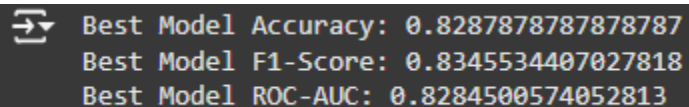
3.1 Model Evaluation

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.

4. Solutions Implemented

In order to develop an effective sentiment analysis model for the IMDB Movie Review dataset, we implemented a structured approach encompassing data processing, model building, and optimization. Here's an outline of each solution component and the methods applied:

4.1 Data Preprocessing Solutions

1. Class Balancing:

- **Problem:** The dataset was imbalanced, with a higher number of positive reviews than negative ones. Class imbalance can skew model predictions towards the majority class, which is undesirable in sentiment analysis.
- **Solution:** We used an undersampling technique. Specifically, we reduced the dataset to a balanced sample of 9000 positive and 1000 negative reviews and further applied **Random UnderSampling (RUS)** to achieve an equal distribution of positive and negative reviews in the dataset. This step ensured fair representation of both classes, leading to improved model reliability.

2. Text Vectorization:

- **Problem:** Text data is inherently unstructured and cannot be fed directly into a neural network.
- **Solution:** We applied **Term Frequency-Inverse Document Frequency (TF-IDF) vectorization** to convert text data into numerical features, emphasizing the most informative words in each review. This approach allows the model to focus on words that are unique to each review, enhancing its ability to learn from meaningful text patterns.

3. Train-Test Split:

- **Problem:** To evaluate the model fairly, we needed separate data for training and testing.
- **Solution:** We split the balanced dataset into a **training set (67%)** and a **test set (33%)**. This split provided a reliable basis for model training while preserving sufficient data for an unbiased evaluation.

4.2 Model Selection and Development Solutions

1. ANN Architecture for Sentiment Analysis:

- **Problem:** Traditional models like SVC, Naive Bayes, and Decision Trees were not sufficiently capturing the nuanced patterns in the data, leading to limited performance.
- **Solution:** We chose an **Artificial Neural Network (ANN)** due to its capability to learn complex patterns. The ANN architecture was designed as follows:
 - **Input Layer:** Received TF-IDF vectorized data as input with 512 units and ReLU activation.
 - **Hidden Layers:** Two fully connected layers with ReLU activation, each followed by dropout layers to prevent overfitting.
 - **Output Layer:** A single neuron with a sigmoid activation for binary classification.
- This structure allowed the model to learn intricate patterns within the text, resulting in superior performance over baseline models.

2. Training Configuration:

- **Problem:** Choosing an appropriate optimizer and loss function is critical in binary classification to ensure effective learning.
- **Solution:** We configured the ANN with:
 - **Optimizer:** Adam, which is well-suited for text data and complex networks due to its adaptive learning rate.
 - **Loss Function:** Binary Cross-Entropy, ideal for binary sentiment classification tasks, to minimize classification errors and improve prediction accuracy.

4.3 Hyperparameter Tuning Solutions

1. Keras Tuner for Optimization:

- **Problem:** To maximize model performance, we needed to find the optimal configuration of the network's hyperparameters.
- **Solution:** We employed **Keras Tuner with Random Search** to optimize critical hyperparameters:
 - **Number of Layers:** Tested 1-3 hidden layers to assess whether deeper networks improved accuracy.
 - **Neurons per Layer:** Ranged from 64 to 512 units to identify the right balance of complexity and model stability.

- **Activation Functions:** Compared ReLU and tanh to maximize learning efficiency.
- **Learning Rate:** Tuned values like 0.01, 0.001, and 0.0001 to ensure an optimal convergence rate.
- **Batch Size:** Evaluated 32 and 64 to optimize the training batch size and balance memory usage.
- This tuning process significantly boosted the model's performance by optimizing layer configurations, neuron counts, and other key parameters, ensuring better accuracy, F1-score, and ROC-AUC.

4.4 Model Evaluation and Comparison

1. Model Evaluation Metrics:

- **Problem:** Accurately assessing model performance requires robust metrics to measure its classification capability.
- **Solution:** We used **Accuracy**, **F1-Score**, and **ROC-AUC** metrics to evaluate the model. These metrics provided a comprehensive view of the model's effectiveness, particularly its balance between precision and recall (F1-score) and ability to distinguish between classes (ROC-AUC).

2. Comparison with Baseline Models:

- **Problem:** Understanding the value of ANN required comparing it with baseline models.
- **Solution:** We initially tested baseline models, including **Support Vector Classifier (SVC)**, **Decision Tree**, **Naive Bayes**, and **Logistic Regression**. Although these models performed reasonably, the ANN outperformed them, highlighting its ability to capture complex patterns in text data, demonstrating its superior accuracy and generalization.