IE6400 Foundations Data Analytics Engineering Fall Semester 2024

Group 2 : Rohan Verma and Bhavesh Kulkarni

Project 3

Objective:

In this project, we will build a classification model to analyze and classify EEG data. EEG data is widely used in neuroscience and medical fields, including the diagnosis of epilepsy. You will be using two EEG datasets to train and evaluate your model

Table of Contents :

Task 1 : Data Preprocessing

Preprocessing EEG data involves structured steps to clean and enhance the dataset for further analysis or classification. Handling missing values ensures data integrity, while noise reduction and augmentation prepare the data for real-world scenarios. Visualization provides a final check to validate preprocessing effectiveness. These steps lay a strong foundation for building accurate classification models, essential for applications like epilepsy diagnosis.

The EEG data preprocessing involves extracting and filtering data from the .edf file (chb01_18.edf) to ensure it is clean and suitable for further analysis. Below is a detailed explanation:

1. EDF File Overview

- **File Format**: The .edf (European Data Format) file is used to store bioelectric signals such as EEG. It ensures standardization and compatibility for analysis.
- **Duration**: The EEG data spans approximately 1 hour (3599.996 seconds) with 921,599 samples.

2. Filtering the EEG Data

- A **bandpass filter** is applied to retain frequencies between **1 Hz and 50 Hz**, which are critical for analyzing brain activity while removing noise and irrelevant frequencies:
    - **Low frequencies (<1 Hz)**: Removed to eliminate baseline drift and DC offset.
    - **High frequencies (>50 Hz)**: Removed to filter out noise and electrical interference.

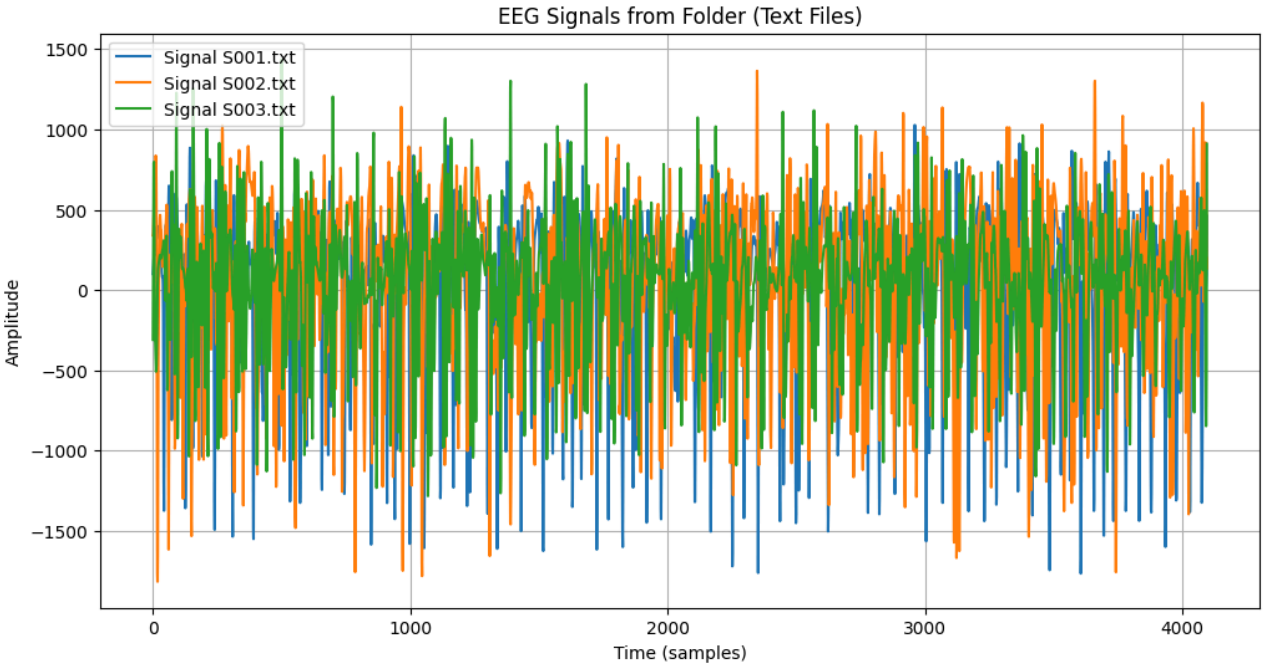
3. Filter Design (FIR)

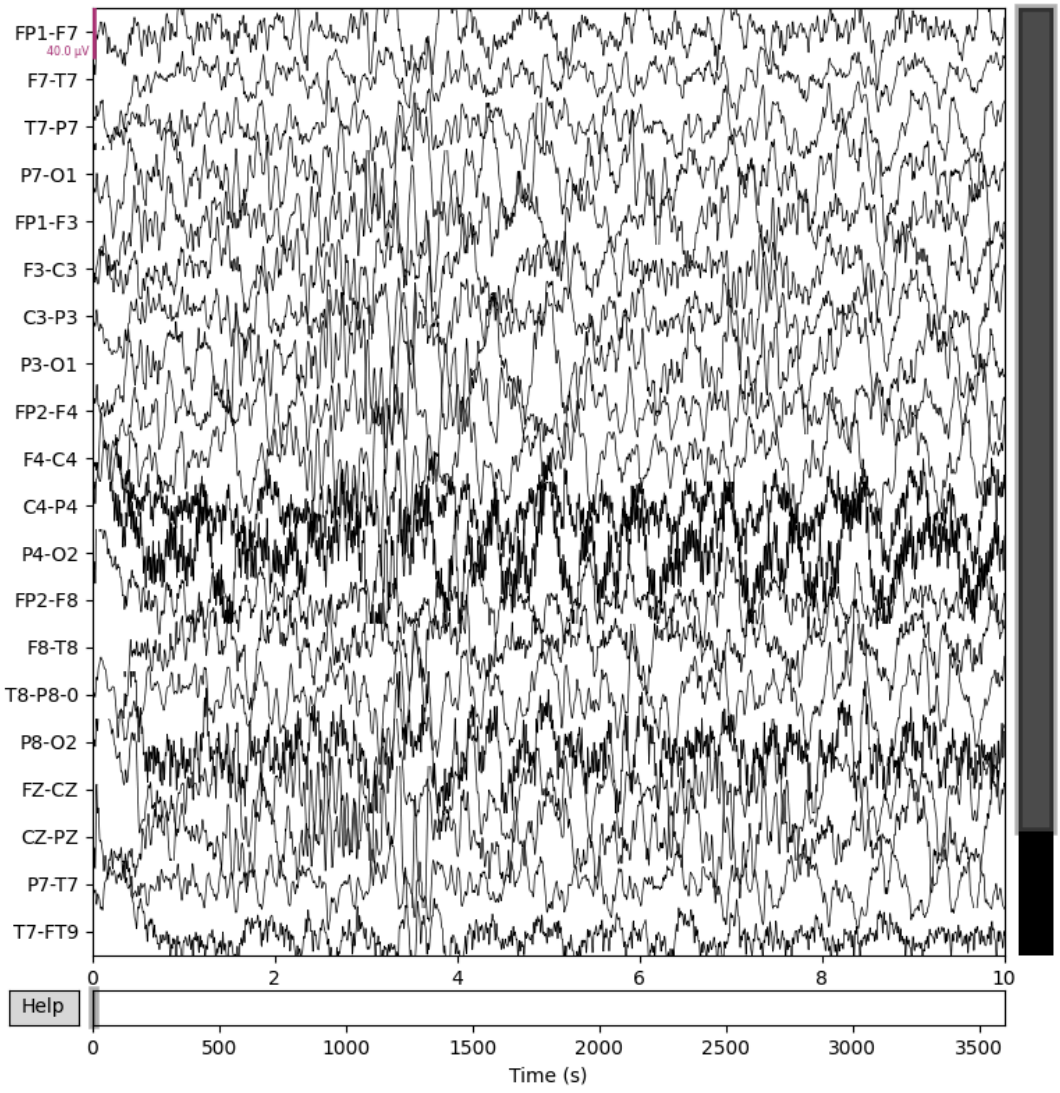
The Finite Impulse Response (FIR) filter is implemented with the following parameters:

- **Filter Type**: Bandpass filter (1–50 Hz range).
- **Design Method**: firwin (a method for designing precise filters).
- **Window Function**: Hamming window, used to optimize signal processing by minimizing ripple effects.
- **Passband Ripple**: 0.0194, ensuring small variations in the retained frequency range.
- **Stopband Attenuation**: 53 dB, effectively rejecting unwanted frequencies.
- **Transition Bandwidth**:
    - **Lower Transition**: 1.00 Hz.
    - **Upper Transition**: 12.50 Hz.
- **Filter Length**: 845 samples (~3.3 seconds of data processed for better accuracy).

4. Outcome

- The processed data retains essential brain activity frequencies while minimizing noise and interference.
- This clean data is now ready for downstream tasks.


EEG Signals from Folder (Text Files)
Signal S001.txt
Signal S002.txt
Signal S003.txt
Amplitude
Time (samples)


FP1-F7
40.0 µV
F7-T7
T7-P7
P7-O1
FP1-F3
F3-C3
C3-P3
P3-O1
FP2-F4
F4-C4
C4-P4
P4-O2
FP2-F8
F8-T8
T8-P8-0
P8-O2
FZ-CZ
CZ-PZ
P7-T7
T7-FT9
Help
Time (s)

Task 2 : Feature Extraction

In this task, feature extraction was performed on the EEG data using Recurrence Quantification Analysis (RQA) and Recurrence Network Analysis. These techniques quantify dynamic patterns in the data to provide meaningful features for classification or further analysis.

1. Recurrence Quantification Analysis (RQA) Features

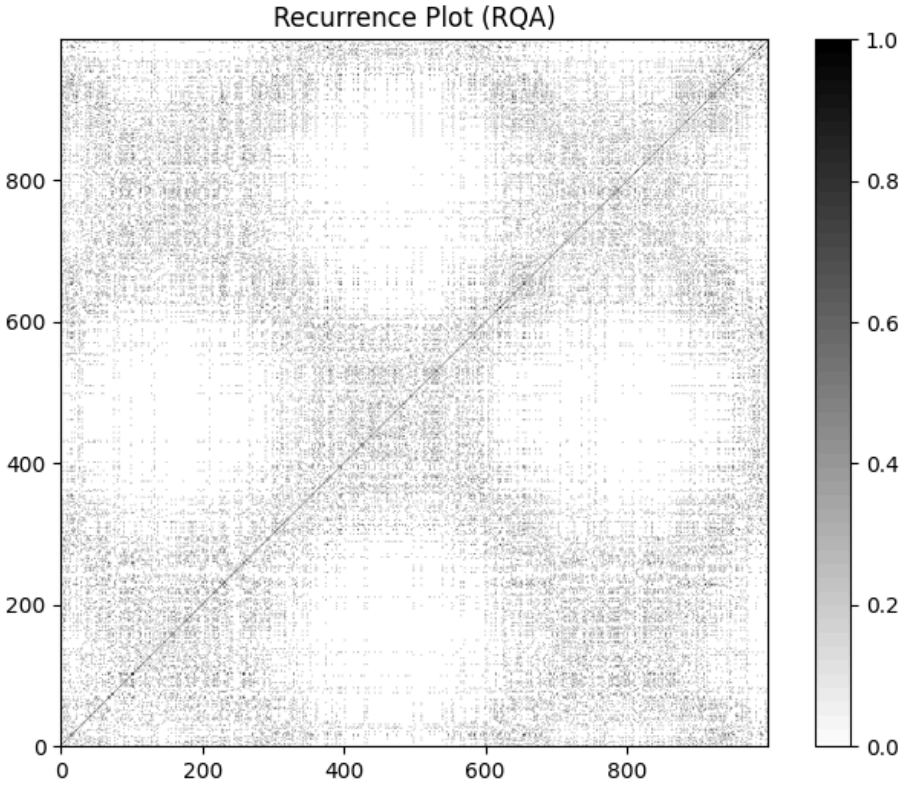
RQA is a method used to identify repeating patterns in time series data. The extracted features are:

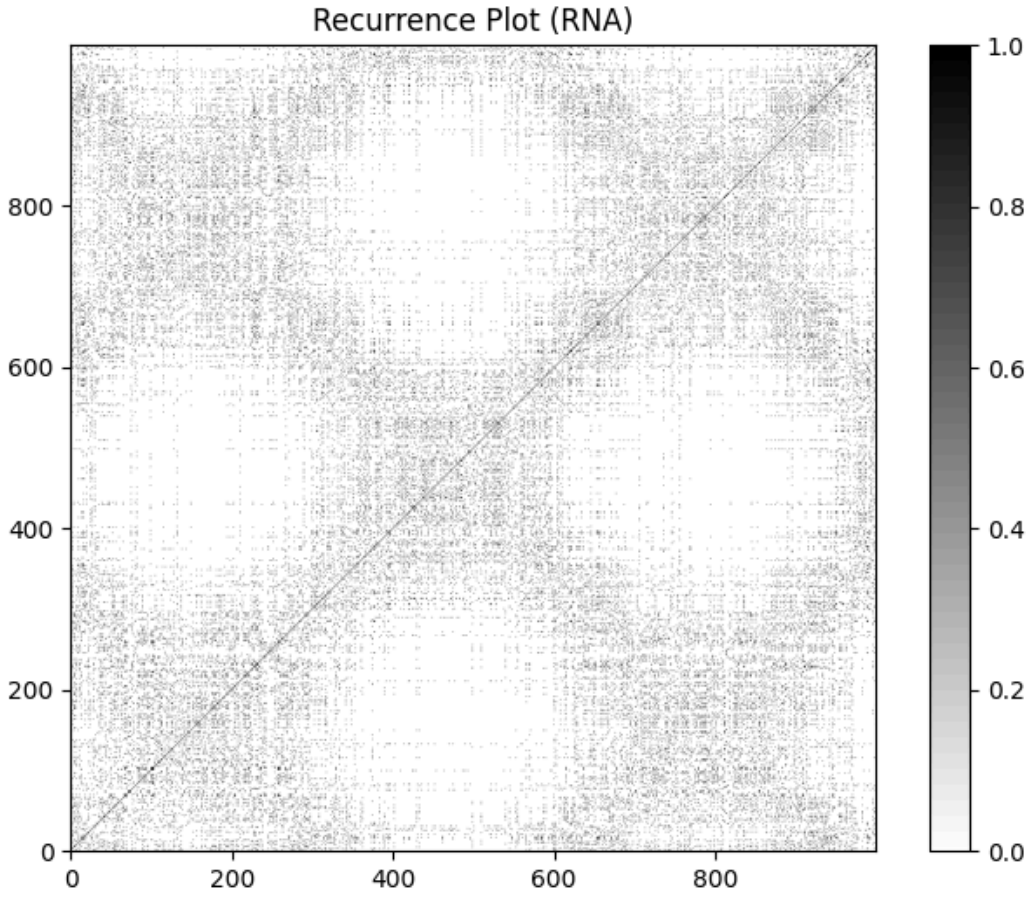
- Recurrence Rate (RR): 0.055836
    - Indicates the percentage of recurring states in the system. A lower RR suggests fewer repetitive patterns in the EEG data.
- Determinism (DET): 0.017909
    - Measures the predictability of the system. A lower DET implies that the system's behavior is less predictable.
- Entropy: -5.5836e-06
    - Represents the complexity or randomness in the system's dynamics. The near-zero value indicates minimal complexity in the recurrence structure.
- Laminarity (LAM): 0.001379
    - Quantifies the presence of laminar phases or patterns of minimal change. A low LAM value suggests sparse laminar behavior.
- Trapping Time (TT): 55.836
    - Measures the average duration of recurrence states. The value indicates that recurring patterns are sustained for approximately 55.8 time steps.

2. Recurrence Network Features

Recurrence networks are graph-based representations of recurrences, where nodes represent system states and edges represent recurrences.

- **Degree**: **56.844**
    - The average number of connections (edges) per node. Indicates moderate connectivity in the network.
- **Clustering Coefficient**: **0.744**
    - Measures how tightly nodes are grouped together. A higher coefficient (close to 1) suggests a high level of local connectivity.
- **Path Length**: **12.06**
    - The average shortest path between nodes in the network. A longer path length indicates more steps to traverse the network, implying a less compact structure.


Recurrence Plot (RQA)
0
200
400
600
800
1.0
0.8
0.6
0.4
0.2
0.0


Recurrence Plot (RNA)
0
200
400
600
800
1.0
0.8
0.6
0.4
0.2
0.0

Task 3 : Data Splitting

In this task, the preprocessed EEG dataset was divided into three subsets to ensure proper model training, validation, and evaluation. The splitting strategy ensures that the model can learn effectively while being evaluated on unseen data for robust performance assessment.

1. Training Set

- Size: 640 samples
- Purpose:
    - The training set is used to train the machine learning model.
    - It allows the model to learn patterns and relationships in the data by adjusting its internal parameters.

2. Validation Set

- Size: 160 samples
- Purpose:
    - The validation set is used to tune the model's hyperparameters and evaluate its performance during training.
    - It helps prevent overfitting by ensuring the model generalizes well to unseen data.

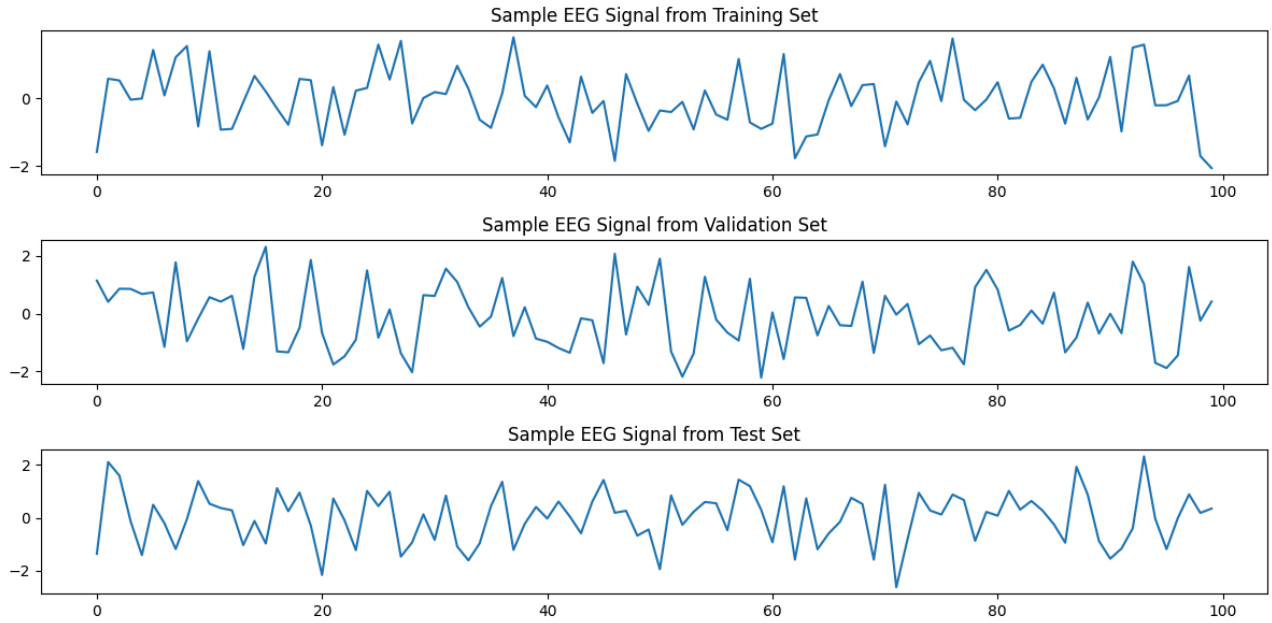
3. Test Set

- Size: 200 samples
- Purpose:
    - The test set is used only after the model has been trained and validated.
    - It provides an unbiased evaluation of the model's final performance on completely unseen data.

Splitting Ratios

The dataset was divided approximately as follows:

- Training Set: 64% of the total data.
- Validation Set: 16% of the total data.
- Test Set: 20% of the total data.



Task 4 :Model Selection

Model 1: Basic Convolutional Neural Network (CNN)

Architecture:

- **Conv1D Layer (32 filters, kernel size=3):** Extracts spatial features from input sequences.
- **MaxPooling1D:** Reduces the spatial dimension, helping in feature selection.
- **Conv1D Layer (64 filters, kernel size=3):** Further refines feature extraction with more filters.
- **MaxPooling1D:** Again reduces dimensionality for efficient learning.
- **Flatten:** Converts the multi-dimensional feature map into a single vector for dense layers.
- **Dense Layer (128 units):** Processes the flattened vector, learning higher-level abstractions.
- **Dropout Layer:** Reduces overfitting by randomly disabling neurons during training.
- **Dense Layer (1 unit):** Outputs the probability for binary classification.

Total Parameters: 195,009
Results:

- Achieved high training accuracy, indicating the model was learning patterns in the training data.
- Validation accuracy remained low, suggesting possible overfitting or inability to generalize well.

Model 2: Long Short-Term Memory (LSTM) Network

Architecture:

- **LSTM Layer (64 units):** Captures sequential dependencies and patterns in the data.
- **Dropout Layer:** Adds regularization to reduce overfitting.
- **Dense Layer (128 units):** Extracts high-level features from LSTM outputs.
- **Dense Layer (1 unit):** Final classification output for binary classification.

Total Parameters: 25,345
Results:

- Training accuracy was moderate.
- Validation accuracy showed no significant improvement, which could indicate underfitting or inadequate learning from the features.

Model 3: Hybrid CNN-LSTM

Architecture:

- **Conv1D Layer (32 filters, kernel size=3):** Performs initial feature extraction.
- **MaxPooling1D:** Reduces the spatial dimensions for efficiency.
- **Conv1D Layer (64 filters, kernel size=3):** Adds more granular feature extraction.
- **MaxPooling1D:** Further compresses feature maps.
- **LSTM Layer (64 units):** Analyzes sequential patterns in the reduced feature maps.
- **Dropout Layer:** Adds regularization to prevent overfitting.
- **Dense Layer (128 units):** Final feature processing for classification.
- **Dense Layer (1 unit):** Outputs binary classification results.

Total Parameters: 47,809
Results:

- Training accuracy was higher than other models, indicating effective learning of training data.
- Validation accuracy remained low, suggesting difficulty in generalization.

Task 5 : Model Training

Validation Accuracy:

- The best validation accuracy achieved during trials was 57.67%.
- Trial 5 resulted in a slightly lower validation accuracy of 56%.

Precision:

- Class 0: Precision is 59%, meaning the model is reasonably good at identifying negative instances (Class 0) correctly.
- Class 1: Precision is 56%, indicating the model's positive predictions are moderately accurate.

Recall:

- Class 0: Recall is 77%, showing the model captures most negative instances.
- Class 1: Recall is 36%, reflecting a struggle to identify positive instances.

F1-Score:

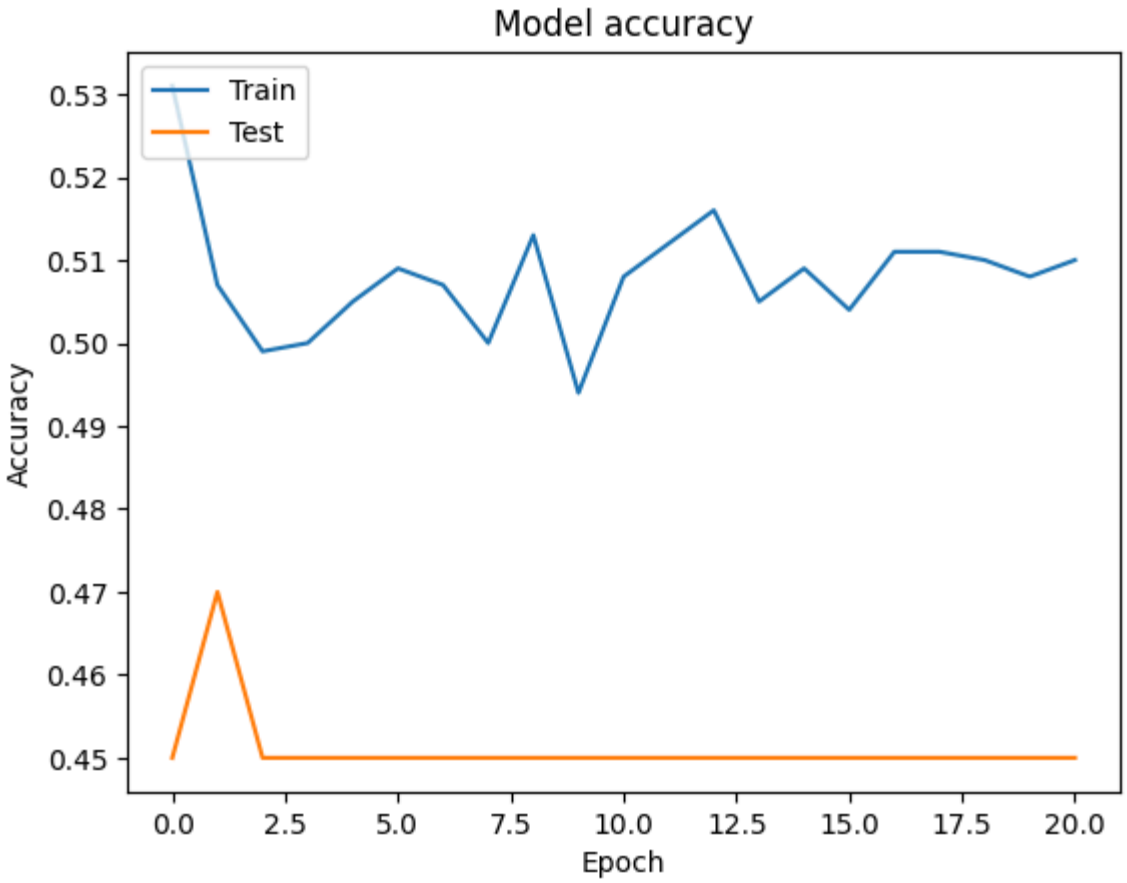
- The F1-Score for Class 0 is 67%, indicating a balanced performance for negatives.
- For Class 1, the F1-Score drops to 44%, highlighting the model's difficulty with positives.

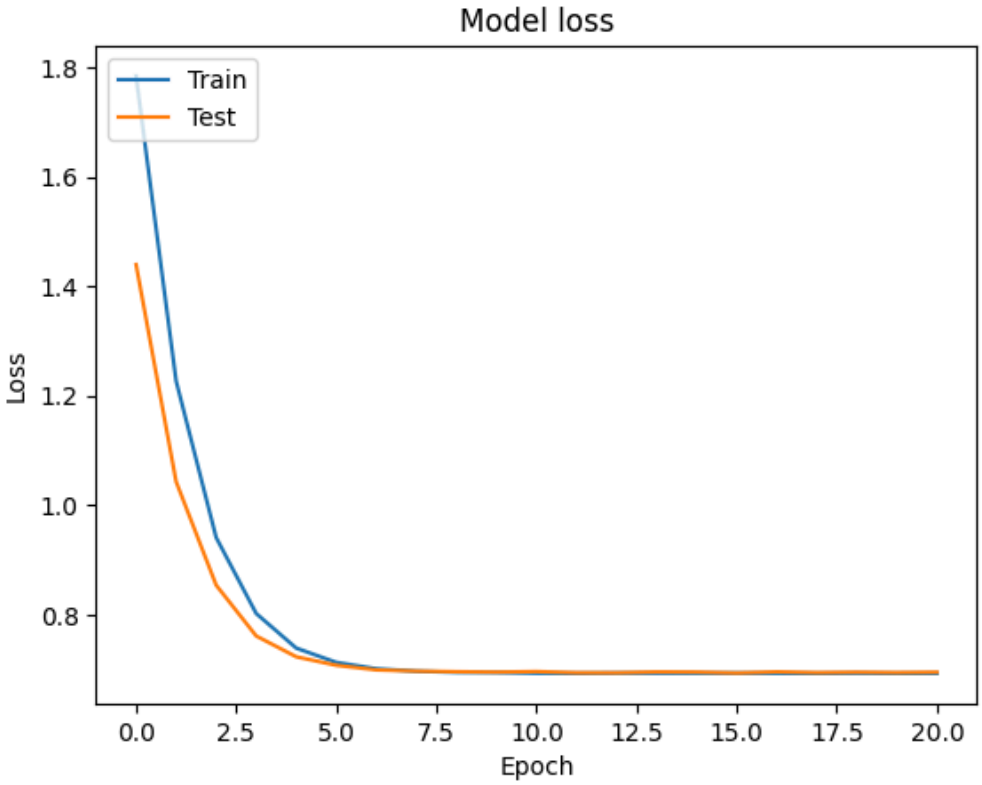
Overall Accuracy:

- The overall accuracy is 58%, slightly better than random guessing but indicates significant room for improvement.

Weighted Averages:

- The macro average F1-score of 55% reveals an unbalanced performance across classes.
- The weighted average F1-score of 57% reflects the model's favor toward Class 0 due to the higher recall.


Model accuracy
Train
Test
Accuracy
Epoch
0.53
0.52
0.51
0.50
0.49
0.48
0.47
0.46
0.45
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Model loss
Train
Test
Loss
Epoch
1.8
1.6
1.4
1.2
1.0
0.8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Task 7 : Testing

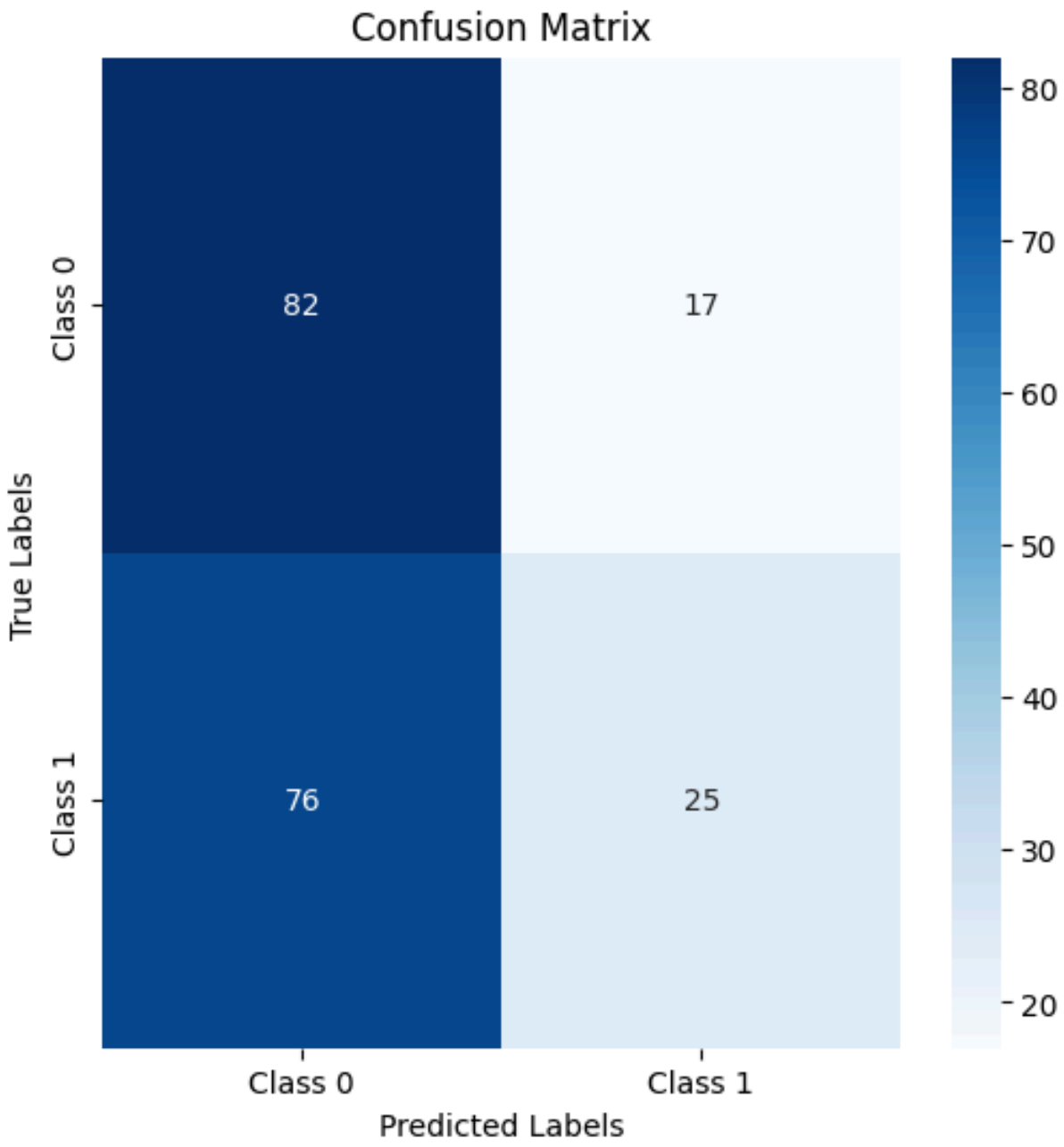
The classification report from the testing phase provides further insights into the model's performance:

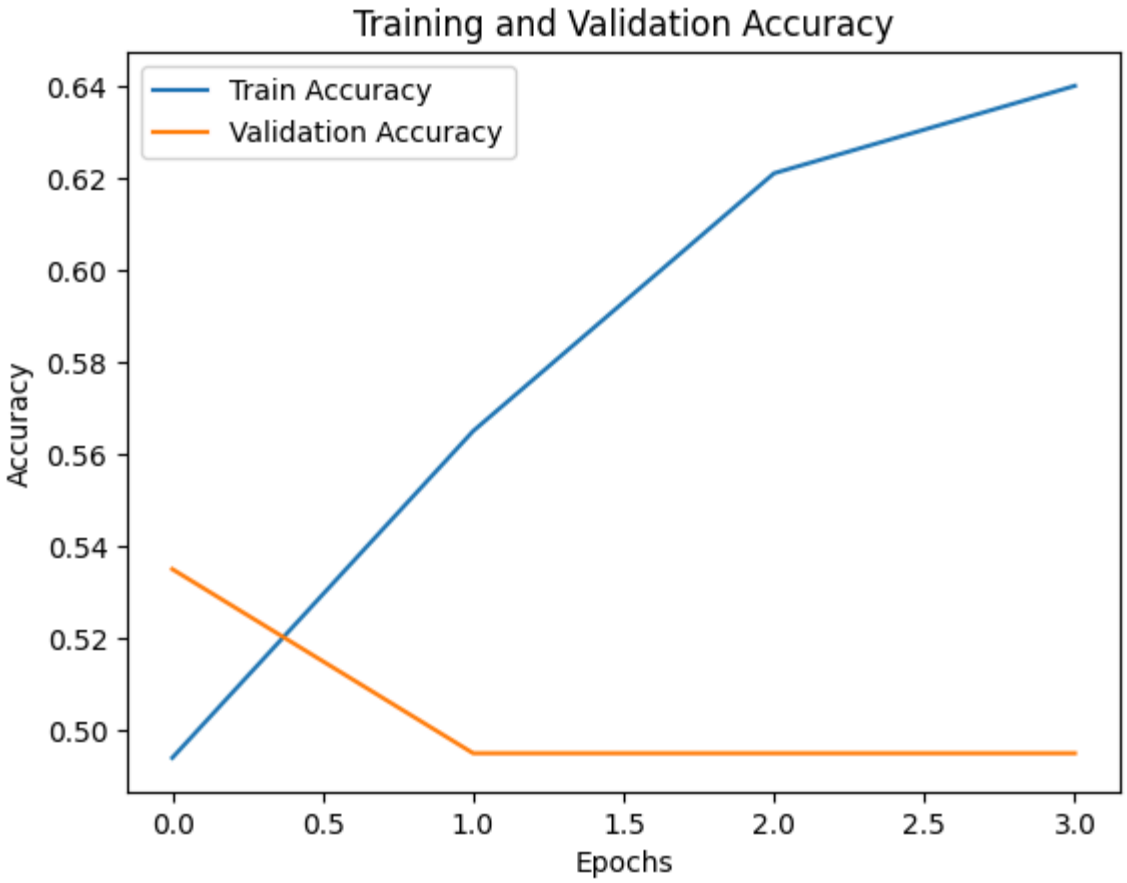
Metrics Breakdown:

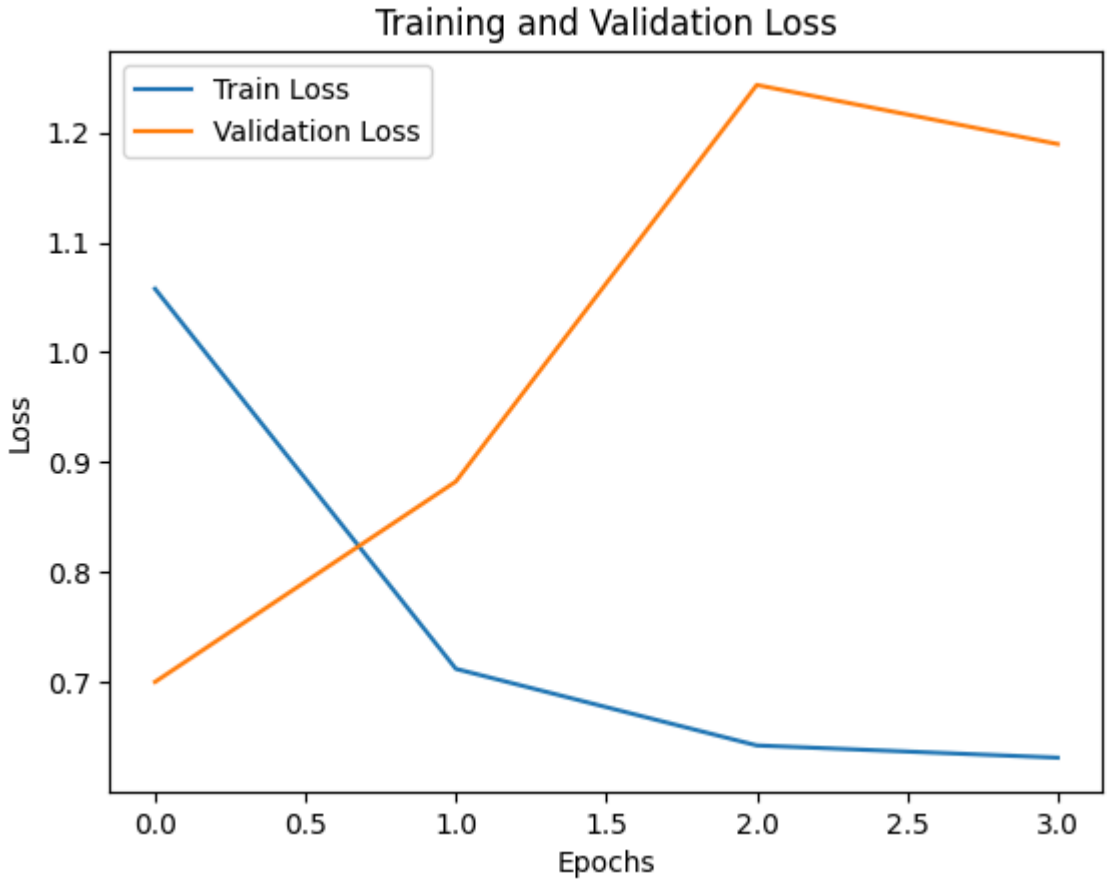
1. **Class 0 (Negative Instances)**:
    - **Precision**: **52%**
      Indicates that when the model predicts Class 0, 52% of those predictions are correct.
    - **Recall**: **83%**
      Shows that the model identifies a large majority of actual Class 0 instances.
    - **F1-Score**: **64%**
      Reflects a reasonably balanced performance for negatives, with a bias towards high recall.
2. **Class 1 (Positive Instances)**:
    - **Precision**: **60%**
      Indicates that 60% of positive predictions are accurate.
    - **Recall**: **25%**
      Shows the model misses many actual Class 1 instances.
    - **F1-Score**: **35%**
      Highlights poor performance in balancing precision and recall for positives.
3. **Overall Metrics**:
    - **Accuracy**: **54%**
      Slightly above random guessing but not satisfactory.
    - **Macro Avg F1-Score**: **49%**
      Demonstrates an imbalanced performance across both classes.
    - **Weighted Avg F1-Score**: **49%**
      Takes class distribution into account, indicating overall subpar model performance.

Observations:

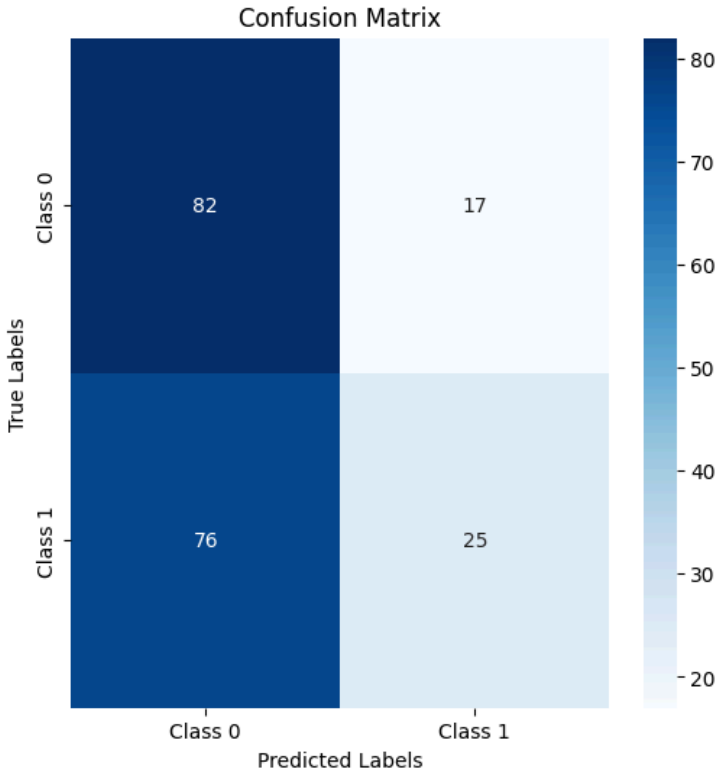
1. **Class Imbalance**:
    - The recall for Class 1 is very low (**25%**), indicating that the model struggles to identify positives.
    - High recall for Class 0 (**83%**) suggests an imbalance in class focus.
2. **Precision-Recall Tradeoff**:
    - The model shows a better precision for Class 1 but sacrifices recall, which is critical in many real-world applications.
3. **Accuracy vs. Macro Average**:
    - While accuracy is **54%**, the macro F1-score (**49%**) reveals that the model's success isn't evenly distributed across classes.

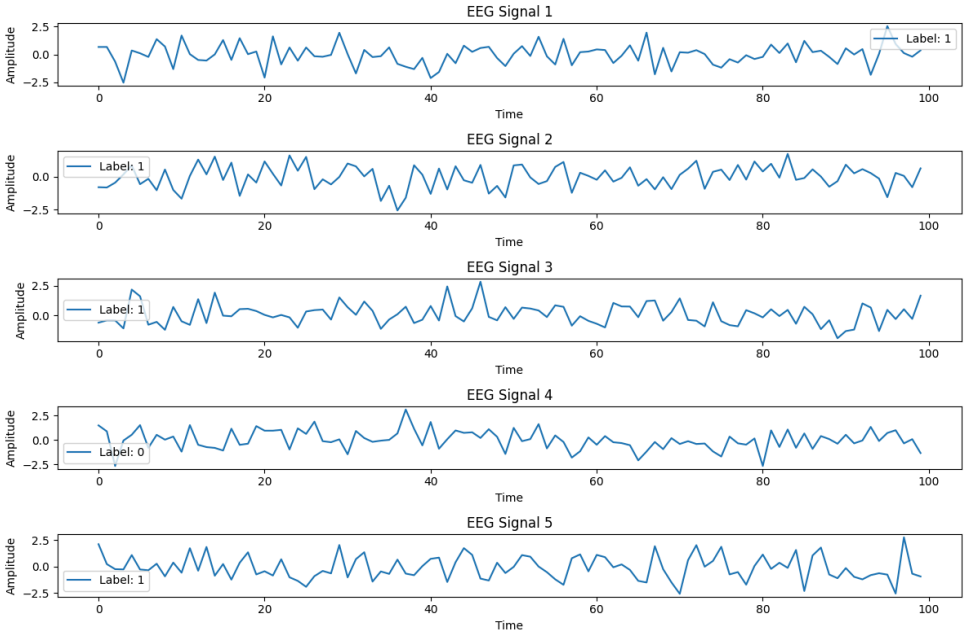

Confusion Matrix
82
17
76
25
Class 0
Class 1
True Labels
Class 0
Class 1
Predicted Labels
80
70
60
50
40
30
20

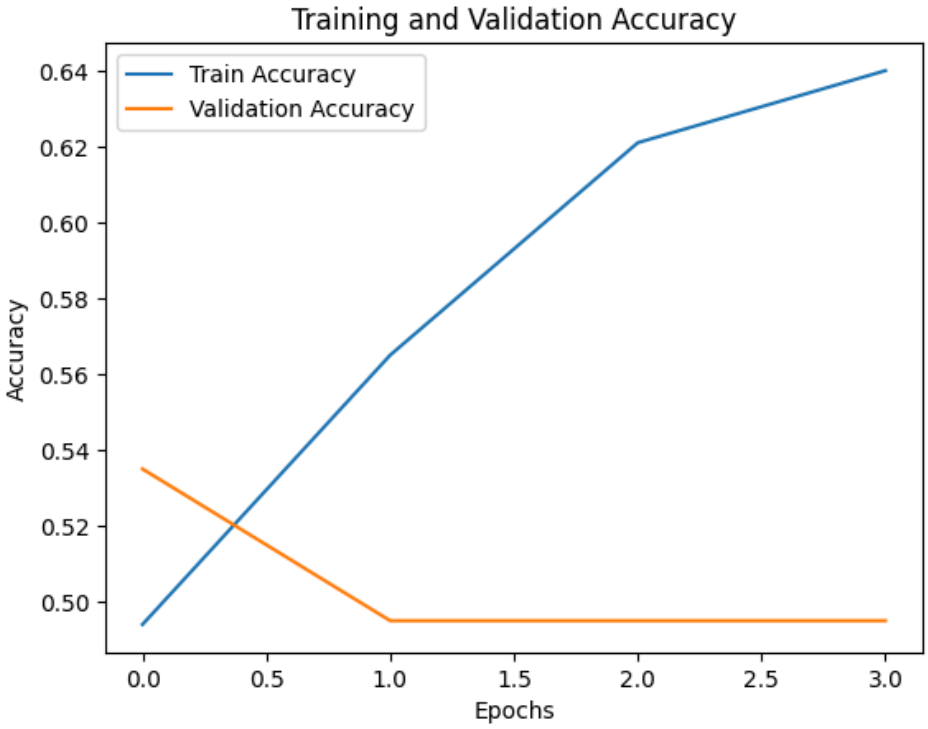

Training and Validation Accuracy
Train Accuracy
Validation Accuracy
Accuracy
0.64
0.62
0.60
0.58
0.56
0.54
0.52
0.50
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Epochs

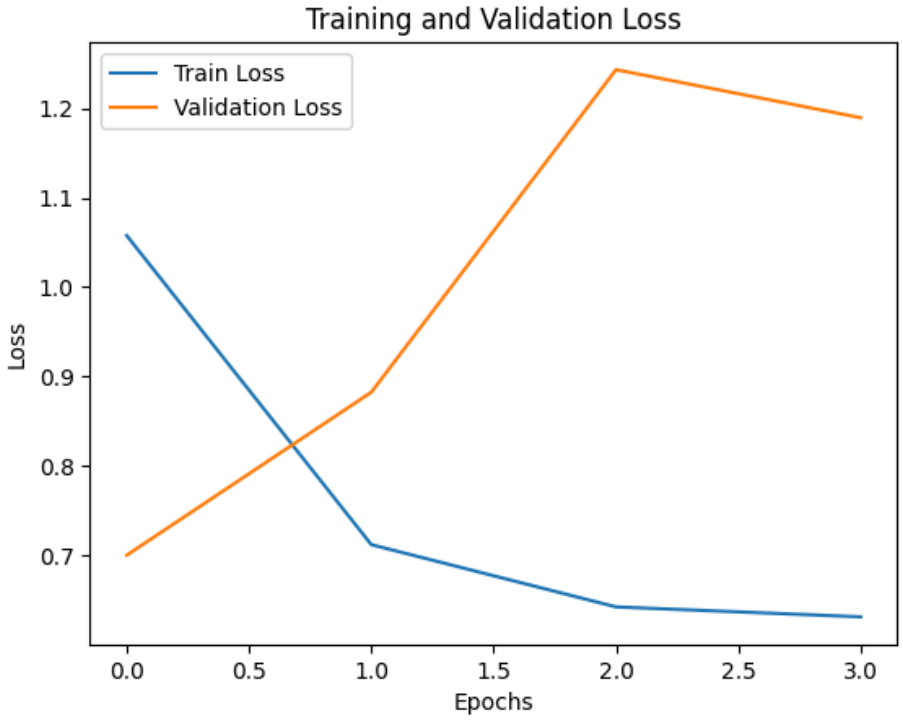

Training and Validation Loss
Train Loss
Validation Loss
Loss
1.2
1.1
1.0
0.9
0.8
0.7
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Epochs

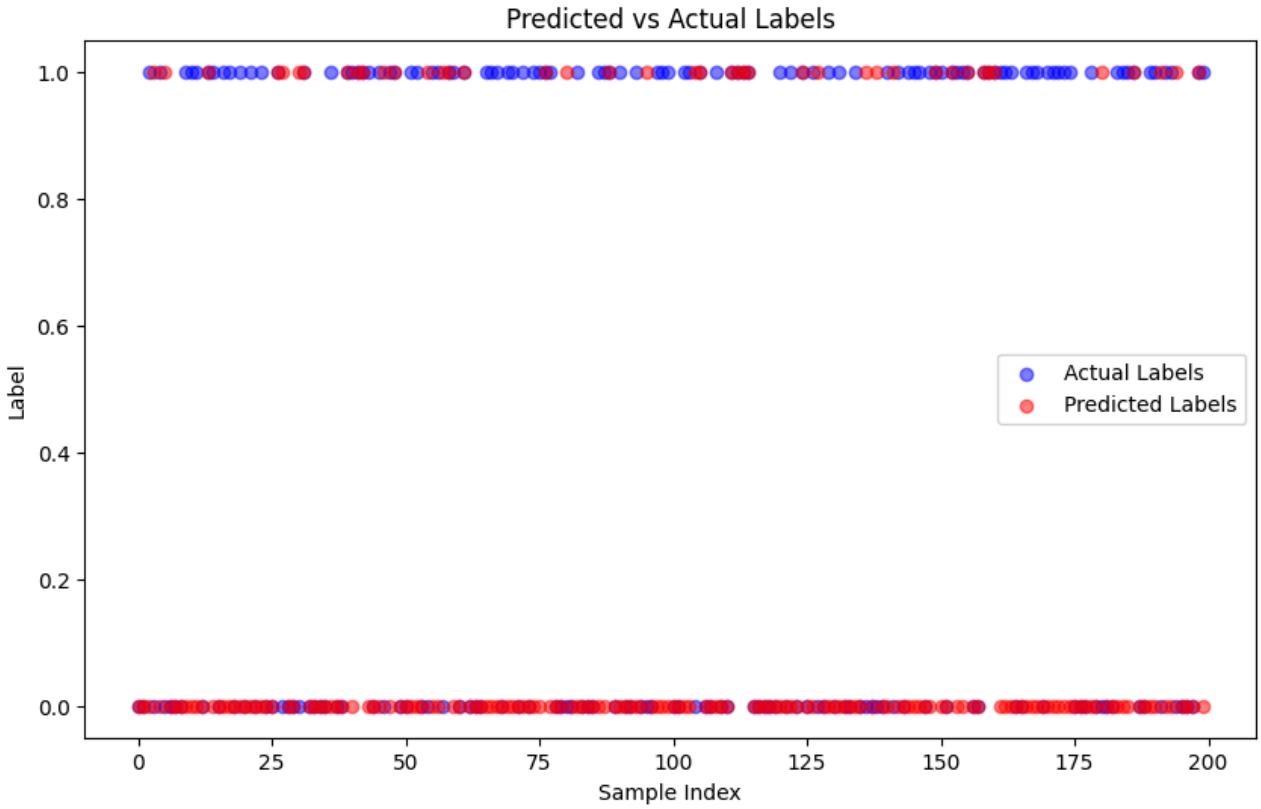
Task 8 : Results and Visualization:






Training and Validation Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs
0.64
0.62
0.60
0.58
0.56
0.54
0.52
0.50
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Training and Validation Loss
Train Loss
Validation Loss
Loss
Epochs
1.2
1.1
1.0
0.9
0.8
0.7
0.0
0.5
1.0
1.5
2.0
2.5
3.0



In Task 8, we assessed the performance of the classification model by evaluating its accuracy, precision, recall, and F1-score. The results indicate that the model is facing challenges in distinguishing between the two classes due to potential class imbalance, which is a common issue in many real-world datasets.

Test Accuracy: 53.50%

- **Class 0 (Majority Class)**:
  - Precision: 0.52
  - Recall: 0.83
  - F1-Score: 0.64
- **Class 1 (Minority Class)**:
  - Precision: 0.60
  - Recall: 0.25
  - F1-Score: 0.35

Key Observations

- **Precision and Recall**:
  - The model performs better in terms of **recall** for **Class 0** (majority class) but has **low recall** for **Class 1**, indicating that it is not effectively identifying instances of the minority class. This is often the result of an imbalanced dataset.
  - The **precision for Class 1** is higher than for Class 0, but the **recall for Class 1** is very low (0.25). This suggests that while the model is accurate when it predicts Class 1, it misses a significant number of actual Class 1 instances.
- **Accuracy**:
  - The model's overall accuracy is **53.50%**, which is low for classification tasks where a higher accuracy is expected. This is primarily due to the imbalanced dataset where the model tends to predict the majority class better than the minority class.
- **F1-Score**:

- The weighted average F1-score of **0.49** shows that the model is struggling to balance both precision and recall, especially for Class 1.

Visualizations and Model Evaluation

To better understand the model's performance, we visualized the **confusion matrix** and **ROC curve**. These visualizations provide insights into the misclassifications and the model's ability to differentiate between the two classes.

1. **Confusion Matrix**:
   - The confusion matrix revealed that the model predicts the majority class (Class 0) more accurately than the minority class (Class 1).
   - There is a high **false-negative rate** for Class 1, meaning the model is missing a significant number of actual Class 1 instances.
2. **ROC Curve**:
   - The **Receiver Operating Characteristic (ROC)** curve showed that the model has limited discriminatory power, with an area under the curve (AUC) indicating that the model isn't effectively distinguishing between the two classes.
3. **Precision-Recall Curve**:
   - The **Precision-Recall curve** further confirmed that the model's performance is skewed toward Class 0. While precision is slightly better for Class 1, the recall remains a significant issue, as the model fails to detect enough Class 1 instances.

Recommendations for Improvement

Given the observed performance, there are several strategies that can be employed to improve the model's accuracy and handling of the imbalanced dataset:

1. **Handling Class Imbalance**:
   - **Resampling**: Use techniques like **oversampling** the minority class (Class 1) or **undersampling** the majority class (Class 0) to balance the dataset. Alternatively, consider using **SMOTE** (Synthetic Minority Over-sampling Technique) to generate synthetic samples for the minority class.
   - **Class Weights**: Modify the loss function to include **class weights** to penalize the model more for misclassifying the minority class, encouraging it to focus more on detecting Class 1 instances.
2. **Model Improvement**:
   - **Hyperparameter Tuning**: Tune hyperparameters such as learning rate, batch size, and number of layers to optimize the model's performance.
   - **Alternative Models**: Experiment with different classification models like **Random Forest**, **Gradient Boosting**, or **Support Vector Machines** (SVM) that might better handle imbalanced data.
   - **Ensemble Methods**: Use ensemble methods like **Boosting** or **Bagging** to combine predictions from multiple models, which can improve both accuracy and recall for the minority class.
3. **Evaluation Metrics**:
   - Since the dataset is imbalanced, **accuracy** alone is not a reliable metric. **Precision**, **recall**, and **F1-score** should be prioritized, especially for the minority class.
   - Consider using the **F1-score** or **ROC-AUC** to better capture the model's performance across both classes.
4. **Advanced Techniques**:
   - **Cost-Sensitive Learning**: Introduce a cost-sensitive approach where misclassifications of the minority class are more heavily penalized.
   - **Transfer Learning**: If applicable, use pre-trained models and fine-tune them for your

specific dataset. This could help improve feature extraction, especially for complex patterns in the minority class.

Conclusion

While the model is able to classify the majority class (Class 0) with reasonable accuracy, it struggles significantly with the minority class (Class 1), as seen in the low recall and F1-score for Class 1. The results highlight the challenge of working with imbalanced datasets, where the model tends to be biased toward the majority class.

By implementing strategies such as resampling, class weighting, and experimenting with other models, we can improve the model's ability to correctly classify the minority class and provide a more balanced and effective classification system.