Alankriti Dubey (50604200)

Nidhi Parab (50606993)

Checkpoint

Part 1

Dataset Overview

The dataset includes a

1. Binary target and
2. Numerical features (f1 - f7)

Number of samples: 766

Statistics:

	f1	f2	f3	f4	f5	f6	f7	target
count	766.00	766.00	766.00	766.00	766.00	766.00	766.00	766.00
mean	3.85	120.88	69.12	20.52	79.99	32.00	0.47	0.35
std	3.37	31.94	19.38	15.97	115.34	7.89	0.33	0.48
min	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00
25%	1.00	99.00	62.50	0.00	0.00	27.30	0.24	0.00
50%	3.00	117.00	72.00	23.00	34.00	32.00	0.37	0.00
75%	6.00	140.00	80.00	32.00	127.75	36.60	0.63	1.00
max	17.00	199.00	122.00	99.00	846.00	67.10	2.42	1.00

Number of missing samples: 0

Preprocessing:

The dataset contains no null values

The datatype object is changed to int using: pd.to_numeric

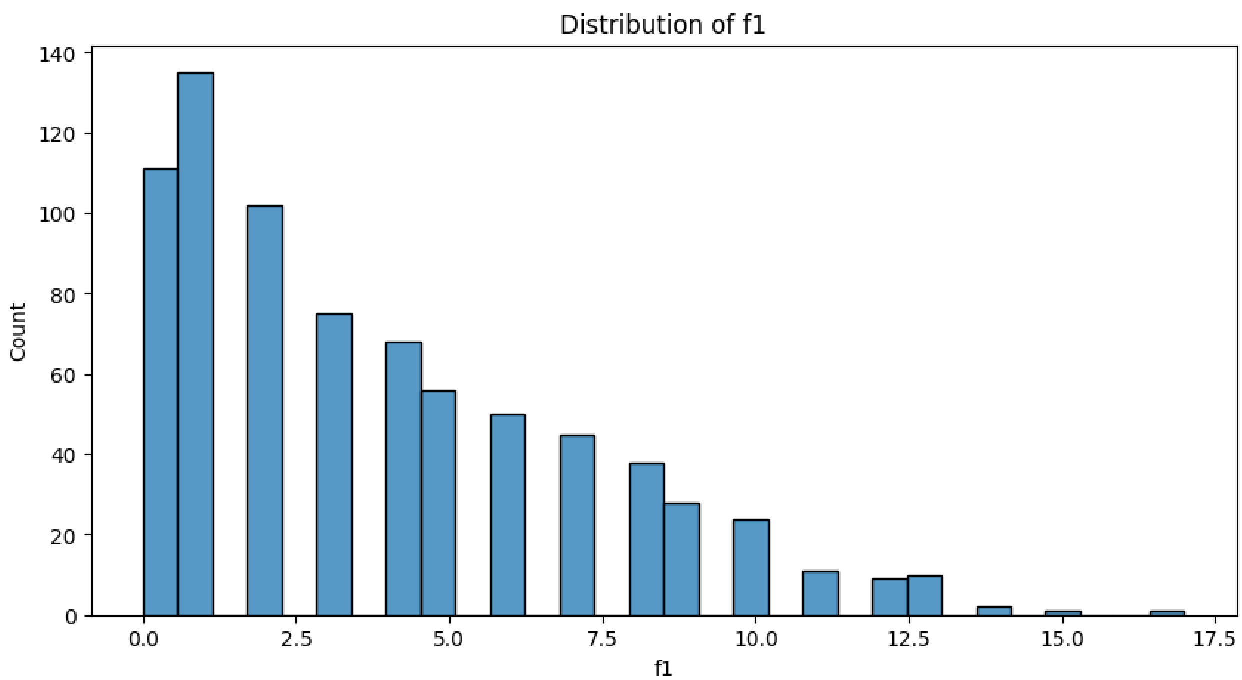
During this conversion NaN values are created

NaN is replaced using mode method

Graphs:

The first graph represents the frequency of f1 data

The histogram is right-skewed indicating that a large majority of the data is clustered near the lower range.



In the second graph, distribution of the variable f2 across two categories of a binary variable target (0 and 1) is displayed

For target 0:

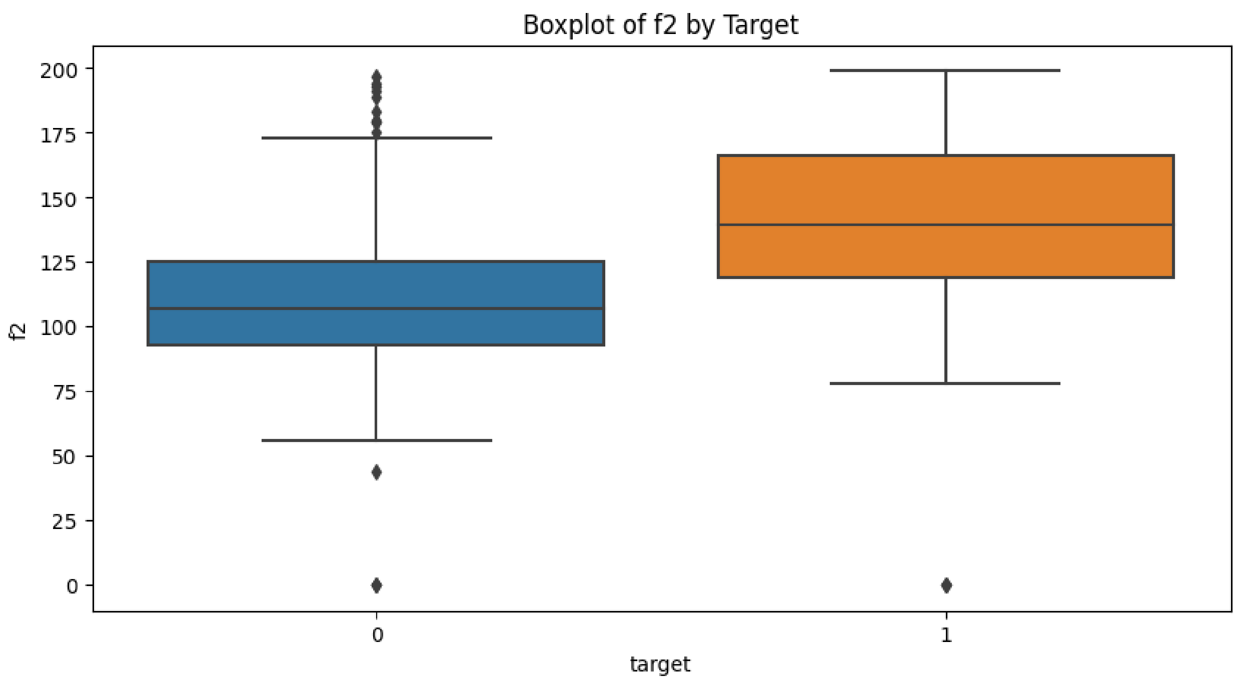
Median: 125

A few outliers are higher than 175 but are not as severe as the lesser outliers.

For target 1:

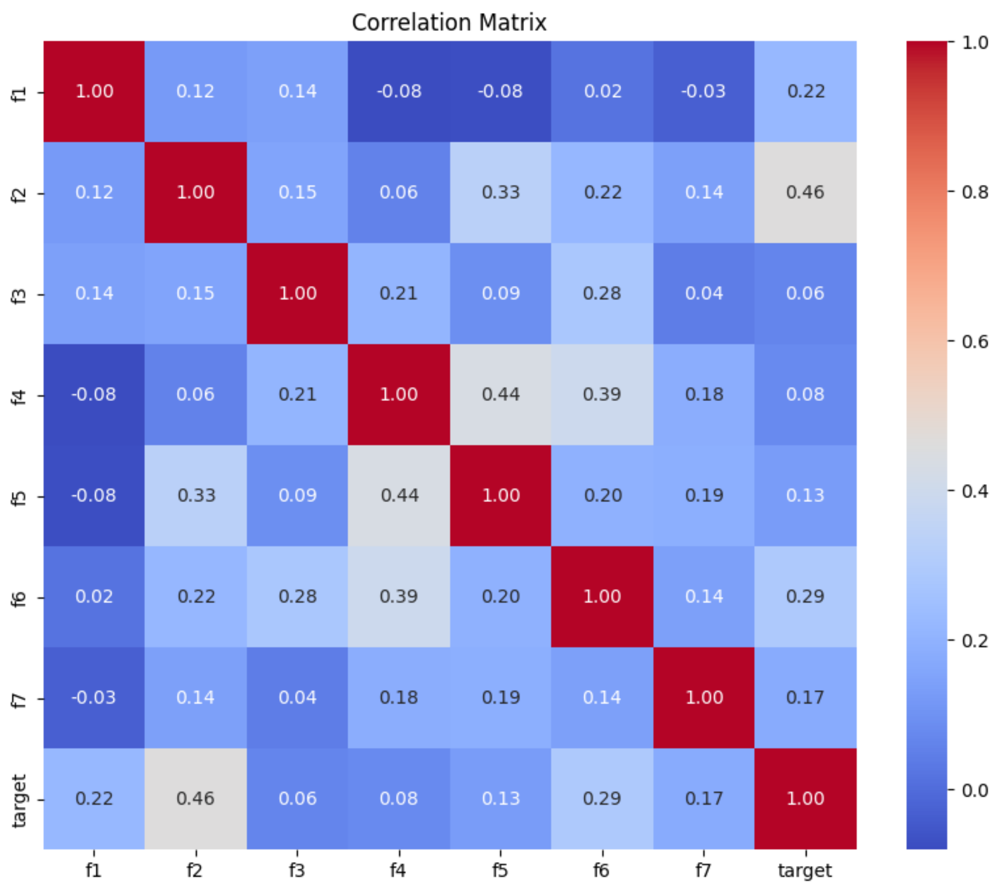
Median: 150

There is just one outlier that can be seen below 75.



f2 and Target: The highest correlation with the target variable is 0.46.

The other variables (f3, f4, f5, and f7) have little to no linear connection with the target variable, as seen by their extremely weak correlations (all below 0.2).



NN architecture

The model takes an input shape of [7, 1] and outputs predictions of the same shape [7, 1]

4,737 Parameters are trainable

The model is appropriate for small-scale classification problems because to its low computational and memory needs. It has a sigmoid activation for binary output predictions and uses dropout to avoid overfitting.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
SimpleNN                                 [7, 1]                    --
├─Linear: 1-1                            [7, 64]                   512
├─Dropout: 1-2                           [7, 64]                   --
├─Linear: 1-3                            [7, 64]                   4,160
├─Dropout: 1-4                           [7, 64]                   --
├─Linear: 1-5                            [7, 1]                    65
├─Sigmoid: 1-6                           [7, 1]                    --
==========================================================================================
Total params: 4,737
Trainable params: 4,737
Non-trainable params: 0
Total mult-adds (M): 0.03
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 0.02
Estimated Total Size (MB): 0.03
==========================================================================================
```

Analysis of the results

Performance metrics:

Accuracy: 0.7857142857142857

With a 78.57% accuracy rate, almost 79 out of 100 samples were properly identified by the model.

Precision: 0.7631578947368421

This metric is important when the cost of false positives is high, as it indicates the model's ability to avoid false alarms.

Recall: 0.5471698113207547

A comparatively poor recall suggests that a huge portion of real positive examples are being missed by the model.

F1 Score: 0.6373626373626373

The trade-off between recall and accuracy is reflected in the F1 score of 63.74%, suggesting that there is potential for improvement.

Loss Graph:

Training Loss:

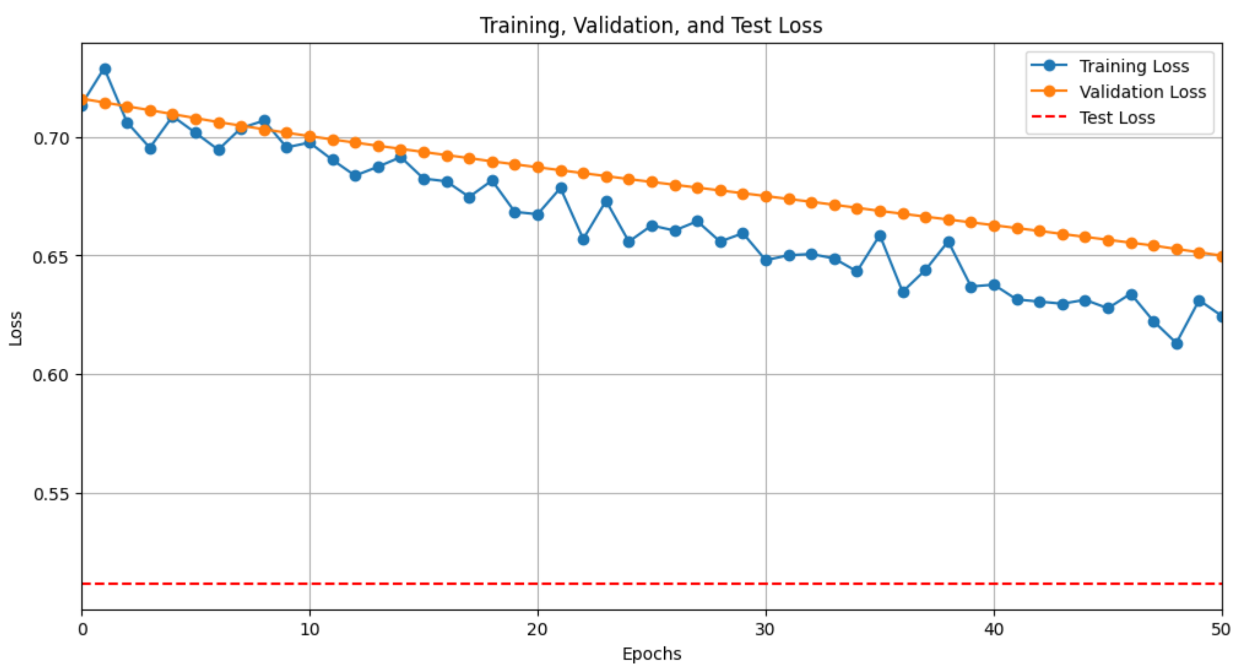
The variations could indicate minor overfitting or model modifications, but as observed the loss generally keeps getting better until the training period is over.

Validation Loss:

Though it is still not as effective as the training set performance, the validation loss exhibits a smoother drop than the training loss, indicating that the model is becoming more capable of generalisation.

Test Loss:

Throughout, the test loss stays constant and is less than the training and validation losses.



Accuracy graph

Training accuracy:

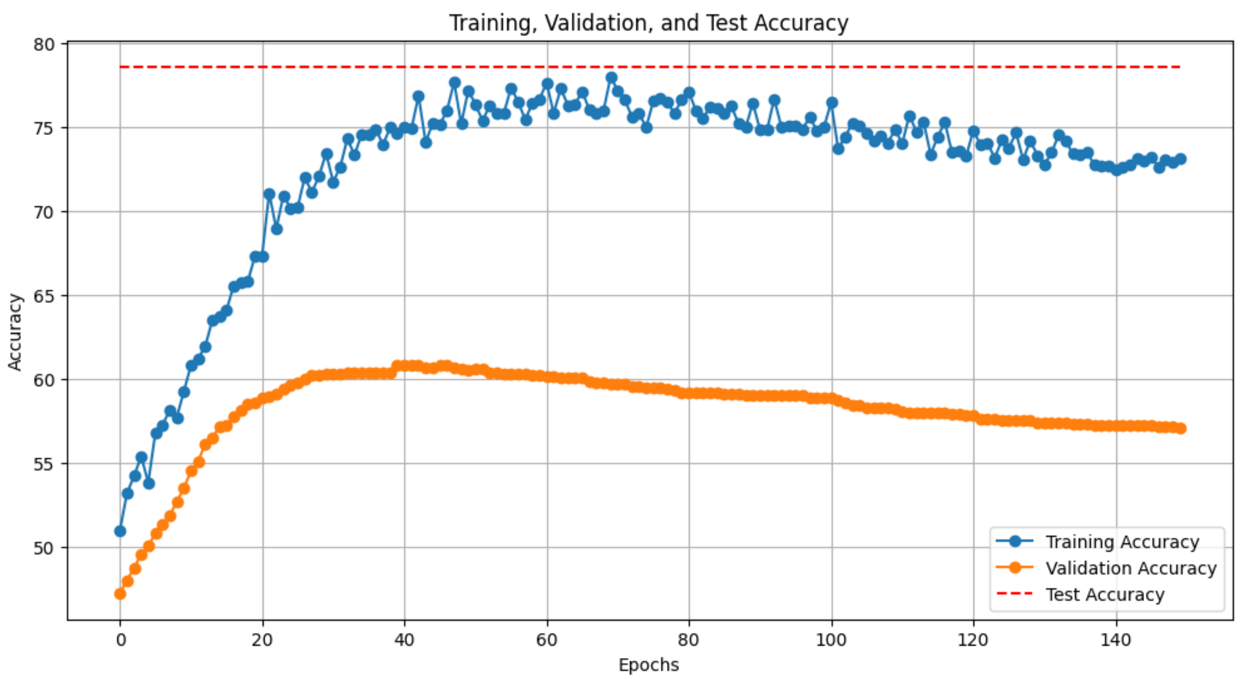
The training accuracy increases rapidly at the beginning and reaches around 75% by epoch 40 this indicates that the model is learning and fitting the training data well

Validation accuracy:

The decline in validation accuracy despite the increase in training accuracy is a sign of overfitting, where the model is becoming too specialized to the training data and is losing its ability to generalize to unseen data.

Test accuracy:

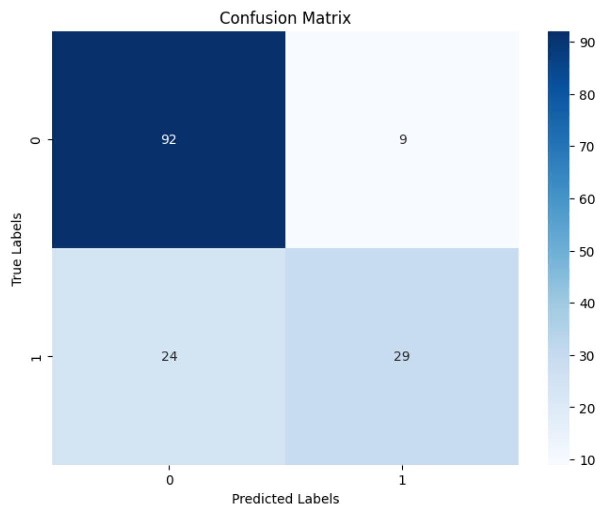
The test accuracy is fixed at around 77-78% throughout the training process.



Confusion matrix

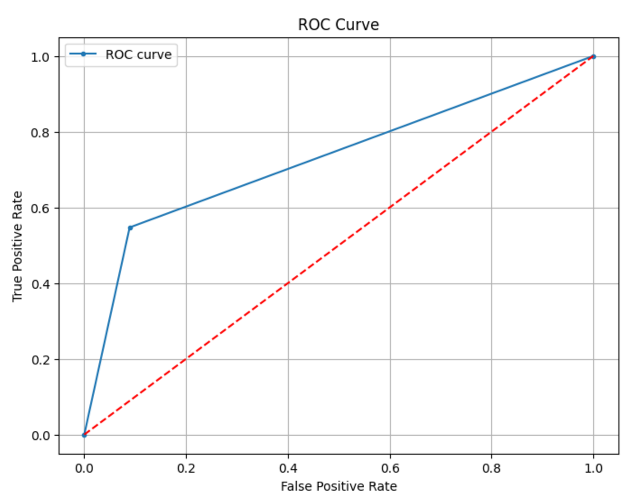
The model has a high precision, meaning most of the positive predictions are correct.

With an overall accuracy of 78.6%, the model performs decently, but there is room for improvement, especially in identifying more true positives and reducing false negatives.



The sharp rise early on suggests that the model is able to achieve a relatively high true positive rate with a low false positive rate, which is a sign of good model performance.

Given the shape of this ROC curve, the AUC is likely moderate, suggesting the model performs reasonably well but has room for improvement.



Part 2

Hyperparameter setups

Dropout Rate Tuning Results

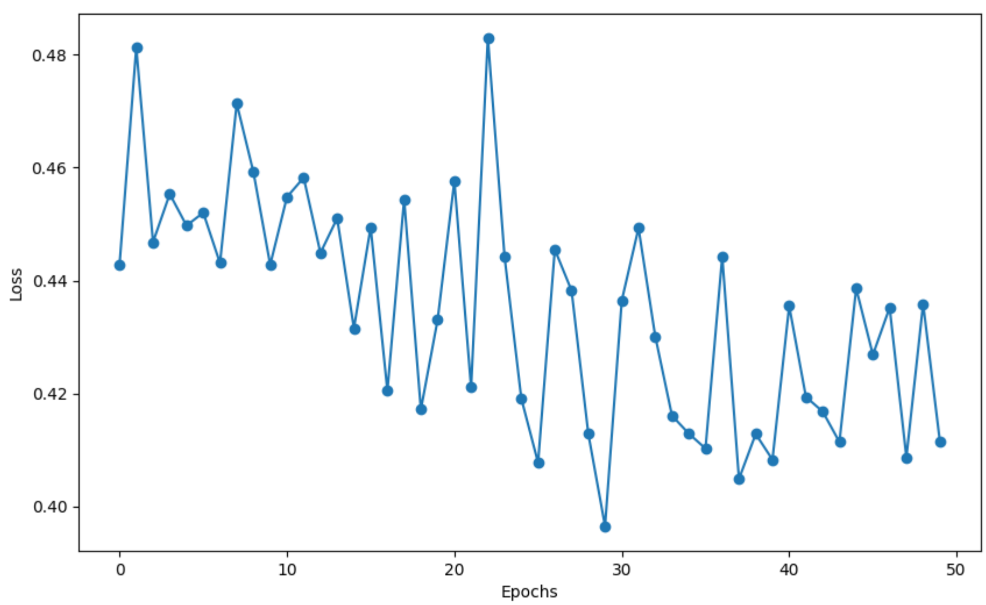
Increasing the dropout rate from 0.1 to 0.5 improved test accuracy by approximately 9%.

This suggests that regularization via dropout helped the model generalize better by preventing overfitting, especially at 0.5 dropout as observed

A loss graph for dropout 5 is shown below. The loss generally keeps getting better until the training period is over.

Dropout Rate Tuning Results:

	Dropout Rate	Test Accuracy
0	0.1	0.688312
1	0.3	0.720779
2	0.5	0.720779



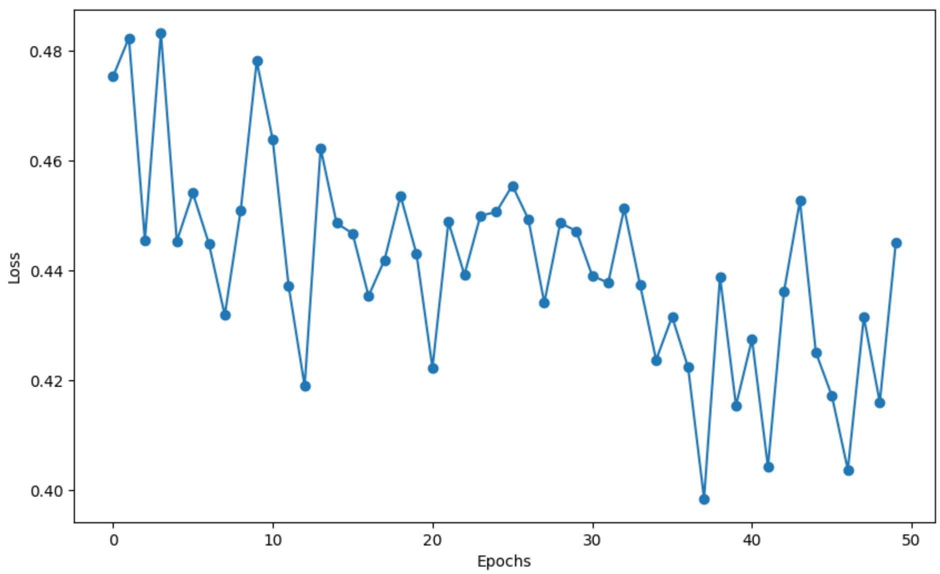
Learning Rate Tuning Results

Accuracy dropped to 74.68% as learning rates rose to 0.01 and 0.1, suggesting that the model could have overshot ideal weight modifications during training because of higher learning rates.

A loss graph for learning rate 0.01 is shown below. The loss generally keeps getting better until the training period is over.

Learning Rate Tuning Results:

	Learning Rate	Test Accuracy
0	0.001	0.740260
1	0.010	0.733766
2	0.100	0.727273



Number of Hidden Layers Tuning Results

Adding more layers does not necessarily improve the model's performance and could lead to overfitting

Number of Hidden Layers Tuning Results:

	Number of Layers	Test Accuracy
0	1	0.746753
1	2	0.740260
2	3	0.720779

Methods used that help to improve the accuracy

K-Fold Cross-Validation:

Average Test Accuracy: 0.7511

Average Test Precision: 0.6310

Average Test Recall: 0.6667

Average Test F1 Score: 0.6476

By training and verifying the model on several subsets of the dataset, K-fold cross-validation made it possible to obtain a more accurate assessment of the model's performance. A little improvement over the original model was shown by the average test accuracy, which rose to 0.7511.

Learning Rate Scheduler

Test Accuracy: 0.7532

Test Precision: 0.6471

Test Recall: 0.6226

Test F1 Score: 0.6346

By modifying the learning rate in response to the model's performance during training, a learning rate scheduler was implemented, which improved the training process. In comparison to the basic model and the k-fold model, the test accuracy improved to 0.7532.

Early Stopping

Epoch: 24

Test Accuracy: 0.7597

Test Precision: 0.6600

Test Recall: 0.6226

Test F1 Score: 0.6408

By tracking validation loss, early stopping prevented overfitting by terminating training at epoch 24. The accuracy increased to 0.7597, demonstrating a significant improvement in model performance while preserving a balance between recall and precision.

Batch Normalization

Accuracy: 0.7662

Precision: 0.6977

Recall: 0.5660

F1 Score: 0.6250

By reducing the internal covariate shift, this method facilitated quicker convergence. Progress from 0.746753 (base model) to 0.7662 (with batch normalization).

Analysis:

Test accuracy:

The accuracies across all models are fairly similar, with slight variations. The Batch Normalized Model model has a better accuracy than all the models.

Training Time:

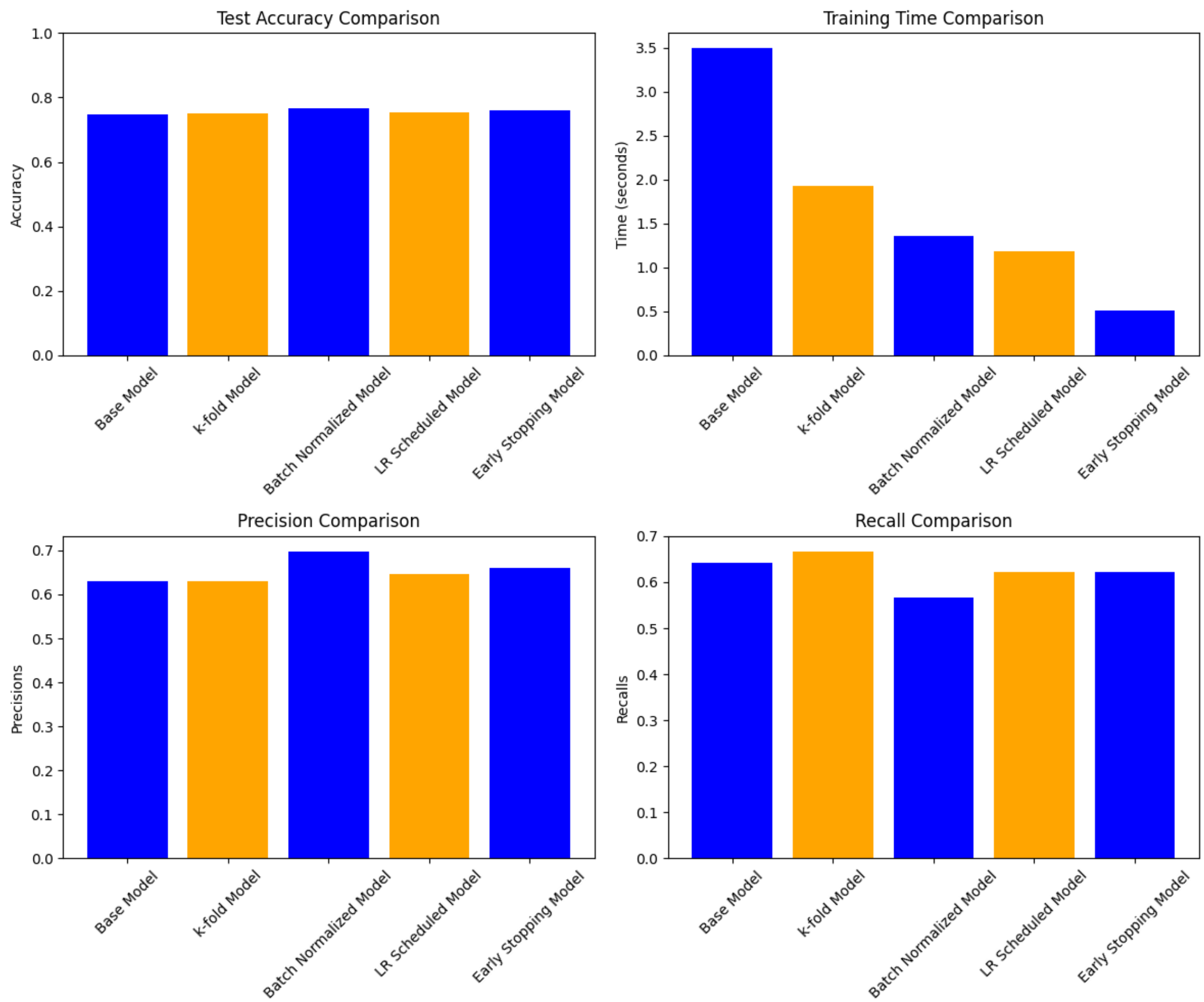
Compared to the other models, the Base Model requires the most training time. Although it is slower than the Base Model, the K-fold Model is still the second slowest. Training periods for the LR-Scheduled Model and Batch Normalised Model are gradually reduced. The Early Stopping Model performs the fastest and requires the least amount of training time.

Precision:

Batch Normalized Model has the highest precision among all models. Other models, including the Base Model, K-fold Model, LR-Scheduled Model, and Early Stopping Model, show slightly lower precision values but are still close to each other.

Recall:

The K-fold Model has the highest recall. The Base Model and Early Stopping Model have moderate recall values, closely following the K-fold Model. The Batch Normalized Model has the lowest recall, while the LR-Scheduled Model also performs similarly but slightly better than the Batch Normalized Model.


Test Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
Base Model
k-fold Model
Batch Normalized Model
LR Scheduled Model
Early Stopping Model
Training Time Comparison
Time (seconds)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Base Model
k-fold Model
Batch Normalized Model
LR Scheduled Model
Early Stopping Model
Precision Comparison
Precisions
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Base Model
k-fold Model
Batch Normalized Model
LR Scheduled Model
Early Stopping Model
Recall Comparison
Recalls
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Base Model
k-fold Model
Batch Normalized Model
LR Scheduled Model
Early Stopping Model

Best Model = Batch Normalized Model

After applying the test data on the model we get:

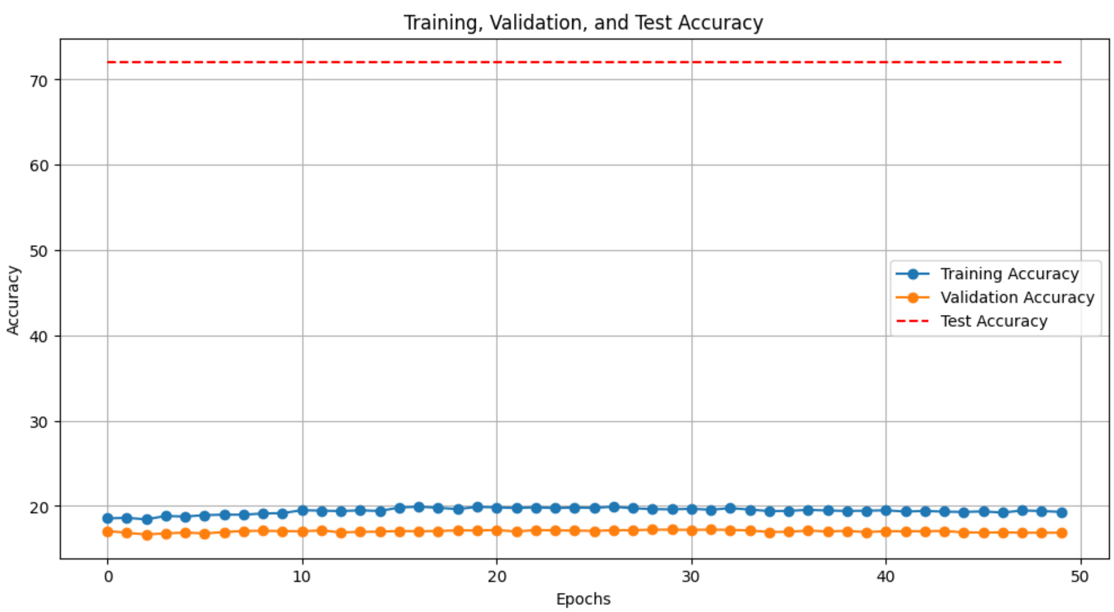
Accuracy: 0.7662337662337663

Precision: 0.6976744186046512

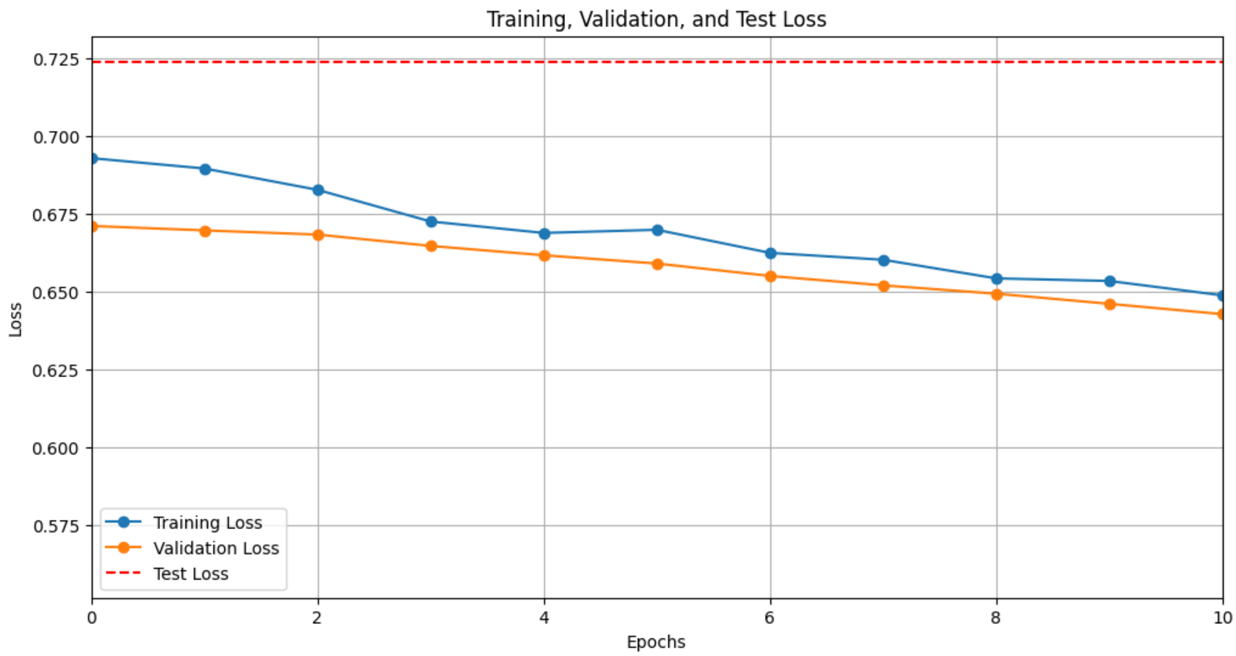
Recall: 0.5660377358490566

F1 Score: 0.625

Both training and validation accuracies being very low and constant suggest the model is underfitting, it is not learning well from the training data. The model may be too simple or not tuned properly, resulting in poor performance on training and validation, but better on the test set.

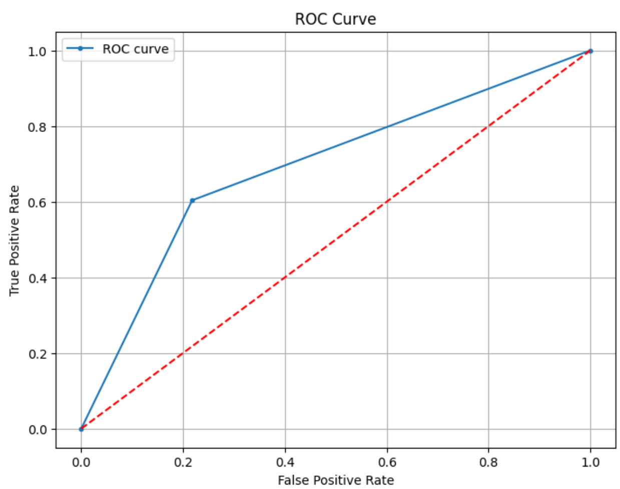


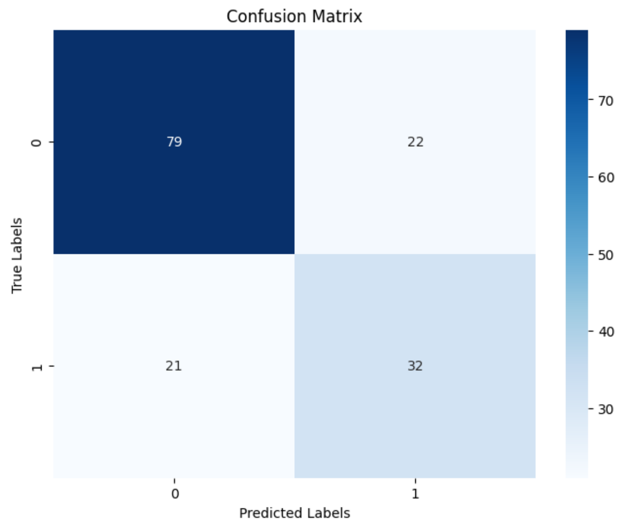
The Training and Validation Loss plots show a steady decline over epochs, which indicates that the model is indeed learning and improving. While the accuracy might be low, the consistent reduction in loss is a good sign that with some adjustments (like more epochs or better optimization), the model can improve further.



Confusion matrix and ROC Curve:

There are only a few misclassifications, and the model is correctly classifying most of the data. The model is slightly better at classifying the negative class than the positive class. ROC curve is above the random guessing line, which indicates that the model is performing better than random guessing. However, the AUC is not very high, which suggests that the model is not performing particularly well. The ROC curve is also quite steep, which means that the model is very sensitive to changes in the classification threshold.





Contributions

Team Member	Assignment Part	Contribution (%)
Alankriti Dubey	Part 1- Defining the Neural Network Part 2 – Early Stopping, Batch Normalization, and Learning Rate Scheduler. Report – Part 1	50
Nidhi Parab	Part 1 – Training the Neural Network Part 2 – Hyperparameters tuning and K-Fold Cross Validation	50