AI foundations Coursework

Table of contents

Part 1: Heart Attack Classification

1.1 Data Preparation

Robust data preparation is essential for accurate machine learning models. It establishes a reliable foundation, ensuring trustworthy outcomes and enhancing the overall quality and credibility of findings.

Outlier handling:

Outliers, data points deviating significantly from the majority, can distort analyses and affect result reliability. After a thorough examination, outliers were identified in 'BMI,' 'MentHlth,' and 'PhysHlth.' The 'Diabetes' boxplot (Figure 1) indicates outliers, mainly because the majority are non-diabetic (Figure 2). Diabetes types 1 and 2, correlated with the target attribute, are not treated as outliers. To address these outliers, a tailored strategy was devised after assessing their prevalence in each attribute. 'BMI' outliers (less than 4% of the dataset) were removed, while for 'MentHlth' and 'PhysHlth' (approximately 15% of the data), a capping approach was applied, setting the maximum value for outliers at 5. Figures 3-8 depict the impact on boxplot graphs before and after handling outliers.

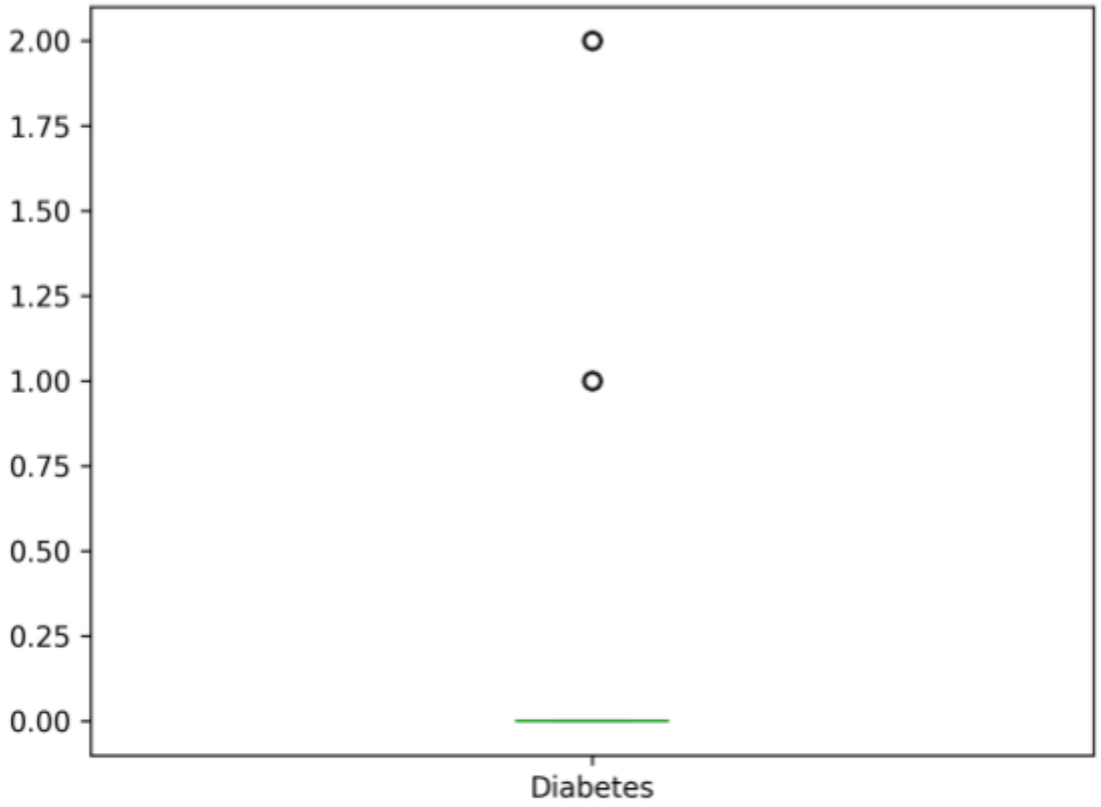

Figure 1: Diabetes Box Plot

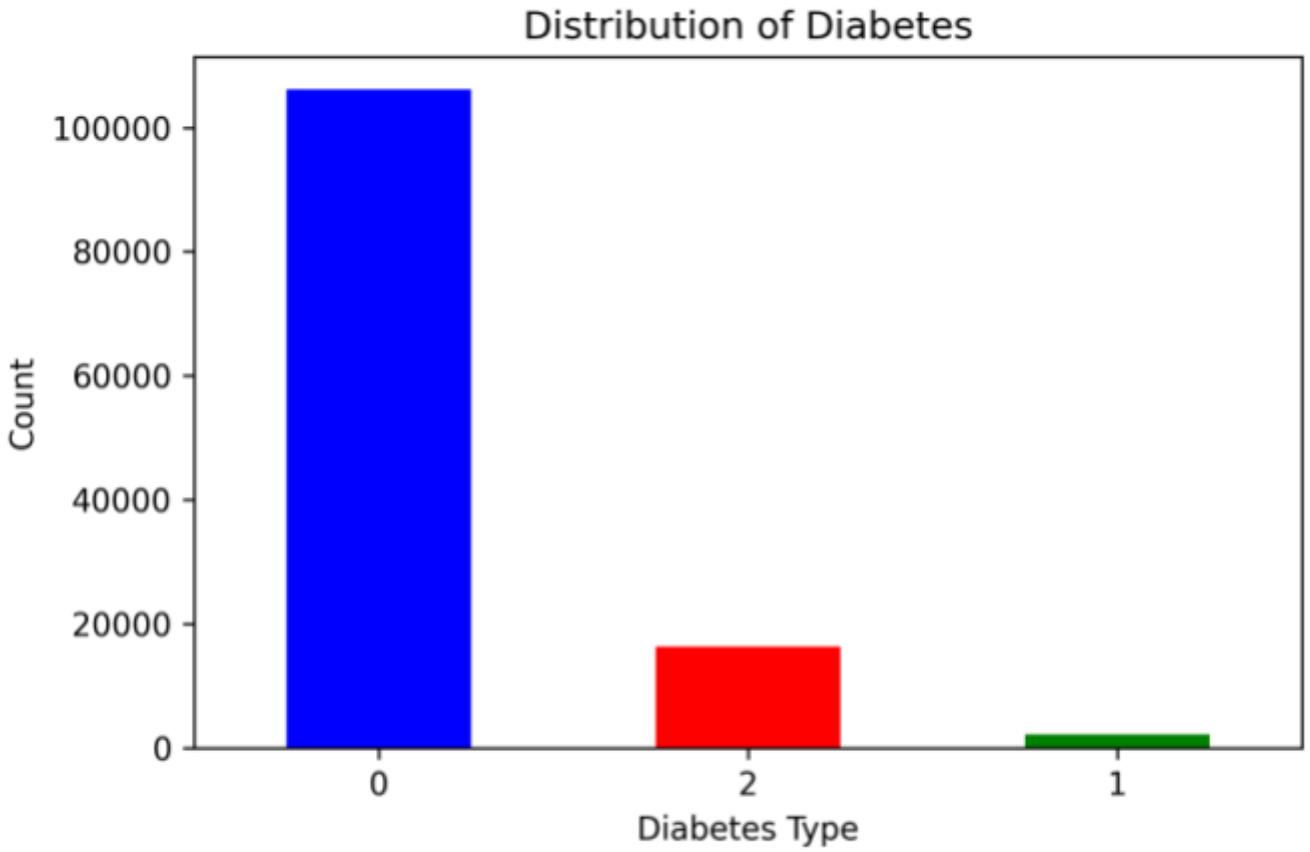

Figure 2: Diabetes Distribution

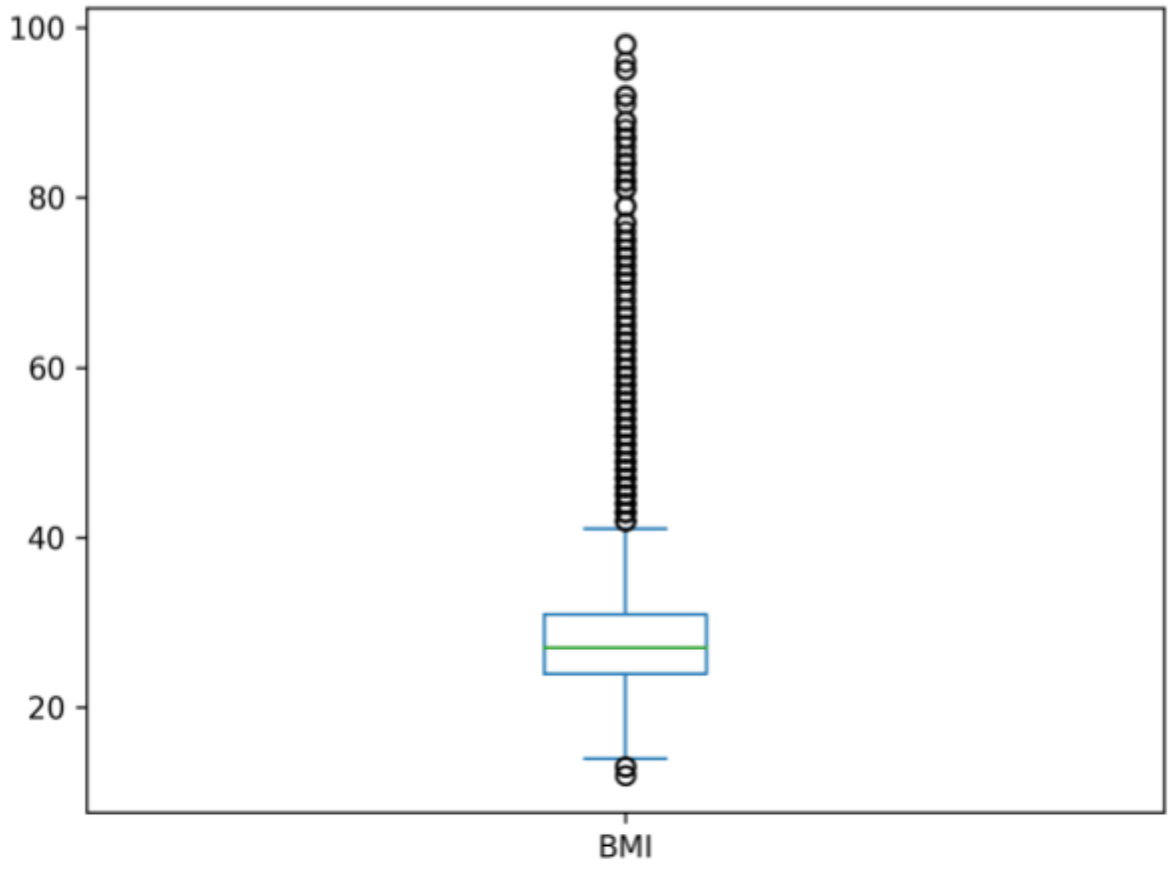

Figure 3: BMI boxplot before outlier handling

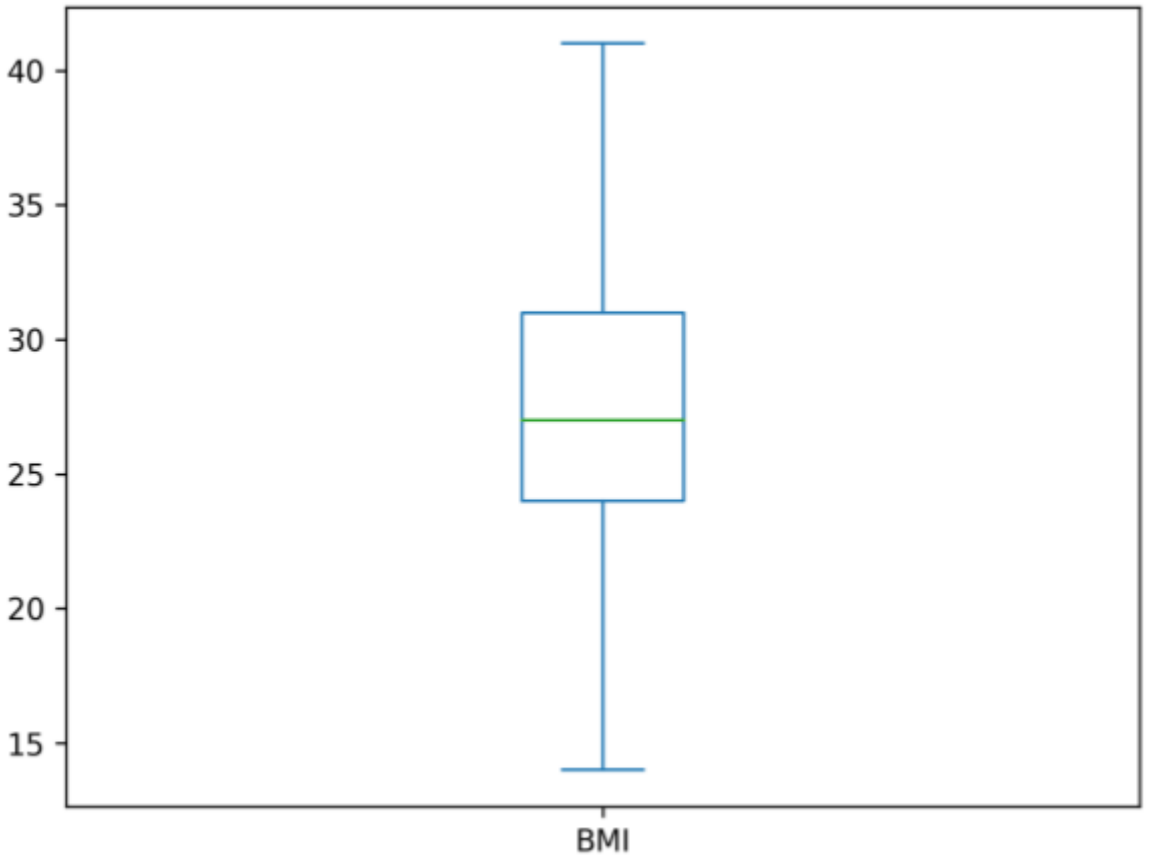

Figure 4: BMI boxplot after outlier handling

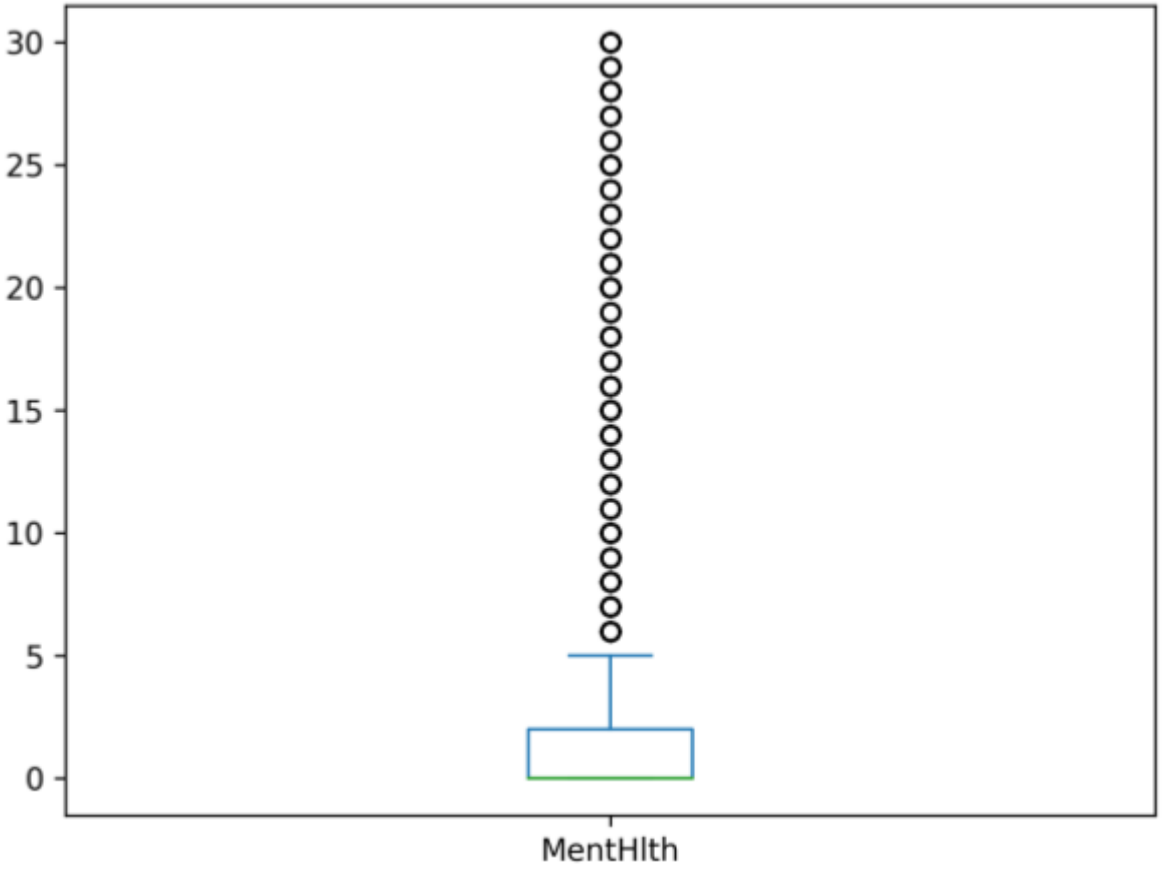

Figure 5: MentHlth before outlier handling

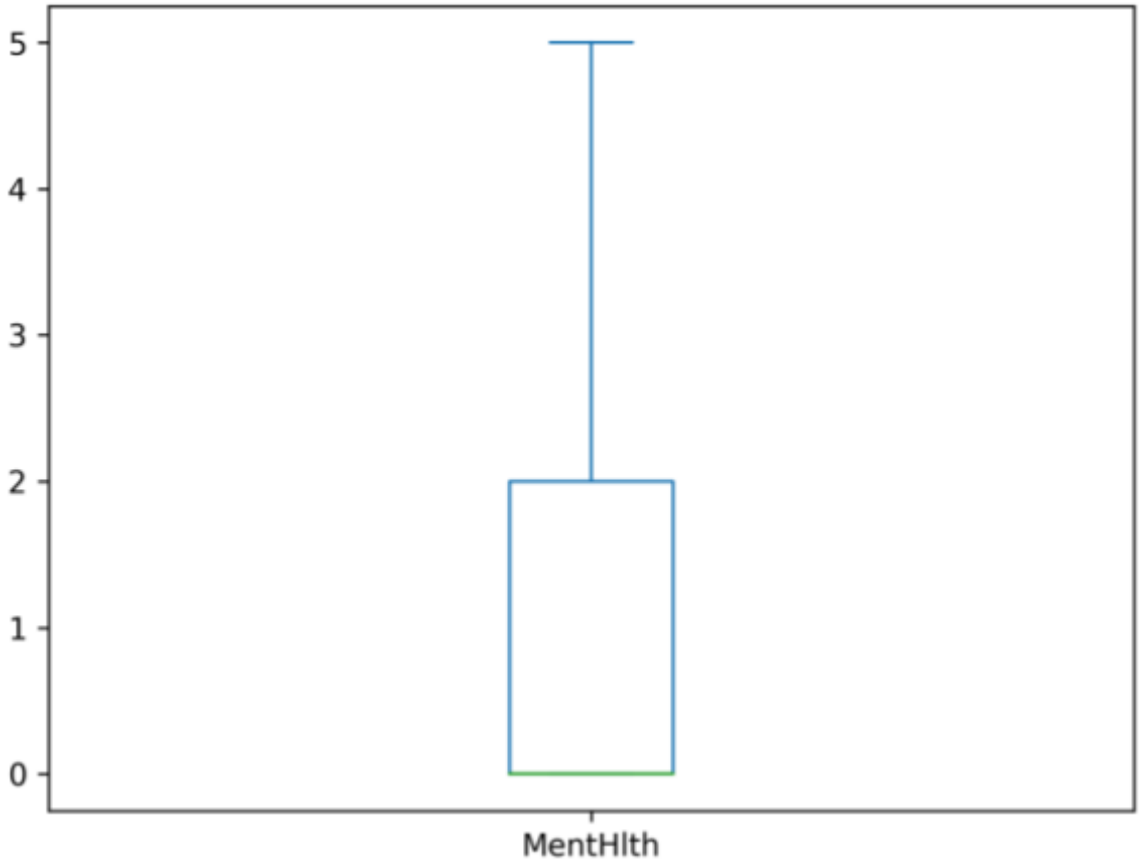

Figure 6: MentHlth after outlier handling

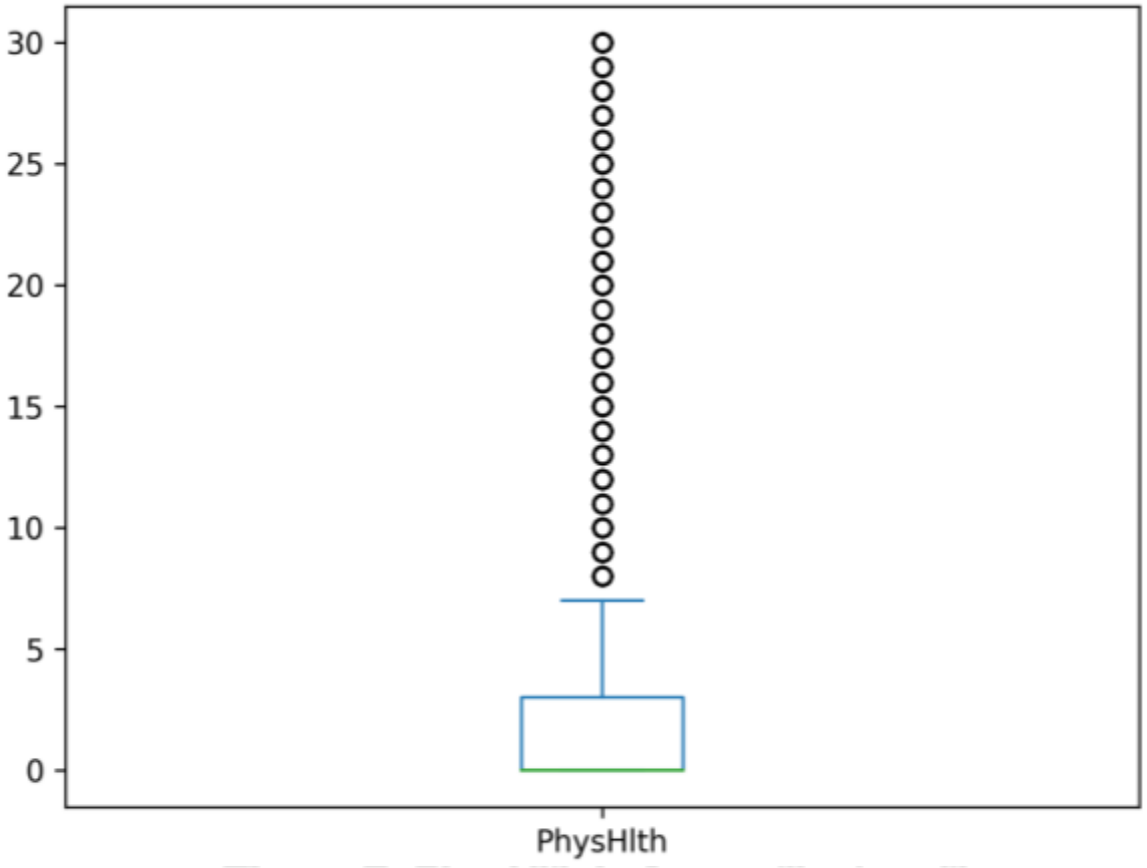

Figure 7: PhysHlth before outlier handling

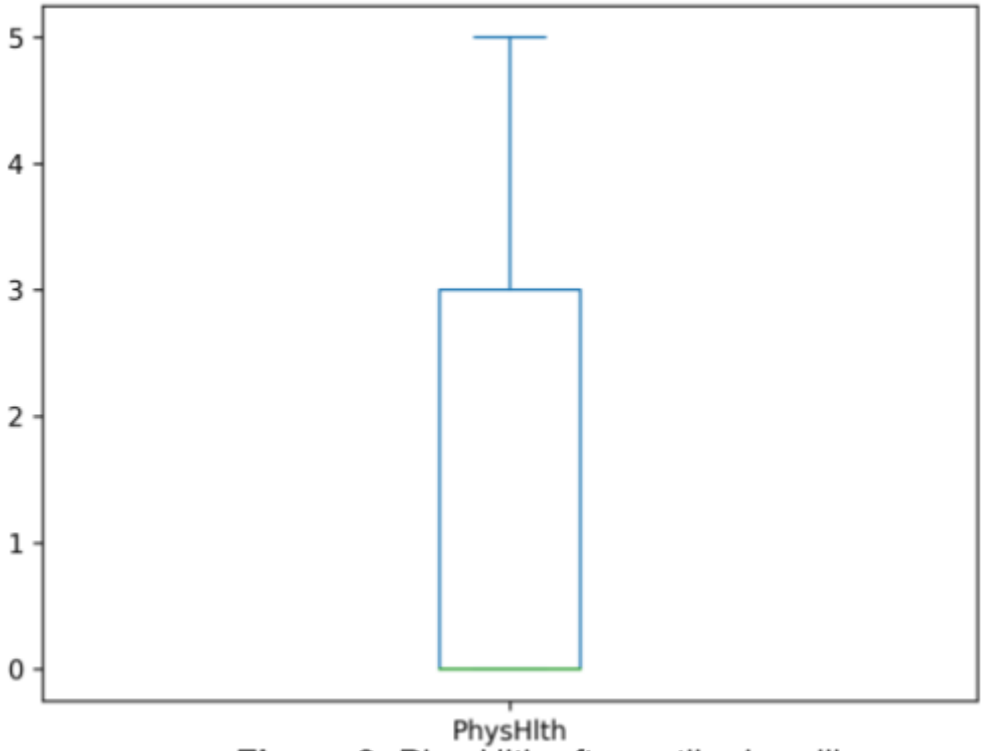

Figure 8: PhysHlth after outlier handling

Missing values handling:

Upon scrutinising the dataset for missing values, it was observed that only the attribute 'BMI' contained any such instances, as illustrated in Figure 9. The count of missing values constituted less than 0.01% of the total records in the dataset. Consequently, the decision was made to eliminate the rows with missing values, ensuring a meticulous handling of the minimal missing data in the dataset.

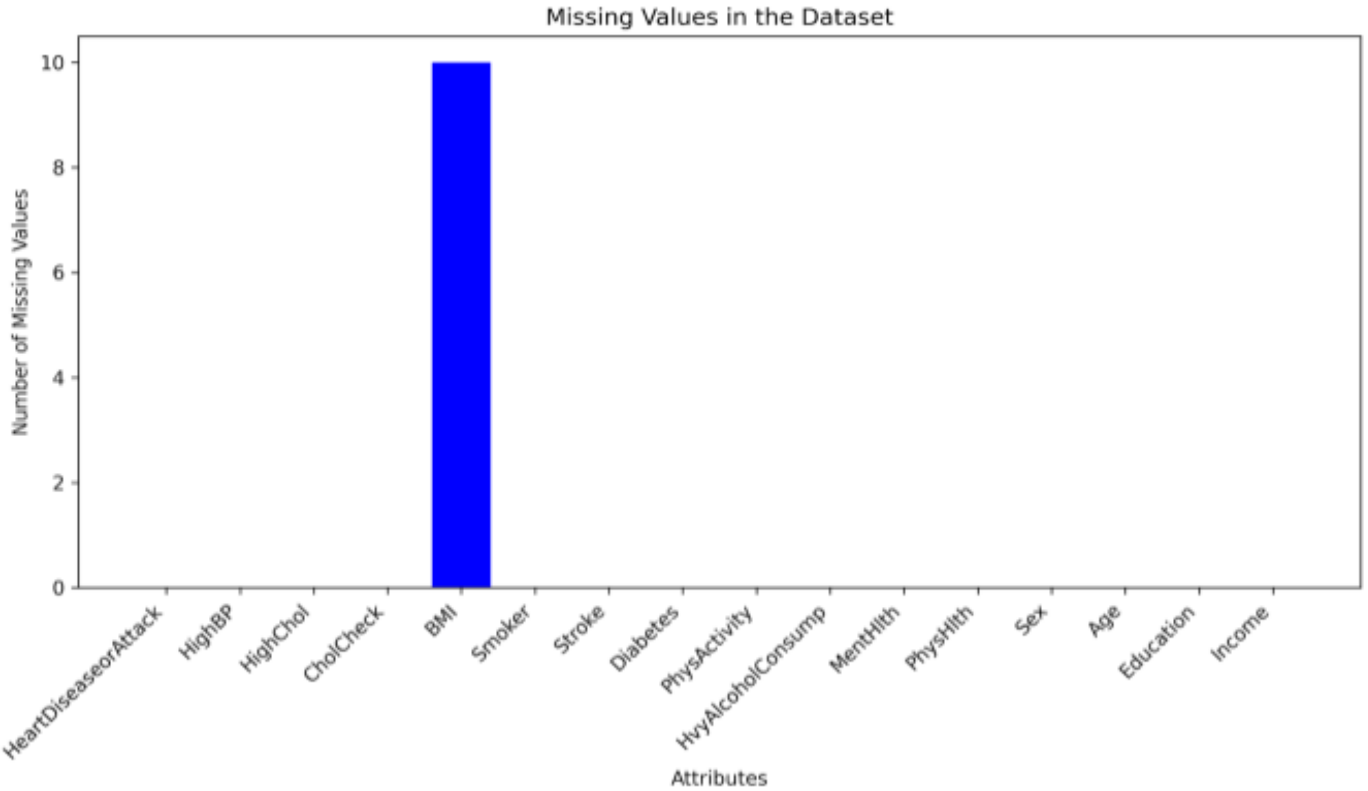

Figure 9: Missing values table

Normalisation,Target Definition, and Balancing Strategy:
All models were trained on the whole dataset so that all data would be used but also to ensure fair model comparisons.The dataset was then divided into target ('HeartDiseaseorAttack') and predictors(all other attributes). Dimensionality reduction was omitted due to strong correlations between all predictors and the target.Unbalanced target classes were identified(Figure 10), posing a risk of bias towards the majority class in binary classification. To overcome this, the Synthetic Minority Over-sampling Technique (SMOTE) was applied, yielding a balanced distribution (Figure 11). Subsequently, predictors underwent normalisation to a standardised scale for enhanced stability and convergence during model training.

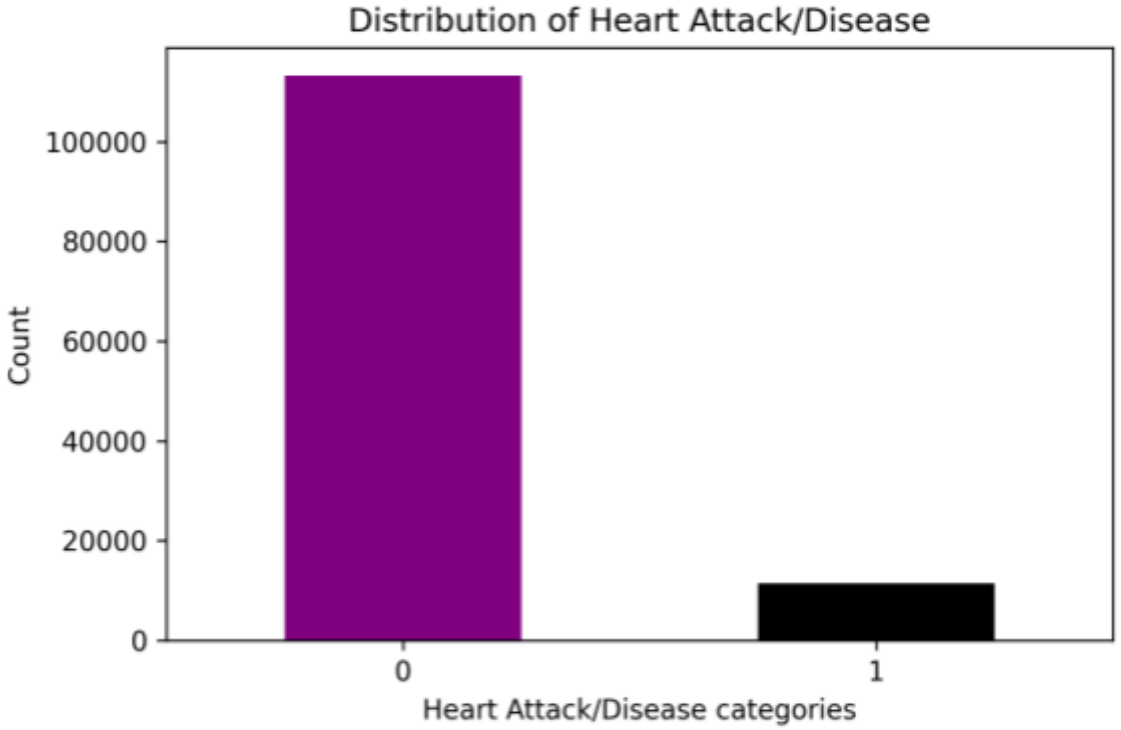

Figure 10: Target data distribution

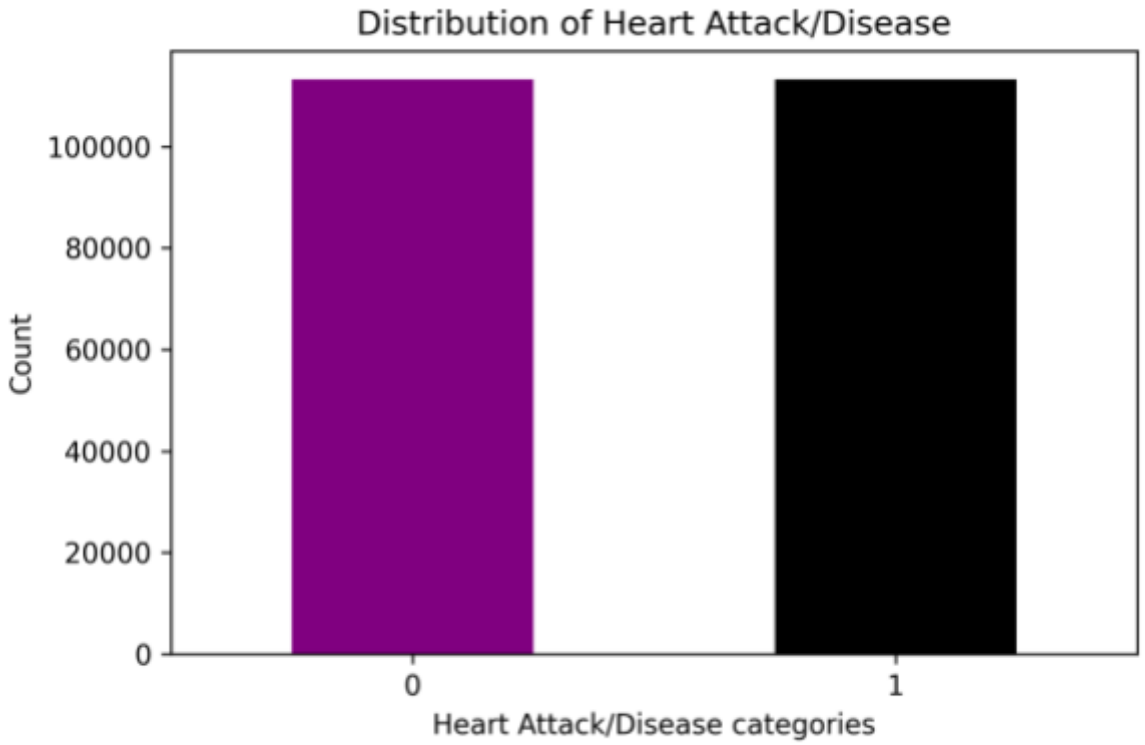

Figure 11: After Smote

1.2 Model Training

Cross-validation was chosen for robust model evaluation, ensuring a reliable performance estimate. However, due to the time-intensive nature of training Artificial Neural Networks (ANN) with cross-validation on a substantial dataset of around 125k entries, a train-test split was employed for ANN to balance computational efficiency while maintaining a reasonable evaluation approach.Also the accuracy results from the grid search method(2) explained below may appear low as a limited number of cross-folds were used due to time constraints, and the actual results are anticipated to be higher with a more exhaustive exploration.

K-Nearest Neighbours(1)

To optimise the algorithm's performance on the provided dataset, hyperparameter fine-tuning was crucial.GridSearch(2) method was employed, a machine learning hyperparameter tuning technique (Figure 12). The selected hyperparameters are as follows::

N_neighbors: Sets the number of neighbours for predicting new data points. Optimal performance, identified through simulation and graph analysis (Figure 13 and 14), was achieved with a single neighbour, guiding the choice for model construction.

Weights: Controls neighbour influence. 'Distance' gives more weight to closer neighbours, while 'uniform' assigns equal weight to all. Testing both options revealed superior performance with 'distance,' guiding its selection for model training.

P: The 'p' hyperparameter in the Minkowski distance metric determines the distance measure type. 'p' at 1 employs Manhattan distance (L1 norm), and at 2, it uses Euclidean distance (L2 norm). The choice of 'p' influences the algorithm's sensitivity to diverse data patterns.

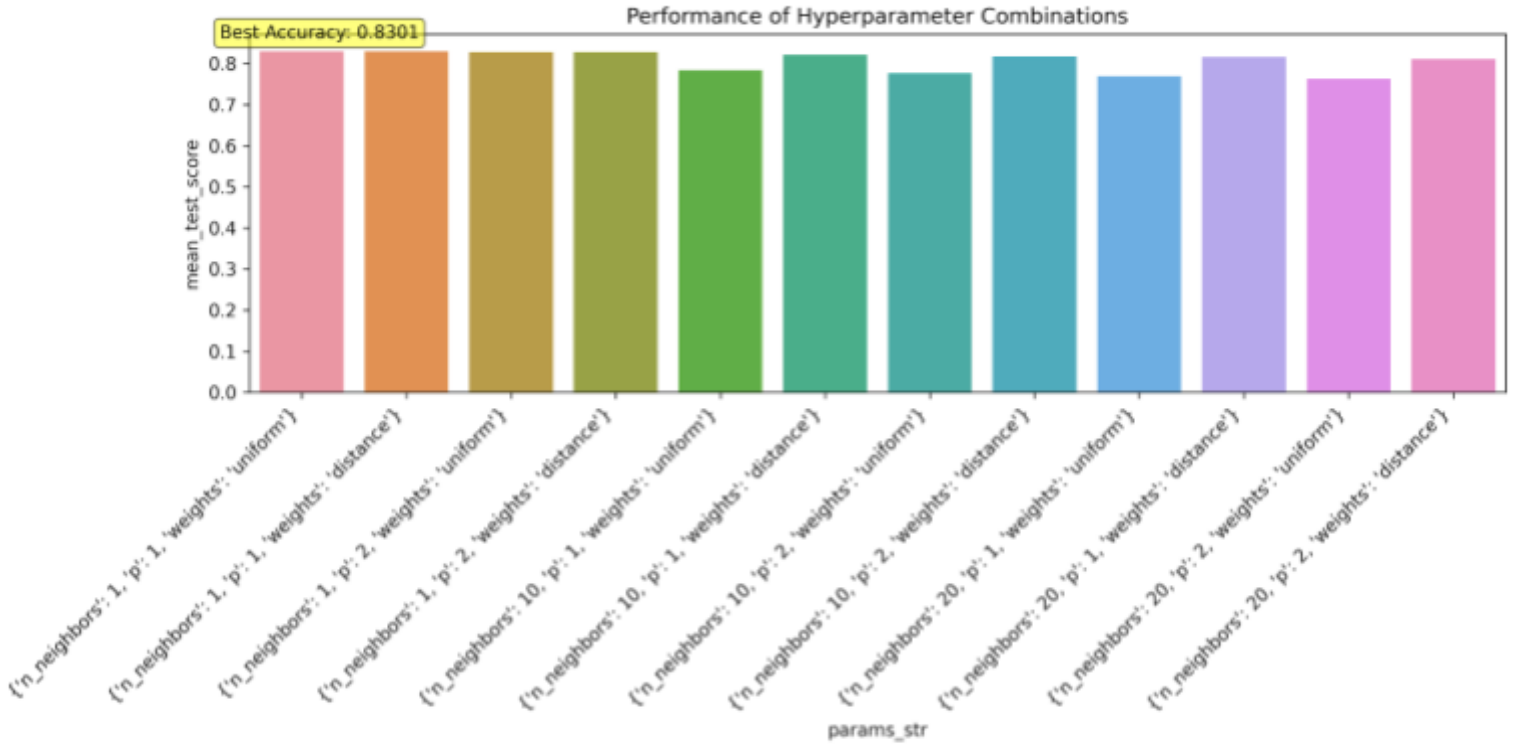

Figure 12: GridSearch results of KNN

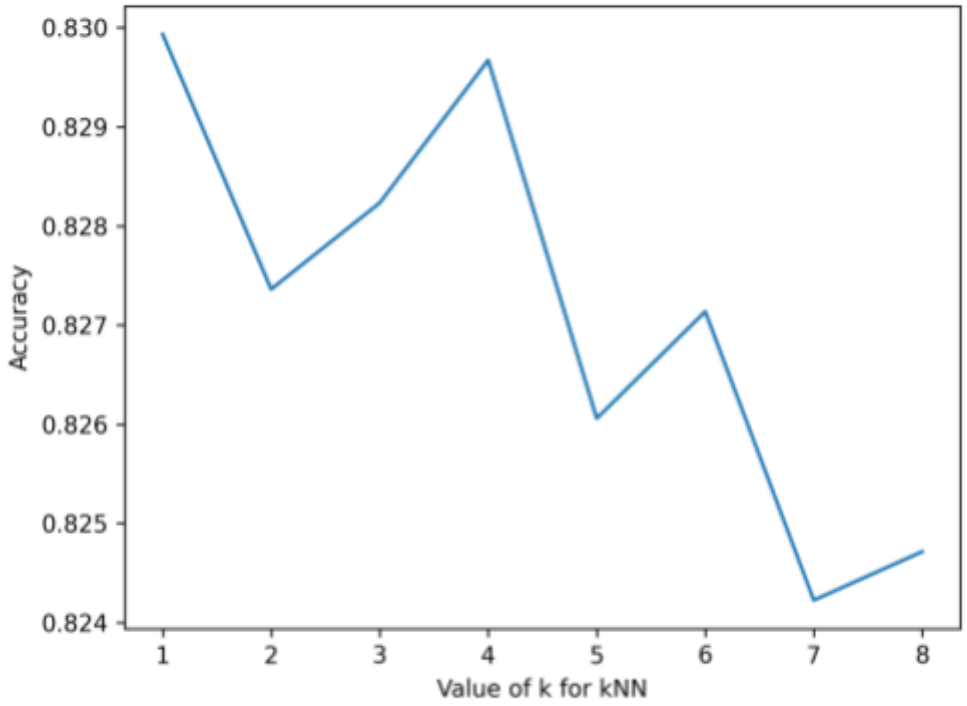

Figure 13: KNN performance for 1-10 neighbours

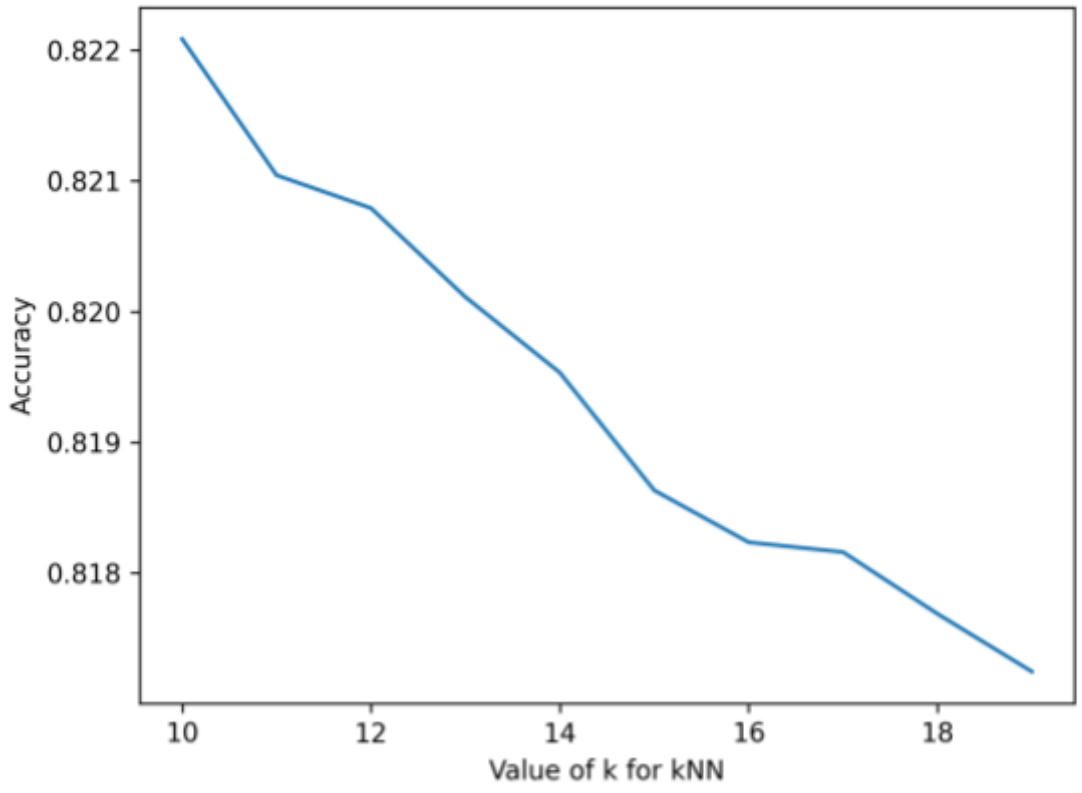

Figure 14: KNN performance for 10-20 neighbours

Decision Tree(3)

To optimise the model's performance on the provided dataset, the Grid Search method was employed as before(Figure 15). The hyperparameters considered in this approach include:

criterion: Specifies the criteria for evaluating split quality—options include Gini impurity ('gini'), entropy ('entropy'), or logarithmic loss ('log_loss').
min_samples_split: Establishes the minimum number of samples required to split an internal node, influencing the tree's depth and structure.
min_samples_leaf: Sets the minimum number of samples needed for a leaf node, affecting the smallest leaf size in the tree.
max_features: Determines the subset of features considered when finding the best split—'None' includes all features, 'sqrt' considers the square root, and 'log2' considers the logarithm base 2.

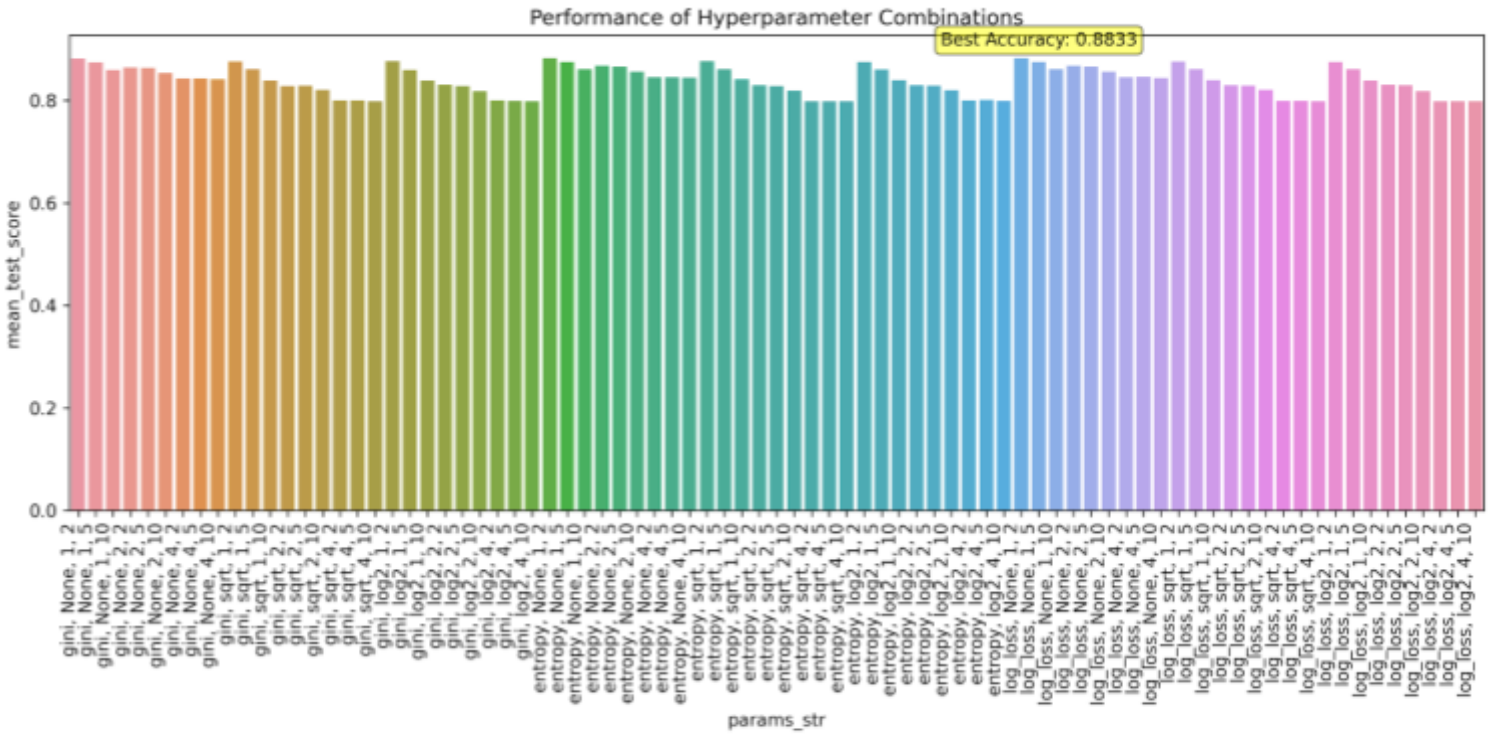

Figure 15: DecisionTree GridSearch Accuracy

Random Forest:

The Random Forest algorithm exhibited satisfactory performance without the need for explicit hyperparameter tuning, so it was retained in its default configuration.

Artificial Neural Network(ANN)(4):

To optimise the performance of the Artificial Neural Network (ANN) on the given dataset, a meticulous fine-tuning of its hyperparameters was crucial to achieve the highest attainable accuracy. The hyperparameters chosen for this optimization process are as follows:

Number of Neurons and Layers: The architecture of the neural network, including the number of neurons and layers, plays a significant role in its performance. The chosen architecture consists of multiple dense layers with varying numbers of neurons (350, 400, 450, 500, 560), each employing the ReLU activation function instead of the output layer which uses the softmax function. This architecture was fine-tuned to strike a balance between complexity and performance.

Dropout Rate: Dropout is a regularisation technique used to prevent overfitting. In this model, dropout layers with a rate of 0.05 were added between dense layers to enhance generalisation.

1.3 Model Evaluation

KNN:

The KNN model demonstrates balanced performance with an accuracy of 86%. The precision values for both classes (0 and 1) indicate that the model correctly identifies instances belonging to each class, with 87% precision for class 0 and 85% for class 1. The recall values reveal the model's ability to capture the majority of instances for each class, achieving 84% recall for class 0 and 88% for class 1. The F1-score, which harmonises precision and recall, is consistently high at 86% for both classes. Overall, these results suggest a well-rounded performance of the KNN model in classification tasks(Figure 16).

In the confusion in figure 17 the model demonstrates good accuracy (86%) but has a notable number of false negatives, indicating a tendency to miss identifying patients at risk. This suggests a potential area for improvement, as false negatives could have significant consequences in this medical context.

	precision	recall	f1-score	support
0	0.87	0.84	0.86	113176
1	0.85	0.88	0.86	113176
accuracy			0.86	226352
macro avg	0.86	0.86	0.86	226352
weighted avg	0.86	0.86	0.86	226352

Accuracy: 0.8584814801724747

Figure 16: KNN Classification Report

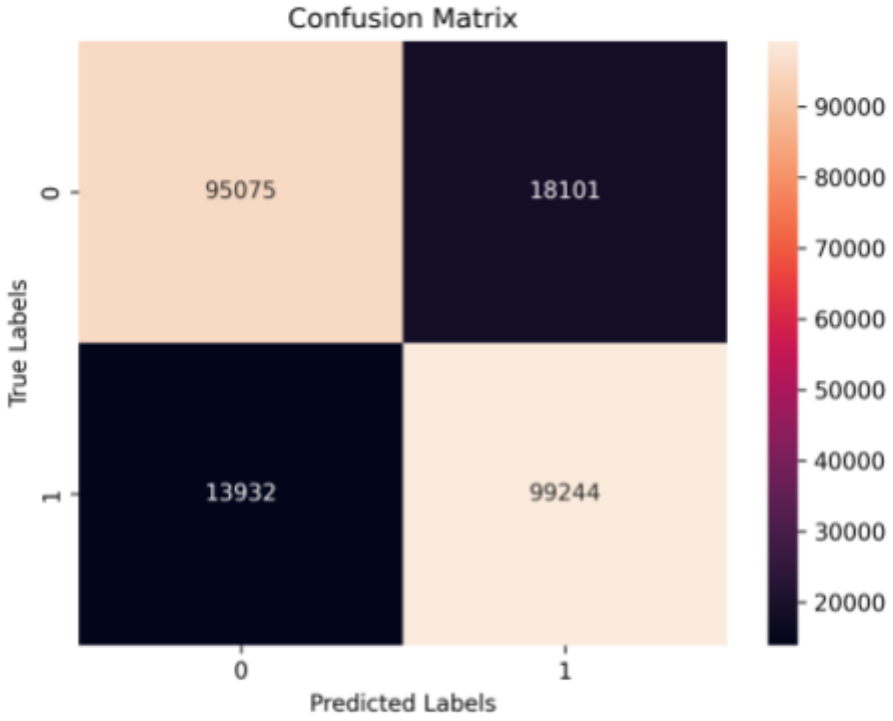

Figure 17: KNN Confusion Matrix

Decision Tree:

The Decision Tree model exhibits robust performance with an accuracy of 88.2%. Precision values for both classes (0 and 1) indicate high accuracy in identifying instances for each class, achieving 91% precision for class 0 and 86% for class 1. The recall values demonstrate the model's effectiveness in capturing the majority of instances for each class, achieving 85% recall for class 0 and 91% for class 1. The F1-score, harmonising precision and recall, is consistently high at 88% for both classes, suggesting a well-rounded performance in classification tasks(Figure 18).

The model demonstrates a high accuracy of 88%, with balanced precision and recall scores for both classes. However, similar to the KNN model, there are still notable false negatives, suggesting a potential area for improvement(Figure 19).

	precision	recall	f1-score	support
0	0.91	0.85	0.88	113176
1	0.86	0.91	0.89	113176
accuracy			0.88	226352
macro avg	0.88	0.88	0.88	226352
weighted avg	0.88	0.88	0.88	226352

Accuracy: 0.8820553827666643

Figure 18: Decision Tree Classification Report

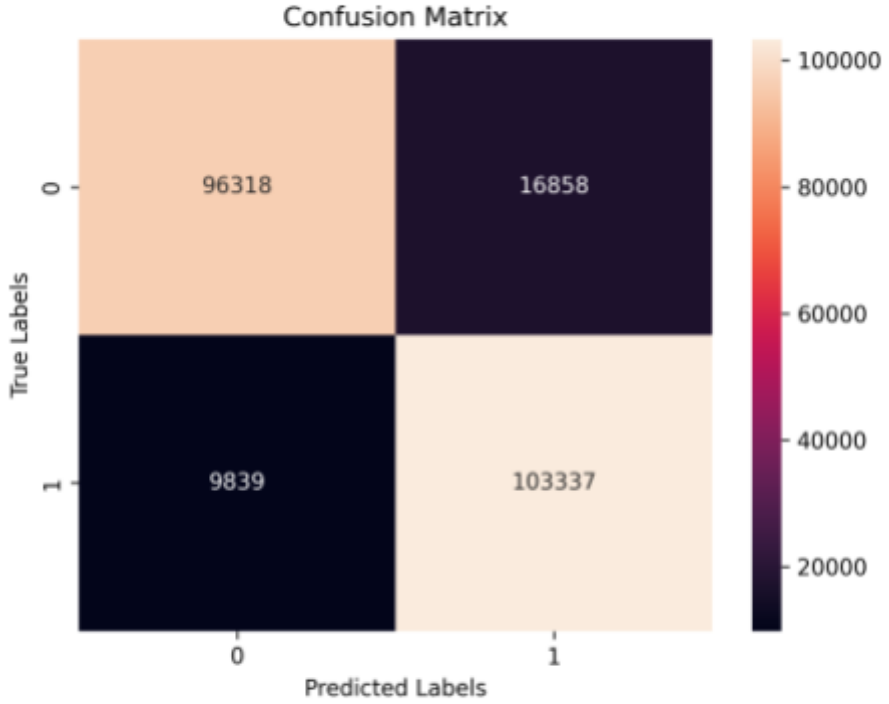

Figure 19: Decision Tree Confusion Matrix

Random Forest:

The Random Forest model excels with a remarkable accuracy of 90.4%. Impressive precision values (93% for class 0, 88% for class 1) signify accurate identification of instances for both classes. The model effectively captures the majority of instances, as evidenced by recall values of 87% for class 0 and 93% for class 1. F1-scores consistently hover around 90%, indicating a well-rounded performance(Figure 20). In the confusion matrix (Figure 21), the model achieves 90% accuracy, with a relatively small number of false negatives (FN: 14,396). This suggests a minor tendency to miss identifying patients at risk, but overall performance remains strong.

	precision	recall	f1-score	support
0	0.93	0.87	0.90	113176
1	0.88	0.93	0.91	113176
accuracy			0.90	226352
macro avg	0.91	0.90	0.90	226352
weighted avg	0.91	0.90	0.90	226352

Accuracy: 0.90357496288896586

Figure 20: Random Forest Classification Report

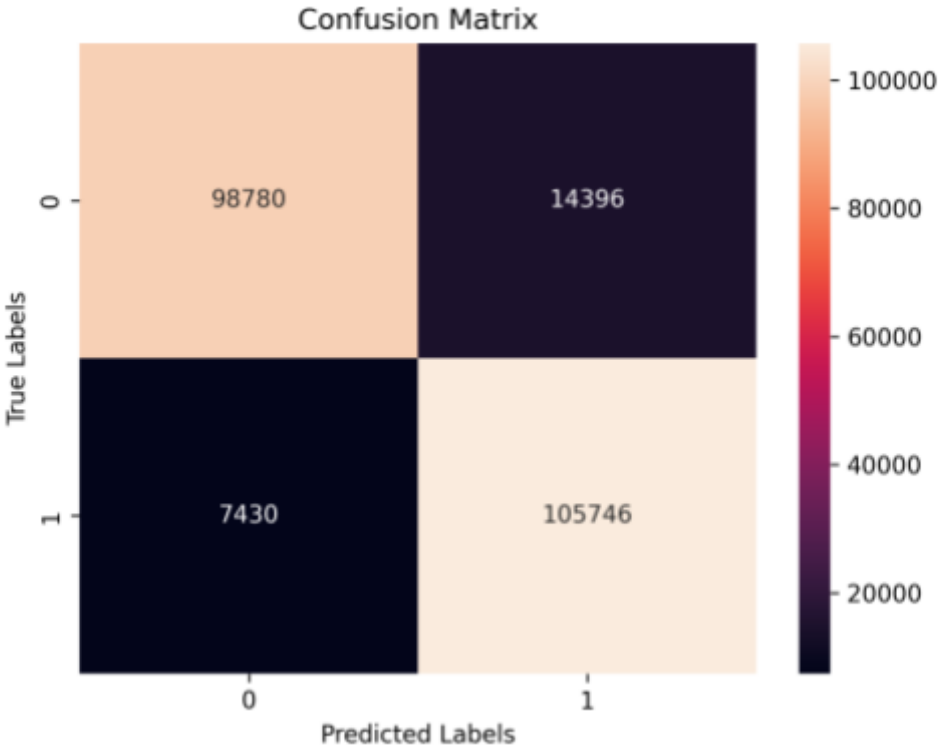

Figure 21: Random Forest Confusion Matrix

Artificial Neural Network:

The Artificial Neural Network (ANN) model exhibits strong performance with an accuracy of 83%. Although precision for class 0 is slightly lower (89%), the model excels with 91% precision for class 1, showcasing its ability to accurately identify instances at risk. Additionally, recall values of 76% for class 0 and 91% for class 1 highlight the model's effectiveness in capturing instances for both classes. The F1-scores, harmonising precision and recall, consistently hover around 83%, indicating well-rounded performance(Figure 22).

In the confusion matrix (Figure 23), the model achieves an 83% accuracy with a relatively small number of false negatives (FN: 5,497). Again this suggests a tendency to miss identifying instances not at risk.

	precision	recall	f1-score	support
0	0.89	0.76	0.82	22651
1	0.79	0.91	0.85	22620
accuracy			0.83	45271
macro avg	0.84	0.83	0.83	45271
weighted avg	0.84	0.83	0.83	45271

Figure 22: ANN Classification Report

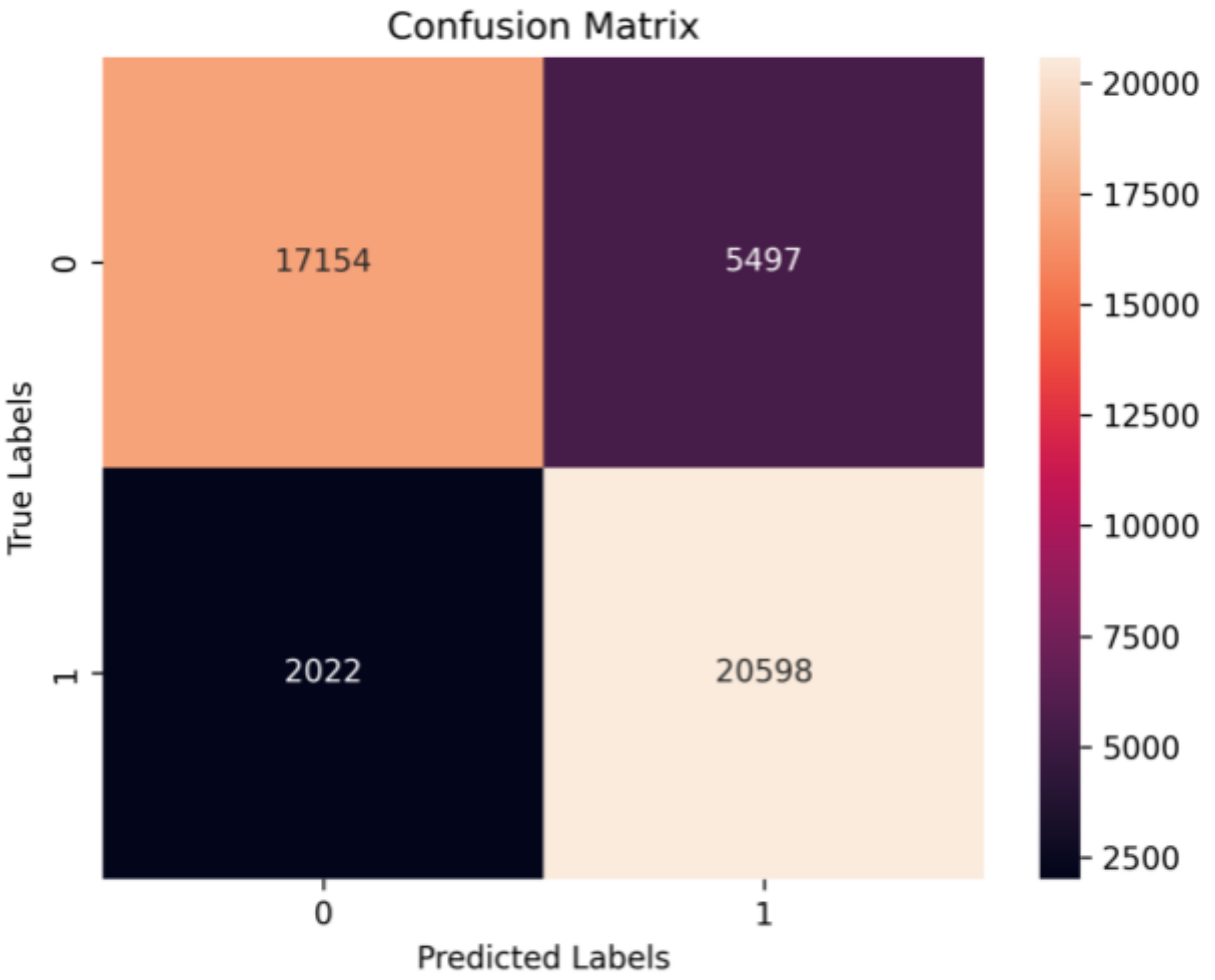

Figure 23: ANN Confusion Matrix

1.4 Model Comparison

Among all four models evaluated, it can be seen in Figure 24 that the Random Forest model stands out as the most accurate for the specific purpose of identifying instances related to heart attacks. It achieves the highest accuracy of 90%, demonstrating superior performance compared to the KNN, Decision Tree, and ANN models.The Random Forest model's precision, recall, and F1-score values consistently outperform the other models, making it the most reliable choice for this classification task.

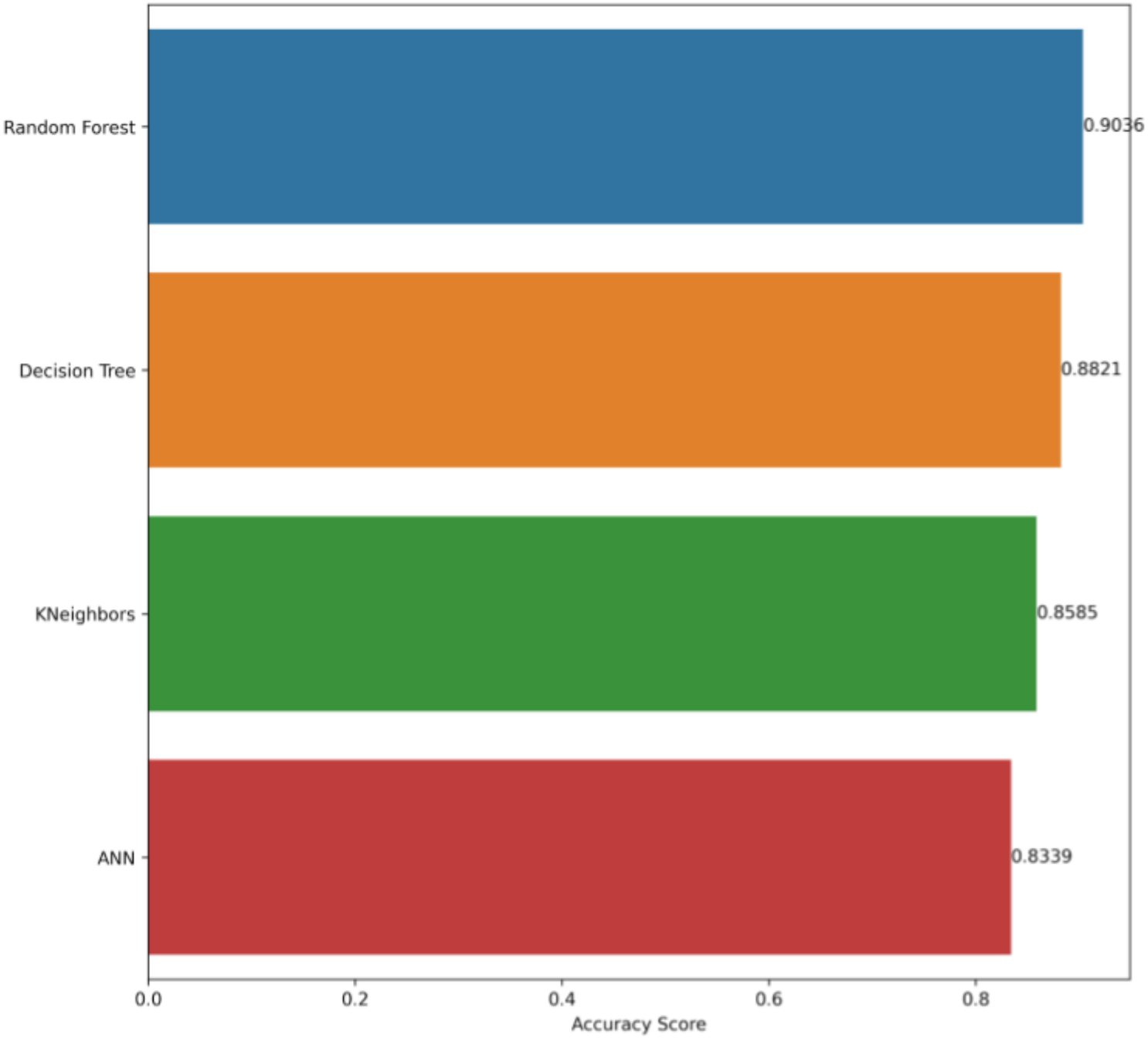

Figure 24: Model Comparison

1.5 Best Model Improvement

Improvement 1:

To enhance the accuracy of the Random Forest model, the initial approach involved augmenting the number of folds during cross-validation. The comparison in Figure 25 illustrates a discernible improvement, with a notable 1% increase in recall values for both classes. Consequently, this adjustment contributed to an overall boost in the model's accuracy.It can also be seen in Figure 26 that the model correctly classified more instances while decreasing the number of incorrectly classified instances in contrast with before .

Before Improvement

	precision	recall	f1-score	support
0	0.93	0.87	0.90	113176
1	0.88	0.93	0.91	113176
accuracy			0.90	226352
macro avg	0.91	0.90	0.90	226352
weighted avg	0.91	0.90	0.90	226352

Accuracy: 0.9035749628896586

After Improvement

	precision	recall	f1-score	support
0	0.93	0.88	0.90	113176
1	0.88	0.94	0.91	113176
accuracy			0.91	226352
macro avg	0.91	0.91	0.91	226352
weighted avg	0.91	0.91	0.91	226352

Accuracy: 0.9074406234537358

Figure 25: Before and After First Improvement

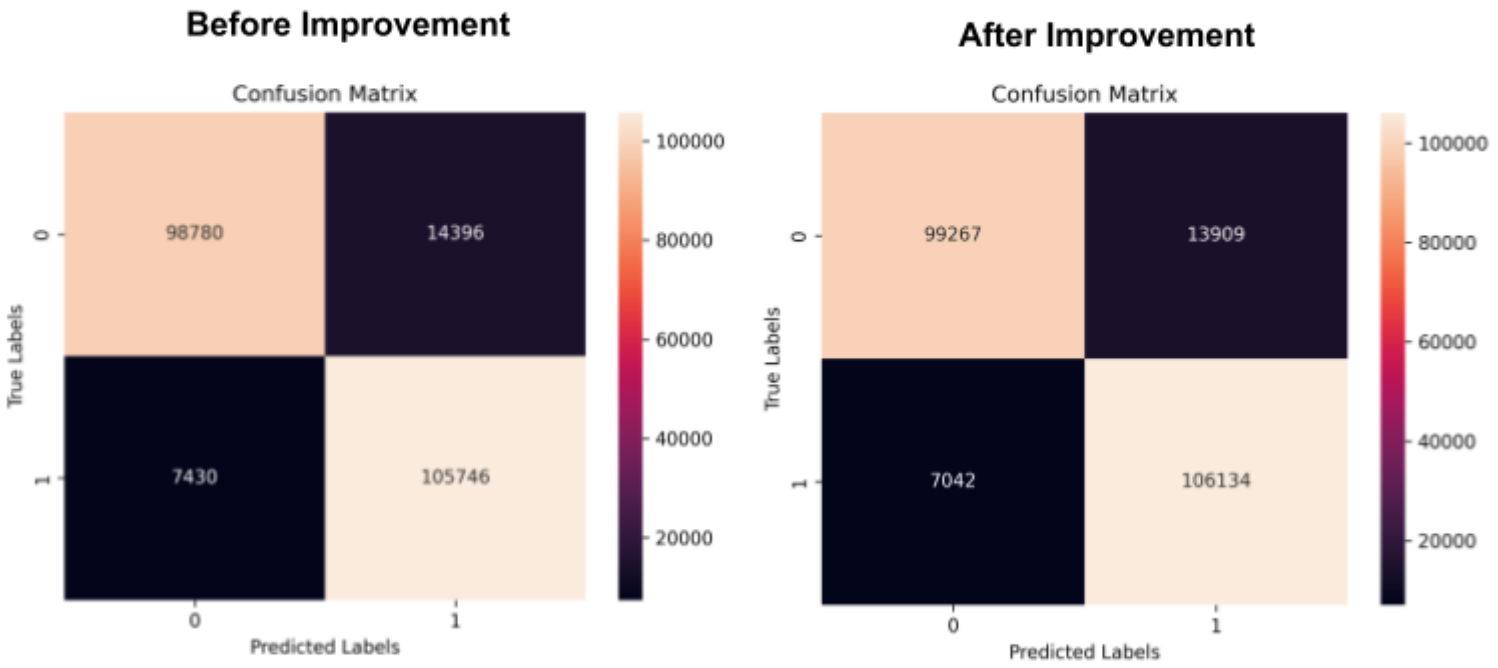

Figure 26: Before and After First Improvement

Improvement 2:

In the second attempt to enhance the original Random Forest model, hyperparameter tuning via the GridSearch method was employed. Optimal hyperparameters led to a modest improvement in overall accuracy (Figure 28), but the key advancement was observed in Figure 27, where False Positives increased, offset by a decrease in False Negatives. In the context of predicting heart attacks or heart disease, this strategic adjustment prioritises sensitivity, encouraging the model to prompt more check-ups for healthy individuals to minimise the risk of falsely reassuring those who may be at serious health risk. This underscores the model's commitment to public safety in heart health predictions.

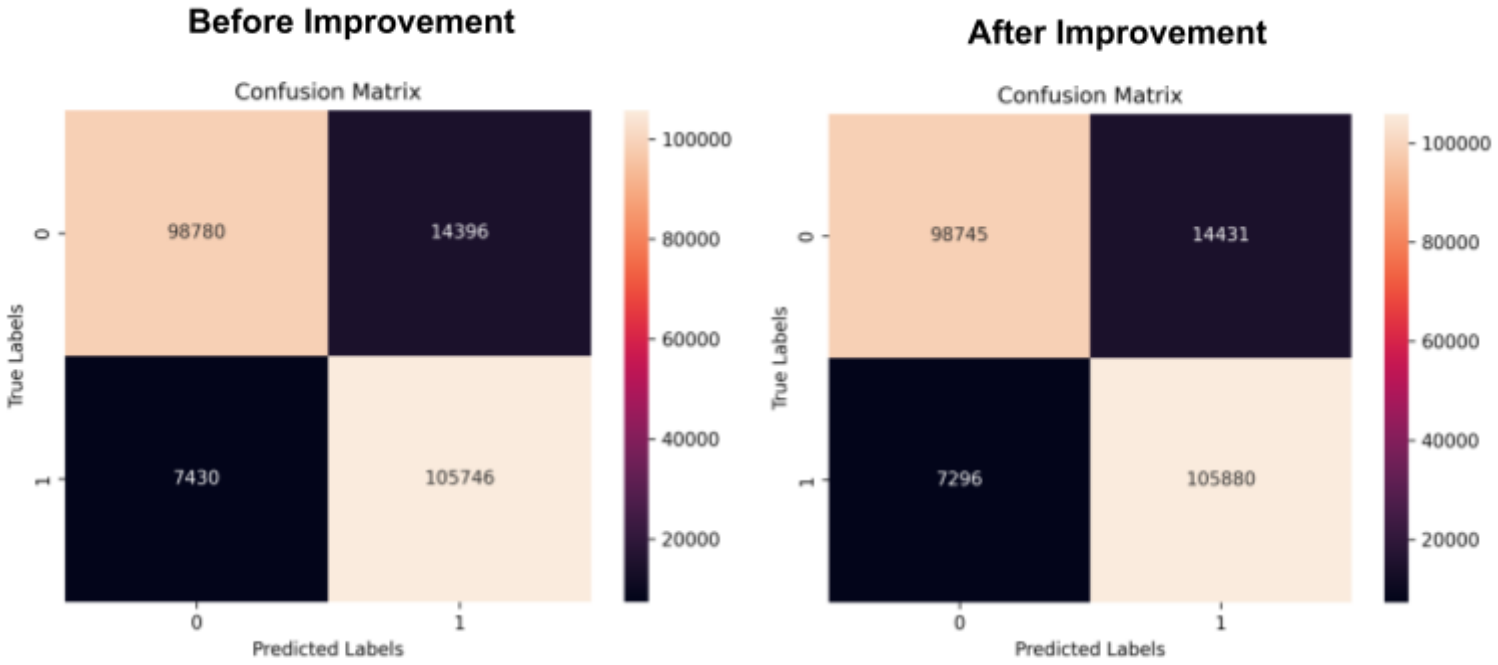

Figure 27: Before and After First Improvement

Before Improvement

	precision	recall	f1-score	support
0	0.93	0.87	0.90	113176
1	0.88	0.93	0.91	113176
accuracy			0.90	226352
macro avg	0.91	0.90	0.90	226352
weighted avg	0.91	0.90	0.90	226352

Accuracy: 0.9035749628896586

After Improvement

	precision	recall	f1-score	support
0	0.93	0.87	0.90	113176
1	0.88	0.94	0.91	113176
accuracy			0.90	226352
macro avg	0.91	0.90	0.90	226352
weighted avg	0.91	0.90	0.90	226352

Accuracy: 0.9040123347706227

Figure 28: Before and After First Improvement

Part 2: Traffic Light Recognition

2.1 Data Preparation

The data preparation for traffic light recognition using images and a Convolutional Neural Network (CNN) involved the following steps. Firstly, the dataset was mounted from Google Drive and unzipped. The necessary libraries, including TensorFlow and OpenCV, were imported. Two directories were defined for 'Red' and 'Green' traffic lights, and images were loaded and resized to a uniform size of 250x250 pixels. A function, `assign_label`, was created to assign labels based on the traffic light colour. The `make_train_data` function iterated through the images in each directory, assigned the corresponding label, and appended the resized images and labels to lists 'X' and 'Z,' respectively. Label encoding was performed using scikit-learn's `LabelEncoder`. The transformation from categorical to numeric labels using LabelEncoder is essential for machine learning models like CNNs, as they require numerical input for effective training. Assigning numeric values (1 for 'Red' and 0 for 'Green') enables the model to comprehend and learn patterns associated with different traffic light colours. The images were then normalised by dividing them by 255, normalisation is crucial to stabilise training, prevent disproportionate influence from large-scale features, and enhance model generalisation. The dataset was split into training and testing sets using `train_test_split`. To enhance the model's robustness, data augmentation was applied using Keras' `Sequential` API, including random horizontal flipping, rotation, zooming, and contrast adjustment. This augmented dataset was ready for training a CNN for traffic light recognition.

2.2 2D Convolutional Neural Network Design

The selected Convolutional Neural Network (CNN) architecture(Figure 29 and 30)(5) is purposefully designed to excel in the task of classifying the colour of traffic lights, distinguishing between red and green. Here's a rationale for each design choice:

Convolutional Layers with Increasing Filters:
Starting with 32 filters and progressively increasing to 96 allows the network to learn hierarchical features of varying complexities. Lower-level filters capture basic features like edges, while higher-level filters discern more intricate patterns crucial for discriminating between red and green traffic lights.
Kernel Sizes:
Larger kernel sizes (e.g., 5x5) in the initial layers enable capturing broader spatial features, while smaller kernel sizes (e.g., 3x3) in subsequent layers focus on more localised details. This is advantageous for traffic light images, as capturing both global and local features is essential for accurate classification.
MaxPooling Layers:
Employing max pooling layers after each convolutional layer reduces spatial dimensions, retains essential information, and enhances translational invariance. This is beneficial for recognizing colour patterns in traffic lights regardless of their position within the image.
Flatten Layer and Dense Layers:
The flatten layer is positioned to transform the spatial hierarchies learned by convolutional layers into a format suitable for dense layers. Dense layers, with increasing units, capture complex, non-linear relationships in the data, allowing the model to discern subtle differences in colour tones indicative of red or green traffic lights.

Dropout Layers:
Dropout layers with a rate of 0.1 introduce regularisation, preventing overfitting by randomly dropping connections during training. This is crucial for ensuring the model generalises well to new, unseen traffic light images and minimises the risk of memorising training samples.
Softmax Activation in Output Layer:
The softmax activation in the output layer produces a probability distribution over the two classes (red and green). This is essential for interpreting model predictions probabilistically and aiding decision-making based on the highest probability, enhancing the model's interpretability.

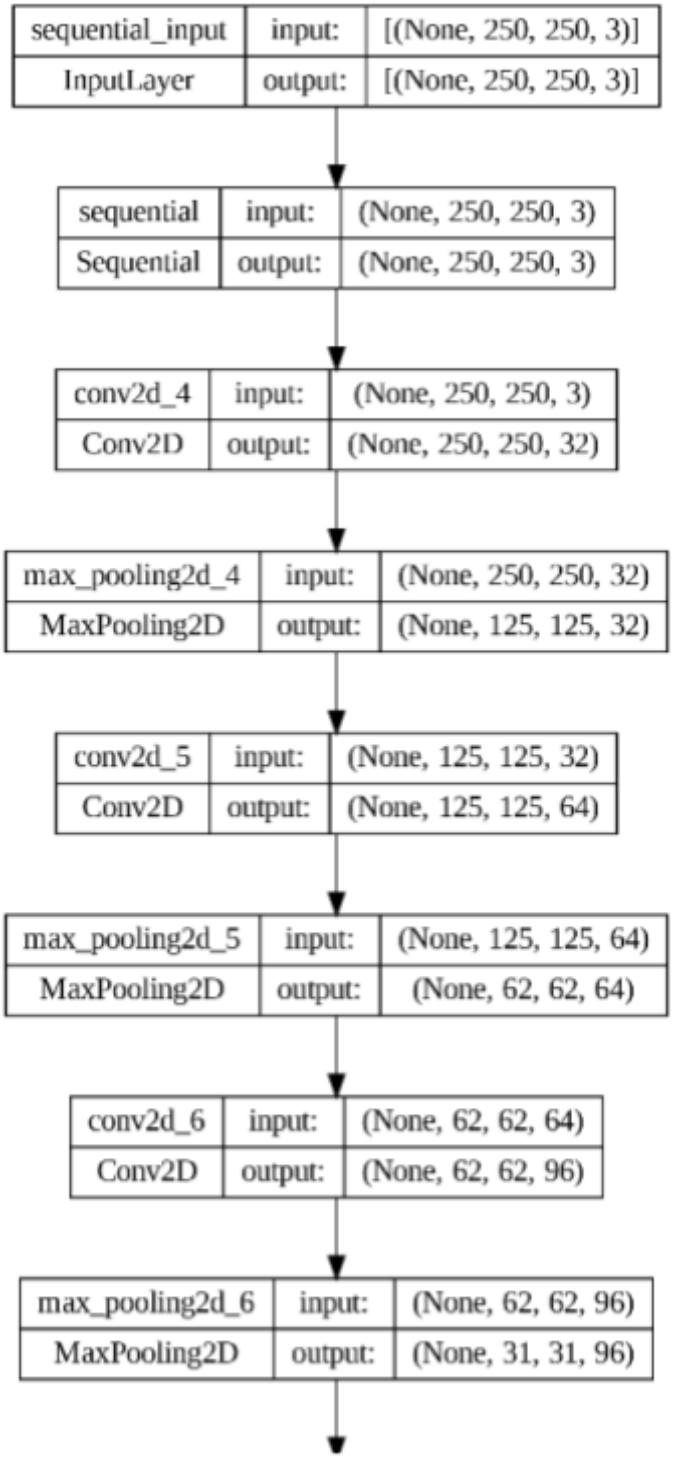

Figure 29: CNN Architecture Part 1

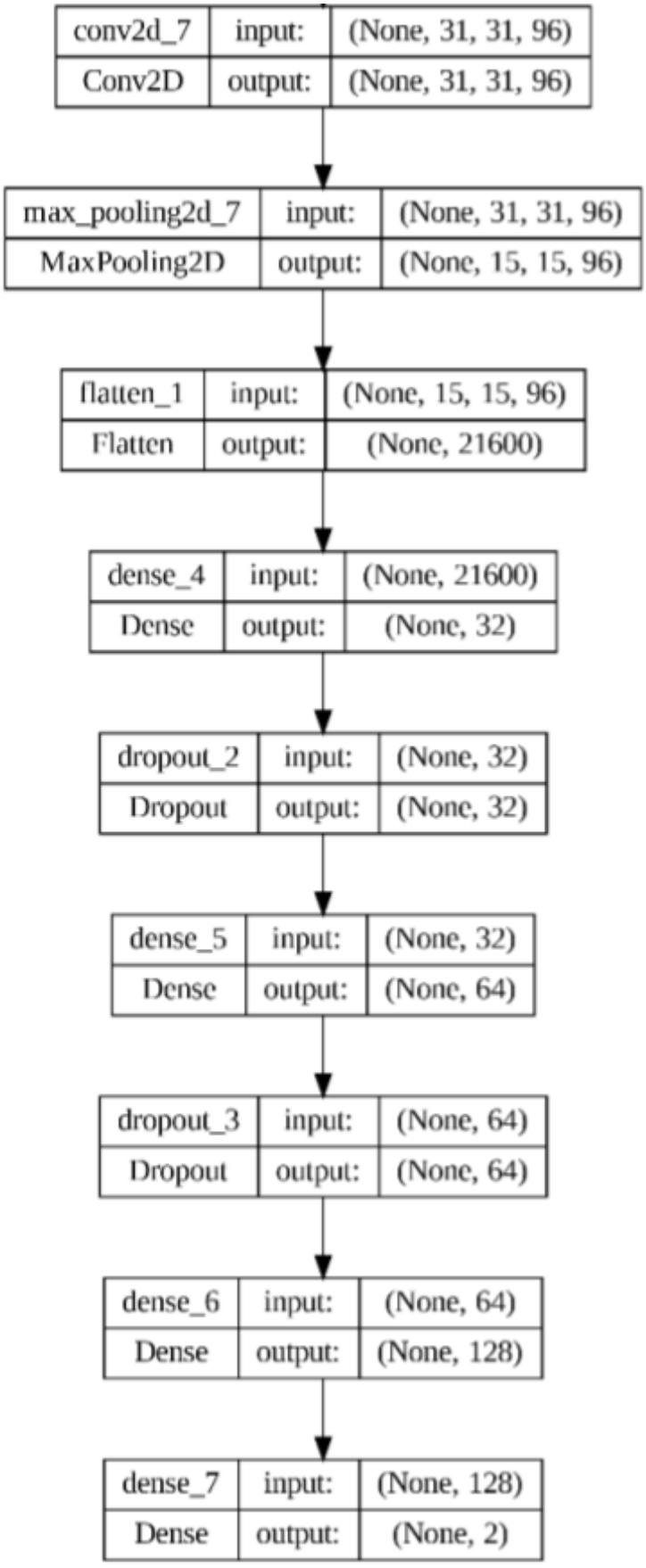

Figure 30: CNN Architecture Part 2

2.3 Model Training Loss and Accuracy

Training and Validation Accuracy(Figure 31):

The training accuracy starts at 43.75% and undergoes fluctuations in the initial epochs.

From epochs 8 to 13, there is a noticeable improvement, reaching 73.44%. This indicates that the model is learning well from the training data.

Epochs 14-22: The training accuracy fluctuates, ranging from 50% to 73.44%. This variability suggests that the model may be sensitive to certain patterns in the data.

Epochs 23-35: There is a steady increase in training accuracy, reaching a peak of 90.62%. This indicates that the model continues to learn and converge to a better solution.

Validation Accuracy: The validation accuracy shows similar patterns to the training accuracy, suggesting that the model is generalising reasonably well to unseen data.

The overall trend in both training and validation accuracy is positive, indicating that the model is capable of learning, improving and generalising over time.

The model's overall performance after the 35 epochs is 84% which can be seen in Figure 32.

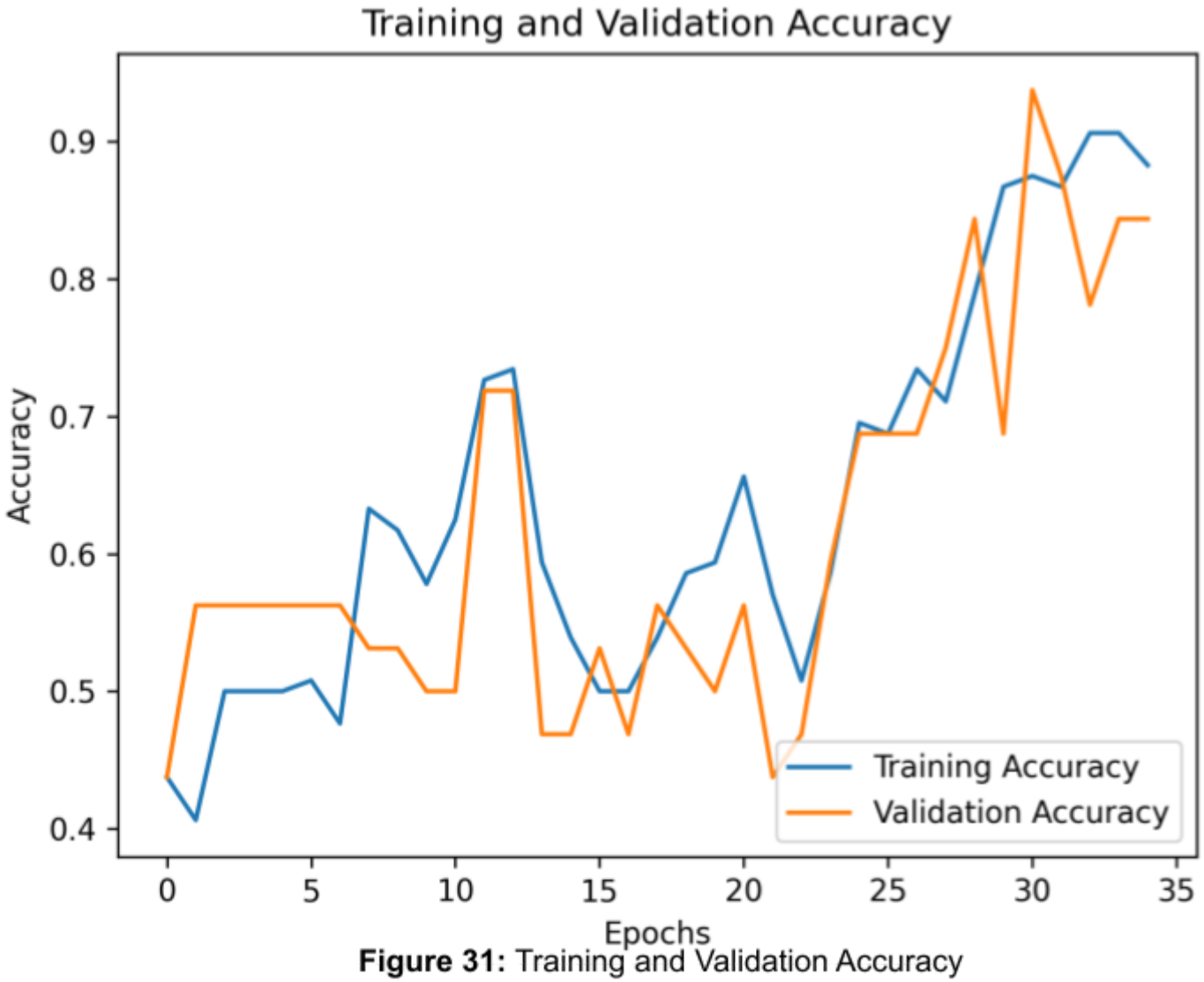

Figure 31: Training and Validation Accuracy

	Accuracy	Loss
Training	88%	33%
Validation	84%	43%

Figure 32: Model Performance at the end of Epoch 35

Training and Validation Loss(Figure 33):

Training Loss:The training loss starts relatively high, indicating that the model's initial predictions deviate significantly from the actual values. There is a gradual decrease in training loss over the initial epochs, suggesting that the model is improving its ability to fit the training data. However, there is a noticeable increase in training loss around epoch 11, leading to a subsequent fluctuation in the following epochs. This could be indicative of challenges in model convergence or sensitivity to specific data patterns. In the later epochs, the training loss exhibits a downward trend, reaching a minimum value, which is a positive sign of the model's capacity to learn from the training data. Validation Loss: The validation loss follows a similar pattern to the training loss, indicating that the model's performance on unseen data aligns with its performance on the training set. There is a significant spike in validation loss around epoch 13, coinciding with the increase in training loss. This might be a critical point where the model encounters challenges in generalising to new data. The subsequent decrease in validation loss indicates a recovery, with the model adjusting to the characteristics of the validation set. Towards the later epochs, the validation loss fluctuates but remains relatively lower compared to the earlier epochs, indicating sustained learning.

In conclusion, while facing challenges in convergence, the model demonstrates the ability to learn and adapt, as indicated by the decreasing trend in both training and validation loss towards the later epochs.

Figure 33: Training and Validation Loss

2.4 Image Testing

In this segment, five randomly selected images from the test set were presented to evaluate the model's proficiency in recognizing the colours of traffic lights. Drawing from the test set ensures that the model encounters previously unseen images, preventing any potential memorization and assessing its ability to generalise. The predictions yielded a binary classification, with 1 denoting red traffic lights and 0 indicating green traffic lights. As depicted in Figures 34-38, the model accurately identified both green and red traffic lights within the test set. This successful performance on previously unseen data underscores the model's robust generalisation and the absence of overfitting, affirming its reliability in real-world applications.

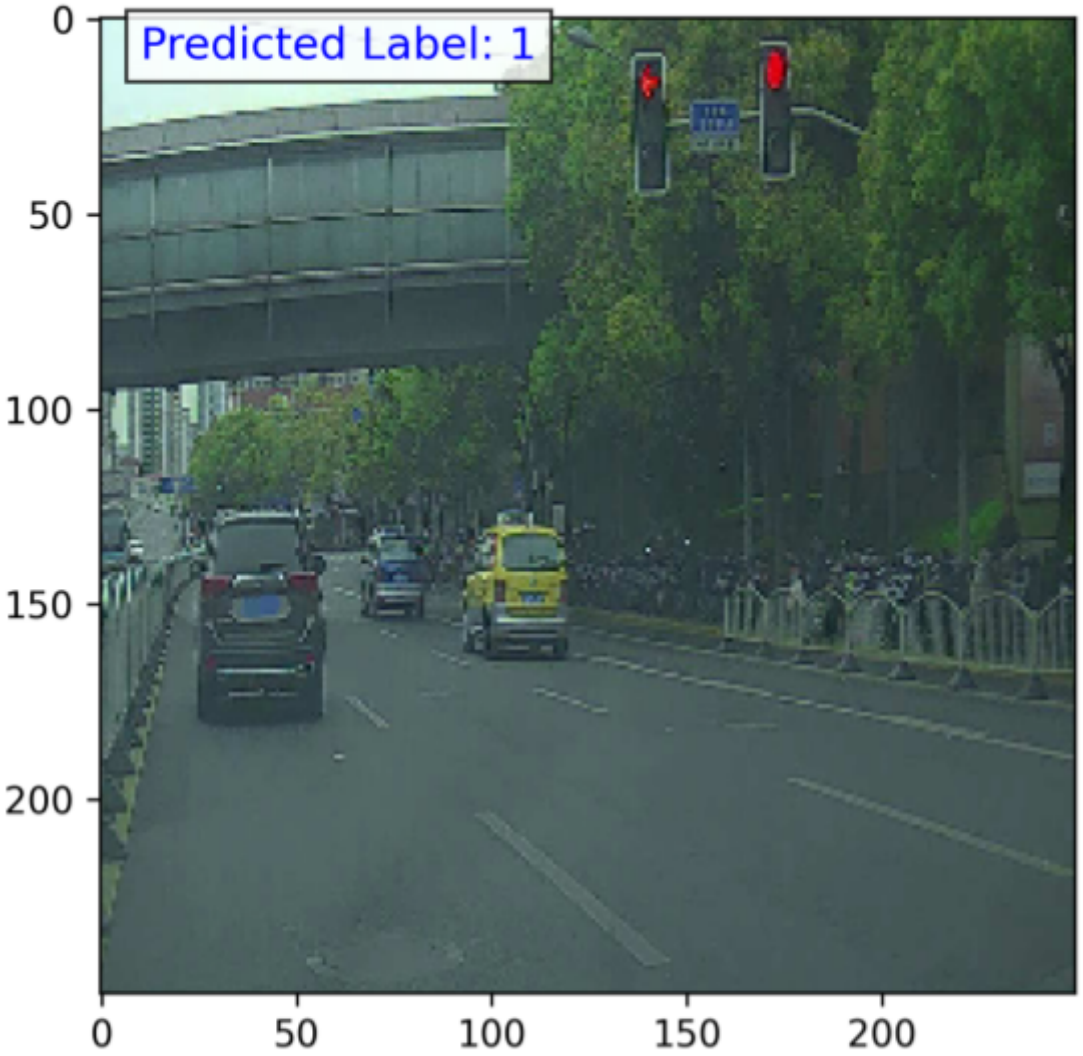

Figure 34: Test Image 1

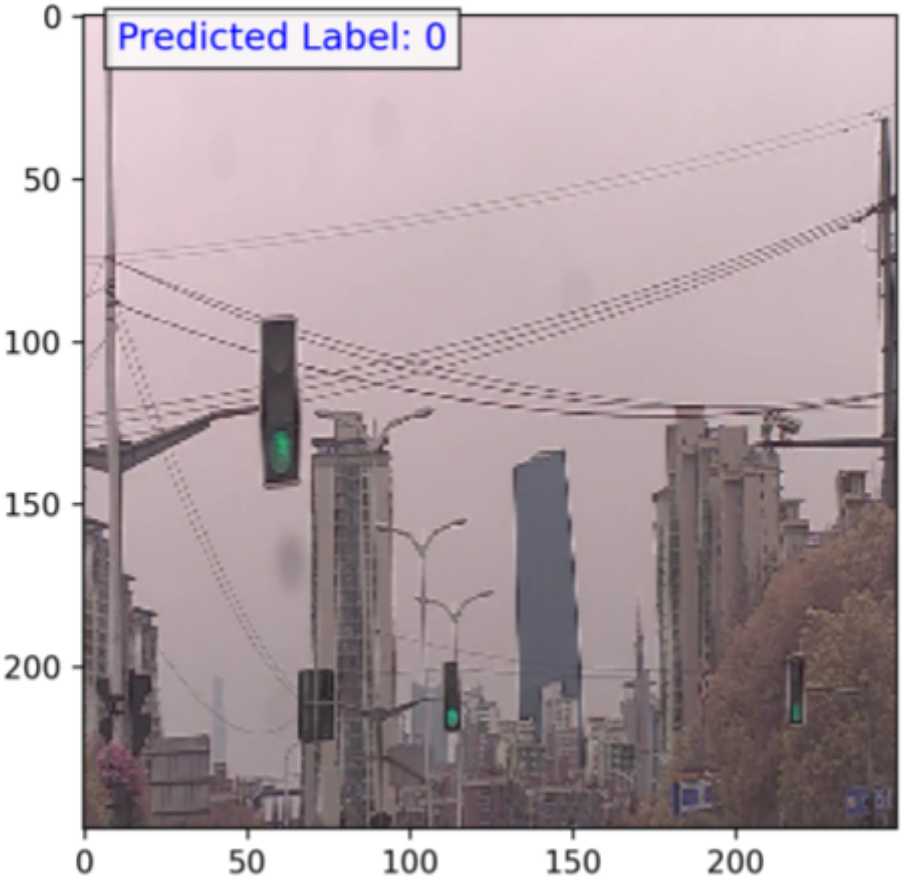

Figure 35: Test Image 2

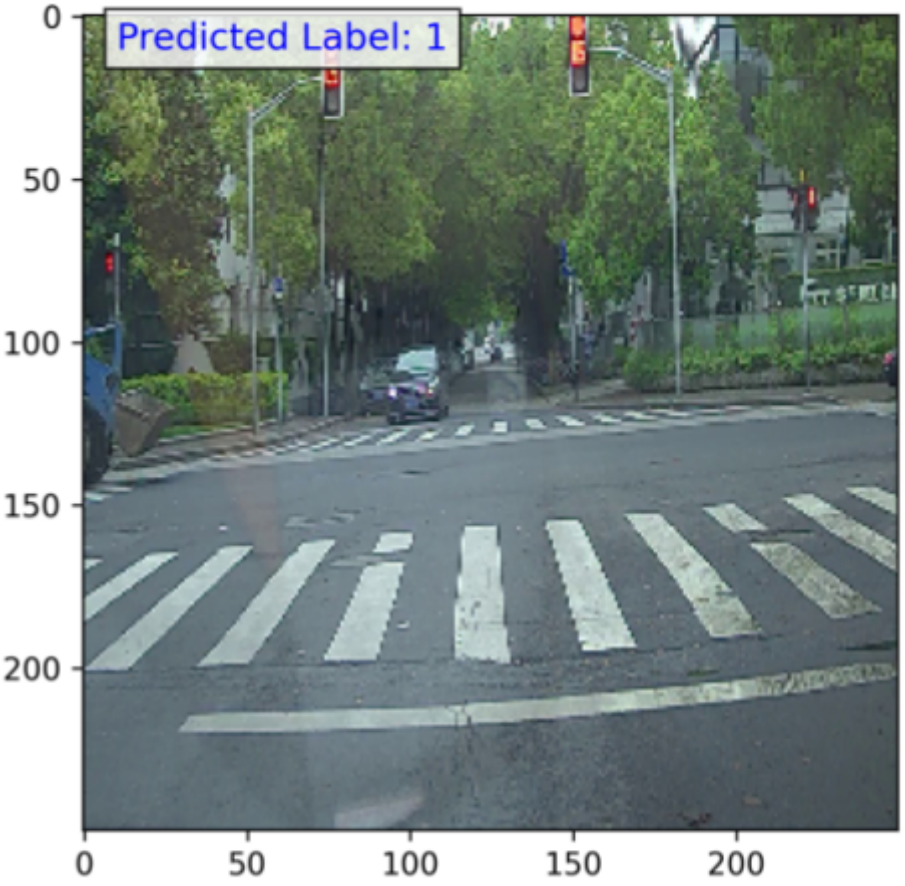

Figure 36: Test Image 3

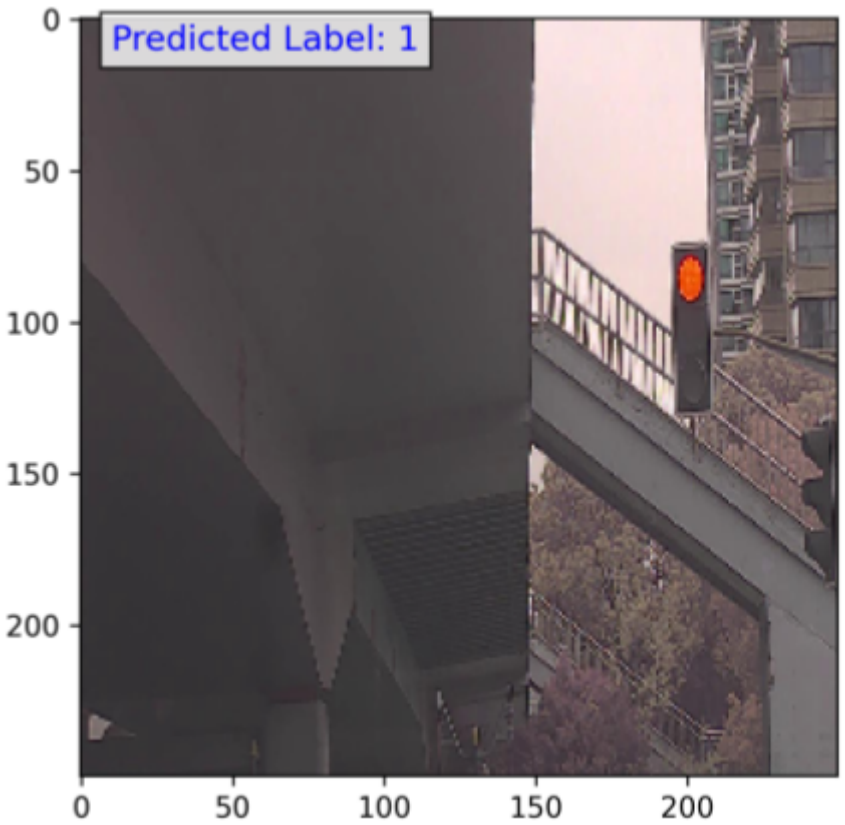

Figure 37: Test Image 4

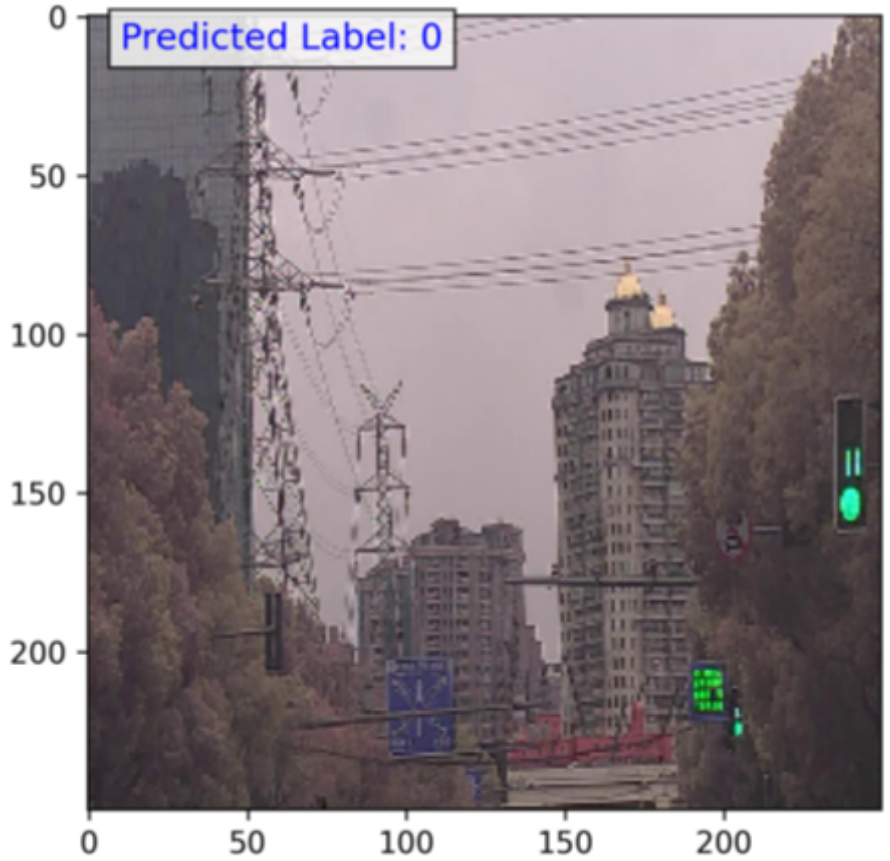

Figure 38: Test Image 5

2.5 Model Improvement

Improvement 1:

In the pursuit of an improved model, several modifications were made. A key adjustment involved increasing the filter count in the last convolutional layer from 96 to 128, enhancing the model's capacity to discern intricate features within input images. This deliberate augmentation aimed to bolster the model's proficiency in recognizing diverse traffic light states. The heightened filter count introduced additional learning parameters, potentially improving the model's ability to generalise across a wider array of patterns. Additionally, the number of training epochs was extended from 35 to 50, providing the model with more learning opportunities. Recognizing the increased risk of overfitting with longer training durations, the dropout rate was concurrently raised from 0.1 to 0.5. This adjustment serves as a preventative measure against overfitting, ensuring the model's learning process encourages a diverse set of features while mitigating the risk of memorising specific patterns in the training data. As it can be seen in Figures 39,40 and 41 the model's accuracy successfully increased from 84% to 94%.

	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Original Model	88%	33%	84%	43%
Improved Model 1	95%	10%	94%	17%

Figure 39: Performance comparison

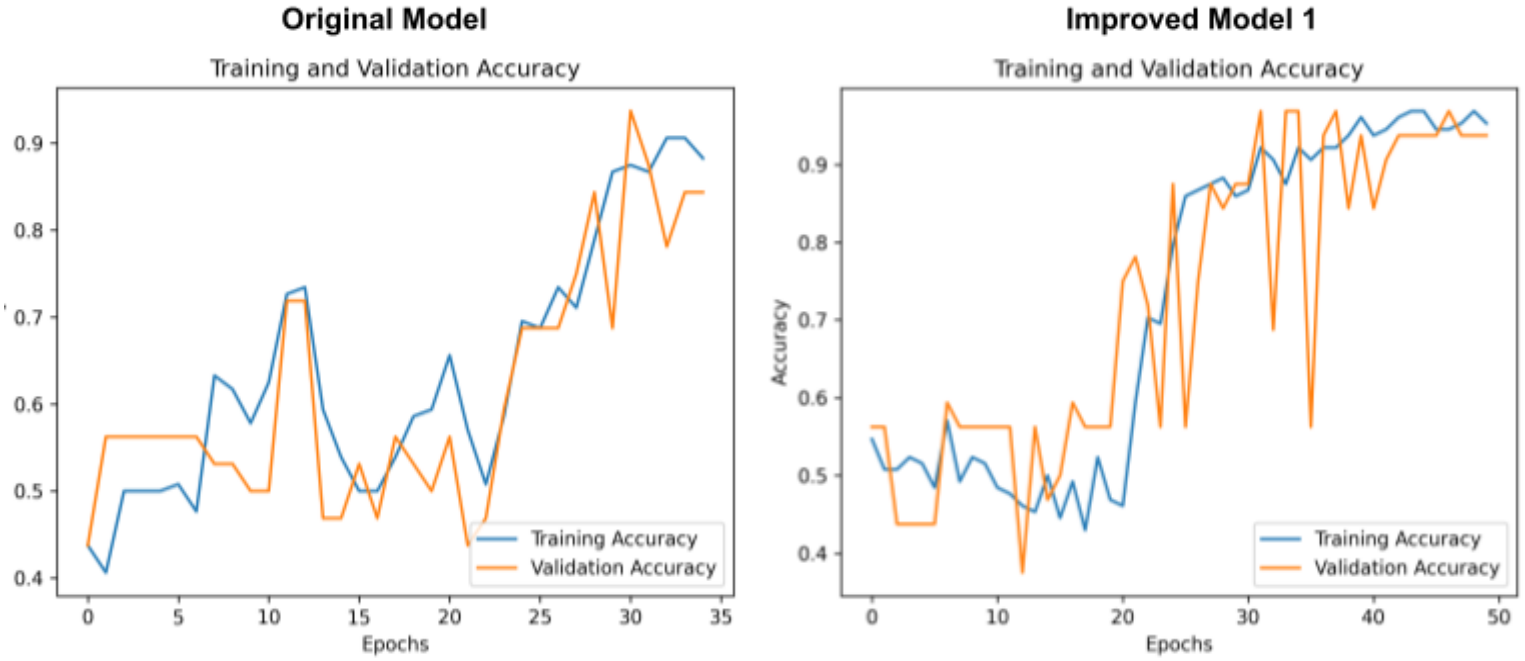

Figure 40: Training and Validation Accuracy Comparison

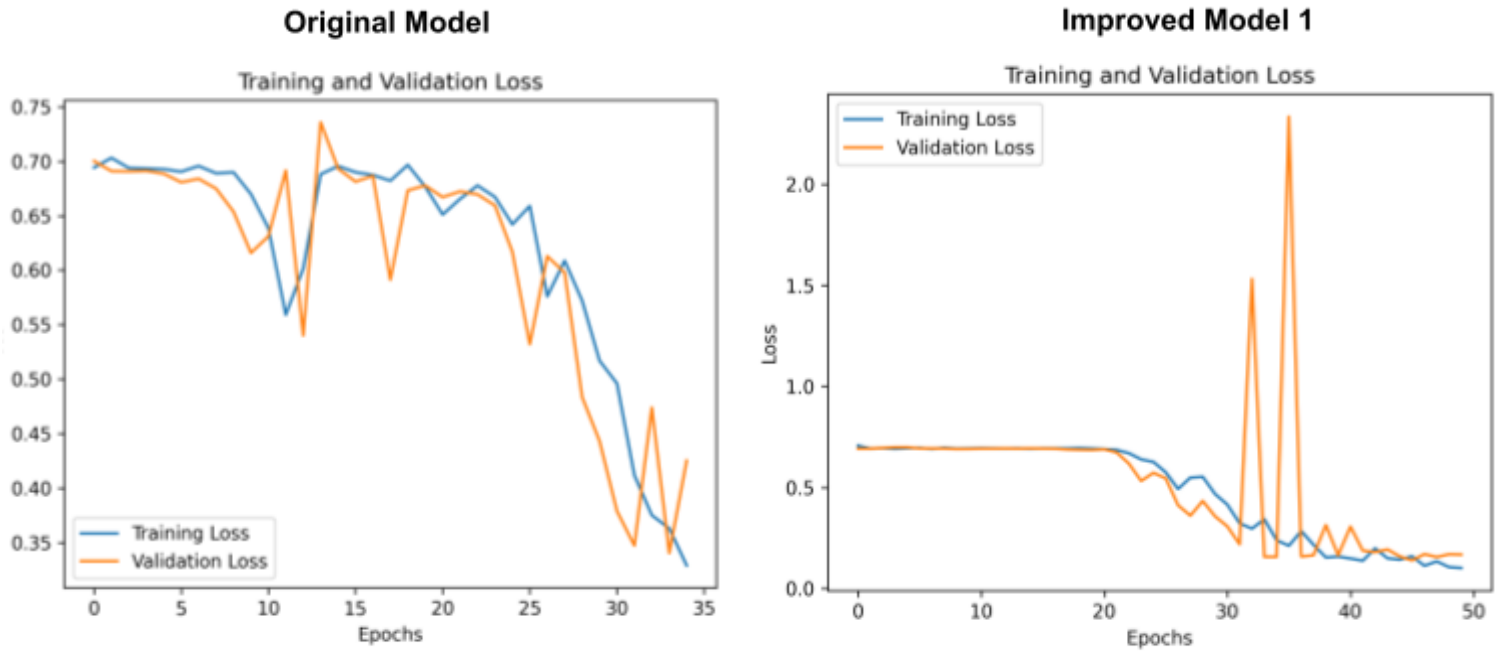

Figure 41: Training and Validation Loss Comparison

Improvement 2:

In this improvement the most significant modification that took place was the addition of a hidden layer with 508 neurons, the addition of a hidden layer could enhance the model's capacity to capture hierarchical and abstract features, potentially improving its ability to discern complex patterns in traffic light images. And as the previous improvement this model's numbers of epochs increased from 35 to 50 in order to give the model more time to learn the dataset and along with this the dropout rate also increased to reduce the chance of the model memorising the data and then overfit. It can be seen in Figures 42,43 and 44 that those modifications lead to a more accurate model as the accuracy increased from 84% to 97%.

	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Original Model	88%	33%	84%	43%
Improved Model 2	95%	17%	97%	18%

Figure 42: Metrics Comparison

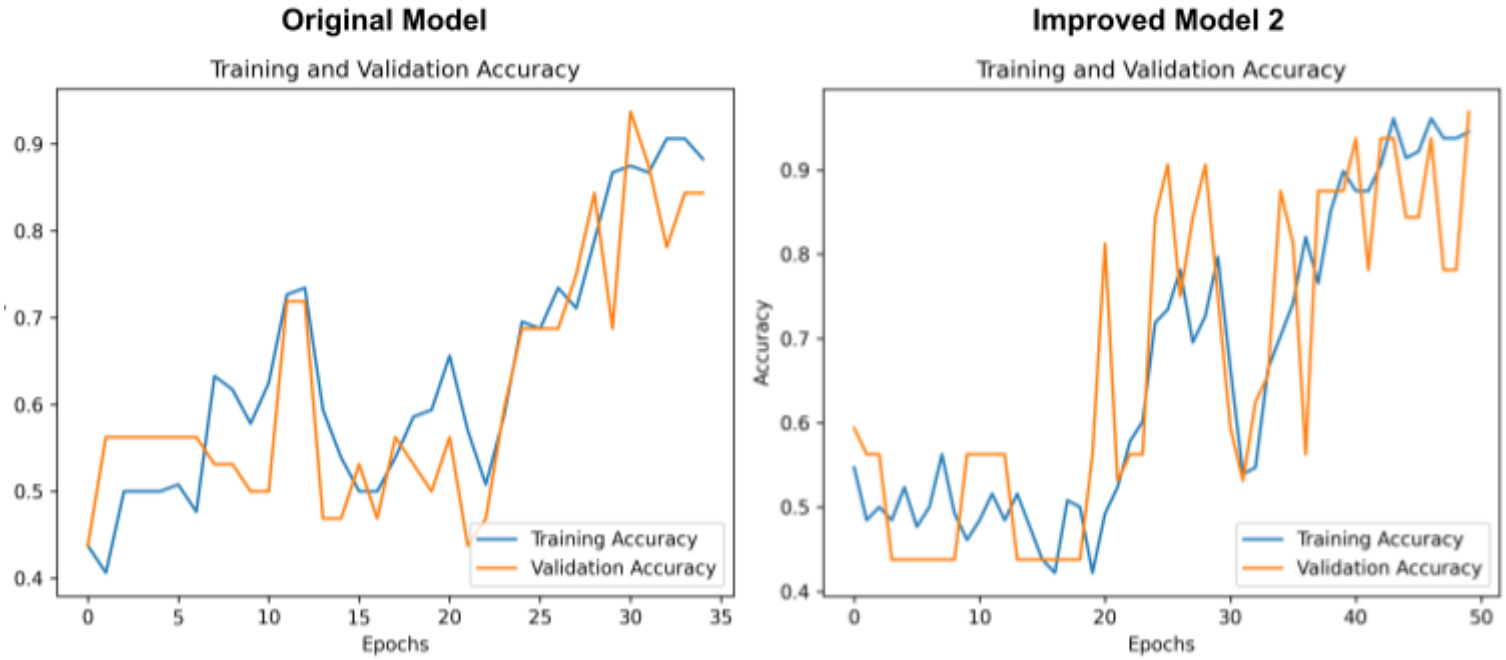

Figure 43: Training and Validation Accuracy Comparison

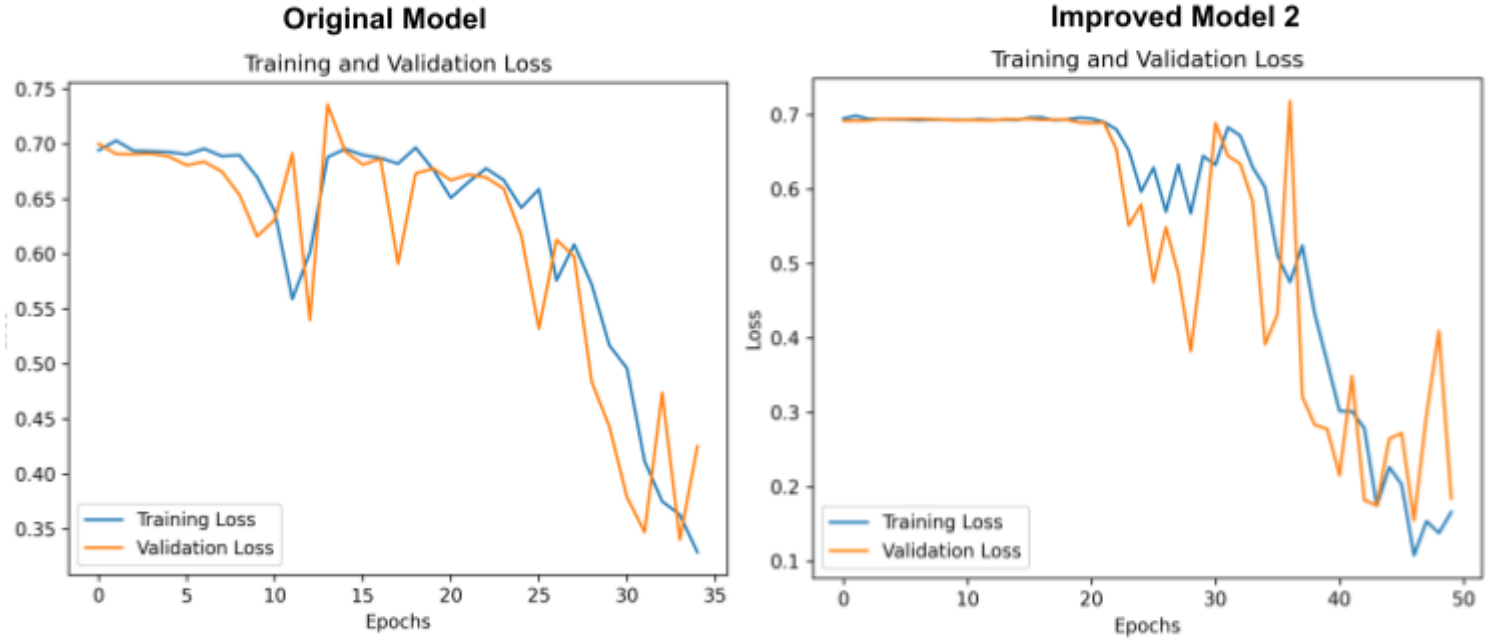

Figure 44: Training and Validation Loss Comparison

References

1. Sklearn.neighbors.kneighborsclassifier. scikit https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier
2. 3.2. tuning the hyper-parameters of an estimator. scikit. https://scikit-learn.org/stable/modules/grid_search.html#exhaustive-grid-search
3. Sklearn.tree.decisiontreeclassifier. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn-tree-decisiontreeclassifier
4. Team, K. Keras Documentation: The sequential class. https://keras.io/api/models/sequential/
5. Tf.keras.utils.plot_model : tensorflow V2.14.0. TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model