Pima Indians Diabetes Database

Kaggle Link- Pima Indians Diabetes Database

Google Colab Notebook Link- Pima Indians Diabetes Database.ipynb

Dataset Explanation:

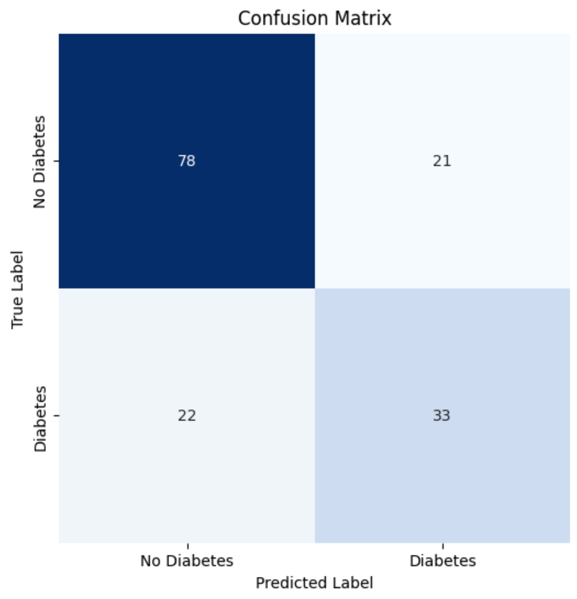
The dataset consists of columns like number of pregnancies, glucose levels, blood pressure, skin thickness, body mass index, diabetes and age of the person along with a final column named as "outcome" which consists of values 0 or 1 representing whether the person contains diabetes or not. This dataset is strictly with respect to females at least 21 years old of Pima Indian heritage. The dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. It has 768 rows.

Different approaches towards building accuracy:

1. Basic Neural Network Implementation

First, I applied a basic neural network on the selected dataset which gave 72.08% validation accuracy. It has 100 epochs.It was implemented using 25 neurons using relu and sigmoid activations.

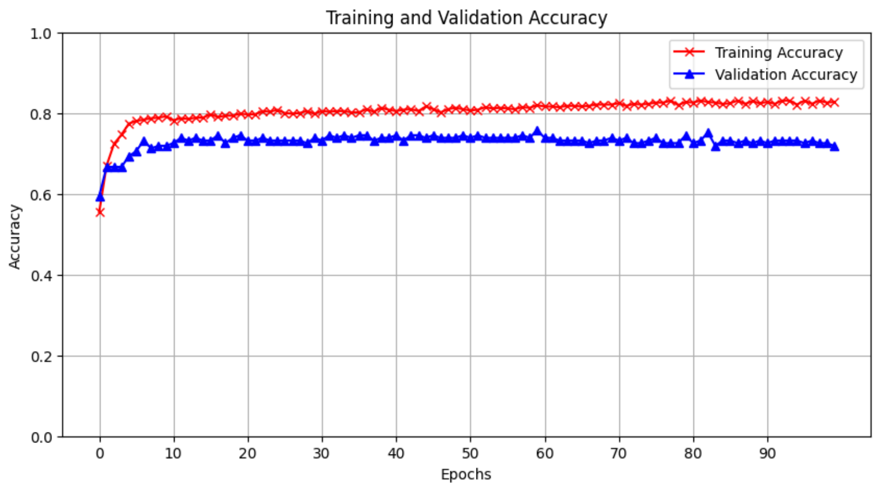
Confusion Matrix:



True Negatives (Top-left: 78): These are cases where the model correctly predicted "No Diabetes." The true label is "No Diabetes," and the model predicted it correctly.
False Positives (Top-right: 21): These are cases where the model incorrectly predicted "Diabetes," but the actual outcome was "No Diabetes." These are false alarms, where the model is over-predicting diabetes.

False Negatives (Bottom-left: 22): These are cases where the model incorrectly predicted "No Diabetes," but the actual outcome was "Diabetes." These are missed cases of diabetes, where the model is under-predicting diabetes.
True Positives (Bottom-right: 33): These are cases where the model correctly predicted "Diabetes." The true label is "Diabetes," and the model predicted it correctly.

Training and Validation Accuracy Graph

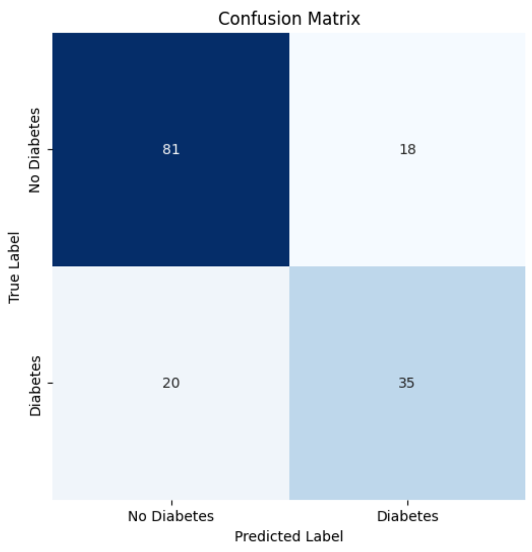


The graph shows the training and validation accuracy over 90 epochs. The red line represents the training accuracy, which improves quickly and stabilizes around 85%. The blue line represents validation accuracy, which initially increases but stabilizes at a lower value, around 65%. The gap between training and validation accuracy suggests potential overfitting, as the model performs better on the training data than on the validation data.

2. L2 Regularization to Improve Accuracy

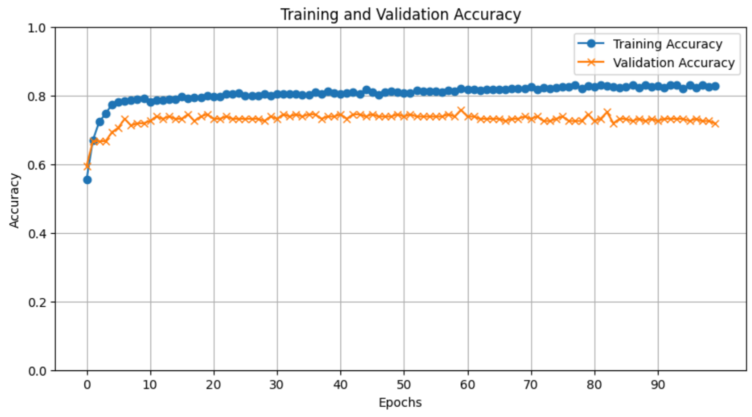
Following this, the next approach was applying L2 regularization with the lambda value being 0.01. The epochs were 100. The accuracy increased to 75.32%.

Confusion Matrix:



True Negatives (Top-left: 81): These are cases where the model correctly predicted "No Diabetes." The true label is "No Diabetes," and the model predicted it correctly.
False Positives (Top-right: 18): These are cases where the model incorrectly predicted "Diabetes," but the actual outcome was "No Diabetes." These are false alarms, where the model is over-predicting diabetes.
False Negatives (Bottom-left: 20): These are cases where the model incorrectly predicted "No Diabetes," but the actual outcome was "Diabetes." These are missed cases of diabetes, where the model is under-predicting diabetes.
True Positives (Bottom-right: 35): These are cases where the model correctly predicted "Diabetes." The true label is "Diabetes," and the model predicted it correctly.

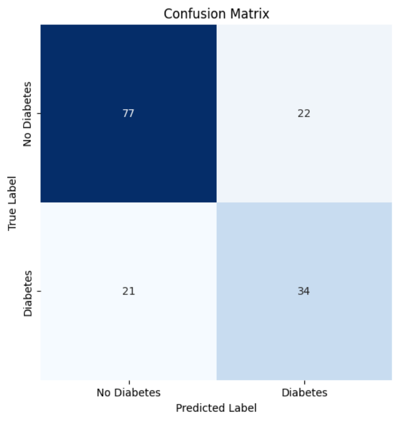
Training and Validation Accuracy Graph



The plot displays the training and validation accuracy over the epochs during the model training process. The gap between the training and validation accuracy indicates that while the model performs well on the training set, it may not generalize effectively to unseen data, as evidenced by the relatively lower validation accuracy. This discrepancy could prompt further tuning of the model as we have done in the next step.

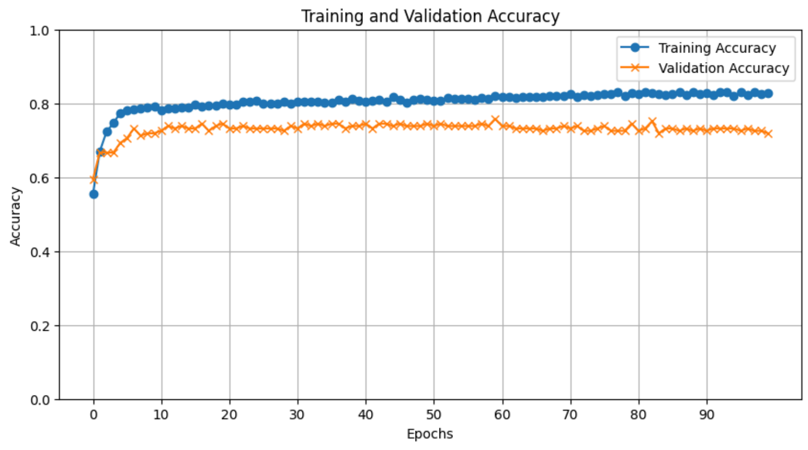
3. Random Forest Classifier for Improved Performance

Next, I decided to apply a Random Forest classifier expecting the accuracy to increase which did increase but by a very small amount. The validation accuracy was 72.08%.
Confusion Matrix:



True Negatives (Top-left: 77): These are cases where the model correctly predicted "No Diabetes." The true label is "No Diabetes," and the model predicted it correctly.
False Positives (Top-right: 22): These are cases where the model incorrectly predicted "Diabetes," but the actual outcome was "No Diabetes." These are false alarms, where the model is over-predicting diabetes.
False Negatives (Bottom-left: 21): These are cases where the model incorrectly predicted "No Diabetes," but the actual outcome was "Diabetes." These are missed cases of diabetes, where the model is under-predicting diabetes.
True Positives (Bottom-right: 34): These are cases where the model correctly predicted "Diabetes." The true label is "Diabetes," and the model predicted it correctly.

Training and Validation Accuracy Graph



Overfitting: The significant gap between training and validation accuracy suggests that the model is likely overfitting. This means it has learned the training data too well, including its noise and outliers, but does not perform as well on new, unseen data.
Generalization: A model that generalizes well will have training and validation accuracy that are closer in value. The current situation indicates that further tuning might be necessary to improve the model's ability to generalize.
Hence, we are proceeding to the next approach.

4. K-Fold Cross-Validation to Enhance Generalization

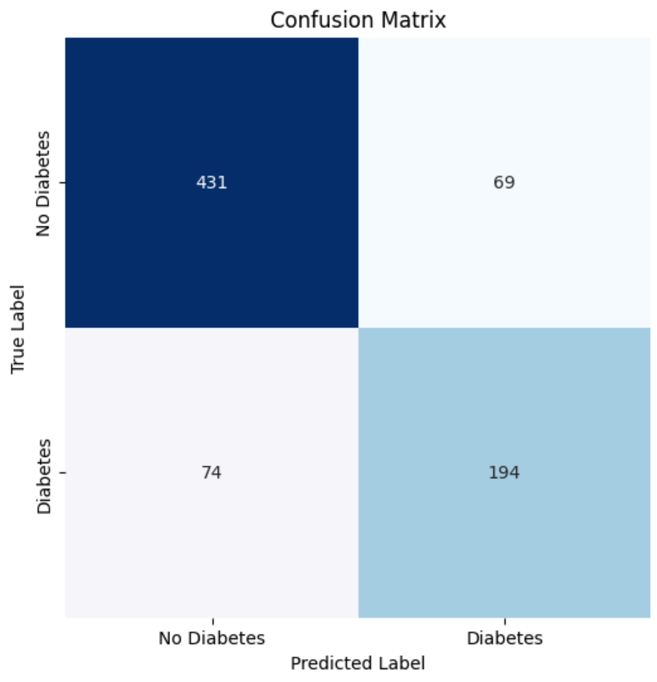
After this, the next approach was k fold iterations where k was taken as 5 with 100 epochs each.

The validation accuracies are as follows:
```
Training on fold 1...
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7468 -
loss: 0.5788
Validation Accuracy for fold 1: 72.08%

Training on fold 2...
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7576 -
loss: 0.6105
Validation Accuracy for fold 2: 77.27%
```

```
Training on fold 3...
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6912 -
loss: 0.5960
Validation Accuracy for fold 3: 73.38%

Training on fold 4...
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7258 -
loss: 0.4792
Validation Accuracy for fold 4: 71.90%

Training on fold 5...
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7054 -
loss: 0.4760
Validation Accuracy for fold 5: 71.90%

24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.7863
- loss: 0.4161
Final Accuracy on the entire dataset: 81.38%
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Confusion Matrix:
[[431  69]
 [ 74 194]]
```

So, 81.38% is a good improvement!

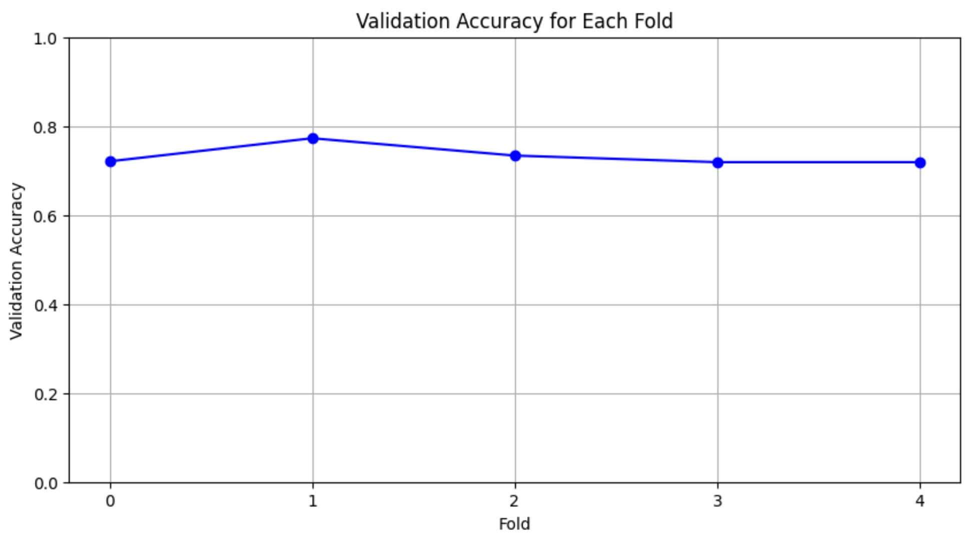
Confusion Matrix



True Negatives (Top-left: 431): These are cases where the model correctly predicted "No Diabetes." The true label is "No Diabetes," and the model predicted it correctly.

False Positives (Top-right: 69): These are cases where the model incorrectly predicted "Diabetes," but the actual outcome was "No Diabetes." These are false alarms, where the model is over-predicting diabetes.
False Negatives (Bottom-left: 74): These are cases where the model incorrectly predicted "No Diabetes," but the actual outcome was "Diabetes." These are missed cases of diabetes, where the model is under-predicting diabetes.
True Positives (Bottom-right: 194): These are cases where the model correctly predicted "Diabetes." The true label is "Diabetes," and the model predicted it correctly.

Accuracy Graph



The plot illustrates the validation accuracy for each fold in a cross-validation process. Each point represents the accuracy achieved on the validation set during that fold, while the connecting line shows the trend across folds. The validation accuracy appears to fluctuate slightly, suggesting some variability in model performance across different subsets of the data. Overall, it indicates that the model maintains a reasonably high accuracy throughout the folds, hovering around 0.8.

5. SMOTE and Hyperparameter Tuning with Random Forest

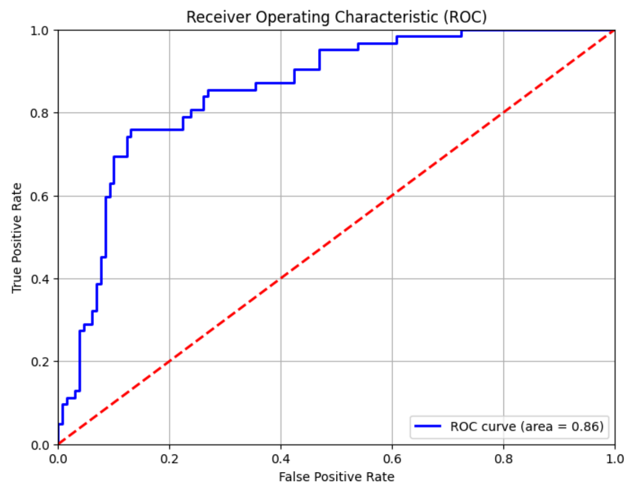
In the final approach, SMOTE (Synthetic Minority Over-sampling Technique) is used to oversample the minority class (diabetic cases). This helps to ensure the model does not get biased toward the majority class. Feature scaling is used with StandardScaler that normalizes the data, which is essential for models like Random Forest to ensure all features contribute equally. I have chosen Random Forest Classifier as it is robust for tabular data and can handle non-linear relationships. Further, Hyperparameter Tuning is done using GridSearchCV, the code performs a grid search over several hyperparameters (e.g., n_estimators, max_depth) to find the best-performing model. This gives us a validation accuracy of 83%.

ROC(Receiver Operator Characteristic Curve) Curve Interpretation:

The area under the ROC curve (AUC) is calculated to quantify the overall performance of the model. In this case, the AUC is 0.86.

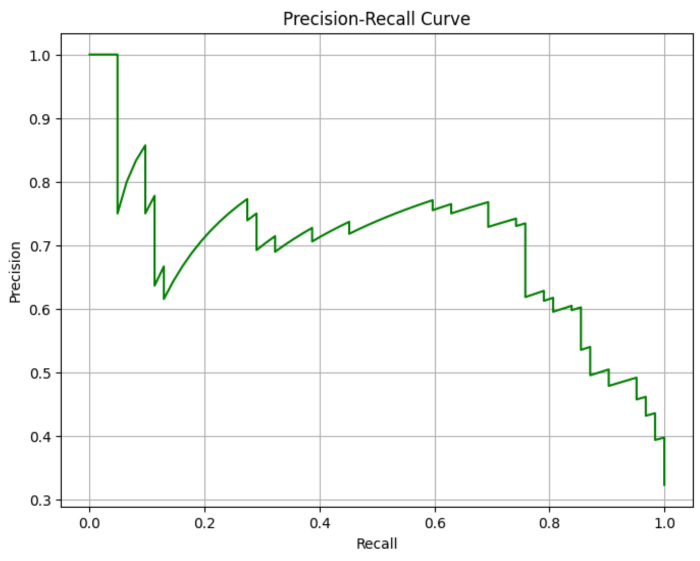
An AUC of 0.86 indicates that the model is quite good at distinguishing between the positive and negative classes. An AUC of 1.0 represents a perfect classifier, while an AUC of 0.5 represents a model that performs no better than random guessing.

So, an AUC of 0.86 suggests that the model has a strong ability to discriminate between patients with and without diabetes. It’s generally considered a good score.

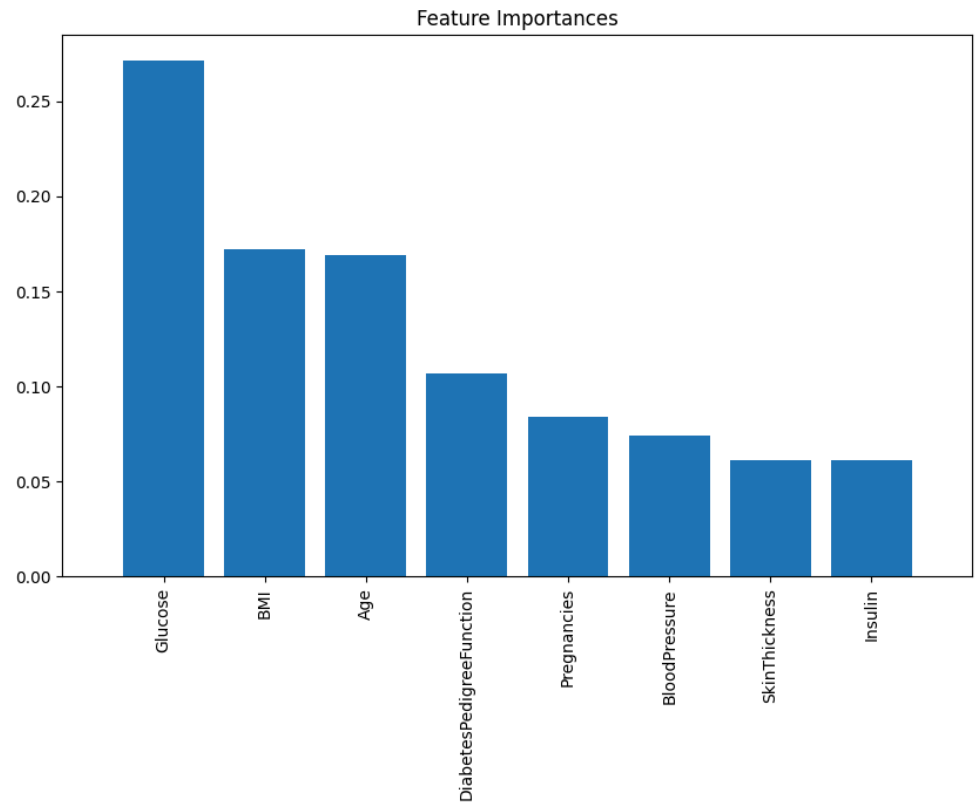


Precision Recall Curve

The Precision-Recall (PR) curve illustrates the trade-off between precision and recall for diabetes prediction model. Initially, precision is high (near 1.0) with low recall, indicating that the model is confident in its predictions but identifies few positive cases. As recall increases, precision drops, reflecting the inclusion of more false positives. The curve stabilizes around a precision of 0.4 to 0.5 as recall approaches 1.0, suggesting that while the model captures most positive cases, it also misclassified many.



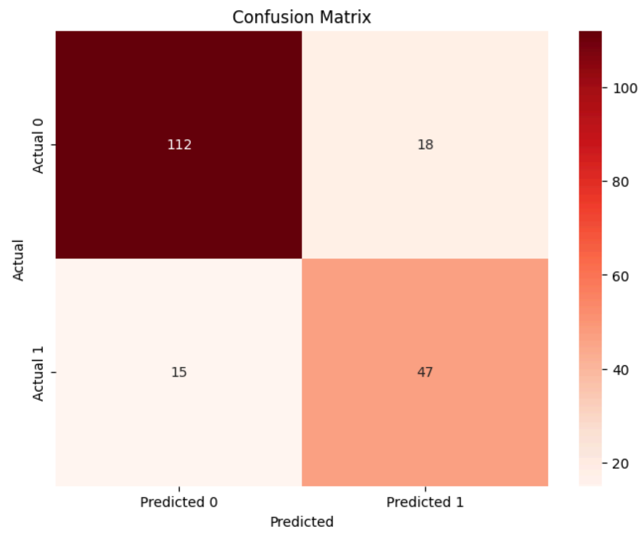
Feature Importances



```
Feature ranking:

1. Glucose (0.2713)

2. BMI (0.1722)

3. Age (0.1689)

4. DiabetesPedigreeFunction (0.1069)

5. Pregnancies (0.0841)

6. BloodPressure (0.0743)

7. SkinThickness (0.0613)

8. Insulin (0.0611)
```

It shows the relative importance of each feature in predicting diabetes. Glucose which has 0.2713 has most power in predicting diabetes hence it is strongly correlated with diabetes followed by BMI, Age and so on.

Confusion Matrix:



True Negatives (TN): The top-left cell (112) indicates that 112 instances were correctly predicted as negative (no diabetes).

False Positives (FP): The top-right cell (18) shows that the model incorrectly predicted 18 instances as positive (diabetes) when they were actually negative.

False Negatives (FN): The bottom-left cell (15) represents 15 instances that were incorrectly predicted as negative when they were positive.

True Positives (TP): The bottom-right cell (47) indicates that the model correctly predicted 47 instances as positive (diabetes).

Overall Approaches Observations:

- **Neural Network + Regularization:** Starting with a basic neural network and adding L2 regularization improved generalization and reduced overfitting.
- **Random Forest:** Initially showed minor improvements but needed tuning and class balancing.
- **Cross-Validation:** Helped provide a more reliable estimate of model performance and improved accuracy to over 81%.
- **SMOTE & GridSearchCV:** The use of SMOTE for handling class imbalance, along with hyperparameter tuning, further refined the model to achieve the highest validation accuracy (83%).

Conclusions:

By applying a combination of techniques—starting with regularization in neural networks to improve generalization, using Random Forest with tuning, and employing Cross-Validation to ensure robust performance metrics—you systematically improved model accuracy. The use of SMOTE to handle class imbalance, combined with GridSearchCV for hyperparameter optimization, ultimately led to the highest validation accuracy of 83%. This iterative approach highlights the importance of addressing overfitting, class imbalance, and tuning parameters to enhance model performance and reliability.