International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 10, pp 1-8, October-2025

https://doi.org/10.47001/IRJIET/2025.910001

Application of Feed Forward and Convolutional Neural Network Models in Diabetes Detection and Prediction Using Classification Algorithms

[1]*Kelvin Nnamani, [2]Kenneth Akpado, [3]Stephen Ufoaroh

[1,2,3]Department of Electronic and Computer Engineering, Nnamdi Azikiwe University, Awka, Anambra State, Nigeria

***Abstract* - This study investigates the use of feedforward and convolutional neural networks for diabetes detection and prediction through classification algorithms. It utilizes a dataset of 495 records from various medical centers in Nigeria, which includes factors such as age, cholesterol levels, blood glucose levels, body mass index, physical activity, family history, and alcohol consumption. The data was split into 70% for training and 30% for testing, employing a multi-layer perceptron (MLP) classifier with Python and TensorFlow Keras. The MLP achieved an accuracy of 70%, while the convolutional neural network reached an accuracy of 71% after 500 epochs. The findings also indicated that 31.52% of alcohol consumers and 68.48% of non-consumers were affected by diabetes, highlighting alcohol consumption as a significant risk factor. The MLP recorded a precision of 49%, recall of 70%, and F1-score of 58% while the convolutional network had a loss of 0.5554, validation accuracy of 58% and validation loss of 1.0068.**

Keywords: Diabetes, Feedforward neural networks (FNNs), convolutional neural networks (CNNs), Multi-layer Perceptron (MLP) classifier, Python, vs code, tensorFlow keras, ROC curve.

I. INTRODUCTION

Diabetes is a chronic metabolic disorder that occurs when the pancreas cannot effectively produce insulin [1], characterized by high blood sugar levels, which can lead to severe health complications if not managed effectively. The increasing prevalence of diabetes worldwide has necessitated the development of advanced diagnostic and predictive tools to facilitate early detection and intervention [2]. Traditional methods of diabetes diagnosis often rely on clinical assessments and laboratory tests, which can be time-consuming and may not always provide timely results.

Recent advancements in artificial intelligence, particularly in machine learning and deep learning, have opened new avenues for improving diabetes detection and prediction. Among these, feedforward neural networks (FNNs) and convolutional neural networks (CNNs) have emerged as powerful tools for analyzing complex datasets and identifying patterns that may not be apparent through conventional methods [5]. FNNs are particularly effective in processing structured data, while CNNs excel in handling unstructured data such as medical images, making them suitable for a wide range of applications in diabetes care [3].

The integration of these neural network models with classification algorithms has shown promising results in enhancing the accuracy of diabetes predictions. Studies have demonstrated that these models can significantly improve diagnostic performance by leveraging large datasets, thereby facilitating timely interventions and personalized treatment plans [6]. This paper aims to explore the applications of FNNs and CNNs in diabetes detection and prediction, highlighting their effectiveness and potential impact on diabetes management.

II. LITERATURE REVIEW

The application of feedforward neural networks (FNNs) and convolutional neural networks (CNNs) in diabetes detection and prediction has garnered significant attention in recent years. These advanced machine learning techniques have demonstrated their potential to enhance diagnostic accuracy and facilitate timely interventions.

2.1 Feedforward Neural Networks in Diabetes Prediction

FNNs have been widely utilized for predicting diabetes by analyzing structured datasets, such as patient demographics, medical history, and laboratory results. A study by [7] employed an FNN model to predict the onset of type 2 diabetes using a dataset that included various risk factors. The model achieved an accuracy of over 90%, highlighting the effectiveness of FNNs in identifying at-risk individuals. Similarly, another research demonstrated that FNNs could effectively classify diabetes patients based on their clinical features, providing a reliable tool for early diagnosis [5].

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 10, pp 1-8, October-2025

https://doi.org/10.47001/IRJIET/2025.910001

2.2 Convolutional Neural Networks for Medical Imaging

CNNs have revolutionized the analysis of medical images, particularly in the context of diabetes-related complications. For instance, [3] developed a CNN model to detect diabetic retinopathy from retinal fundus photographs. The model achieved a sensitivity of 90% and specificity of 98%, showcasing its potential to assist ophthalmologists in diagnosing this common diabetes complication. Furthermore, a study by [4] utilized CNNs to analyze optical coherence tomography images for early detection of diabetic macular edema, demonstrating the versatility of CNNs in various imaging modalities.

2.3 Integration of Neural Networks with Classification Algorithms

The integration of FNNs and CNNs with classification algorithms has further enhanced their predictive capabilities. A study by [6] explored the combination of CNNs with support vector machines (SVM) for diabetes prediction, resulting in improved accuracy compared to using either method alone. This hybrid approach allows for better feature extraction and classification, making it a promising avenue for future research.

According to [8], ANN models were used to predict diabetes dataset based on multi-layer prediction classifier with batch size: 100, hidden layer sizes: 5, learning rate; 0.001 and maximum iteration of 500. In his analysis, the model proposed a high accuracy of 76.6% and by varying the hyper parameters, the best accuracy of 81% was achieved.

2.4 Real-World Applications and Impact

The real-world applications of these neural network models extend beyond prediction to include personalized treatment plans and patient management. For example, a study by [9] implemented an FNN-based decision support system that provided personalized recommendations for diabetes management based on individual patient data. This system not only improved patient adherence to treatment but also resulted in better glycemic control.

2.5 Challenges and Future Directions

Despite the promising results, challenges remain in the widespread adoption of FNNs and CNNs in clinical practice. Issues such as data privacy, the need for large annotated datasets, and the interpretability of model predictions pose significant hurdles [10]. Future research should focus on addressing these challenges, exploring transfer learning techniques, and developing explainable AI models to enhance trust and usability in clinical settings.

III. METHODOLOGY

3.1 Data Collection and Loading

The data used was collected from the Pima Indian Diabetes Dataset, Bio-Statistics Diabetes Dataset, Nnamdi Azikiwe University Teaching Hospital (NAUTH), Enugu State University Teaching Hospital (ESUTH) Parklane Enugu state, University of Nigeria Teaching Hospital (UNTH), Federal Teaching Hospital Abakaliki, Ebonyi State. (FETHA) and Federal Medical Centre Owerri, Imo state, Nigeria.

The data collected constituting of 495 dataset including Age, cholesterol, Blood Glucose Level (BGL), Body Mass index (BM INDEX), Physical Activity level (PHY AL), Family History of Patients (FHP) and Alcohol Consumption (ALCO CON).

3.2 Data Preprocessing

The features (X) and target variable (y) are separated. The dataset is then split into training and testing sets using train_test_split.

3.2.1 Splitting the Dataset into Training and Test Sets

- X: This represents the feature matrix (input data).
- y: This represents the target variable (output labels).
- test_size=0.3: This parameter specifies that 30% of the data should be allocated to the test set, while the remaining 70% will be used for the training set.
- random_state=0: This parameter ensures that the split is reproducible. Setting a specific seed (in this case, 0) means that every time you run the code, you will get the same split of the data.

3.2.2 Splitting the Test Set into Validation and Test Sets

- X_test and y_test: These are the test sets obtained from the previous split (70% of the original data).
- test_size=0.4: This indicates that 40% of the remaining test set (which is 40% of the original dataset) will be allocated to the new test set, while the remaining 60% will be used as the validation set.
- Resulting Sizes:
  - After the first split:
    - Training set: 70% of the original data
    - Test set: 30% of the original data
  - After the second split:
    - Validation set: 24% of the original data (60% of the 40% test set)
    - New test set: 16% of the original data (40% of the 40% test set)

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 10, pp 1-8, October-2025

https://doi.org/10.47001/IRJIET/2025.910001

3.3 Data Training

The supervised training of the dataset was done using feed forward neural network (multi-layer perception-MLP) and Convolutional Neural Network.

3.3.1 Multi-Layer Perceptron (MLP)

A typical application of a deep learning model is the feedforward deep network, or multilayer perceptron (MLP) which is a mathematical function of mapping some set of input values to output values [11].The MLPclassifier, a type of feed forward neural network was imported from scikit-learn.

3.3.1.1 Creating the Neural Network (MLP Classifier)

- **MLP Classifier:** This is a class in **scikit-learn 1.7.1** that implements a Multi-layer Perceptron (MLP) for classification tasks. It is a type of feedforward neural network.

Parameters Explained:

- hidden_layer_sizes=(100, 100):
  - This parameter specifies the architecture of the neural network.
  - It indicates that there are **two hidden layers**, each containing **100 neurons**.
  - The structure can be visualized as:
    - Input Layer → Hidden Layer 1 (100 neurons) → Hidden Layer 2 (100 neurons) → Output Layer.
- max_iter=500:
  - This parameter sets the maximum number of iterations (epochs) for training the model.
  - The training process will stop after **500 iterations** unless convergence is achieved earlier.
- activation='relu':
  - This specifies the activation function to be used in the hidden layers.
  - **ReLU (Rectified Linear Unit)** is a popular activation function that helps the model learn complex patterns by allowing it to capture non-linear relationships. It outputs the input directly if it is positive; otherwise, it outputs zero.
- solver='adam':
  - This parameter specifies the optimization algorithm to be used for training the model.
  - **Adam (Adaptive Moment Estimation)** is an efficient optimization algorithm that combines the advantages of two other extensions of stochastic gradient descent. It is widely used for training deep learning models.
- random_state=0:
  - This parameter ensures reproducibility of the results by controlling the randomness of the weight initialization and the shuffling of the data.
  - Setting a specific seed (in this case, 0) means that every time you run the code, you will get the same results.

3.3.1.2 Training the Model (fit() method)

The parameters of the classifier method of training are categorized into: X_train and y_train.

- **X_train**: This is the feature matrix containing the training data. It consists of the input features used to train the model.
- **y_train.ravel()**:
  - This flattens the target array **y_train** if it has a shape of **(n_samples, 1)** to a shape of **(n_samples,)**. This is necessary because **scikit-learn** expects the target variable to be a 1D array for classification tasks.
- **What happens during training?**:
  - The **fit()** method trains the neural network on the provided training data (**X_train** and **y_train**).
  - During training, the model adjusts its weights and biases to minimize the classification error using backpropagation and the specified optimizer (Adam).
  - The training process involves multiple iterations (up to 500) where the model learns from the data.

3.3.2 Convolutional Neural Network Model (Cnn Model)

The model provided is used to create a Convolutional Neural Network (CNN) model using TensorFlow's Keras API. Here tensorflow 2.20 and keras 3.11.1 were used

Components used:

- **tf.keras.models.Sequential**:
  - This creates a **sequential model** in Keras, which is a linear stack of layers. Each layer has exactly one input tensor and one output tensor.
  - The layers are added in the order they are defined, making it easy to build simple models.

Layers used:

1. **FirstLayer: tf.keras.layers.Dense(16, activation='relu')**:

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 10, pp 1-8, October-2025

https://doi.org/10.47001/IRJIET/2025.910001

- **Dense(16)**: This is a fully connected (dense) layer with **16 neurons**.
- **activation='relu'**: The activation function used is **ReLU (Rectified Linear Unit)**. This function outputs the input directly if it is positive; otherwise, it outputs zero. ReLU helps the model learn complex patterns and mitigates the vanishing gradient problem.

2. **Second Layer: tf.keras.layers.Dense(16, activation='relu')**:
   - This is another fully connected layer with **16 neurons** and the same **ReLU activation function**. It allows the model to learn more complex representations of the data by stacking layers.
3. **Output Layer: tf.keras.layers.Dense(1, activation='sigmoid')**:
   - **Dense(1)**: This is the output layer with **1 neuron**. It is typically used for binary classification tasks.
   - **activation='sigmoid'**: The activation function used here is **sigmoid**, which outputs a value between 0 and 1. This is suitable for binary classification, where the output can be interpreted as the probability of the positive class.

3.3.2.1 Compiling the Model

Parameters used:

1. optimizer=tf.keras.optimizers.Adam(learning_rate=0.005):

- optimizer: This parameter defines the optimization algorithm used to update the model's weights during training.
- tf.keras.optimizers.Adam: Adam (Adaptive Moment Estimation) is a popular optimization algorithm that combines the advantages of two other extensions of stochastic gradient descent. It adapts the learning rate for each parameter based on the first and second moments of the gradients.
- learning_rate=0.005: This sets the initial learning rate for the Adam optimizer. A learning rate of 0.005 means that the optimizer will make relatively small updates to the model's weights during training. The choice of learning rate can significantly affect the training process and convergence.

2. loss=tf.keras.losses.BinaryCrossentropy():

- loss: This parameter specifies the loss function used to evaluate how well the model's predictions match the true labels.
- tf.keras.losses.BinaryCrossentropy(): This loss function is used for binary classification tasks. It measures the difference between the predicted probabilities (output of the model) and the actual binary labels (0 or 1). The binary cross-entropy loss is defined as: where is the true label and is the predicted probability for each sample. The goal during training is to minimize this loss.

3. metrics=['accuracy']:

- metrics: This parameter specifies the metrics used to evaluate the model's performance during training and testing.
- ['accuracy']: This indicates that accuracy will be tracked as a performance metric. Accuracy is defined as the ratio of correctly predicted instances to the total instances. It provides a straightforward measure of how well the model is performing.

3.3.2.2 Model Summary

The summary of the model is shown in Table 1.

Table 1: Model Summary

Layer (type)	Output shape	Param #
dense (Dense)	(None, 16)	80
Dense_1 (Dense)	(None, 16)	272
Dense_2 (Dense)	(None, 1)	17

3.3.2.3 Training the Model

The data visualization of the training set based on the accuracy and loss is shown in Figure 1 and Figure 2 respectively.

Explanation of Components used:

- **hist = cnn.fit(...)**:
  - The **fit()** method trains the model on the provided training data (**X_train** and **y_train**).
  - The result of the training process is stored in the variable **hist**, which is an object containing information about the training process, including loss and accuracy metrics for each epoch.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 10, pp 1-8, October-2025

https://doi.org/10.47001/IRJIET/2025.910001

Parameters used:

1. X_train:

- This is the feature matrix containing the training data. It consists of the input features used to train the model.

2. y_train:

- This is the target variable (output labels) corresponding to the training data. It contains the true labels for each sample in **X_train**.

3. epochs=500:

- This parameter specifies the number of complete passes through the training dataset. In this case, the model will be trained for **500 epochs**. Each epoch consists of one forward pass and one backward pass of all training samples.

4. batch_size=20:

- This parameter defines the number of samples that will be used in each iteration (or batch) during training. In this case, the model will process **20 samples at a time** before updating the weights. Using mini-batches helps in stabilizing the training process and can lead to faster convergence.

5. validation_data=(X_valid, y_valid):

- This parameter specifies the validation dataset to evaluate the model's performance during training.
- **X_valid**: The feature matrix for the validation set.
- **y_valid**: The target variable for the validation set.
- The model will evaluate its performance on this validation set at the end of each epoch, allowing you to monitor how well the model generalizes to unseen data.

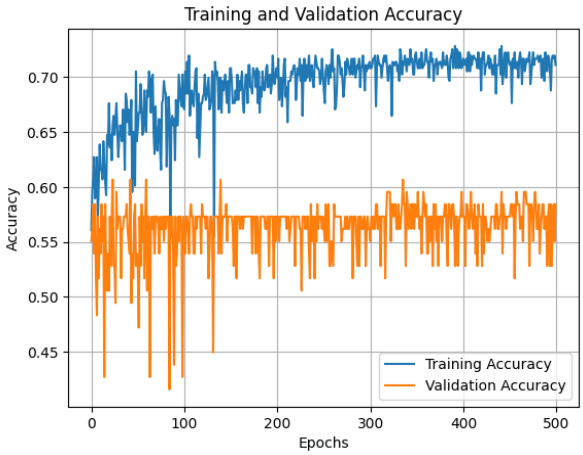

Figure 1: Training and Validation Accuracy versus 500 Epochs

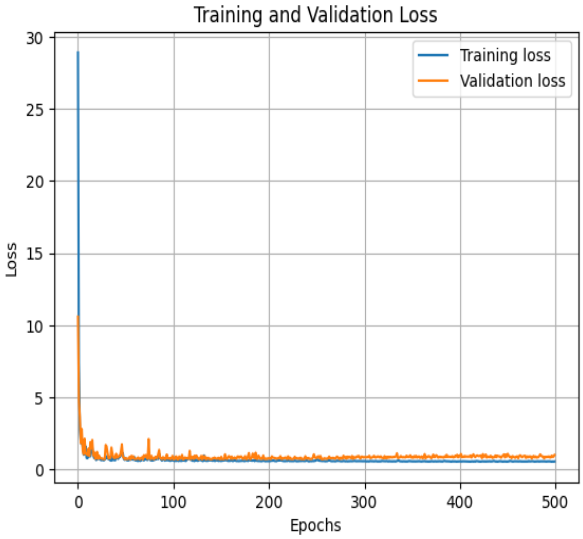

Figure 2: Training and Validation Loss versus 500 Epochs

IV. TEST AND RESULTS

The test result of diabetes is based on the consumption of Alcohol as a determinant for the evaluation of the diabetes detection and prediction.

4.1 Analysis of the Health DataFrame Column

'ALCO CON': This column likely represents whether an individual consumes alcohol or not. The values in this column are expected to be binary as shown in Figure 3. The probability of Alcohol consumption based on Age, cholesterol, BGL, and BM INDEX is shown in Figure 4, Figure 5, Figure 6 and Figure 7 respectively:

- 0: Indicates that the individual does not consume alcohol which constitutes 339 (68.48%).
- 1: Indicates that the individual does consume alcohol which constitutes 156 (31.52%). The hue plot of the alcohol consumption is shown in Figure8.

4.1.1 Counting non-consumers

- HEALTH[' ALCO CON'] == 0: This creates a boolean mask that filters the DataFrame to include only the rows where the value in the ' ALCO CON' column is 0.
- HEALTH[HEALTH[' ALCO CON'] == 0]: This returns a new DataFrame containing only the rows where individuals do not consume alcohol.
- len(...): The len() function calculates the number of rows in this filtered DataFrame, which represents the count of individuals who do not consume alcohol.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 10, pp 1-8, October-2025

https://doi.org/10.47001/IRJIET/2025.910001

4.1.2 Counting Consumers

- HEALTH[' ALCO CON'] == 1: This creates a boolean mask that filters the DataFrame to include only the rows where the value in the ' ALCO CON' column is 1.
- HEALTH[HEALTH[' ALCO CON'] == 1]: This returns a new DataFrame containing only the rows where individuals do consume alcohol.
- len(...): The len() function calculates the number of rows in this filtered DataFrame, which represents the count of individuals who consume alcohol.

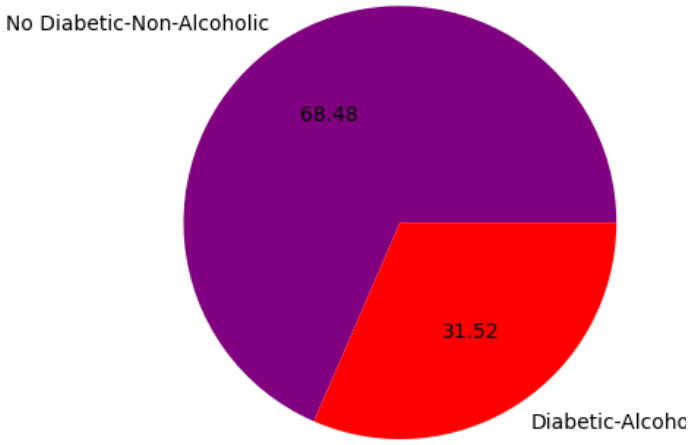

Figure 3: Alcohol consumption performances

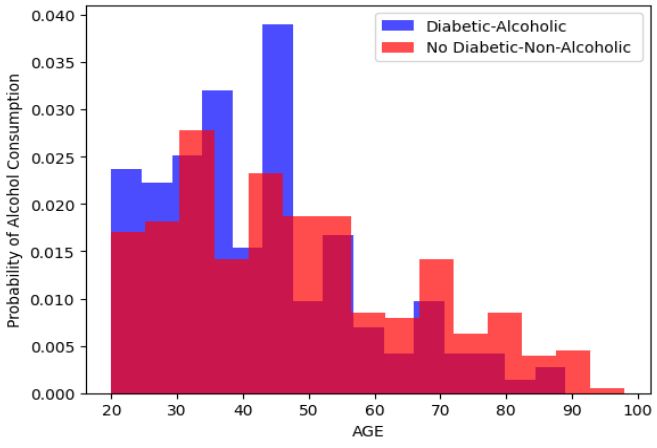

Figure 4: Probability of Alcohol based on Age

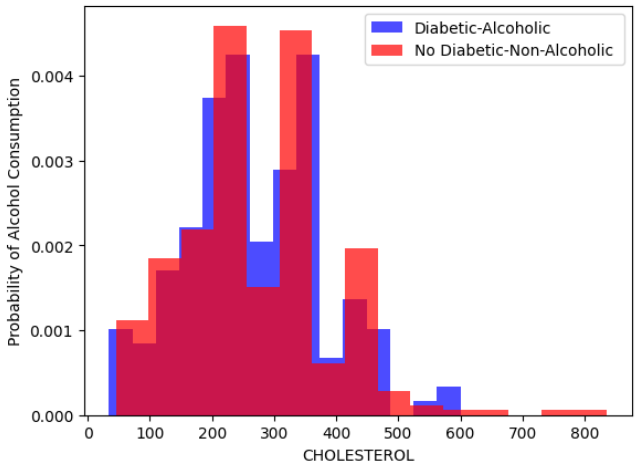

Figure 5: Probability of Alcohol based on cholesterol

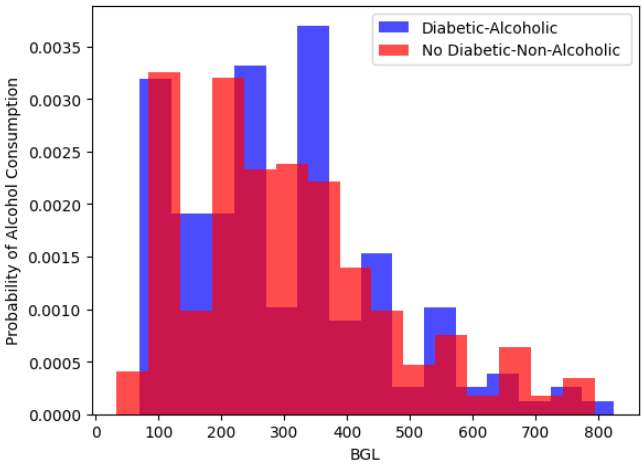

Figure 6: Probability of Alcohol based on BGL

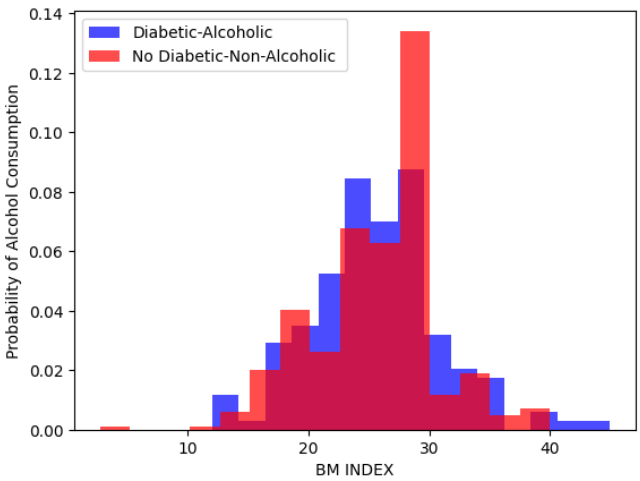

Figure 7: Probability of Alcohol based on BM INDEX

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 10, pp 1-8, October-2025

https://doi.org/10.47001/IRJIET/2025.910001

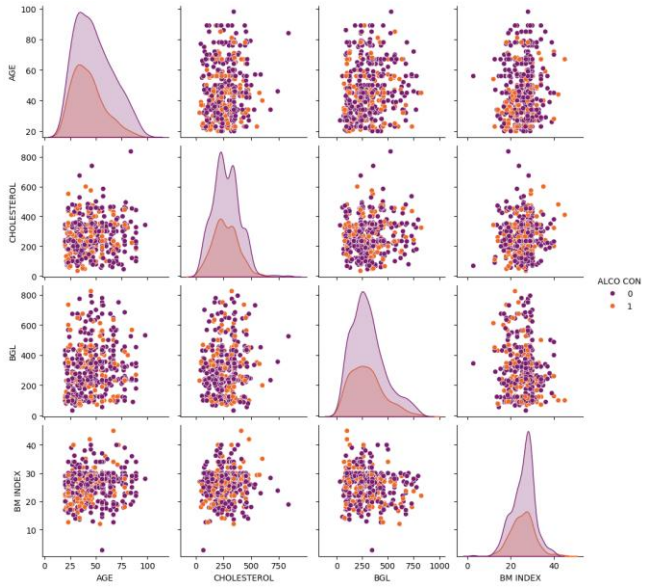

Figure 8: Hue Plot of Alcohol Consumption Based on Age, Cholesterol, BGL and BM INDEX

4.2 Evaluation of Test set

The prediction of the test set for Multi-layer Perception (MLP) and Convolutional Neural Networks (cnn) is based on Accuracy, precision, recall, F1-score, support, cross-validation, confusion Matrix and ROC-curve.

4.2.1 Accuracy of MLP: The Accuracy of the test set during prediction is 64.44%

4.2.2 Classification Report of MLP: The Classification report of MLP showing the Accuracy, precision, recall, F1-score, and support is shown below. From the report the accuracy is 70%, the precision is 49%, the recall is 70%, the F1-score is 58% with the average support of 60.

4.2.3 MLP Cross Validation results: 20-folds cross-validation of the test set gave a mean cross-validation score of 0.68

4.2.4 MLP and CNN ROC-curve: The predicted model evaluates the performance of a binary classification model by calculating the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC). The Area under the curve, Auc for MLP is 0.5 while a greater improvement was made in convolutional neural network with AUC of 0.55.

ROC Curve: The ROC curves as shown in Figure9 and Figure 10 are graphical representations of a classifier's performance across different threshold values. It plots the **True Positive Rate (TPR)** against the **False Positive Rate (FPR)** at various threshold settings.

- **True Positive Rate (TPR)**, also known as sensitivity or recall, is the ratio of correctly predicted positive observations to all actual positives:

$$TPR = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

- **False Positive Rate (FPR)** is the ratio of incorrectly predicted positive observations to all actual negatives:

$$FPR = \frac{False\ Positives}{False\ Positives + True\ Negatives}$$

- **AUC (Area Under the Curve)**: The AUC quantifies the overall performance of the classifier. It represents the area under the ROC curve. An AUC of 1 indicates a perfect classifier, while an AUC of 0.5 indicates a classifier with no discrimination ability (equivalent to random guessing).

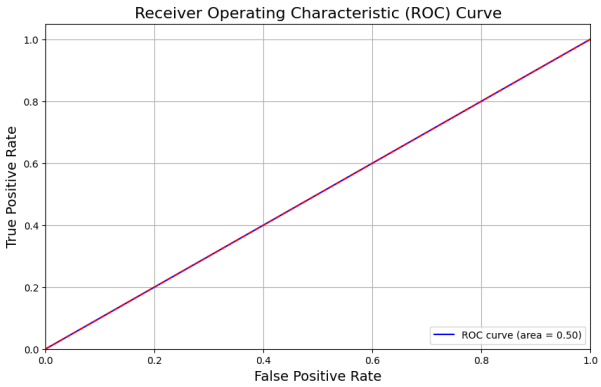

Figure 9: ROC curve of the MLP predicted Model

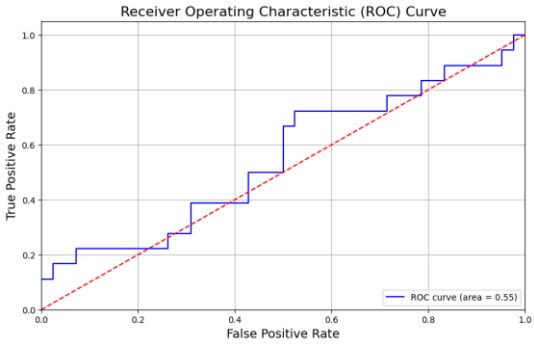

Figure 10: ROC curve of the Convolutional Neural Network predicted Model

V. CONCLUSION

In summary, Feedforward Neural Networks and Convolutional Neural Networks are two of the most commonly used ANN models for intelligent systems, each with distinct architectures and applications. Evaluation techniques such as cross-validation, confusion matrices, and ROC curves with AUC are essential for assessing the

ISSN (online): 2581-3048
Volume 9, Issue 10, pp 1-8, October-2025
https://doi.org/10.47001/IRJIET/2025.910001

performance and validity of ANN models, ensuring that they generalize well to unseen data. These methodologies are crucial for developing robust and reliable intelligent systems across various domains.

REFERENCES

[1] Kenneth A. Akpado, Ugochukwu J. Njonu, P.C Obioma and Anthony N. Isizoh. (2021). An Improved Method for Predicting Diabetes Mellitus Using Adaptive Neuro-Fuzzy Inference System. *International Journal of Scientific Engineering and Science.* Volume 5, Issue 11, pp. 19-32.

[2] Cho, N. H., Shaw, J. E., Karuranga, S., Huang, Y., da Rocha Fernandes, J. D., Ohlrogge, A. W., and Malanda, I. (2018). IDF Diabetes Atlas: Global estimates of diabetes prevalence for 2011 and projections for 2030 and 2040. *Diabetes Research and Clinical Practice*, 87(1), 4-14.

[3] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., and Narayanaswamy, A. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22), 2402-2410.

[4] Ting, D. S. W., Cheung, C. Y., Lim, G., and Wong, T. Y. (2019). Deep learning in ophthalmology: The future is here. *The Lancet Digital Health*, 1(2), e100-e102.

[5] Khan, M. A., Bashir, A., and Khan, M. A. (2020). A survey on machine learning techniques for diabetes prediction. *International Journal of Advanced Computer Science and Applications*, 11(6), 1-8.

[6] Bashir, A., Khan, M. A., and Khan, M. A. (2021). A comprehensive review on machine learning techniques for diabetes prediction. *Journal of Healthcare Engineering*, 2021, 1-15.

[7] Alshahrani, M. M., Alzahrani, A. A., and Alshahrani, A. A. (2020). Predicting type 2 diabetes using feedforward neural networks. *Journal of Healthcare Engineering*, 2020, 1-10.

[8] Ibrahim, E. I. A. (2022). Diabetes prediction by using artificial neural network. *Sudan University of Science and Technology College of graduate Studies.*

[9] Dey, N., Ashour, A. S., and Balas, V. E. (2020). A decision support system for diabetes management using feedforward neural networks. *Journal of Ambient Intelligence and Humanized Computing*, 11(1), 1-10.

[10] Rajkomar, A., Dean, J., and Kohane, I. (2019). Machine learning in medicine. *New England Journal of Medicine*, 380(14), 1347-1358.

[11] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. *MIT Press.* http://www.deeplearningbook.org/

Citation of this Article:

Kelvin Nnamani, Kenneth Akpado, & Stephen Ufoaroh. (2025). Application of Feed Forward and Convolutional Neural Network Models in Diabetes Detection and Prediction Using Classification Algorithms. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 9(10), 1-8. Article DOI https://doi.org/10.47001/IRJIET/2025.910001

****

