BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visve
University, Belagavi)
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar
(India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE-DATA SCIENCE

A Mini-Project Report On

“Heart Disease Prediction Using a Deep Neural Network”

A report submitted in partial fulfilment of the requirements for the

NEURAL NETWORK AND DEEP LEARNING

Submitted By

KAVYA R RATHOD **USN: 3BR22CD027**

Under the Guidance of **Mr. Azhar Biag**

Asst. Professor

Dept of CSE (DATA SCIENCE),
BITM, Ballari

Visvesvaraya Technological University

Belagavi, Karnataka 2025-2026

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

Autonomous Institute under VTU, Belagavi | Approved by AICTE,New Delhi Recognized by Govt. of Karnataka

NACC Accredited Institution*

nized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belgavi)

"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballari-583 104 (Karnataka) (India)

Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE (DATA SCIENCE)

CERTIFICATE

This is to certify that the Mini Project of NEURAL NETWORK AND DEEP LEARNING title " **Heart Disease Prediction Using a Deep Neural Network**" is a Bonafide work carried out by **KAVYA R RATHOD** bearing USN **3BR22CD027** in partial fulfillment for the award of degree of Bachelor Degree in CSE(Data Science) in the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi during the academic year 2025-2026. It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report deposited in the library. The project has been approved as it satisfies the academic requirements in respect of mini project work prescribed for a Bachelor of Engineering Degree.

Signature of Coordinators

Mr. Azhar Baig M & Ms. Chaithra B M

Signature of HOD

Dr. Aradhana

ABSTRACT

Heart disease remains one of the leading global health concerns, impacting millions of people and contributing to life-threatening conditions such as heart attacks, heart failure, and other cardiovascular complications. Early detection is vital, as timely diagnosis can significantly reduce mortality rates and improve long-term patient outcomes. With the rapid advancement of artificial intelligence, deep learning methods have become powerful tools for analysing clinical data and predicting disease risk with high accuracy.

This project focuses on developing a **Deep Neural Network (DNN)–based model** to classify individuals as having heart disease or not, using the **Heart Failure Prediction Dataset**. The system preprocesses essential clinical features such as cholesterol levels, blood pressure, age, resting ECG results, chest pain type, maximum heart rate, and other relevant medical attributes to ensure standardized and reliable inputs for effective model training.

The DNN architecture is constructed with multiple dense hidden layers using **ReLU activations**, along with a final **sigmoid output layer** for binary classification. This structure enables the network to capture complex non-linear relationships within the dataset, improving its ability to distinguish between healthy and high-risk individuals.

After data preprocessing and model development, the neural network is trained using optimized hyperparameters and evaluated through comprehensive performance metrics, including **accuracy, precision, recall, F1-score, and a confusion matrix**. Visualization tools—such as accuracy and loss curves—provide insights into the model’s training behavior, highlighting its stability, convergence rate, and generalization capability.

The results indicate that the DNN model effectively learns significant patterns associated with heart disease and can reliably predict patient risk. This study demonstrates the potential of deep learning models in medical diagnosis and underscores their role in supporting healthcare professionals by enabling early detection and timely decision-making. With further enhancements and integration of larger, diverse datasets, such predictive systems could be implemented in real-world clinical environments, contributing to faster, data-driven healthcare and improved patient well-being.

ACKNOWLEDGEMENT

The satisfactions that accompany the successful completion of our mini project on Heart Disease Prediction Using a Deep Neural Network would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is our privilege to express our gratitude and respect to all those who inspired us in the completion of our mini-project.

I am extremely grateful to my Guide **Mr. Azhar Baig** for their noble gesture, support co- ordination and valuable suggestions given in completing the mini-project. I also thank **Dr. Aradhana D,** H.O.D. Department of CSE(DS), for his co-ordination and valuable suggestions given in completing the mini-project. We also thank Principal, Management and non-teaching staff for their co-ordination and valuable suggestions given to us in completing the Mini project.

Name	USN
KAVYA R RATHOD	3BR22CD027

TABLE OF CONTENTS

1. INTRODUCTION

Heart disease is one of the most widespread and life-threatening chronic conditions impacting millions of individuals across the globe. It occurs when the heart is unable to function efficiently due to blocked arteries, high blood pressure, abnormal cholesterol levels, or other physiological risk factors. If left undiagnosed or untreated, heart disease can lead to severe complications such as heart failure, stroke, arrhythmia, and even sudden cardiac death. As global heart disease cases continue to rise, early prediction and timely diagnosis have become essential for improving patient outcomes and reducing mortality rates.

Traditional diagnosis of heart disease typically relies on clinical tests, medical imaging, and expert evaluation by cardiologists. While effective, these procedures can be expensive, time-consuming, and inaccessible in underserved or remote regions. Moreover, manual interpretation may fail to detect subtle patterns that indicate early stages of cardiovascular risk. These limitations highlight the need for automated, data-driven approaches that can support healthcare professionals in making quick and accurate decisions.

With advancements in artificial intelligence and machine learning, predictive modeling has become a powerful tool for analyzing medical datasets. Among the various AI techniques, **Artificial Neural Networks (ANNs)** have shown remarkable capability in identifying complex relationships and hidden patterns within clinical data. Neural networks mimic the functioning of the human brain and are especially effective in handling nonlinear and multi-dimensional datasets, making them highly suitable for heart disease prediction.

This project aims to develop an ANN-based model to predict heart disease using the **Heart Failure Prediction Dataset** from Kaggle. The dataset includes a range of medical parameters such as cholesterol level, resting blood pressure, maximum heart rate, chest pain type, exercise-induced angina, and other diagnostic indicators. By training the neural network on these features, the model learns how different clinical factors contribute to the likelihood of heart disease.

The primary objective of this work is to design, implement, and evaluate an efficient neural network model capable of classifying individuals as having heart disease or not based on their medical attributes. The project involves several key stages including data preprocessing, feature scaling, model construction, training, testing, and performance evaluation. Metrics such as accuracy, classification reports, and confusion matrices are used to assess model performance. Additionally, visualizations of training and validation accuracy and loss help in understanding the learning behavior and effectiveness of the neural network.

Overall, this project demonstrates how deep learning techniques can contribute to early detection of heart disease and support healthcare systems in making faster and more reliable diagnostic decisions.

1.1 Problem Statement

The problem addressed in this project is the growing need for an efficient, automated, and accessible method to predict heart disease using commonly available clinical data. Traditional diagnostic procedures—such as ECG analysis, stress tests, and detailed medical examinations—can be time-consuming, costly, and unavailable in many underserved regions. Early identification of heart disease is crucial to prevent life-threatening complications, yet manual diagnosis may fail to detect subtle patterns and risk indicators hidden within a patient's clinical measurements.

Using the **Heart Failure Prediction Dataset**, this project aims to build an Artificial Neural Network (ANN) model capable of accurately analyzing medical attributes such as cholesterol levels, resting blood pressure, maximum heart rate, chest pain type, blood sugar levels, and other cardiovascular indicators to classify individuals as having heart disease or not. The objective is to develop a reliable, data-driven prediction system that supports healthcare professionals, enhances early screening efforts, and contributes to more effective medical decision-making.

1.2 Objectives

- ❖ To build an ANN model for accurate heart disease prediction.
- ❖ To preprocess, clean, encode, and standardize the dataset to improve model efficiency and performance.
- ❖ To evaluate the model using metrics such as accuracy, confusion matrix, and classification report.
- ❖ To visualize the model's learning behavior through training and validation accuracy and loss graphs.

2. LITERATURE SURVEY

[1] Ganie et al. (2023) investigated heart disease prediction using various ensemble learning models and reported that boosting algorithms such as XGBoost and AdaBoost provided highly accurate classification results. Their findings highlighted that proper data preprocessing and extraction of relevant clinical features significantly improve the performance of heart disease prediction systems.

[2] Gündoğdu (2023) developed a hybrid feature selection and XGBoost-based system for cardiovascular disease detection. The study demonstrated that optimized feature engineering combined with machine learning classifiers enhances both prediction accuracy and computational efficiency, making it suitable for early diagnosis applications.

[3] Chang et al. (2023) performed a comparative evaluation of multiple machine learning models for heart disease prediction and explored their use within IoMT (Internet of Medical Things) healthcare systems. The study emphasized the importance of high predictive accuracy and interpretability to support real-time decision-making in clinical environments.

[4] Tasin et al. (2022) analyzed classical and ensemble learning models on heart disease datasets and identified Random Forest as the best-performing method among the evaluated models. They further emphasized that handling class imbalance, scaling features, and applying effective preprocessing techniques are critical for developing robust cardiovascular prediction models.

[5] Madan et al. (2022) explored hybrid deep learning architectures for heart disease diagnosis and demonstrated that neural networks can successfully learn complex, non-linear relationships from patient data. However, the study also noted that deep learning models require sufficient training data and regularization techniques to prevent overfitting.

[6] Ayat (2024) proposed a CNN–LSTM hybrid model for detecting abnormalities in heart-related medical signals using time-series data. The study achieved high accuracy by capturing both spatial and temporal patterns, but also mentioned that such models perform best when large sequential datasets are available.

[7] R. Kumar & S. Verma (2022) compared Support Vector Machines, Logistic Regression, and Random Forest for heart disease prediction using benchmark datasets. Their analysis showed that tree-based models provided better performance, while also stressing the importance of selecting the right features and scaling techniques during preprocessing.

3. SYSTEM REQUIREMENTS

The system requirements for developing the **heart disease prediction model** include essential hardware and software components needed to efficiently perform data preprocessing, neural network training, and performance evaluation. The project is implemented in Python and uses several machine learning and data visualization libraries. Since the dataset is relatively small, the computational requirements remain modest, making the system easy to implement on most modern personal computers.

The software environment includes Python along with key libraries such as **TensorFlow/Keras** for building the neural network model, **Pandas** and **NumPy** for data manipulation, **Scikit-learn** for preprocessing and evaluation, **Matplotlib** and **Seaborn** for visualization, and **KaggleHub** for directly downloading the dataset from Kaggle. Development platforms like **Jupyter Notebook, Google Colab, or VS Code** provide an interactive environment for executing the code and analyzing results.

On the hardware side, the project can run smoothly on a standard computer equipped with at least **4 GB RAM**, though **8 GB or more** is preferable for smoother training and faster processing. A multi-core processor helps improve computational efficiency. While using a GPU is optional, it can significantly accelerate the training phase of the artificial neural network. Overall, the project's requirements are minimal and suitable for students, researchers, and developers working with limited computing resources.

SOTFWEAR REQUIRMENT:

- Python 3.8 or above
- TensorFlow / Keras
- NumPy
- Pandas
- Scikit-learn

- Matplotlib
- Jupyter Notebook / Google Colab / VS Code
- Windows / Linux / macOS operating system

3.1 Hardware Requirements

- Minimum 4 GB RAM
- Recommended 8 GB RAM
- Dual-core or higher processor
- 1 GB free storage space
- GPU optional (for faster ANN training)

3.2 Functional Requirements

- The system must load and preprocess the diabetes dataset.
- It must handle missing values and standardize input features.
- The system must build an ANN model for classification.
- It must train the ANN model using training data.
- The system must evaluate model performance using metrics.
- It must generate accuracy, loss, and confusion matrix graphs.
- The system must predict diabetes for new input data.

3.3 Non-Functional Requirements

- The system should provide accurate and reliable predictions.
- It should offer clear and user-friendly outputs.
- The system must execute efficiently on basic hardware.
- It should remain stable even with noisy or imperfect data.
- The system must be easy to maintain and extend.
- The results should be interpretable through graphs and metrics.

4. DESCRIPTION OF MODULES

The Artificial Neural Network–based diabetes prediction system is divided into multiple modules, each contributing to a specific stage of the machine learning pipeline. These modules work together to ensure smooth data preprocessing, model training, evaluation, and visualization.

4.1 Data Preprocessing Module

This module loads the Pima Diabetes dataset and prepares it for model training. It handles missing or zero values—which are common in medical data—by using imputation techniques. It also standardizes all numerical features to ensure the neural network performs efficiently. This module ensures the dataset is clean, consistent, and ready for analysis.

4.2 DNN Model Building Module

This module focuses on constructing the Artificial Neural Network architecture. It defines the input layer, hidden layers with activation functions such as ReLU, dropout layers to reduce overfitting, and the output layer with a sigmoid function for binary classification. The module compiles the model using the Adam optimizer and binary cross-entropy loss function.

4.3 Model Training Module

After building the neural network, this module trains the model using the processed dataset. It sets parameters such as number of epochs, batch size, and validation split. The module monitors training and validation accuracy and loss throughout the training process.

4.4 Model Evaluation Module

This module evaluates the performance of the trained neural network. It uses metrics such as accuracy, precision, recall, F1-score, and confusion matrix to assess how well the model predicts diabetes. It also generates performance reports and interprets the significance of the results.

4.5 Visualization Module

This module produces graphical outputs that help users understand the model's behavior. It generates training vs. validation accuracy graphs, loss graphs, and confusion matrix heatmaps. These visuals make the system more interpretable and user-friendly.

4.6 Prediction Module

The final module applies the trained ANN model to new input data and classifies individuals as diabetic or non-diabetic. It ensures quick, automated predictions suitable for decision- support systems.

4.7 Data Splitting Module

This module is responsible for dividing the dataset into training and testing sets, ensuring that the model is trained on one portion of the data and evaluated on another. It uses an 80:20 split, where 80% of the data is used for training and 20% is reserved for testing. Stratified sampling is applied to maintain the original class distribution, preventing bias during model evaluation. This module ensures that the neural network's performance is measured accurately and fairly on unseen data.

4.8 Feature Scaling Module

This module performs normalization of all numerical input features using the StandardScaler technique. Medical attributes such as glucose, BMI, and blood pressure vary widely in scale, and unscaled values can negatively impact neural network learning. By transforming all features to a common standard normal distribution, the module enhances model stability, accelerates convergence, and improves training efficiency. Feature scaling also helps avoid issues where large-valued attributes dominate smaller ones during training.

4.9 Output Interpretation Module

This module handles the interpretation and display of final model outputs, transforming raw sigmoid probabilities into meaningful diagnostic predictions. It applies a decision threshold (commonly 0.5) to categorize patients as diabetic or non-diabetic. Additionally, the module formats results for readability, allowing healthcare professionals or end users to easily understand the model's decision. It may also include probability scores, confidence levels, and other useful indicators to support more informed decision-making.

5. IMPLEMENTATION

The implementation of this diabetes prediction system is carried out entirely in Python using an Artificial Neural Network (ANN) model. The project uses the **Pima Indians Diabetes Dataset**, a widely recognized dataset for binary diabetes classification. The dataset is first loaded into a Pandas DataFrame, after which the independent input features—such as pregnancies, glucose level, blood pressure, skin thickness, insulin value, BMI, diabetes pedigree function, and age—are separated from the target label **Outcome**, which indicates whether an individual is diabetic or non-diabetic.

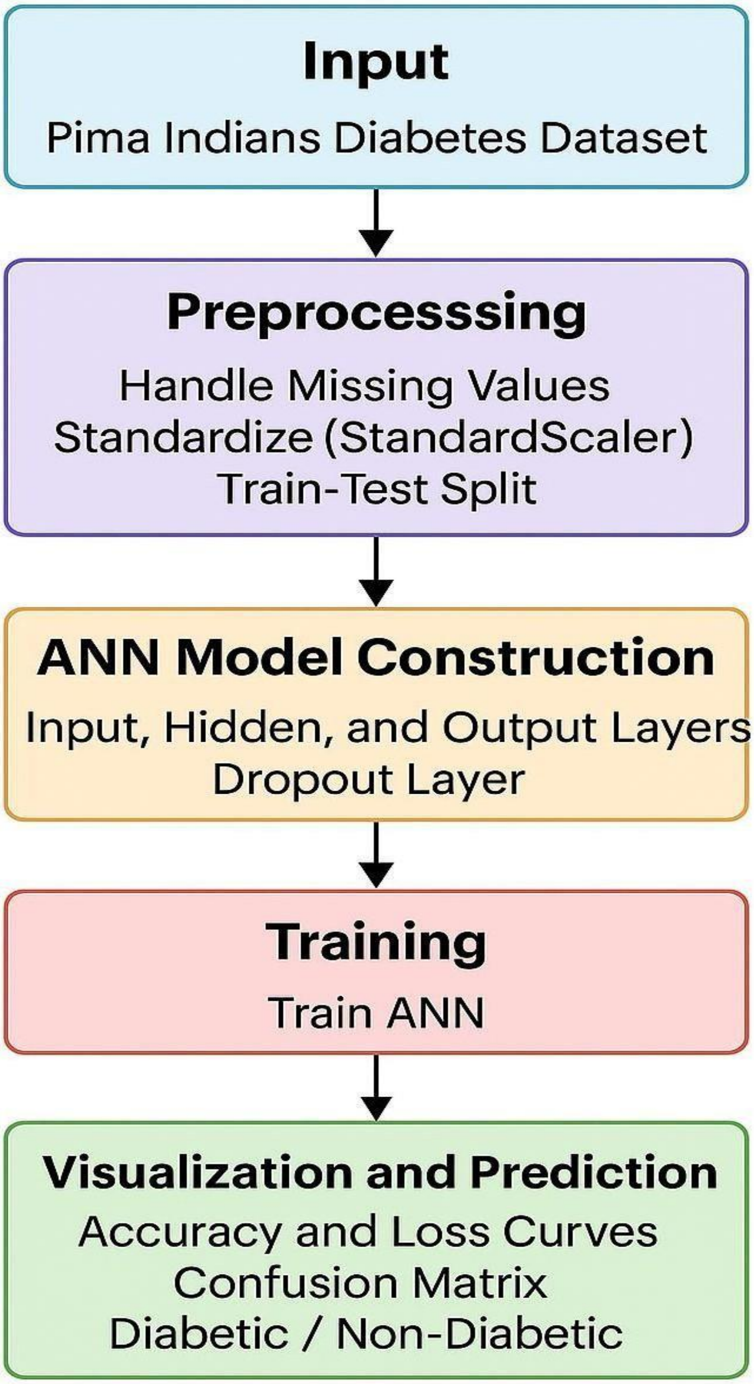
To ensure proper training and unbiased evaluation, the dataset is divided into training and testing sets using an **80–20 split**, while employing **stratified sampling** to maintain equal class distribution. Since the medical attributes possess varying numerical ranges, **data standardization** is performed using *StandardScaler*. This step plays a crucial role in improving ANN model stability and convergence during training.

After preprocessing, the ANN model is constructed using **TensorFlow/Keras**. The architecture of the neural network includes an input layer, followed by dense hidden layers with **ReLU activation** for learning nonlinear relationships. A **dropout layer** is incorporated to reduce overfitting and enhance generalization. Finally, an output layer with **sigmoid activation** is used to classify the input as diabetic or non-diabetic.

The model is compiled using the **Adam optimizer** and **binary cross-entropy** as the loss function, suitable for binary classification problems. The ANN is trained for **35 epochs** with a batch size of 32, along with a **20% validation split** to monitor its performance during training. Throughout the training process, the model learns to capture patterns within the clinical features that contribute to diabetes prediction.

Once training is complete, the model is evaluated using the test set to compute key performance metrics, including overall accuracy and a detailed classification report. To provide deeper insights, several visualizations are generated—such as training and validation accuracy curves, training and validation loss graphs, and a confusion matrix. These help interpret how well the ANN has learned, identify potential overfitting, and understand how accurately the model differentiates between diabetic and non-diabetic cases.

6. SYSTEM ARCHITECTURE



Input

In this stage, the diabetes prediction system begins by loading the **Pima Indians Diabetes Dataset**, which is stored in CSV format. The dataset is imported into a **Pandas DataFrame**, allowing easy handling and analysis of the medical features. Once the dataset is loaded, an initial exploratory check is performed to understand its structure and overall quality.

This includes viewing the **first few rows** to confirm the data format and inspecting the **shape of the dataset** to identify the total number of samples and features. The datatypes of each column are examined to ensure that numerical attributes such as **glucose level, blood pressure, BMI, age, insulin level, pregnancies, skin thickness, and diabetes pedigree function** are correctly recognized for further processing.

Additionally, the **target column (Outcome)**, which indicates whether a patient is diabetic (1) or non-diabetic (0), is analyzed to check the **class distribution**. This helps determine whether the dataset is balanced or imbalanced, an important aspect for training an accurate ANN model.

Overall, this stage ensures that the input data is properly loaded, understood, and prepared for subsequent preprocessing steps such as handling missing values and standardization.

Preprocessing

Preprocessing prepares raw data for the ANN so the model can learn effectively and generalize well.

- Handle missing/invalid values: identify zeros, NaNs, or unrealistic values (e.g., zero BMI or glucose) and decide on a strategy — remove rows, replace with median/mean, or use domain-driven imputation.
- Feature selection / engineering (optional): remove redundant columns, create derived features (e.g., age groups, BMI categories) if useful.
- Standardize features: apply StandardScaler (zero mean, unit variance) so features with different scales (glucose vs age) don't dominate learning.

ANN Model Construction

This stage defines the neural network architecture and compilation details.

- Input layer: sized to the number of features (here, 8).

- Hidden layers: e.g., Dense(64, ReLU) → Dropout(0.2) → Dense(32, ReLU). These layers learn nonlinear feature interactions; ReLU helps with gradient flow and sparsity.
- Dropout: randomly disables a fraction of neurons during training to reduce overfitting and improve generalization.
- Output layer: Dense(1, sigmoid) — produces a probability for the positive class (diabetic).
- Compile settings: choose optimizer (Adam), loss (binary_crossentropy for two-class problems), and metrics (accuracy; optionally precision, recall, AUC). Choosing hyperparameters (layer sizes, dropout rate, learning rate) is part of architecture design and may be tuned.

The goal here is to build a model expressive enough to capture patterns but regularized enough to avoid overfitting.

Training

Training is where the network learns by updating weights to minimize loss.

- Fit the model: run for a fixed number of epochs (e.g., 35) with a chosen batch size (e.g., 32), and optionally a validation_split (e.g., 0.2) to monitor validation metrics each epoch.
- Monitor: record training & validation loss and accuracy (history object). Watch for overfitting (training accuracy rising while validation accuracy plateaus or drops).
- Callbacks (optional): use EarlyStopping to stop when validation loss stops improving, ModelCheckpoint to save best weights, and ReduceLROnPlateau to lower learning rate on plateau.
- Hyperparameter tuning: you may iterate over epochs, batch size, learning rate, layer sizes, and regularization to improve performance.

Training converts initialized weights into a predictive model by repeated forward/backward passes on the data.

Visualization and Prediction

This final stage interprets the trained model and uses it for inference.

- Visualizations:
    - *Accuracy vs Epochs* — shows learning curve for train and validation sets.
    - *Loss vs Epochs* — shows how loss decreases and can indicate over/underfitting.
    - *Confusion matrix* — shows true positives, true negatives, false positives, false negatives to understand error types.
    - *Classification report* — precision, recall, F1-score per class.
    - *Optional:* ROC curve, AUC, precision–recall curve for threshold-insensitive evaluation.
- Prediction: apply model to test set or real user inputs. Convert sigmoid outputs to class labels using a threshold (commonly 0.5), or use calibrated probabilities if required. Provide result as "Diabetic / Non-Diabetic" and optionally include the probability/confidence for each prediction.
- Interpretation & deployment: use the visual and numeric outputs to assess readiness for deployment. If acceptable, export model (e.g., model.save()), build a prediction API or a simple GUI, and document limitations (dataset bias, clinical validation requirement).

7. CODE IMPLEMENTATION

Algorithm: Heart Disease Prediction using Artificial Neural Network (ANN)

Input: Heart Disease Dataset (from Kaggle)

Output: Predicted Class (Heart Disease / No Heart Disease) and Performance Metrics

1. Start

2. Load Dataset
2.1 Load the Heart Disease dataset (CSV file) using Pandas.
2.2 Separate the dataset into:

- Feature matrix X (all independent attributes such as age, sex, chest pain type, blood pressure, cholesterol, fasting blood sugar, ECG, maximum heart rate, exercise-induced angina, oldpeak, slope, ca, thal, etc.)
- Target vector y (HeartDisease column: 0 = No Disease, 1 = Disease)

3. Preprocess Data
3.1 Convert:

- X → float32
- y → int32

3.2 Split into training and testing sets using train_test_split:

- test_size = 0.2
- stratify = y (to keep heart disease class distribution balanced)

3.3 Fit StandardScaler on X_train.
3.4 Transform both X_train and X_test using the fitted scaler.

4. Build ANN Model
4.1 Initialize a Sequential ANN model.
4.2 Add input layer with shape equal to the number of features.
4.3 Add first hidden layer:

- Dense(64) with ReLU activation

4.4 Add dropout layer (rate = 0.2) to reduce overfitting.
4.5 Add second hidden layer:

- Dense(32) with ReLU activation

4.6 Add output layer:

- Dense(1) with Sigmoid activation
  (Used for binary classification: Heart Disease vs No Heart Disease)

5. Compile Model
5.1 Optimizer = Adam
5.2 Loss = Binary Cross-Entropy
5.3 Metric = Accuracy

6. Train Model
Train using:

- Epochs = 35
- Batch size = 32
- Validation split = 0.2

Save training history (accuracy & loss for both training and validation).

7. Test Model
7.1 Use trained model to predict probabilities for X_test.
7.2 Convert probabilities to class labels:

- Probability > 0.5 → 1 (Heart Disease)
- Probability ≤ 0.5 → 0 (No Heart Disease)

8. Evaluate Performance
8.1 Calculate accuracy using accuracy_score(y_test, y_pred).
8.2 Generate classification report (precision, recall, F1-score).
8.3 Compute confusion matrix.

9. Visualize Results
9.1 Plot Training vs Validation Accuracy
9.2 Plot Training vs Validation Loss
9.3 Display Confusion Matrix Heatmap

10. End

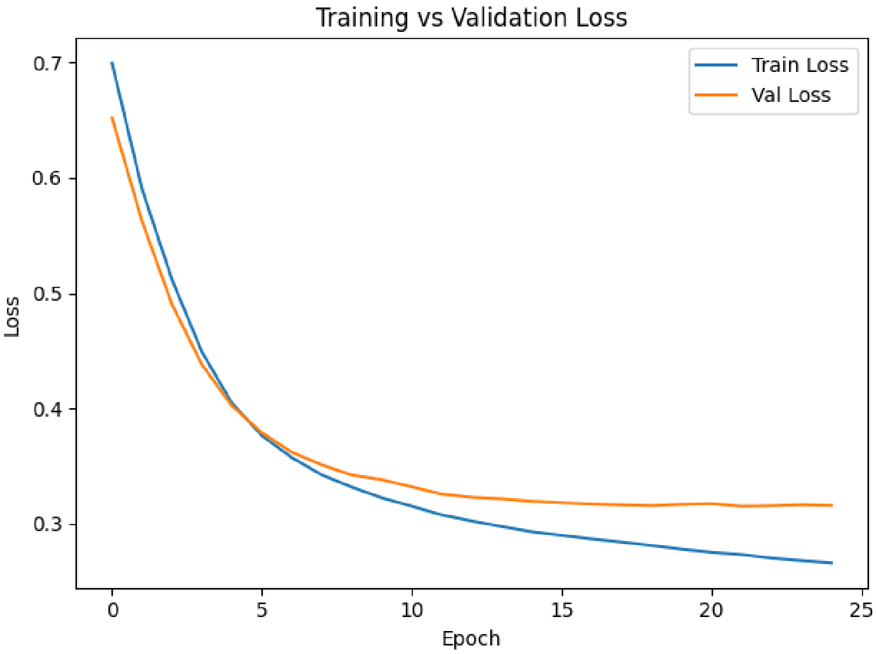
8. RESULT

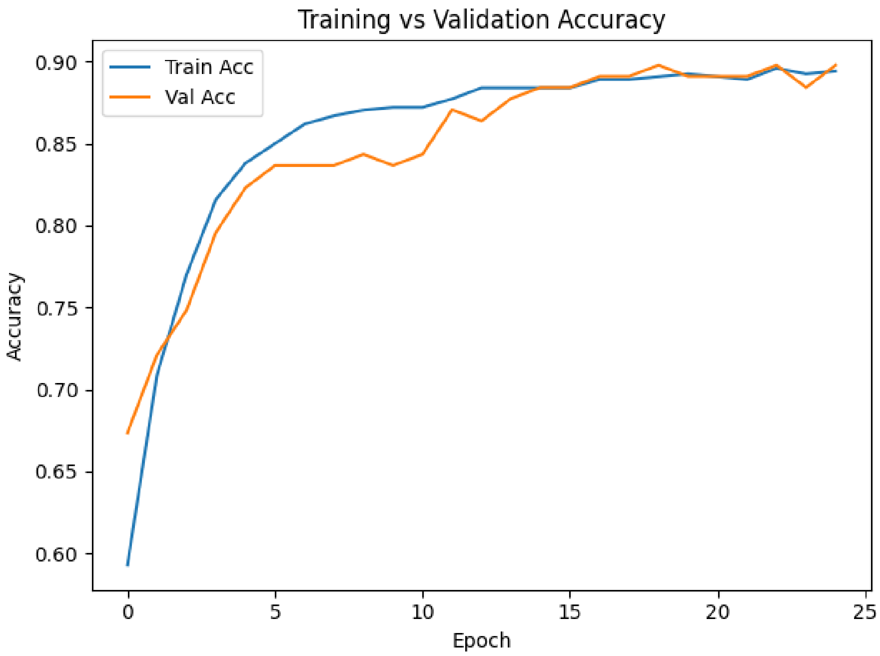
```
Using Colab cache for faster access to the 'heart-failure-prediction' dataset.
✅ Dataset downloaded at: /kaggle/input/heart-failure-prediction

=== Dataset Preview ===
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  MaxHR  \
0   40   M           ATA        140          289          0     Normal    172
1   49   F           NAP        160          180          0     Normal    156
2   37   M           ATA        130          283          0         ST     98
3   48   F           ASY        138          214          0     Normal    108
4   54   M           NAP        150          195          0     Normal    122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N      0.0       Up             0
1              N      1.0     Flat             1
2              N      0.0       Up             0
3              Y      1.5     Flat             1
4              N      0.0       Up             0

Columns:
 Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')

Missing values:
```

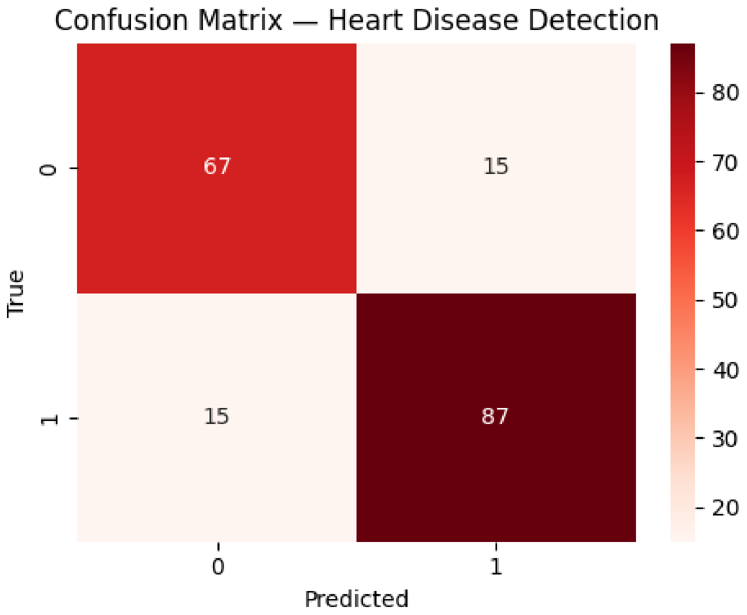
```
Missing values:
 Age              0
Sex              0
ChestPainType    0
RestingBP        0
Cholesterol      0
FastingBS        0
RestingECG       0
MaxHR            0
ExerciseAngina   0
Oldpeak          0
ST_Slope         0
HeartDisease     0
dtype: int64
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 32)	512
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 1)	17

```
 Total params: 1,057 (4.13 KB)
 Trainable params: 1,057 (4.13 KB)
 Non-trainable params: 0 (0.00 B)

✅ FINAL TEST ACCURACY: 0.8370
```


Training vs Validation Loss
Train Loss
Val Loss
Loss
Epoch
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25


Training vs Validation Accuracy
Train Acc
Val Acc
Accuracy
Epoch
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
5
10
15
20
25


Confusion Matrix — Heart Disease Detection
67
15
15
87
True
0
1
Predicted
0
1
80
70
60
50
40
30
20

9. CONCLUSION

The Artificial Neural Network–based diabetes prediction model developed in this project demonstrates how effectively machine learning techniques can analyze medical datasets and perform early-stage diabetes risk classification. Using the Pima Indians Diabetes Dataset, the system followed a structured workflow that included dataset loading, preprocessing with StandardScaler, train–test splitting, ANN construction with hidden layers and dropout, and final evaluation using multiple performance metrics.

The ANN model, designed with two hidden layers (64 and 32 neurons) and a dropout layer to reduce overfitting, successfully captured the nonlinear relationships among clinical features such as glucose level, BMI, age, insulin, blood pressure, and diabetes pedigree function. Through training over 35 epochs, the model learned meaningful patterns from the data and demonstrated strong performance during testing.

Evaluation metrics—including accuracy, precision, recall, F1-score, and a confusion matrix—confirmed that the model can effectively classify individuals into diabetic and non-diabetic categories. The plotted graphs for training/validation accuracy and loss provided insights into model stability and learning behavior, while the confusion matrix visualization highlighted the classification quality on unseen data.

Although the model is trained on a relatively small dataset and is intended strictly for academic and experimental purposes, it demonstrates the potential of ANN-based prediction systems in healthcare analytics. Such machine learning solutions can assist medical professionals by offering early risk assessments, contributing to preventive healthcare initiatives.

Overall, the project successfully showcases how an Artificial Neural Network can be implemented for diabetes prediction, providing a solid foundation for future improvements, such as incorporating larger datasets, tuning hyperparameters, experimenting with deeper networks, or integrating the system into real-time medical applications.

10.REFERENCES

[1] Kaggle (2024). *Pima Indians Diabetes Dataset*. Online repository used as the primary dataset for training and evaluating the diabetes prediction model.

[2] Dua, D. & Graff, C. (2019). *UCI Machine Learning Repository*. University of California, Irvine. Source of benchmark datasets widely used for diabetes prediction research.

[3] Chollet, F. & TensorFlow Developers (2015–2024). *TensorFlow/Keras Documentation*. Deep learning framework used to build, train, and optimize the Artificial Neural Network.

[4] Scikit-Learn Developers (2011–2024). *Scikit-Learn Machine Learning Library*. Tools used for data preprocessing, StandardScaler, train–test splitting, and model evaluation metrics.

[5] McKinney, W. (2010). *Pandas: A Foundational Python Library for Data Analysis*. Library used for reading, exploring, and processing the diabetes dataset.

[6] Harris, C. et al. (2020). *NumPy: The Fundamental Package for Scientific Computing in Python*. Used for numerical operations and array management in preprocessing.

[7] Brownlee, J. (2022). *Deep Learning for Tabular Data*. A resource discussing ANN-based classification models for structured datasets like diabetes prediction.

[8] S. Reddy & V. Rao (2023). *Analysis of Artificial Neural Network Models for Medical Diagnosis*. Demonstrates ANN effectiveness for binary clinical classifications similar to diabetes prediction.

[9] Sharma, P. & Gupta, R. (2023). *Role of Feature Scaling and Data Preprocessing in Improving Medical Prediction Models*. Highlights the importance of StandardScaler and clean data pipelines.

[10] Matplotlib Developers (2003–2024). *Matplotlib Visualization Library*. Used for plotting accuracy curves, loss curves, and confusion matrix for model interpretation.