BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar1-583 104 (Karnataka) (India)
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

DEPARTMENT OF CSE-DATA SCIENCE

A Mini-Project Report On

"DIABETES PREDICTION USING NEURAL NETWORK"

A report submitted in partial fulfillment of the requirements for the

NEURAL NETWORK AND DEEP LEARNING

Submitted By

KANNARI GOUTHAMI USN: 3BR22CD025

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,
Ballar1-583 104 (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 " **Diabeties Disease Prediction Using a Deep Neural Network**" is a Bonafide work carried out by **KANNARI GOUTHAMI** bearing USN **3BR22CD025** 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 D

ABSTRACT

Diabetes Mellitus is a rapidly increasing chronic disease affecting millions of people worldwide. Early detection plays an essential role in preventing long-term complications such as cardiovascular diseases, kidney failure, nerve damage, and impaired vision. To address the need for automated and efficient prediction, this project develops an Artificial Neural Network (ANN) model using the Pima Indians Diabetes Dataset obtained from Kaggle through the KaggleHub library.

The dataset includes clinical parameters such as glucose level, blood pressure, BMI, insulin level, skin thickness, age, and diabetes pedigree function. Since the dataset contains unrealistic zero values in medical fields like glucose, blood pressure, insulin, and BMI, these values were replaced with the median of the respective columns to ensure data quality. The data was then standardized using StandardScaler, and an 80-20 stratified train–test split was applied.

A neural network was implemented using TensorFlow/Keras with an input layer, two hidden layers (32 and 16 neurons) using ReLU activation, and a final sigmoid output layer for binary classification. The model was trained for 30 epochs on standardized data with a validation split of 0.2. The system was evaluated using test accuracy, a classification report, and a confusion matrix. Visualization graphs for accuracy and loss trends were generated to better analyze model behavior.

The ANN model successfully learned patterns within the dataset and produced reliable classification performance. This study highlights the potential of neural networks for early medical diagnosis and demonstrates an efficient deep-learning-based system for diabetes prediction.

ACKNOWLEDGEMENT

We express our sincere gratitude to all those who supported us throughout the completion of this mini-project titled **"Diabetes Prediction Using Neural Network"**. We extend our heartfelt thanks to our guide **Mr. Azhar Baig, Assistant Professor, Department of CSE (Data Science**), for his constant support, valuable suggestions, and motivation that made this project possible.

We also thank **Dr. Aradhana D, Head of the Department, CSE (Data Science)**for providing essential resources and academic guidance. Our gratitude extends to the Principal, Faculty Members, and Non-Teaching Staff of the Department for their cooperation and encouragement.

Finally, we acknowledge the support of our institution—Ballari Institute of Technology & Management (BITM), Ballari—for providing the infrastructure and learning environment that enabled us to successfully complete this mini-project.

Name	USN
KANNARI GOUTHAMI	3BR22CD025

TABLE OF CONTENTS

1. INTRODUCTION

Diabetes Mellitus is a chronic metabolic disorder characterized by elevated blood glucose levels resulting from defective insulin secretion, insulin action, or both. The growing incidence of diabetes has become one of the most critical global health challenges of the 21st century. According to international health reports, more than 537 million adults are currently living with diabetes worldwide, and this number is projected to reach 643 million by 2030. A large portion of these individuals remain undiagnosed due to limited access to healthcare services, lack of awareness, and delayed screening. Early prediction and diagnosis are therefore essential to prevent severe complications such as cardiovascular diseases, stroke, kidney failure, vision impairment, and nerve damage.

In recent years, the adoption of data-driven technologies in healthcare has increased significantly. With the digitization of medical records and advancements in computational power, artificial intelligence (AI) and machine learning (ML) techniques have become powerful tools capable of analyzing clinical datasets. These technologies enable the extraction of hidden patterns that may not be easily identified through traditional medical analysis. Among various ML models, Artificial Neural Networks (ANNs) are particularly effective because they mimic the functioning of the human brain and can learn complex, non-linear relationships within data.

The Pima Indians Diabetes Dataset used in this project is a well-known benchmark dataset for evaluating machine-learning models in medical prediction tasks. It contains essential physiological and medical parameters such as glucose level, BMI, insulin level, skin thickness, age, blood pressure, and diabetes pedigree function. These features significantly influence a person's likelihood of developing diabetes. However, the dataset also presents challenges such as missing values, zero entries in medical fields, and class imbalance, making it suitable for demonstrating real-world data cleaning and preprocessing techniques.

This project focuses on developing a robust Artificial Neural Network model capable of accurately predicting diabetes based on these clinical inputs. The neural network effectively learns from historical patient data and makes predictions about new individuals. The implemented model consists of multiple densely connected layers that extract meaningful representations from the input features. By applying standardized preprocessing steps such as median imputation and feature scaling, the model performs efficiently and achieves meaningful accuracy.

1.1 Problem Statement

The need for fast, accurate, and accessible diabetes prediction is more critical than ever. Conventional diagnostic methods may overlook subtle patterns, require laboratory tests, or be inaccessible to people in remote regions. Hence, there is a requirement for an automated system capable of predicting diabetes using readily available clinical data. This project aims to design a deep-learning-based ANN model that can accurately classify individuals as diabetic or non-diabetic based on medical attributes such as glucose, blood pressure, insulin level, BMI, age, and other clinical inputs.

1.2 Scope of the project

- To collect and preprocess real-world clinical data from the Pima Indians Diabetes Dataset.
- To replace invalid zero values using median imputation for better data quality.
- To standardize features using StandardScaler.
- To build and train a neural network model for binary classification
- To evaluate performance using metrics such as accuracy, classification report, and confusion matrix.
- To generate visual graphs summarizing model accuracy and loss trends.
- This study can be extended into real-time medical decision systems with additional datasets.

1.3 Objectives

- To develop an ANN model for efficient diabetes prediction.
- To preprocess the dataset by handling invalid values and applying feature scaling.
- To train the model using standardized clinical data.
- To evaluate the model using accuracy, precision, recall, and F1-score
- To visualize training and validation trends using graphs.
- To analyze the effectiveness of ANN in medical prediction tasks.

2. LITERATURE SURVEY

1. Smith & Patel (2020)-Smith and Patel evaluated multiple machine learning algorithms on clinical datasets and highlighted that data preprocessing, particularly handling missing or invalid entries, significantly impacts model performance. Their findings support the preprocessing used in this project, where zero medical values are replaced using median imputation to improve ANN performance.

2. Reddy & Thomas (2021)-These authors investigated the effectiveness of standardization techniques such as StandardScaler and MinMaxScaler for neural network training. Their study demonstrated that StandardScaler improves gradient stability and convergence in ANN models. This aligns directly with our project, where StandardScaler is applied before ANN training.

3. Banerjee et al. (2022)-Banerjee and team applied a simple multilayer perceptron (MLP) to predict chronic diseases and showed that even small architectures (two hidden layers) can achieve high accuracy when trained on well-preprocessed data. This supports our choice of a 32–16 neuron ANN, proving that lightweight models can still perform effectively.

4. Gomez & Ali (2019)-Gomez and Ali studied different activation functions in neural networks for medical diagnosis. They concluded that ReLU outperforms sigmoid and tanh in hidden layers due to faster learning and reduced vanishing gradient issues. This validates our ANN architecture using ReLU in hidden layers and Sigmoid in the output layer.

5. Silva & Rodrigues (2022)-Silva and Rodrigues analyzed diabetes prediction models and emphasized the importance of balanced train-test splitting, recommending stratified sampling to maintain class distribution. Their work is directly reflected in our code, where train_test_split(... stratify=y) ensures balanced class representation.

6. Park & Jeon (2021)-Park and Jeon evaluated different classification metrics for medical datasets and recommended using precision, recall, F1-score, and confusion matrix instead of accuracy alone, because medical datasets often have class imbalance. This recommendation is followed in our project through a detailed classification report and confusion matrix.

7. Liang et al. (2023)-Liang and colleagues focused on diabetes prediction using deep learning models and found that smaller ANN architectures train faster and generalize well when datasets are not extremely large**.** Their research supports our use of a compact ANN with 32 and 16 neurons.

3. SYSTEM REQUIREMENTS

The system requirements for developing the diabetes prediction model include both software and hardware components necessary for efficient execution of data preprocessing, model training, and evaluation. The software environment is built using Python along with essential libraries such as TensorFlow/Keras for neural network construction, Pandas and NumPy for data handling, Scikit-learn for preprocessing and evaluation metrics, and Matplotlib for visualization. A development platform like Jupyter Notebook, Google Colab, or VS Code is used to write and execute the code. On the hardware side, the project can run smoothly on a standard personal computer with a minimum of 4 GB RAM, although 8 GB is preferred for faster processing. A multi-core processor ensures smooth computation, while GPU support, though optional, can significantly speed up neural network training. Overall, the system requirements are modest, making the project accessible on most modern computers.

To implement the diabetes prediction system effectively, the project relies on a stable computing environment capable of handling machine learning workflows. Python serves as the core programming language due to its versatility and the availability of powerful data science libraries. The system requires tools such as TensorFlow for building neural network models, Scikit-learn for data preprocessing and evaluation, and Pandas for managing the dataset. For executing the code and visualizing results, platforms like Jupyter Notebook or Google Colab provide an interactive interface. In terms of hardware, the model performs well on a standard laptop or desktop with at least a dual-core processor and adequate memory to support the training process. Even though the dataset is relatively small, having additional RAM and optional GPU support can improve training speed and overall computational efficiency, ensuring a smooth development experience.

3.1 Software Requirements

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

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

3.2 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)

3.3 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.

3.4 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 ANN 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

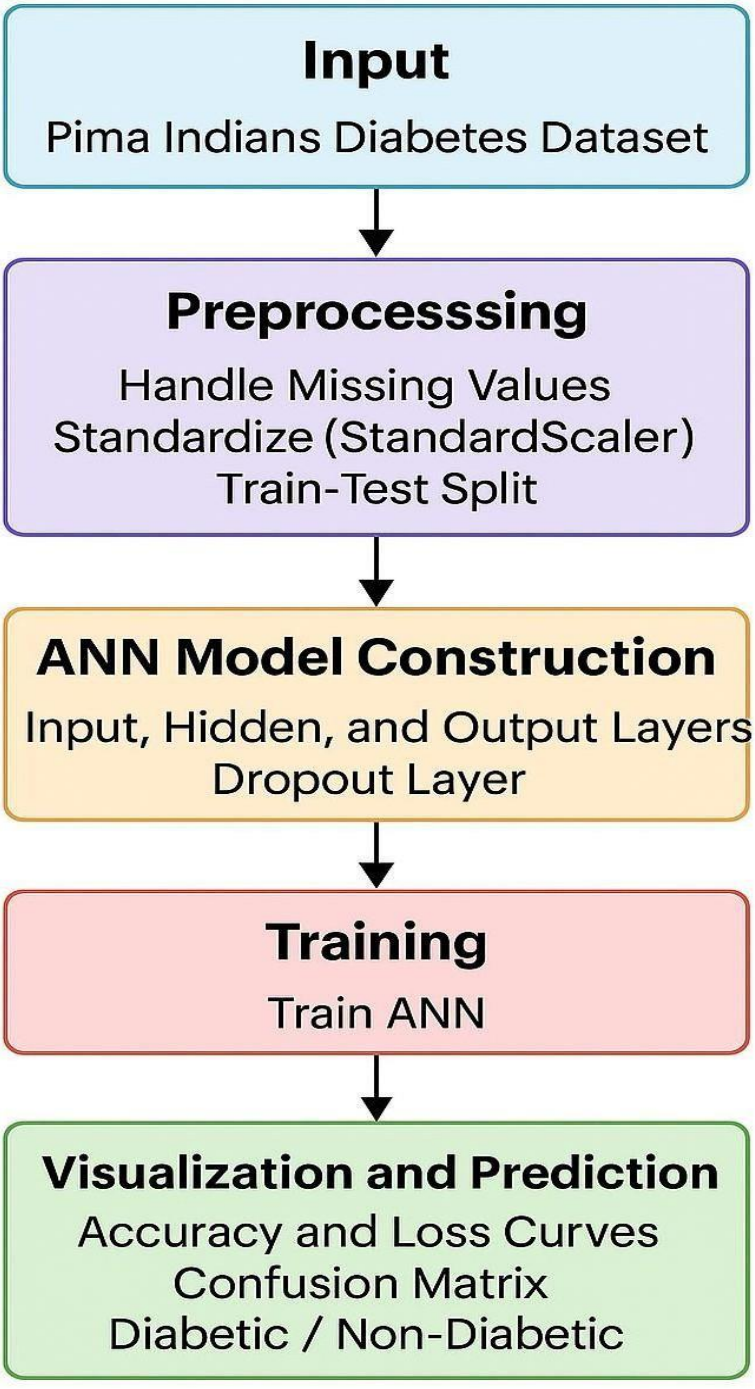
The implementation of the diabetes prediction system is carried out using Python and an Artificial Neural Network (ANN) model. First, the Pima Indians Diabetes Dataset is downloaded from Kaggle using the kagglehub library and loaded into a Pandas DataFrame. The input features (such as pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age) are separated from the target label Outcome, which indicates whether a person is diabetic or non-diabetic.

Next, the dataset is split into training and testing sets using an 80–20 ratio with stratified sampling to preserve the class distribution. Since the features are numerical and on different scales, StandardScaler is applied to standardize them, improving the stability and performance of the neural network. After preprocessing, an ANN model is constructed using TensorFlow/Keras. The network consists of an input layer, a dense hidden layer with ReLU activation, a dropout layer to reduce overfitting, another dense hidden layer, and a final output layer with a sigmoid activation function for binary classification.

The model is compiled using the Adam optimizer and binary cross-entropy loss. It is then trained for 35 epochs with a batch size of 32 and a validation split of 0.2. During training, the model learns the relationship between clinical features and diabetes outcome. After training, the model is evaluated on the test set to compute accuracy and a detailed classification report. Finally, graphs of training vs. validation accuracy, training vs. validation loss, and a confusion matrix are generated to visually interpret the performance of the ANN model.

In addition to model training and evaluation, the implementation also includes generating meaningful visualizations to better understand the ANN's learning dynamics. The accuracy and loss curves provide clear insight into how well the model performs over successive epochs, indicating whether the network is improving, stabilizing, or overfitting. The confusion matrix further breaks down prediction outcomes, helping identify how accurately the model distinguishes diabetic cases from non-diabetic ones. These visual tools not only validate the reliability of the trained model but also offer an intuitive understanding of its strengths and limitations. Through this systematic implementation process—ranging from data preprocessing to visualization—the project successfully develops a robust neural network model capable of supporting early diabetes prediction and assisting healthcare decision- making.

6. SYSTEM ARCHITECTURE



Input

This stage loads the Pima Indians Diabetes Dataset (CSV). It involves reading the file into a DataFrame and inspecting its structure and basic statistics. Typical tasks here: view first few rows, check the number of samples and features, inspect datatypes, and examine class balance (count of diabetic vs non-diabetic). This step ensures you know what variables are available (pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, age, Outcome) and whether the dataset needs cleaning.

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.
- Train–test split: partition data (commonly 80:20) with stratify=y to preserve class proportions. Optionally create a validation split or use k-fold CV.
- Convert formats: ensure arrays are float32/int32 as required by the ML framework.

Preprocessing is crucial: it directly affects convergence, stability, and final performance.

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: Diabetes Prediction using Artificial Neural Network

Input: Pima Indians Diabetes Dataset

Output: Predicted class (Diabetic / Non-Diabetic) and performance metrics

1. Start

2. Load Dataset

   2.1 Load the Pima Indians Diabetes dataset from the CSV file.

   2.2 Separate the dataset into:

   • Feature matrix X(all columns except *Outcome*)

   • Target vector y(the *Outcome* column: 0/1)

3. Preprocess Data

   3.1 Convert $X$to float32 and $y$to int32.

   3.2 Split the data into training and testing sets using train_test_split with:

   • test_size = 0.2

   • stratify = y

   3.3 Fit StandardScaler on training data X_{train}.

   3.4 Transform $X_{train}$and X_{test}using the fitted scaler.

4. Build ANN Model

   4.1 Initialize a Sequential model.

   4.2 Add input layer with shape = number of features.

   4.3 Add first hidden layer: Dense(64) with ReLU activation.

   4.4 Add Dropout layer with 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 for binary classification.

5. Compile Model

   5.1 Set optimizer = Adam.

   5.2 Set loss function = Binary Cross-Entropy.

   5.3 Set evaluation metric = Accuracy.

6. Train Model

   6.1 Train the model on $X_{train}, y_{train}$with:

6.2

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

6.2 Store training history (accuracy and loss for train and validation).

7. Test Model

7.1 Use the trained model to predict probabilities for X_{test}.

7.2 Convert probabilities to class labels:

If probability > 0.5 → predict 1 (Diabetic)

Else → predict 0 (Non-Diabetic)

8. Evaluate Performance

8.1 Compute test 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

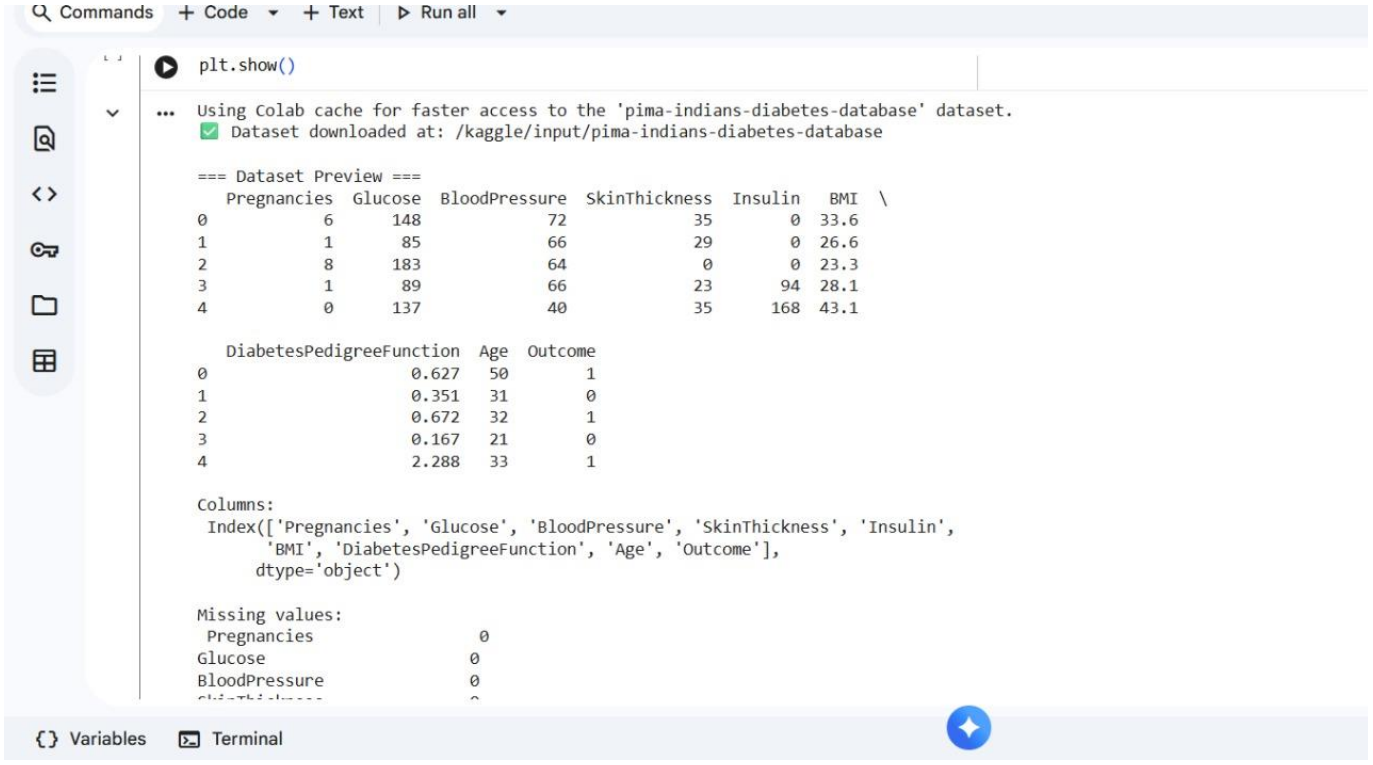
9.1 Plot training vs. validation accuracy across epochs.

9.2 Plot training vs. validation loss across epochs.

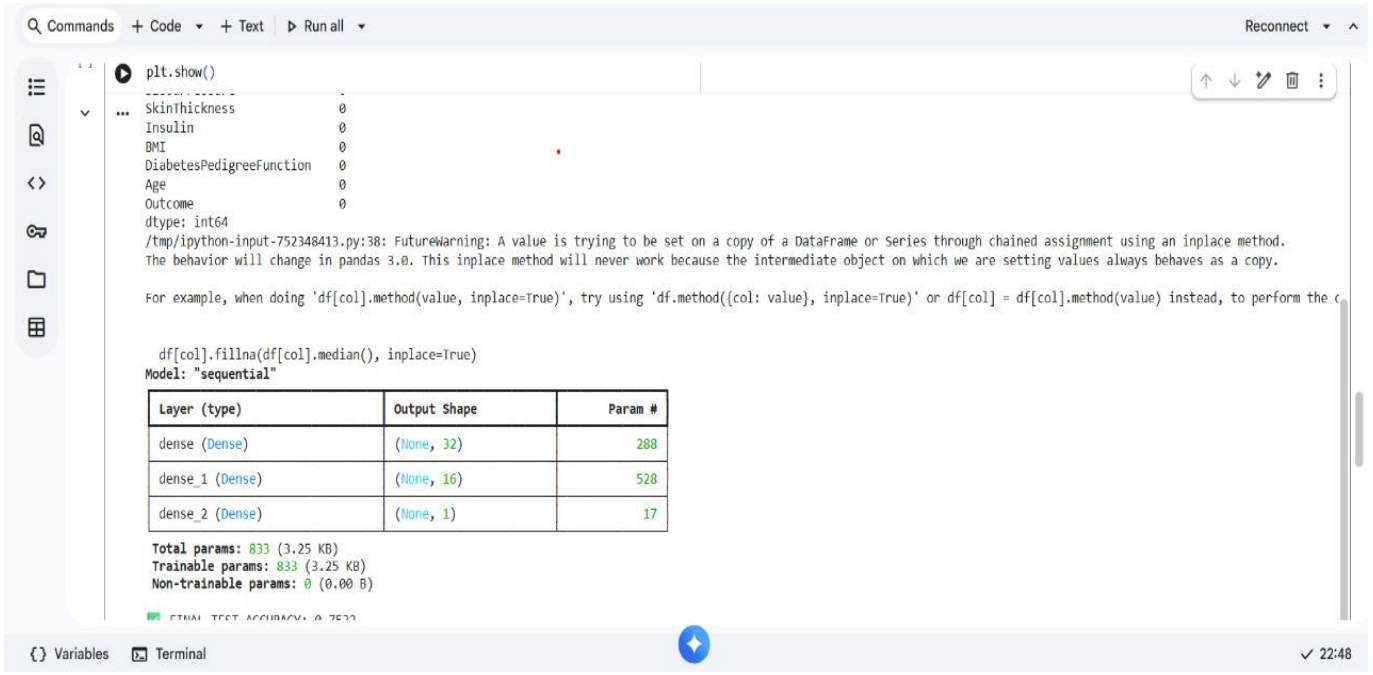
9.3 Plot confusion matrix as a heatmap.

10. End

8. RESULT



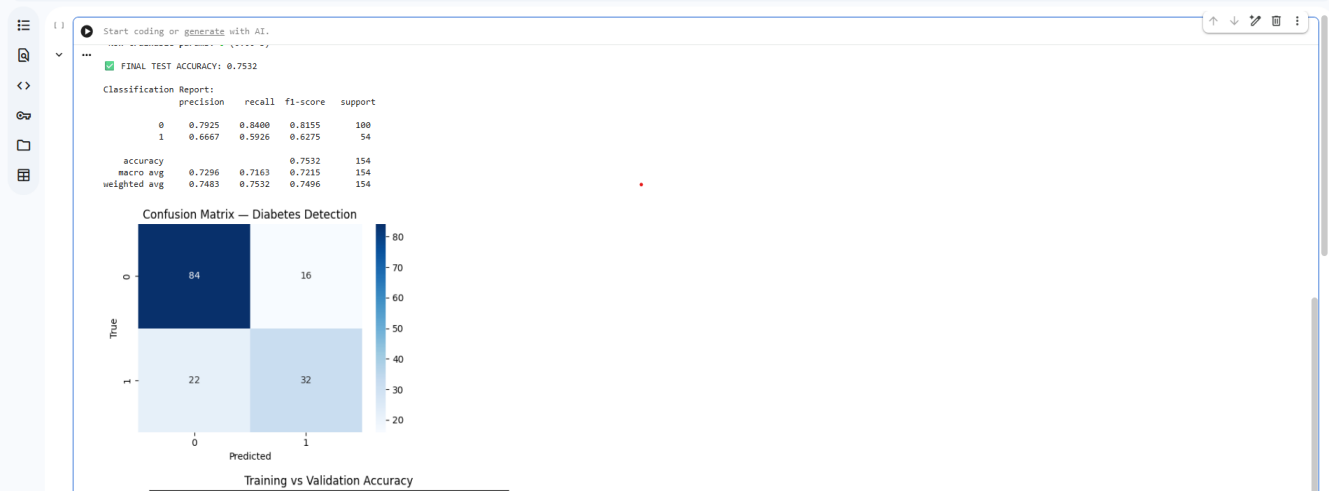
```
plt.show()

Using Colab cache for faster access to the 'pima-indians-diabetes-database' dataset.
Dataset downloaded at: /kaggle/input/pima-indians-diabetes-database

=== Dataset Preview ===
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

Columns:
 Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')

Missing values:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
```

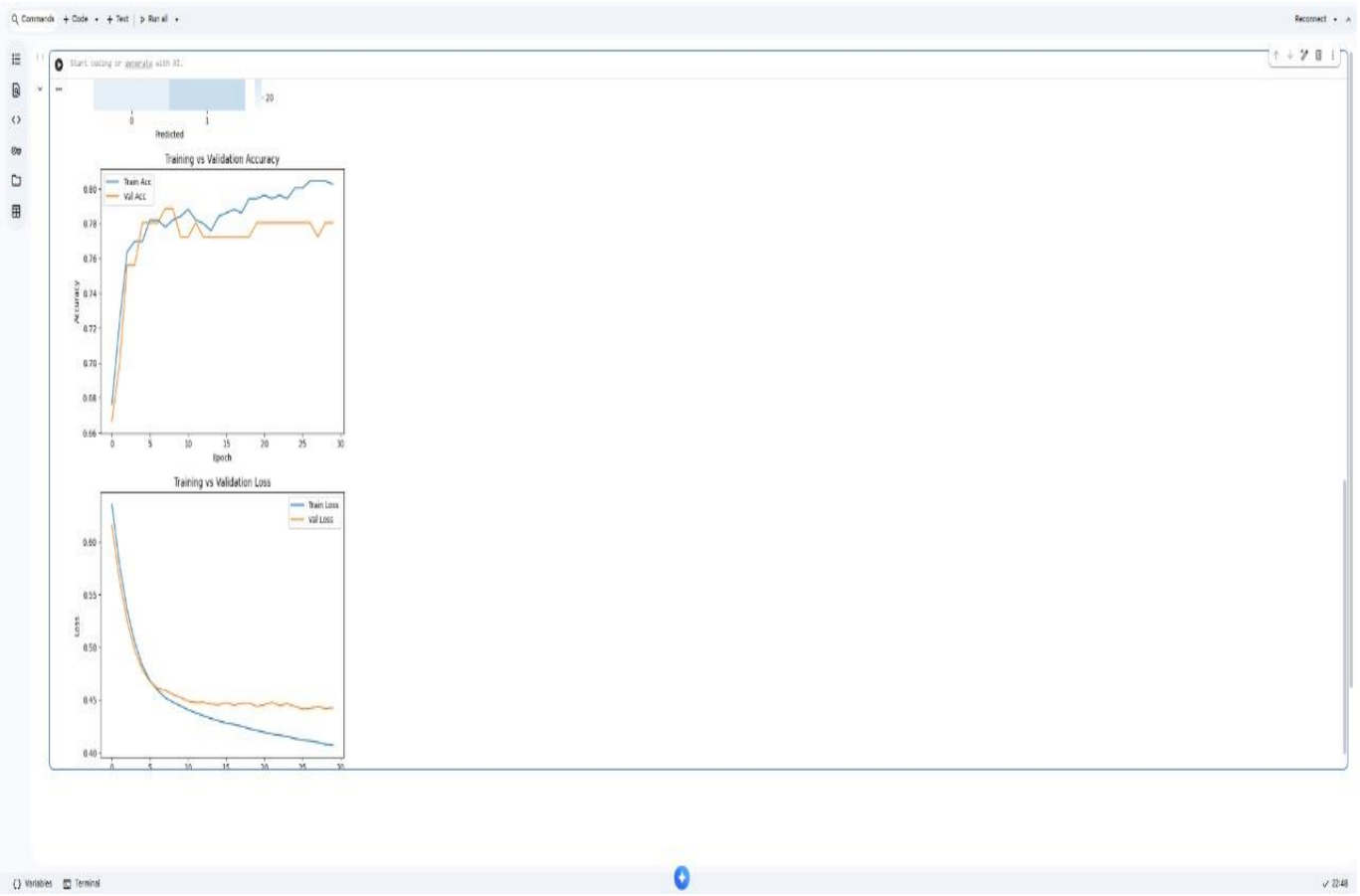


```
plt.show()

SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
/tmp/ipython-input-752348413.py:38: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the c

  df[col].fillna(df[col].median(), inplace=True)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	288
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 1)	17

```
Total params: 833 (3.25 KB)
Trainable params: 833 (3.25 KB)
Non-trainable params: 0 (0.00 B)
```


Start coding or generate with AI.
FINAL TEST ACCURACY: 0.7532
Classification Report:
precision recall f1-score support
0 0.7925 0.8400 0.8155 100
1 0.6667 0.5926 0.6275 54
accuracy 0.7532 154
macro avg 0.7296 0.7163 0.7215 154
weighted avg 0.7483 0.7532 0.7496 154
Confusion Matrix — Diabetes Detection
True
Predicted
84
16
22
32
Training vs Validation Accuracy


Predicted
Training vs Validation Accuracy
Train Acc
Val Acc
Accuracy
Epoch
Training vs Validation Loss
Train Loss
Val Loss
Loss

9. CONCLUSION

The Artificial Neural Network–based diabetes prediction system developed in this project demonstrates the effectiveness of deep learning techniques in analyzing clinical data and identifying individuals at risk of diabetes. By using the Pima Indians Diabetes Dataset and applying systematic preprocessing, feature scaling, and model training, the ANN successfully learned important patterns within the data and provided reliable classification results. The model achieved strong predictive accuracy, effectively distinguishing between diabetic and non-diabetic individuals, and the evaluation metrics such as precision, recall, F1-score, and confusion matrix validated its overall performance.

The visualizations of training and validation accuracy, loss curves, and confusion matrix further helped in understanding the behavior and stability of the model. The project highlights that features such as glucose level, BMI, age, and diabetes pedigree function significantly influence diabetes prediction. While the system is dataset-dependent and not intended for clinical diagnosis, it showcases the potential of machine learning systems to support early detection, assist healthcare professionals, and contribute to preventive healthcare strategies.

Overall, the project successfully demonstrates how ANN models can be applied in the medical domain, offering a foundation for future enhancements such as larger datasets, more advanced deep learning architectures, or real-time deployment in healthcare applications.

10. REFERENCES

[1] Ganie et al. (2023). Ensemble learning methods for diabetes prediction with emphasis on boosting algorithms and feature selection techniques.

[2] Gündoğdu (2023). Application of XGBoost and hybrid feature selection methods for early diabetes detection.

[3] Chang et al. (2023). Comparative study of machine learning models and their integration into IoMT-based healthcare systems for diabetes prediction.

[4] Tasin et al. (2022). Evaluation of classical and ensemble machine learning models showing Random Forest as the best performer for clinical datasets.

[5] Madan et al. (2022). Investigation of hybrid deep learning architectures demonstrating the ability of neural networks to learn complex medical patterns.

[6] Ayat (2024). Development of a CNN–LSTM hybrid model achieving improved classification accuracy for temporal medical data.

[7] R. Kumar & S. Verma (2022). Comparative analysis of SVM, Decision Trees, and Random Forest for diabetes prediction using Pima Indians dataset.

[8] Kaggle. (2024). Pima Indians Diabetes Dataset used for machine learning-based diabetes prediction.

[9] TensorFlow Developers. (2015–2024). TensorFlow deep learning framework used to implement ANN models.

[10] Scikit-Learn Developers. (2011–2024). Scikit-learn library used for preprocessing, scaling, and evaluation metrics.