S. No	CONTENT	PAGE.NO
1	INTRODUCTION 1.1 Introduction 1.2 Overview of Diabetes 1.3 Keywords	1 - 3
2	LITERATURE SURVEY	4 - 5
3	EXPERIMENTAL OR REQUIREMENTS AND METHODS 3.1 SYSTEM SPECIFICATION Hardware Requirements Software Requirements 3.2 Project Description 3.3 Libraries Used	6
4	SYSTEM ANALYSIS 4.1 System Analysis 4.2 Proposed System 4.3 System Study 4.4 Dataset 4.5 Data Processing	7 - 10
5	AIM & SCOPE OF THE PRESENT INVESTIGATION 5.1 Aim 5.2 Scope	11
6	OVERVIEW DIABETES 6.1 What is diabetes 6.2 Types 6.3 Symptoms 6.4 Complication 6.5 Diagnosis & Tests 6.6 Prevention	12 - 20
7	PREDICTION USING DEEP LEARNING 7.1 Introduction	21 – 32

	7.2 ML Approaches	
	7.3 Methods	
	7.4 Phases	
8	INTRODUCTION TO PYTHON FOR DATA ANALYSIS	33 – 41
	8.1 Introduction python	
	8.2 Need for the ML	
	8.3 Data analysis	
	8.4 Anaconda Navigator	
	8.5 Jupiter Notebook	
9	LIBRARIES USED	42 – 46
10	METHODOLOGIES	47 – 50
11	SYSTEM DESIGN OR ARCHITECTURE	51 – 54
	11.1 System Architecture	
	11.2 Use Case Diagram	
	11.3 Flow Chart Diagram	
12	IMPLEMENTATION	55 – 69
	12.1 Random Forest	
	12.2 K-Nearest Neighbor	
	12.3 Classifier using gradient Boosting	
	12.4 Deep Neural Networks	
13	RESULTS AND SCREENSHOTS	70 – 84
14	CONCLUTION	85
15	FUTURE ENHANCEMENT	86
16	REFERENCES	87

LIST OF FIGURES

LIST OF FIGURES Figure No	Titles	Page
Fig 1	Symptoms of Diabetes	14
Fig 2	System Architecture	51
Fig 3	Use Case Diagram	53
Fig 4	Flow Chart Diagram for Diabetes Prediction Using Neural Networks	54

INTRODUCTION

This is a classical classification problem of supervised machine learning. Goal: to perform a diagnostic study on whether or not a patient has diabetes or not inconsistent with the data given. Background Diabetes Infection is a common chronic disease that causes great threats to human health. Diabetes is an ordinarily identified illness which includes abnormal quantities of blood glucose and malfunctioning insulin secretion characteristics in it. It can cause chronic impairment and failure of a kind which hurt particular tissues primarily; to e.g., eyes (retinopathy), kidneys (nephropathy), heart (cardiomyopathy) blood vessels (atherosclerosis). nerves (neuropathy). What Are the Exceptional Forms of Diabetes? An extended existence changes within the living conditions, a common incidence in contemporary age reared its sharp teeth in the subject of what have become as soon as coined form 2 diabetes. As a result, it is a research field which should be paid more attention to how to quickly and accurately diagnose type 2 diabetes. Based on the daily physical examination data, people can use machine learning to have an inkling of whether they are diabetic or not, as a reference for doctors. The most pressing problems for machine learning are feature selection and the choice of an appropriate classifier. So Logistic Regression and Decision Tree are used for predicting the diabetes in this study.

Overview of Diabetes

Diabetes is a chronic disease that does a lot of serious harm, millions of people are affected by it throughout the world. Its hallmark feature is high blood glucose(sugar) levels, that can lead to a number of health problems. complications like heart disease, organ failure, blindness and amputation. With the devastating health impacts of diabetes, much activity has gone into trying to determine how to predict whether a person will develop the disease. Machine learning particularly deep neural networks especially in recent years have shown great potential towards predicting diabetes.

A class of machine-learning algorithms called deep neural networks — which are well-suited for the kind of complex, high-dimensional data encountered in medical research — made a significant impact.

There are many machine learning algorithms which can be used to predict diabetes. One method is supervised learning, applying a machine model to the dataset from a diabetes patients with labels and which were without for thousands of registered diabetics and non-diabetics. This model can then be applied to predict diabetes risk of other patients based on their d and train an artificial intelligence model using a labelled data set that include diabetic and non-diabetic patient. The model can then be used to predict how likely a person is to develop diabetes based on their demographic characteristics, medical history, among other variables. Age, sex, race / ethnicity medical history other aspects of your health.

Yet another approach is unsupervised learning, here you will have to train the model using an unlabelled patient data dataset. From there, the model can identify patterns in the data missing to human analysts like the effect of different risk factors for diabetes on one another.

Why DNNs: A good use case for deep neural networks would be working with large and complex datasets. These networks are built of layers of linked nodes that process different things in the input data. It enables the network to identify more complex structures in the data once each layer's output has been relayed free of error to its subsequent higher level.
The benefit to machine learning and deep neural networks for diabetes prediction is that they can pick up on patterns in the data that may be too subtle or complex for humans. Such models have to consider a myriad of risk factors that span demographic, medical history, lifestyle and genetic predisposition mean for diabetes.

Keywords:

Deep Learning, Deep Neural Networks, AUC, Hyperparameter tuning, Diabetes prediction, Biomarkers,

Gradient Boosting Algorithm (GBM),

TensorFlow, Keras

Prediction

AIM: To accurately predict whether or not the patients in the dataset have diabetes or not.

SCOPE: The accuracy of the model is obtained by comparing the predicted values against the original set of values. Identifying diabetics or predicting the upcoming of a diabetic life can be propelled by using various machine learning techniques.

In this project we are going to use a dataset named Diabetes.csv file which contains the details such as, Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age and Outcome.

LITRATURE SURVEY

Birjais *et al*. [8] experimented on PIMA Indian Diabetes (PID) data set. It has 768 instances and 8 attributes and is available in the UCI machine learning repository. They aimed to focus more on diabetes diagnosis, which, according to the World Health Organization (WHO) in 2014, is one of the world's fastest-growing chronic diseases. Gradient boosting, logistic regression, and naive Bayes classifiers were used to predict whether a person is diabetic or not, with gradient boosting having an accuracy of 86%, logistic regression having a 79% accuracy, and naive Bayes having a 77% accuracy.

Amina Azar, Yasir Ali, Muhammad Awais, Khuram Zaheer (2018) [1] For the prediction of diabetes, compare data mining models.
Volume 9 Issue 8 of the Advanced Computer Science and Applications International Journal (IJACSA) contains an article.
This article compares Using the PIDD dataset, decision tree and K-Nearest Neighbor algorithms are used. However, decision trees are prone to overfitting and instability of predictions due to small changes in data. KNN algorithms are computationally expensive and sensitive to irrelevant features and the choice of k.

Dilip Singh Sisodia and Deepti Sisodia (ICCIDS 2018) Diabetes Prediction Using A Classifier Algorithm. Article Published The decision tree approach was used to the PIDD dataset in the International Conference on Computational Intelligence and Data Science. The decision tree algorithm also has drawbacks such as overfitting, instability due to small changes in data, difficulty in interpretation, bias towards features with many outcomes, step-like decision boundaries, and limited to binary splits.

Veena In 2015 [5], Vijayan.V. and Anjali.C. Decision Support Systems for Diabetic Mellitus Prediction: A Review at the World Conference on Communication Technologies (GCCT) used a neural network model, However, neural networks are computationally expensive, prone to overfitting, and difficult to interpret.

Dong-Ping Rao, Qiug Liu, Yi-Xiang Huang, and Xue-Hui Meng (2016) [6] On the basis of risk variables, three data mining methods are evaluated for predicting diabetes or prediabetes. Article published in 10.1016/j.kjms. compare KNN on the other hand, Random Forest algorithms PIDD dataset. However, KNN algorithms are computationally expensive for large datasets and sensitive to irrelevant features and difficulty in determining the optimal value of k. Neural networks are prone to overfitting, difficult to interpret, and sensitive to noisy data.

Mujumdar and Vaidehi [27] presented a diabetes prediction model for better diabetes classification that included a few extrinsic factors that caused diabetes, 2 4 as well as regular components such as glucose, BMI, age, insulin, and so on. The new data set enhanced classification accuracy when compared to the old data set. Multiple ML approaches were used on the data set, and classification was done with a variety of algorithms, with LR yielding the highest accuracy at 96%. The AdaBoost classifier was found to be the most accurate, with a 98.8% accuracy rate. They used two separate data sets to compare the accuracy of ML techniques. When compared to the existing data set, it was clear that the model improved diabetes prediction accuracy and precision.
Mercaldo et al.[28] offered a strategy for classifying diabetic patients based on a set of features chosen according to the WHO criteria. Evaluating real-world data using state of the art machine learning algorithms. The model was trained using six alternative classification approaches, with the Hoeffding Tree method scoring 0.770 in precision and 0.775 in recall. They used data from the PIMA Indian community in Phoenix, Arizona, to evaluate the method.

EXPERIMENTAL OR REQUIREMENTS AND METHODS

3.1 SYSTEM SPECIFICATION

Hardware Requirements:

1. Processor – Intel Core processors or any AMD chips.
2. RAM – 4 GB
3. Hard Disk – 64GB
4. Operating System – Windows 7 and above

Software requirements:

1. Jupiter notebook (anaconda 3).
2. Python 3 or latest version.

3.2 PROJECT DESCRIPTION

We are going to predict diabetes via three different Deep learning methods including: Random Forest, k-Nearest Neighbour, Gradient Boosting and Deep Neural Network. We have got this dataset from Kaggle. We are going to analyze the information using Logistic Regression in this dataset and try to present in a visual manner using different libraries in python programming language like NumPy, Pandas and matplotlib.

3.3 LIBRARIES USED

We will generally be working on this project using the standard libraries such as:

- Pandas
- NumPy
- Matplotlib
- Sklearn
- Tensorflow
- Keras

SYSTEM ANALYSIS

4.1. System Analysis:

To develop a diabetes prediction system using machine and deep neural networks, you would need to go through a similar system analysis process to identify the key components and objectives of the system. This would involve collecting and interpreting relevant data to identify the problem of diabetes prediction and decomposing the system into its components to better understand its objectives.
The system analysis process would also involve examining the performance of existing algorithms used for diabetes prediction and identifying areas where improvements can be made. You would need to define the deep neural network's parameters and hyperparameters to enhance its functionality and make use of methods like transfer learning and hyperparameter tuning to increase precision and shorten training times.

Disadvantages of Existing Systems:
Limited data may lead to overfitting and lower accuracy.
The accuracy of the prediction may be the quality of the data has limitations.
The prediction models may not be easily interpretable, making it difficult to understand the underlying factors contributing to the prediction.
Overall, the system analysis process for developing a diabetes prediction system would involve identifying the key components and objectives of the system, examining the effectiveness of current algorithms and pinpointing opportunities for system performance enhancement.

4.2. Proposed System:

- ✓ The proposed methodology involves developing a hybrid model that combines methods for machine learning and deep neural networks to accurately predict the probability that someone may get diabetes.
- ✓ The model is trained using a large and diverse dataset that includes demographic and health-related features such as age, gender, BMI, blood pressure, glucose levels, and family history of diabetes. 18
- ✓ The proposed methodology includes a feature selection process to determine which attributes are most important for the prediction task, and feature engineering to transform the raw data into meaningful features that can improve the model's performance.

- ✓ The hybrid model uses a deep neural network architecture such as a Long Short-Term Memory (LSTM) network or a Convolutional Neural Network (CNN) can recognise intricate correlations and patterns in the data and provide precise predictions.
- ✓ In order to determine the model's efficacy, it is assessed using established assessment measures including accuracy, precision, recall, F1 score, and AUC-ROC.
- ✓ The proposed methodology is expected to provide high accuracy in diabetes prediction and can be used in clinical settings to help healthcare providers make informed decisions about patient care.

Disadvantages of Proposed System:

❖ Data availability and quality: The accuracy and effectiveness rely significantly on the accessibility and calibre of the data utilised for training and testing the proposed model. As a result of inadequate or flawed data, predictions may be inaccurate.
❖ Model complexity and interpretability: Deep learning models can be complex and difficult to interpret, which can be a limitation in some applications where transparency and interpretability are critical, such as in healthcare.
❖ Overfitting: Deep learning models can be prone to overfitting, leading to reduced generalization performance on new data. Regularization techniques such as dropout and early stopping can be used to mitigate this issue.
❖ Computational resources: Training deep neural networks can require huge volumes of memory and high-performance computer systems are major computational resources. For some applications with constrained resources, this can be a restriction.

4.3. System Study:

The proposed system for diabetes prediction using machine learning and deep neural networks involves developing a model that uses a person's medical history and lifestyle variables to forecast the risk that they may acquire diabetes. To train and evaluate the model, the system needs a collection of 19 medical records comprising details like age, gender, BMI, blood pressure, glucose levels, and other pertinent medical data.
The software environment required for the proposed system includes the following:

- ✓ Python is a programming language, and TensorFlow and Keras are two deep learning libraries.
  Machine learning libraries: Scikit-learn
  Data visualization libraries: Matplotlib, Seaborn

The suggested system's feasibility study can be carried out as follows:

Technical feasibility: The suggested method makes use of sophisticated deep learning algorithms, which need substantial computer power and technological know-how. Most firms have the requisite software and hardware infrastructure, and hiring and training may provide the essential skills.

Operational feasibility: The proposed system requires a dataset of medical records, which can be obtained from medical institutions or publicly available datasets. The system can be integrated with existing medical record systems and can be used to automate the process of diabetes prediction.

Economic feasibility: The proposed system is cost-effective compared to traditional medical diagnosis methods, which can be time-consuming and costly. The system requires the initial investment in software and hardware infrastructure, but the long-term benefits outweigh the costs.

Schedule feasibility: The proposed system development can be completed within a reasonable timeframe, considering the availability of skilled resources and the complexity of the system.

Overall, the proposed system for diabetes prediction using machine learning and deep neural networks is technically, operationally, economically, and schedule feasible. It has the potential to provide accurate and timely predictions of diabetes, leading to better management and treatment of the disease. 20

4.4. Dataset:

A condition known as diabetes mellitus is brought on by an excess of glucose in the blood and a deficiency of the hormone insulin, which is generated by the pancreas in the human body. Due to unhealthy lifestyle, diabetes mellitus has become popular these days. It is important for doctors or hospitals to predict based on the current data and different biomarkers whether person will have diabetes or not and accordingly they can plan treatment and further diagnosis plan. This dataset has been formed by taking data of 768 people.

Dataset consists of 768 records and has 9 attributes. This dataset has following attributes:

- BMI: BMI of a patient
- Insulin: insulin measure in a patient's body.
- Blood Glucose: Glucose content in blood
- Age: Age of a person
- Diabetes Pedigree:
- Skin Thickness: Thickness level of skin in a patient
- Blood Pressure: Blood pressure measured
- Pregnancies:
- Outcome: This is the target

This is a classification dataset. We will be predicting the outcome of the variable Outcome which has values as 1 or 0.

Dataset consisted of various numeric, binary categorical columns.

AIM & SCOPE OF THE PRESENT INVESTIGATION

5.1 AIM:

To accurately predict whether or not the patients in the dataset have diabetes or not.

5.2 SCOPE:

The accuracy of the model is obtained by comparing the predicted values against the original set of values. Identifying diabetics or predicting the upcoming of a diabetic life can be propelled by using various machine learning techniques.

In this project we are going to use a dataset named Diabetes.csv file which contains the details such as, Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age and Outcome.

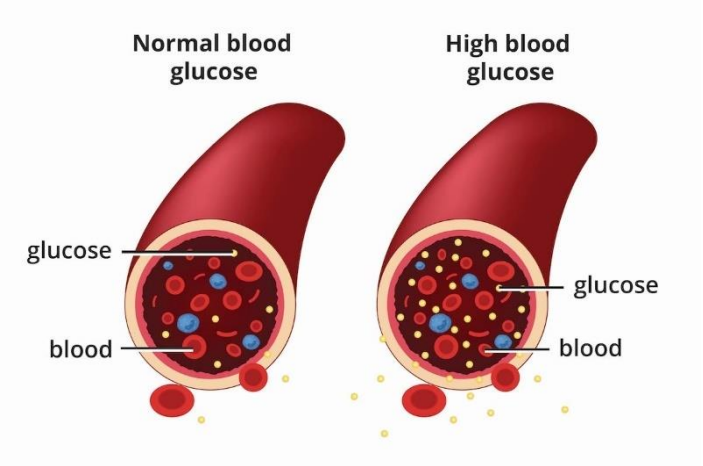
OVERVIEW DIABETES

6.1 What is diabetes

Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Glucose is your body's main source of energy. Your body can make glucose, but glucose also comes from the food you eat.

Insulin is a hormone made by the pancreas that helps glucose get into your cells to be used for energy. If you have diabetes, your body doesn't make enough—or any—insulin, or doesn't use insulin properly. Glucose then stays in your blood and doesn't reach your cells.

Diabetes raises the risk for damage to the eyes, kidneys, nerves, and heart. Diabetes is also linked to some types of cancer. Taking steps to prevent or manage diabetes may lower your risk of developing diabetes health problems.



6.2 Types of diabetes

There are several types of diabetes. The most common forms include:

- **Type 2 diabetes**: With this type, your body doesn't make enough insulin and/or your body's cells don't respond normally to the insulin (insulin resistance). This is the most common type of diabetes. It mainly affects adults, but children can have it as well.

- **Prediabetes**: This type is the stage before Type 2 diabetes. Your blood glucose levels are higher than normal but not high enough to be officially diagnosed with Type 2 diabetes.
- **Type 1 diabetes**: This type is an autoimmune disease in which your immune system attacks and destroys insulin-producing cells in your pancreas for unknown reasons. Up to 10% of people who have diabetes have Type 1. It's usually diagnosed in children and young adults, but it can develop at any age.
- **Gestational diabetes**: This type develops in some people during pregnancy. Gestational diabetes usually goes away after pregnancy. However, if you have gestational diabetes, you're at a higher risk of developing Type 2 diabetes later in life.

Other types of diabetes

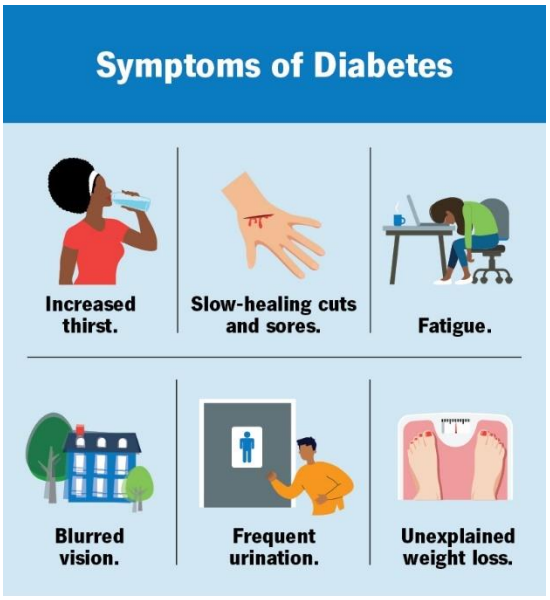
A less common type of diabetes, called monogenic diabetes, is caused by a change in a single gene. Diabetes can also come from having surgery to remove the pancreas, or from damage to the pancreas due to conditions such as cystic fibrosis *NIH external link* or pancreatitis.

Common is diabetes?

Diabetes is common. Approximately 37.3 million people in the United States have diabetes, which is about 11% of the population. Type 2 diabetes is the most common form, representing 90% to 95% of all diabetes cases.

About 537 million adults across the world have diabetes. Experts predict this number will rise to 643 million by 2030 and 783 million by 2045.

6.3 Symptoms of diabetes



Symptoms of diabetes include:

- Increased thirst (polydipsia) and dry mouth.
- Frequent urination.
- Fatigue.
- Blurred vision.
- Unexplained weight loss.
- Numbness or tingling in your hands or feet.
- Slow-healing sores or cuts.
- Frequent skin and/or vaginal yeast infections.

What causes type 1 diabetes?

Type 1 diabetes occurs when your immune system, the body's system for fighting infection, attacks and destroys the insulin-producing beta cells of the pancreas. Scientists think type 1 diabetes is caused by genes and environmental factors, such as viruses, that might trigger the disease. Studies such as TrialNet External link are working to pinpoint causes of type 1 diabetes and possible ways to prevent or slow the disease.

What causes type 2 diabetes?

Type 2 diabetes—the most common form of diabetes—is caused by several factors, including lifestyle factors and genes.

Overweight, obesity, and physical inactivity

You are more likely to develop type 2 diabetes if you are not physically active and are overweight or have obesity. Extra weight sometimes causes insulin resistance and is common in people with type 2 diabetes. The location of body fat also makes a difference. Extra belly fat is linked to insulin resistance, type 2 diabetes, and heart and blood vessel disease. To see if your weight puts you at risk for type 2 diabetes, check out these Body Mass Index (BMI) charts.

Insulin resistance

Type 2 diabetes usually begins with insulin resistance, a condition in which muscle, liver, and fat cells do not use insulin well. As a result, your body needs more insulin to help glucose enter cells. At first, the pancreas makes more insulin to keep up with the added demand. Over time, the pancreas can't make enough insulin, and blood glucose levels rise.

Genes and family history

As in type 1 diabetes, certain genes may make you more likely to develop type 2 diabetes. The disease tends to run in families and occurs more often in these racial/ethnic groups:

- African Americans
- American Indians
- Asian Americans
- Hispanics/Latinos
- Native Hawaiians
- Pacific Islanders

Genes also can increase the risk of type 2 diabetes by increasing a person's tendency to become overweight or have obesity.

What causes gestational diabetes?

Scientists believe gestational diabetes, a type of diabetes that develops during pregnancy, is caused by the hormonal changes of pregnancy along with genetic and lifestyle factors

.

Insulin resistance

Hormones produced by the placenta *NIH external link* contribute to insulin resistance, which occurs in all women during late pregnancy. Most pregnant women can produce enough insulin to overcome insulin resistance, but some cannot. Gestational diabetes occurs when the pancreas can't make enough insulin.

As with type 2 diabetes, extra weight is linked to gestational diabetes. Women who are overweight or have obesity may already have insulin resistance when they become pregnant. Gaining too much weight during pregnancy may also be a factor.

Genes and family history

Having a family history of diabetes makes it more likely that a woman will develop gestational diabetes, which suggests that genes play a role. Genes may also explain why the disorder occurs more often in African Americans, American Indians, Asians, and Hispanics/Latinas.

6.4 Complications of diabetes

Diabetes can lead to acute (sudden and severe) and long-term complications — mainly due to extreme or prolonged high blood sugar levels.

Acute diabetes complications

Acute diabetes complications that can be life-threatening include:

- **Hyperosmolar hyperglycemic state (HHS)**: This complication mainly affects people with Type 2 diabetes. It happens when your blood sugar levels are very high (over 600 milligrams per deciliter or mg/dL) for a long period, leading to severe dehydration and confusion. It requires immediate medical treatment.
- **Diabetes-related ketoacidosis (DKA)**: This complication mainly affects people with Type 1 diabetes or undiagnosed T1D. It happens when your body doesn't have enough insulin. If your body doesn't have insulin, it can't use glucose for energy, so it breaks down fat instead. This process eventually releases substances called ketones, which turn your blood acidic. This causes labored breathing, vomiting and loss of consciousness. DKA requires immediate medical treatment.
- **Severe low blood sugar (hypoglycemia)**: Hypoglycemia happens when your blood sugar level drops below the range that's healthy for you. Severe hypoglycemia is very low blood sugar. It mainly affects people with diabetes who use insulin. Signs include blurred or double vision, clumsiness, disorientation and seizures. It requires treatment with emergency glucagon and/or medical intervention.

Long-term diabetes complications

Blood glucose levels that remain high for too long can damage your body's tissues and organs. This is mainly due to damage to your blood vessels and nerves, which support your body's tissues.

Cardiovascular (heart and blood vessel) issues are the most common type of long-term diabetes complication. They include:

- Coronary artery disease.
- Heart attack.
- Stroke.
- Atherosclerosis.

Other diabetes complications include:

- Nerve damage (neuropathy), which can cause numbness, tingling and/or pain.
- Nephropathy, which can lead to kidney failure or the need for dialysis or transplant.
- Retinopathy, which can lead to blindness.
- Diabetes-related foot conditions.
- Skin infections.
- Amputations.
- Sexual dysfunction due to nerve and blood vessel damage, such as erectile dysfunction or vaginal dryness.
- Gastroparesis.
- Hearing loss.
- Oral health issues, such as gum (periodontal) disease.

Living with diabetes can also affect your mental health. People with diabetes are two to three times more likely to have depression than people without diabetes.

6.5 Diagnosis and Tests

Healthcare providers diagnose diabetes by checking your glucose level in a blood test. Three tests can measure your blood glucose level:

- **Fasting blood glucose test**: For this test, you don't eat or drink anything except water (fast) for at least eight hours before the test. As food can greatly affect blood sugar, this test allows your provider to see your baseline blood sugar.
- **Random blood glucose test**: "Random" means that you can get this test at any time, regardless of if you've fasted.
- **A1c**: This test, also called HbA1C or glycated hemoglobin test, provides your average blood glucose level over the past two to three months.

To screen for and diagnose gestational diabetes, providers order an oral glucose tolerance test.

The following test results typically indicate if you don't have diabetes, have prediabetes or have diabetes. These values may vary slightly. In addition, healthcare providers rely on more than one test to diagnose diabetes.

Type of test	In-range (mg/dL)	Prediabetes (mg/dL)	Diabetes (mg/L)
Fasting blood glucose test	Less than 100.	100 to 125.	126 or higher.
Random blood glucose test	N/A.	N/A.	200 or higher (with classic symptoms of hyperglycemia or hyperglycemic crisis).
A1c	Less than 5.7%.	5.7% to 6.4%.	6.5% or higher.

6.6 Prevention

How can I prevent diabetes?

You can't prevent autoimmune and genetic forms of diabetes. But there are some steps you can take to lower your risk for developing prediabetes, Type 2 diabetes and gestational diabetes, including:

- Eat a healthy diet, such as the Mediterranean diet.
- Get physically active. Aim for 30 minutes a day at least five days a week.
- Work to achieve a weight that's healthy for you.
- Manage your stress.
- Limit alcohol intake.
- Get adequate sleep (typically 7 to 9 hours) and seek treatment for sleep disorders.
- Quit smokSing.
- Take medications as directed by your healthcare provider to manage existing risk factors for heart disease.

In the United States, diabetes is the eighth leading cause of death. A large number of people with diabetes will die from a heart attack or stroke.

However, it's important to know that you can live a healthy life with diabetes. The following are key to a better prognosis:

- Lifestyle changes.
- Regular exercise.
- Dietary changes.
- Regular blood sugar monitoring.

Studies show that people with diabetes may be able to reduce their risk of complications by consistently keeping their A1c levels below 7%.

DIABETES PREDICTION USING DEEP LEARNING

7.1 Introduction

Diabetes rates have been increasing alarmingly each year, particularly when left untreated. Diabetes is a chronic condition caused by excessive glucose in the bloodstream. Individuals with diabetes mellitus are unable to produce enough insulin, a hormone secreted by the pancreas. Insulin plays a vital role in regulating cellular glucose levels, which is essential for energy production. If diabetes remains untreated, various complications may arise, such as visual impairment, cardiovascular problems, dental diseases, stroke, and microvascular complications that can lead to retinopathy, kidney failure, and nerve damage.

Diabetes Management and Monitoring Tools

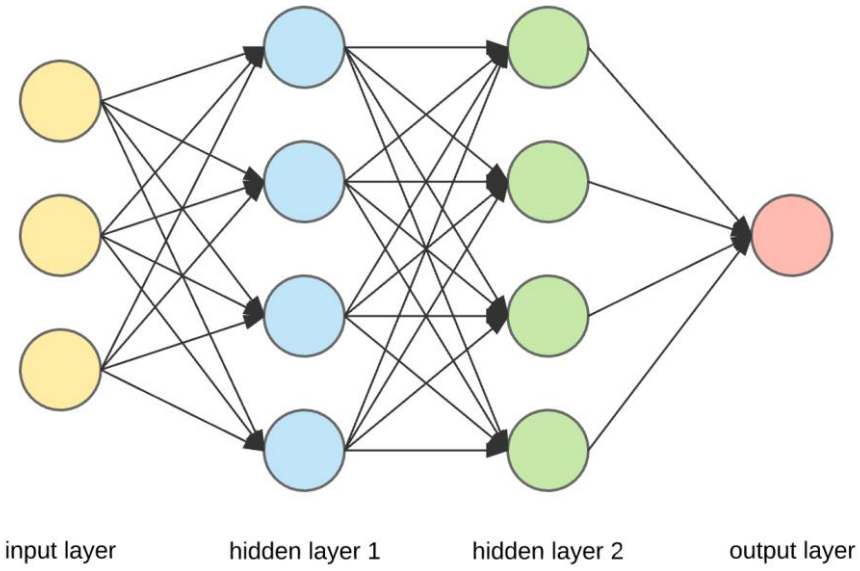
Diabetes management tools are crucial for monitoring glucose levels, insulin usage, and food intake. These tools include activity trackers, glucose meters, continuous glucose monitoring (CGM) devices, and sensor-augmented insulin pumps. The primary goal of glucose management is to prevent unwanted glycemia and its associated complications.

Importance of Early Detection and DL Applications

Early detection of diabetes is critical to prevent the progression of the disease. To help reduce diabetes-related deaths, deep learning (DL) techniques can support automatic disease detection. DL is a subset of machine learning (ML) and artificial intelligence that has made significant advancements in medical applications. Using supervised deep neural networks, DL processes and classifies large volumes of data, enabling early detection of diseases.

Deep Neural Networks (DNNs)

Deep Neural Networks (DNNs) are a type of Artificial Neural Network (ANN) with multiple layers: an input layer, hidden layers, and an output layer. The number of hidden layers is typically two or more, allowing for more complex data processing and learning.



7.2 Machine Learning (ML) Approaches

There are three main types of ML:

1. **Supervised Learning (SL):** Uses labeled input data for model optimization through iterative processes like backpropagation. Popular SL-based DL algorithms include:
    - **Convolutional Neural Networks (CNN):** Primarily used for image recognition tasks by processing multi-dimensional arrays.
    - **Multi-Layer Perceptrons (MLP):** A feed-forward neural network used for classification and regression tasks.
    - **Recurrent Neural Networks (RNN):** Capable of capturing temporal features, making them useful in applications such as speech recognition and natural language processing.

2. **Unsupervised Learning (USL):** Does not require predefined labels for input data. Instead, it uncovers hidden structures in the dataset. Common USL tasks include clustering and dimensionality reduction. Two basic USL architectures are:
    - **Restricted Boltzmann Machines (RBMs):** Estimate probabilistic distributions over input data for mapping representations.
    - **Autoencoders:** Used when the number of output units in the output layer matches the number of input units.
    -
3. **Reinforcement Learning (RL):** Inspired by how humans learn through trial and error, RL algorithms interact with their environment to learn from experiences and achieve the optimal outcome. Common RL tasks include real-time decision-making and robot navigation.

Figure – 1

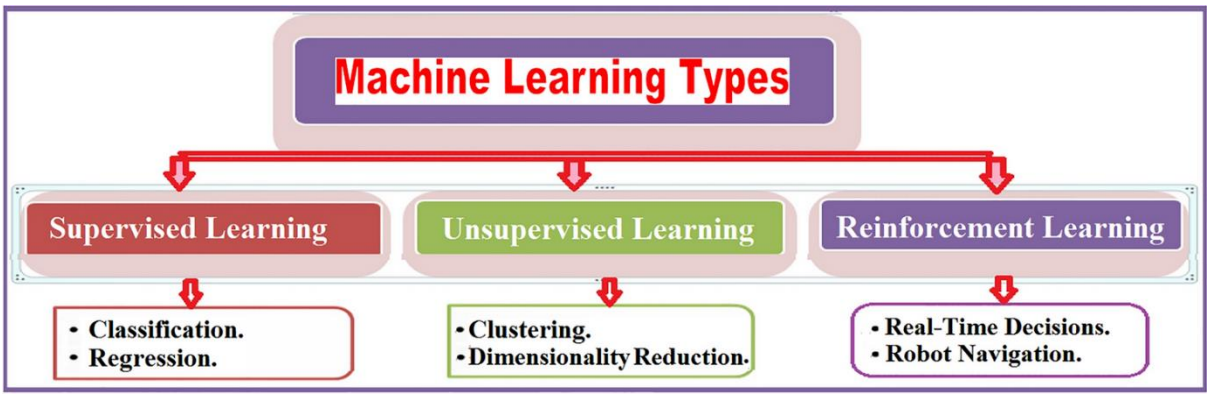
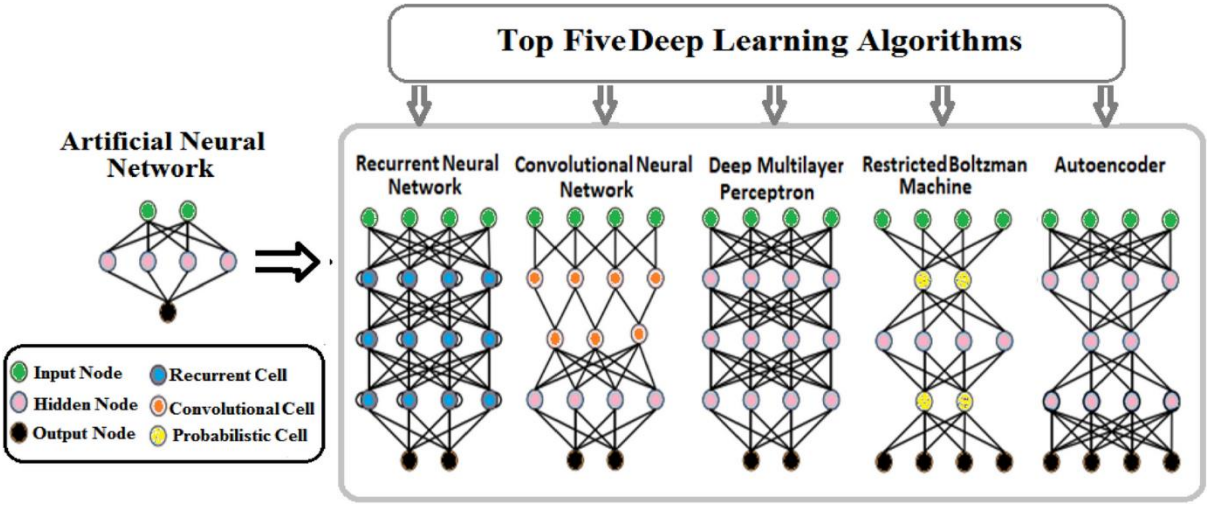


Figure - 2



7.3 Methods

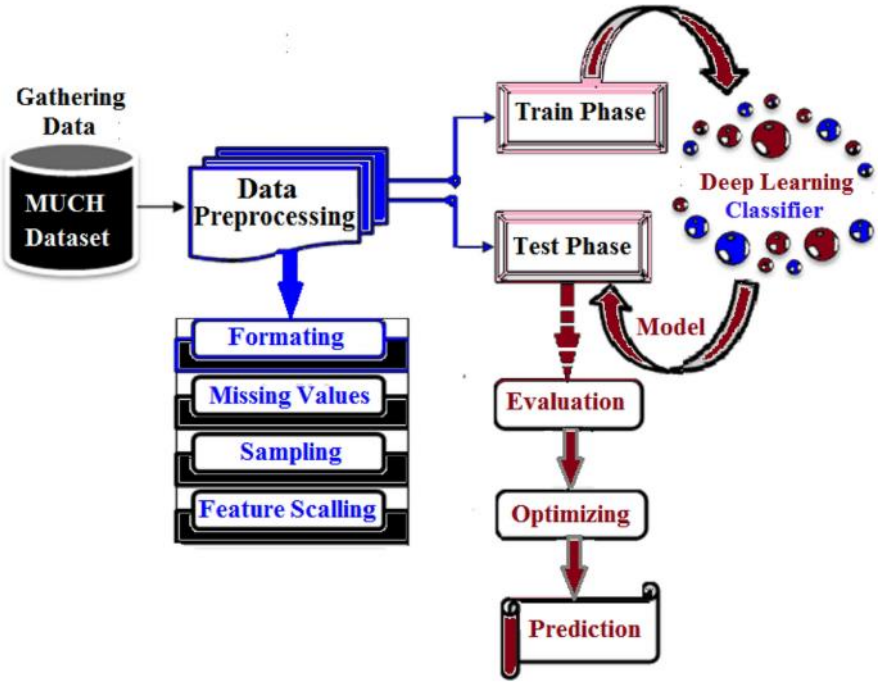
Disease classification assigns patients' information to binary categories based on their medical attributes. This was accomplished by building a model using a train-and-test cycle. It is important to note that this technique is considered a supervised classification technique. A set of predefined labelled attributes is provided as a training set to aid in the classification of new, unknown datasets. However, the size of the dataset can present a significant challenge owing to the high dimensionality of irrelevant attributes, resulting in poor classification performance. This problem can be handled through a preprocessing step by building an aperture classification algorithm designed using statistics and control theory to analyze and retrieve knowledge from experience.

Following the proposed model, it was partitioned into three distinct layers. The input layer is responsible for receiving seventeen raw input data features from the domain without any prior computation. The input features are then subjected to a DNN architecture that guarantees ten hidden layers. The input features are assigned initialized weight values ranging from (0 to 1). All neurons in the hidden layers are attached to both the previous and next layers. At the hidden layers, each node provides an abstraction to the NN with all feature computations. The result is then transferred to the output layer, which assimilates the information learned through the hidden layer and provides the binary classification output. Parameters such as weights and biases, are initially randomized and subsequently adjusted to optimize the prediction model.

The proposed model will yield greater efficiency in the field of medical disease classification field. The model has been presented utilizing the DNN technique, with fine-tuned hyperparameters that assist in building a robust model for the classification of diabetes disease, based on the medical records of the MUCHD dataset.

The model under consideration comprises a sequence of phases, as presented in Figure 3. The system includes a data collection phase, a data preprocessing and classification phase, a train/test phase, an evaluation phase, an optimizing phase, and a prediction phase. The data is sourced from the MUCHD dataset. During the preprocessing phase, it is of utmost importance to enhance the features of missing values and remove insignificant features to avoid misleading results. Subsequently, the features are scaled to a normalized scale. The normalized preprocessed features are then fed to the classifier. The model learns from the trained labeled data and tests its performance using the unlabeled test data. Following The validation phase, the classifier performance is evaluated, followed by the optimizing phase, and ultimately culminating in the prediction phase.

Figure - 3



Six Phases of the Proposed System

1. Data Collection Phase

The quality and quantity of data are crucial factors that significantly impact the effectiveness of the model. The model's performance depends heavily on selecting appropriate and reliable data. Careful consideration is given to selecting relevant features and determining the total number of samples. Assumptions are made about which data features are most relevant to diabetes. In essence, good data leads to better performance and a more successful model.

2. Dataset Availability

The proposed model utilizes the MUCHD dataset, which comprises pediatric patients aged 1 to 19 years, including both male and female individuals. This dataset is sourced from the Mansoura University Children's Hospital (MUCH) repository system, affiliated with the Faculty of Medicine in the Dakahlia Governorate, Egypt. The data is collected from patient medical records, including clinical examinations, laboratory tests, and admission notes.

The MUCHD dataset includes 548 records, classified into two distinct categories: diabetic and non-diabetic (healthy) patients. These records are associated with 18 attributes, which provide detailed medical information. The dataset includes features such as Age, Sex, Duration, Cholesterol, Creatinine, Acetone, Glycated Hemoglobin (HbA1c), Insulin level, Post-Prandial C-Peptide (PCPeptide), Fasting C-Peptide (FCPeptide), 2-hour Post-Prandial Blood Glucose (PBGlucose), Fasting Blood Glucose (FBGlucose), Random Blood Glucose (RBGlucose), Blood Acidosis (pH), Bicarbonate (HCO3), Sodium (Na), Potassium (K), and an output target feature named Diagnosis.

The 'Diagnosis' attribute is a binary output feature that indicates the presence or absence of diabetes. A value of 1 represents diabetic patients (397 cases), while a value of 0 represents non-diabetic patients (151 cases). Table 1 provides a comprehensive description of the 18 attributes. The 'Diagnosis' attribute is considered the dependent output variable, while the remaining 17 attributes serve as independent input features.

Table – 1

The Input Attributes of the MUCHD Dataset.

Attribute	Description
Age	Age (Years)
Sex	Gender
Duration	Period time of diabetes symptoms per week
Cholesterol	Cholesterol
Creatinine	Creatinine
Acetone	Acetone
HbA1c	Haemoglobin(A1c)
Insulin	Insulin Level
PCPeptide	Post Prandial C-Peptide
FCPeptide	Fast C-Peptide
PBGlucose	Post-Prandial Blood Glucose
FBGlucose	Fast Blood Glucose
RBGlucose	Random Blood Glucose
PH	Blood Acidosis of Blood Gases
HCO3	Bicarbonate of Blood Gases
Na	The Sodium of Blood Gases
K	Potassium of Blood Gases
Diagnosis	Diagnosis of Diabetes is 1 for a positive test and 0 for a negative test

3. Data pre-processing and classification phase

This step is performed using DNN. This phase is used to ensure that the input data is presented in a clear and organized format. The primary advantage of feature extraction is its ability to identify the most effective features for the model classifier to learn the representation16. certain errors may arise due to human mistakes during the data collection phase, resulting in labelling errors.

Formatting

The dataset may not be in the correct format such as a database or CSV file. To address this, we meticulously prepared the MUCHD dataset in CSV file format.

Missing *values*

Dealing with noisy missing values poses a significant challenge when gathering data for DL techniques that extremely land with a perfect dataset, which will probably take a significant amount of time. Missing data samples often arise from errors in data collection as a blank space for diagnostic features that are not applicable. Missing values are typically denoted as Nan or null indicators. It is necessary to delete redundant rows and columns. Consequently, two approaches are recommended to address this problem. The first involves eliminating the samples of the missing values, but it is risky to delete relevant information. The second method is to impute the missing values by replacing them with the mean value for each input feature. In Figure 4, the missing values appear as white blank.

Figure 4

Age	Sex	Duration	Cholesterol	Creatinine	Acetone	HbA1c	Insulin	PCPeptide	FCPeptide	PBGlucose	FBGlucose	RBGlucose	PH	HCO3	Na	K
7	1	72	197	0.4	1	13.6	0.6	1.2		350	211	446	7.365	15.1	157	
13	2	192	195	0.6		7.5	0.7		0.2	500	350		7.439	23.8	133.3	4
3	2	2		0.5	3	7.7	0.6	0.774	0.326	587	544	600	7.443	4.6	138.9	4.37
2	2	10	190			4.6	0.8	0.407		525	400	500		24.3	136.6	4.4
8	1	1	110	0.3	4	8.5	1.3	0.37	0.96	490	480	498	7.438	22.5	132.5	
8	2	3		0.7	2	6.7	1.1	1.01	0.294	412	300	498	7.39	24.3	129.6	5.56
7	2	96	200	0.5		6.8	0.7		1.33	498	350	266	7.34	15.2	131.7	4.08
12	2	16	216	0.6	3	5.9	0.6	2.44	1.11	466	390	390	7.408	23.7	149.2	4.69
5	2	2	162	0.9	2	7.1	1	0.578	0.434	500	400	480	7.332	27.9	141.5	4.13
14	1		171			8	0.8	0.376	0.9	326	233	500	7.316	24.3	131.1	3.47
5	1	2		0.8	2	4.5		0.238		322	302	350	7.184	12.7	128	4.81
12	2	8	210	1.1	1	7	0.7	0.01		433	300	450	7.297	19.2	138.7	5.02
5	2	120	200	0.4	1	5.6	1.2	1.82	0.565		400		7.427	17.7	147.9	2.24
3	1	4	145	0.5	4	8	0.7	2.59		525	350	351	7.4	25.7	137.3	3.98
13	1	1.5		0.4		7.3	0.8	0.2	0.1	500		525	7.325	27.7	139.9	4.27
12	1	1	184	0.4	1	7.5	0.7	0.21	0.03	454	362	290	7.349	26.1	139.9	4.29

Table 3 provides a comprehensive overview of the most significant relevant features namely HbA1c, Insulin, PCPeptide, FCPeptide, PBGlucose, FBGlucose, and RBGlucose. The number of missing values for the most significant features is calculated.

Table 3

The Number of Missing Values of the MUCHD Dataset.

Attributes	Missing values
Age	0
Sex	0
Duration	0
Cholesterol	0
Creatinine	0
Acetone	0
HbA1c	194
Insulin	76
PCPeptide	156
FCPeptide	145
PBGlucose	66
FBGlucose	78
RBGlucose	61
PH	0
HCO3	0
Na	0
K	0
Diagnosis	0

Sampling

Sometimes you may have too much data than you require. This can result in increased computational and memory requirements. We consider an appropriate number of samples, which will speed up the processing steps involved in exploring and prototyping the solution. The size of the data samples is determined according to the requirements for faster convergence and reduction of the disk space.

Feature *scaling*

In the preprocessing phase, it is crucial to take a specific step. The majority of DNN algorithms perform significantly better when dealing with features that are on the same scale21. The features are implemented to reduce uncertainty, incorrect results, and cost/processing time. One effective method for achieving this is feature scaling, which involves forming the smallest value of any feature to 0.0 and the largest value to 1.0. There are two common feature scaling techniques. The first is normalization, which rescales features to a range between 0.0 and 1.0. The second technique is standardization, which involves centering the field at a mean of 0.0 and a standard deviation of 1.0. The columns feature has the same parameters as a standard normal distribution, that is, zero mean and unit variance. This makes it much easier for the learning algorithms to learn the weights of the parameters as well as valuable information on the outliers, thereby rendering the algorithms less susceptible to their influence. Generally, when we input data into DNN algorithms, it is customary to manipulate the input data in such a way that the values are adjusted to a balanced scale. The missing values are substituted with the corresponding mean value after normalization. Normalization of data is a crucial step to ensure that the model can be generalized appropriately.

4 & 5 Train/Test phase

The dataset is split into three subsets a training set, a validation set, and a test set, which learns only from the training data, while the validation set is employed for development by fine-tuning the hyperparameters. Finally, the test set to evaluate its performance serves as the ultimate benchmark, allowing us to assess the model's effectiveness in real-world scenarios. We divide the MUCHD dataset into smaller batches with 70% of the data allocated for training and validation purposes, and 30% reserved for testing data. We then feed these beaches into the DNN technique.

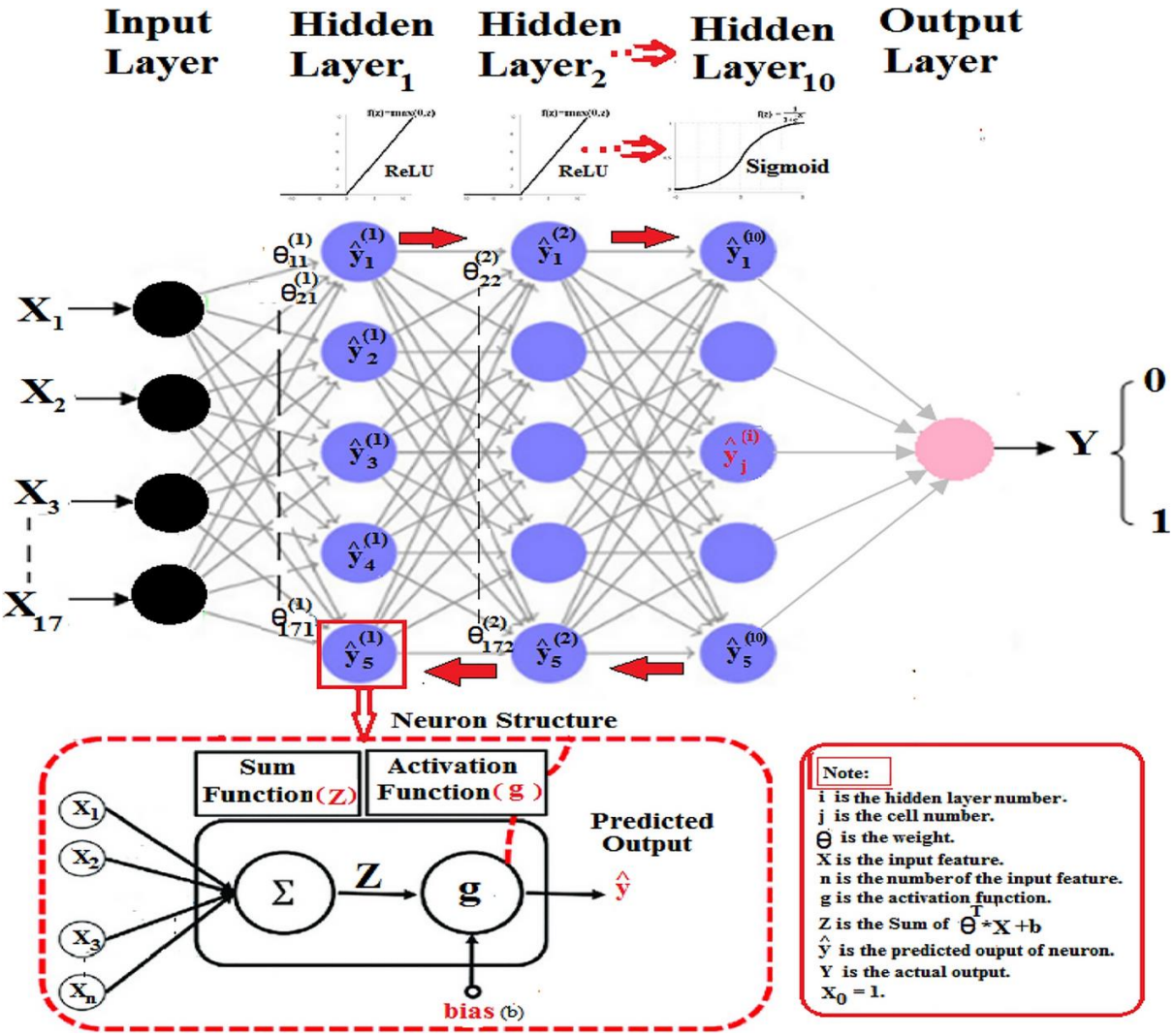
Train algorithm

The first input layer takes the seventeen input features from the MUCHD dataset. The combination of the inputs, weights, and bias are supplied to the activation function as ReLU and Sigmoid with ten hidden layers which are passed further to the next layer. The ReLU function is efficient in computation and scale-invariant. The Sigmoid function is used to fix the data into the range [0,1] for implementation. The DNN algorithm uses a backpropagation technique for classification. The DNN must return through its layers, update the weights to improve itself and minimize the cost function back to the input. The hyperparameters are then fine-tuned. The assigned weight and bias values are updated to reduce the loss function. The Loss function measures how far the predicted output is from the actual output, which determines whether to decrease or increase the weights and bias. Compute the gradient descent which adjusts the parameters by moving to a flat region. the gradient/derivative of the cost function determines whether the weights and bias decrease or increase compared to the optimal weight value. This process is repeated until the proposed system approaches the perfect predicted output as shown in Algorithm 1. MLP is a feedforward neural networks that have various layers of perceptron. In MLP each linear combination is propagated to the next layer based on the result of their computation. the perceptron can only provide outputs in the form of 0 and 1 which will be effective in the binary classification of diabetes. The predicted output is compared with the desired output for the given input.

The construction of a DNN

The Construction of DNN Algorithms and architecture is explained in Figure 5

Figure 5



INTRODUCTION TO PYTHON FOR DATA ANALYSIS

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.
For a description of standard objects and modules, see The Python Standard Library. The Python Language Reference gives a more formal definition of the language. To write extensions in C or C++, read Extending and Embedding the Python Interpreter and Python/C API Reference Manual. There are also several books covering Python in depth.
This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in The Python Standard Library.

8.2 Introduction to MACHINE LEARNING USING PYTHON

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data. In this article, we'll see basics of Machine Learning, and implementation of a simple machine-learning algorithm using python.
Machine learning is a method of teaching computers to learn from data, without being explicitly programmed. Python is a popular programming language for machine learning because it has a large number of powerful libraries and frameworks that make it easy to implement machine learning algorithms.
Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

To get a little overview here are a few popular plotting libraries:

MATPLOTLIB:

It is a multi-platform data visualization library built on NumPy arrays, and designed to work with the broader SciPy stack. It was conceived by John Hunter in 2002, originally as a patch to I Python for enabling interactive MATLAB-style plotting via gnu plot from the I Python command line.

PLOTLY:

Plotlib is the Python Library for interactive data visualizations. Plt allows you to plot superior interactive graphs than either Matplotlib or Seaborn.

Need for the MACHINE LEARNING

The field of artificial intelligence known as machine learning is concerned with the process by which computers attempt to forecast future events using historical data and the information they already possess. There are two distinct forms of machine learning. The first kind of learning is called supervised learning, and in this type of learning, the data itself serve as the instructor, and the system is constructed based on the dataset. The second kind of learning is called unsupervised learning, and it involves the data teaching itself by identifying certain patterns within the dataset and then categorising those patterns. Over the last several years, a large number of writers have reported and discussed their research on diabetes prediction by utilising machine learning algorithms.

MACHINE LEARNING ALGORITHM

The research that has been done on machine learning has resulted in the development of multiple data mining methods. These algorithms may be directly applied to a dataset in order to create some predictions or to derive important inferences and conclusions from such a dataset. Immediate use of these algorithms is possible. Decision tree,NaivE Bayes, k-means, neural network, and other similar algorithms are examples of prominent data mining techniques. In the part that comes after this one, we will talk about them.

The **Naive Bayes (NB)** method is a straightforward approach to the construction of classifiers. It is a Bayesian probabilistic classifier that uses Bayes' theorem as its foundation. Given the class variable, each and every Naive Bayes classifier operates on the assumption that the value of a single feature does not rely in any way on the value of any additional feature. The following statement illustrates Bayes theorem: Here X is the data tuple and C is the class, P(C|X) is calculated by multiplying P(X|C) by P(C)/P(X). This ensures that P(X) remains the same for all classes. Despite the fact that it operates on the assumption that feature values are critically independent, which is not a realistic assumption, it performs very well on huge datasets when this requirement is assumed and is true.

A **decision tree** is a kind of decision support system that utilises a tree-like structure or representation of actions and their potential repercussions, which may include the results of chance events as well as utility considerations. It is one of the methods in which an algorithm may be shown. In the field of operational research, decision trees are often used, particularly in the process of decision analysis, to assist with the identification of a strategy that would most likely result in the achievement of the objective. In the field of machine learning, it is also a widely used tool. The process of mapping first from tree's parent node to each of its leaf nodes individually may quickly and efficiently convert a decision tree to a collection of rules. When everything is said and done, reasonable judgments may 1 3 be obtained by adhering to these criteria.

Support Vector Machine (SVM) - an abbreviation for It is a kind of learning called supervised learning, and it divides data into two categories using a hyper plane as the dividing line. The Support Vector Machine (SVM) accomplishes the same goal as C4.5, with the exception that it does not make any use of Decision Trees. In order to reduce the likelihood of an incorrect classification being made, the support vector machine makes an effort to increase the margin, which is the distance across the hyper plane and the 2 data points that are closest to it from each class. Scikit-learn, MATLAB, and LIBSVM are examples of well-known software packages that may be used to create support vector machines.

A **Network Made of Artificial Neurons (ANN)** A computer model that replicates the architecture and functionality of biological neurons is referred to as an artificial neural network, or ANN for short. Since a neural network adapts or trains in some manner depending on the outputs and inputs, for that specific stage and subsequently for each phase, data that travels thru the network has an effect on the architecture of the ANN. ANNs are recognised as nonlinear statistical data modelling software because of their ability to simulate the intricate interactions that exist across inputs and outputs or to identify trends. ANNs are built using layers that are linked to one another. The artificial neural networks that are being used to improve current data analytics tools are very straightforward mathematical models.

8.3 INTRODUCTION TO PYTHON FOR DATA ANALYTICS

Python is a high level, dynamic programming language. Python3.4 version was used as it is a mature, versatile and robust programming language. It is an interpreted language which makes the testing and debugging extremely quickly as there is no compilation step. There are extensive open-source libraries available for this version of python and a large community of users. Python is simple yet powerful, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data, i.e. spoken English using NLTK. Other high level programming languages such as R and Matplotlib were considered because they have many benefits such as ease of use but they do not offer the same flexibility and freedom that Python can Deliver.

Features in Python

There are many features in Python, some of which are discussed below

- Easy to code.
- Free and Open Source.
- Object-Oriented Language.
- GUI Programming Support.
- High-Level Language.
- Extensible feature.
- Python is Portable language.
- Python is Integrated language.

8.4 ANACONDA

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, [12] as a graphical alternative to the Command Line Interface (CLI).

The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google TensorFlow, can find that it stops working having used pip to install a different package that requires a different version of the dependent NumPy library than the one used by TensorFlow. In some cases, the package may appear to work but produce different results in detail.

Open-source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

8.4 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop Graphical User Interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda 1 6

Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.
The following applications are available by default in Navigator:
- JupyterLab.
- Jupyter Notebook.
- QtConsole.
- Spyder.
- Glue.
- Orange.
- RStudio.
- Visual Studio Code.

8.5 JUPYTER NOTEBOOK

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupiter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupiter web application, Jupiter Python web server, or Jupiter document format depending on context. Usually ending with the ".ipynb" extension. Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release[11][12] (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.
The Notebook interface was added to IPython in the 0.12 release[14] (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0).

Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data. Prior to Pandas, Python was majorly used for data munging and preparation. It had very less contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data—load, prepare, manipulate, model and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

• Fast and efficient Data Frame object with default and customized indexing.
• Tools for loading data into in-memory data objects from different file formats.
• Data alignment and integrated handling of missing data.
• Reshaping and pivoting of date sets.
• Label-based slicing, indexing and sub-setting of large data sets.
• Columns from a data structure can be deleted or inserted.
• Group by data for aggregation and transformations.
• High performance merging and joining of data.
• Time Series functionality.

Matplotlib is the most popular python plotting library. It is a low-level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

Key features of Matplotlib

- Semantic way to generate complex, subplot grids.
- Settings the aspect ratio of the axes box.
- Colored labels in legends.
- Ticks and labels.
- RcParams can be passed as Decorators.
- 3D plots now support minor ticks.

Plotly allows users to import, copy and paste, or stream data to be analyzed and visualized. For analysis and styling graphs, Plotly offers a Python sandbox (NumPy supported), datagrid, and GUI. Python scripts can be saved, shared, and collaboratively edited in Plotly.

Key features of Plotly

- Charts.
- Dashboards.
- File Export.
- App Manager.
- Kubernetes Authentication.
- Jobs Queue.
- Snapshot Engine.

LIBRARIES USED

To create a detailed diabetes prediction report using Pandas, NumPy, Matplotlib, Sklearn, TensorFlow, and Keras, we would follow these steps. I'll break down the roles of each library and how they contribute to the process:

- Pandas
- NumPy
- Matplotlib
- Sklearn
- Tensorflow
- Keras

1. Pandas: Data Handling

Pandas is essential for loading, cleaning, and preparing the dataset. The steps include:

- **Loading the dataset**: Use pd.read_csv() to load the diabetes dataset (e.g., Pima Indians Diabetes Dataset or a similar one).
- **Exploratory Data Analysis** (EDA): Get insights like missing values, mean, median, standard deviation using.describe(), .info(), and checking for correlations using .corr().
- **Data cleaning**: Handle missing data, outliers, and ensure data types are correct for processing.

Example:

```
import pandas as pd

# Load dataset
df = pd.read_csv('diabetes.csv')

# Check for missing values
df.info()

# Summary statistics
df.describe()
```

2. NumPy: Data Manipulation

NumPy is used for efficient numerical operations, particularly for handling arrays and matrices. It's often used behind the scenes in Pandas and TensorFlow for matrix manipulations.

- **Conversion**: Convert DataFrame to NumPy arrays for modeling purposes.

- **Scaling**: Perform operations like normalization, standardization to bring all feature values to the same scale.

Example:

```
import numpy as np

# Convert DataFrame to NumPy array

data = df.to_numpy()

# Separate features and labels

X = data[:, :-1]  # Features
y = data[:, -1]   # Labels
```

3. Matplotlib: Data Visualization

Matplotlib is used for visualizing the data to understand patterns and relationships between features and target variable (e.g., diabetes or not).

- **Visualize distributions**: Plot histograms, boxplots, scatter plots to explore the dataset visually.
- **Correlation heatmaps**: Plot heatmaps to see feature correlations.

Example:

```
import matplotlib.pyplot as plt

# Histogram of one feature (e.g., Glucose levels)

plt.hist(df['Glucose'], bins=30, color='blue')

plt.title('Distribution of Glucose Levels')

plt.xlabel('Glucose')

plt.ylabel('Frequency')

plt.show()
```

4. Sklearn: Machine Learning

Sklearn is crucial for splitting the dataset, pre-processing, and creating machine learning models.

- **Data splitting**: Use train_test_split to split the dataset into training and testing sets.
- **Preprocessing**: Standardize the data using StandardScaler().
- **Modeling**: Create models such as logistic regression, decision trees, or support vector machines for baseline predictions.
- **Evaluation**: Evaluate model performance using metrics like accuracy, precision, recall, and confusion matrix.

Example:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and train a model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# Predictions and evaluation
y_pred = model.predict(X_test_scaled)
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
```

5. TensorFlow and Keras: Deep Learning

TensorFlow and Keras are used to build deep learning models, particularly for complex tasks like diabetes prediction using neural networks.

- **Model creation**: Use Keras API to build a neural network with layers like Dense, Dropout, etc.
- **Compilation**: Compile the model with an optimizer (e.g., Adam), loss function (e.g., binary crossentropy), and metrics (e.g., accuracy).
- **Training**: Train the model using model.fit() on the training data.
- **Evaluation**: Evaluate the model on the test set using model.evaluate().

Example:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Build a neural network
model = Sequential([
    Dense(64, input_dim=X_train_scaled.shape[1], activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')  # For binary classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Train the model
model.fit(X_train_scaled, y_train, epochs=50, batch_size=32,
validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(X_test_scaled, y_test)
print(f'Test accuracy: {accuracy}')
```

Methodologies

Random Forest Classifier:

We used the widely used machine learning method known as the Random Forest Classifier. Using a randomly chosen portion of the training data, this technique generates a collection of decision trees. The final class of the test object is subsequently decided by adding the votes from several decision trees. The total number of trees to be produced as well as factors relevant to decision trees, such as minimum split and split criteria, are included in the hyperparameters for the Random Forests Classifier.

Using the K-Nearest Neighbor Classifier:

The A popular approach for classification issues is K-Nearest Neighbor Classifier. Using the majority of votes received from its neighbours, this method determines the class of a data point. The algorithm uses metrics like Euclidean distance, Hamming distance, cosine distance, etc. to determine the separation between the points. Based on the votes obtained, the classifier assigns a label to the new data point that needs to be predicted.

Gradient Boosting Algorithm:

Machine learning methods for classification and regression issues include gradient boosting. a collection of unreliable prediction models, frequently decision trees, is produced as the prediction model. The model is constructed stage-by-stage using the method, which generalises it by enabling the optimisation of any differentiable loss function.

Deep Neural Network with 1 Hidden Layer:

We developed a neural network model that includes 3 dense layers, with the first being the input layer where we defined an input shape of 8 and specified an activation function. We used the RELU activation function with 12 neurons. 28

Deep Neural Network with 5 Hidden Layers:

We also developed a neural network model that includes 3 hidden dense layers. The input layer has an input shape of 8, and we specified the action of activating RELU. We ran the model for various numbers of epochs and learning rates and utilised the sigmoid activation function in the output layer.

Deep Neural Network with More Hidden Layers:

In addition, we developed a neural network model with 7 hidden dense layers. The input layer has an input shape of 8, and we specified an activation function. We used the RELU sigmoid activation function in the output layer and activation function in the intermediate layers. We tested the model with different epoch counts and learning rates.

CONV1D Layer:

The Conv1D layer in neural networks is a one-dimensional convolutional layer that is used for processing and analyzing data with a sequential or time-series structure. It is typically used in deep learning models for natural language processing, speech recognition, and audio processing, where a series of numbers make up the input data, such as audio signals, time-series data, or sequences of words.
The Conv1D layer applies a set of filters to the input sequence, where each filter slides along the sequence to extract local features. The filter is applied to a window of neighboring values at a time and produces a single output value by performing a between the filter and the input window, use the dot product operation. A collection of feature maps that reflect the existence of particular features at various locations in the input sequence make up the output sequence that is produced.
The Conv1D layer has several hyperparameters that can be tuned, including the filter window's dimensions, the number of filters, the stride, and the padding. The complexity of the model rises as the number of filters is increased, enabling it to capture more intricate aspects in the input sequence. Increasing the size of the filter window increases the receptive field of the layer and allows it to capture longer-range dependencies in the input sequence. The stride determines how much the filter moves along the sequence at each step, and the padding can be used to control the output size of the layer. 29

In order to inject non-linearity into the model and give it the ability to learn more complicated representations of the input sequence, the output of the Conv1D layer is often routed through a non-linear activation function, such as ReLU. The result can then be sent to further layers, such as pooling layers or dense layers, to further process and analyze the features extracted by the Conv1D layer.

Max pooling 1D Layer:

The MaxPooling1D layer in neural networks is a one-dimensional The output of the Conv1D layer is downsampled using a pooling layer. It is typically used in deep learning models for processing sequential or time-series data, where the input sequence is too long or too complex to be processed directly by the subsequent layers.

The MaxPooling1D layer operates on a set of feature maps produced by the Conv1D layer, where each feature map represents the presence of a specific feature at different positions in the input sequence. The layer applies a sliding window over each feature map and computes the maximum value within each window. The resulting output is a downsampled feature map that contains the most significant features in the input sequence.

The MaxPooling1D layer has several hyperparameters that can be tuned, including the stride and the size of the pooling window. While a smaller pooling window maintains more specific information about the input sequence, an increase in the pooling window decreases the dimensionality of the output and raises the level of abstraction in the features. The stride determines how much the pooling window moves along the sequence at each step.

The MaxPooling1D layer reduces the complexity of the input sequence and aids in the extraction of the most important features, while also reducing the risk of overfitting and improving the computational efficiency of the model. The output of the MaxPooling1D layer can be passed to other layers, such as dense layers, for further processing and analysis, or can be used as the final output of the model.

SYSTEM DESIGN OR ARCHITECTURE

11.1 Structure of the System:

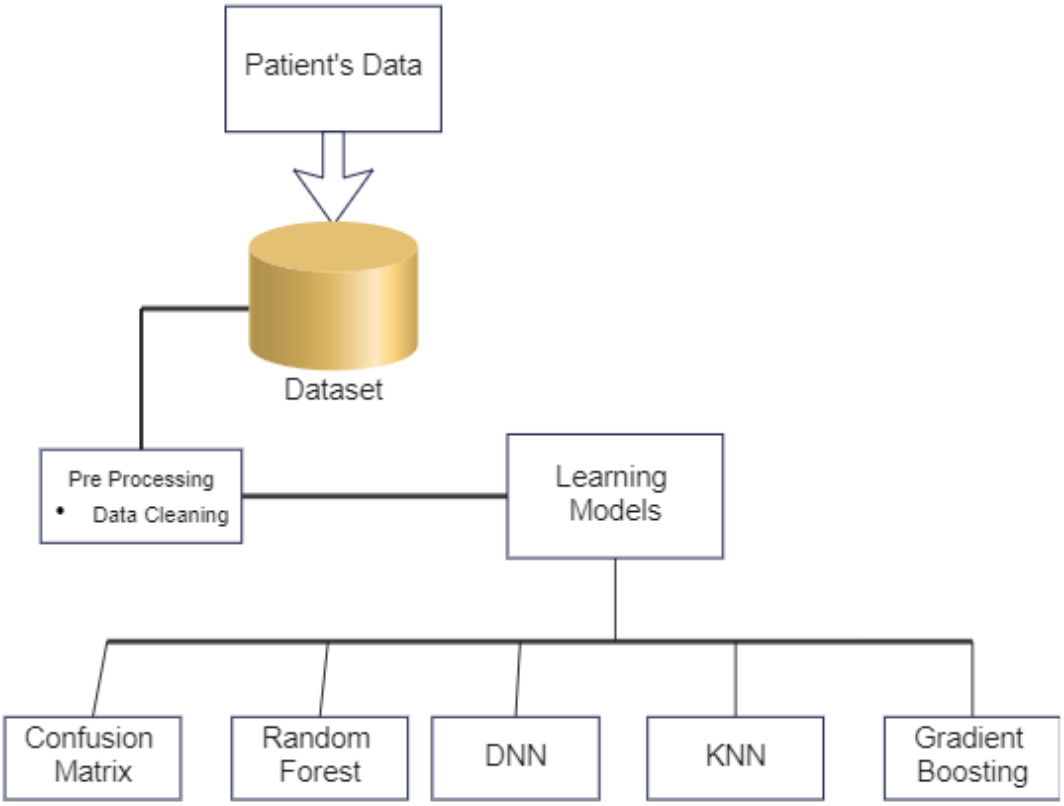


Fig 4.1 Structure of the System

A dataset, which is a structured format appropriate for data analysis, is created once the patient data has been gathered. The dataset then undergoes data cleaning and preprocessing, where missing values, outliers, and other data quality issues are addressed.

Once the data is cleaned and preprocessed, it is ready for analysis using various machine learning models. Confusion matrix, random forest, DNN, KNN, and Gradient Boosting are some of the commonly used models. These models are applied to the preprocessed data to identify patterns, make predictions, and gain insights. The diagram represents a simplified flow of the data analysis process, where patient data is transformed into valuable insights using various machine learning techniques.

11.2 Case Study Diagram

A use case diagram is a particular type of behavioural diagram that originates from and is described by a use-case study in the Unified Modeling Language (UML). Its goal is to provide a graphical representation of the functioning of a system in terms of actors, their goals (represented as use cases), and any connections between those use cases. The main goal of a use case diagram is to show which actor performs which system functions. Roles can serve as representations of the system's actors.

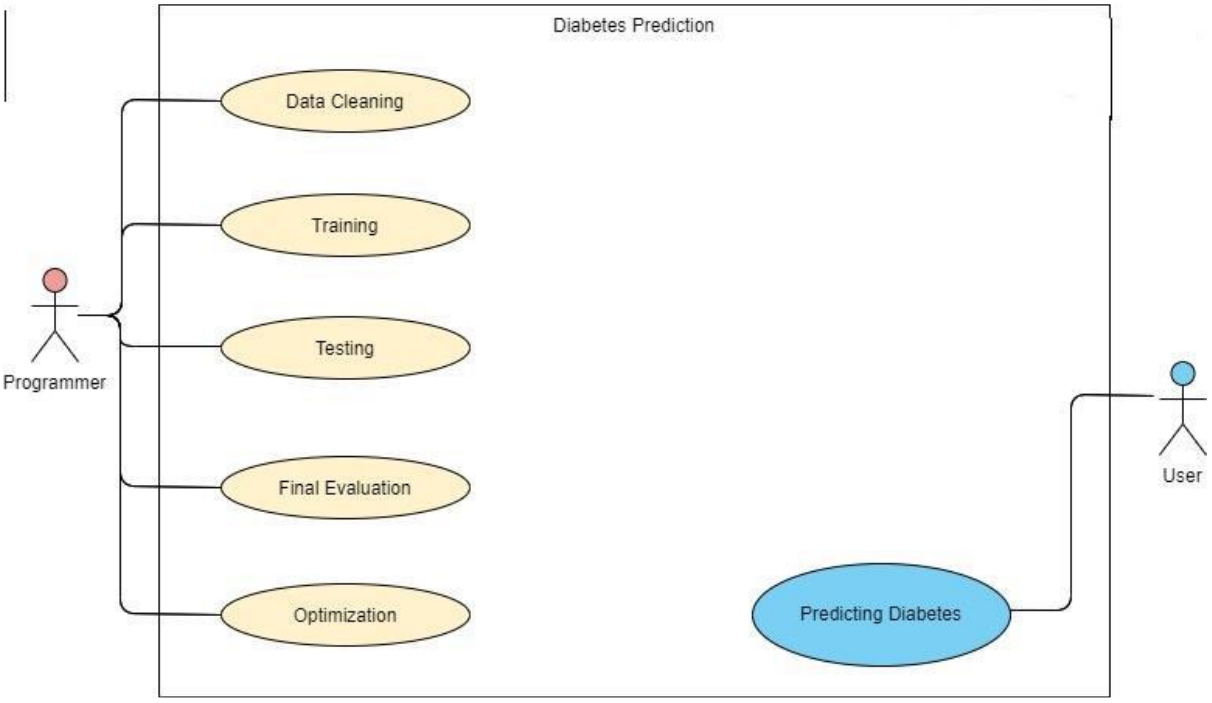


Fig 4.2. Case Study Diagram

11.3 Flowchart Illustration

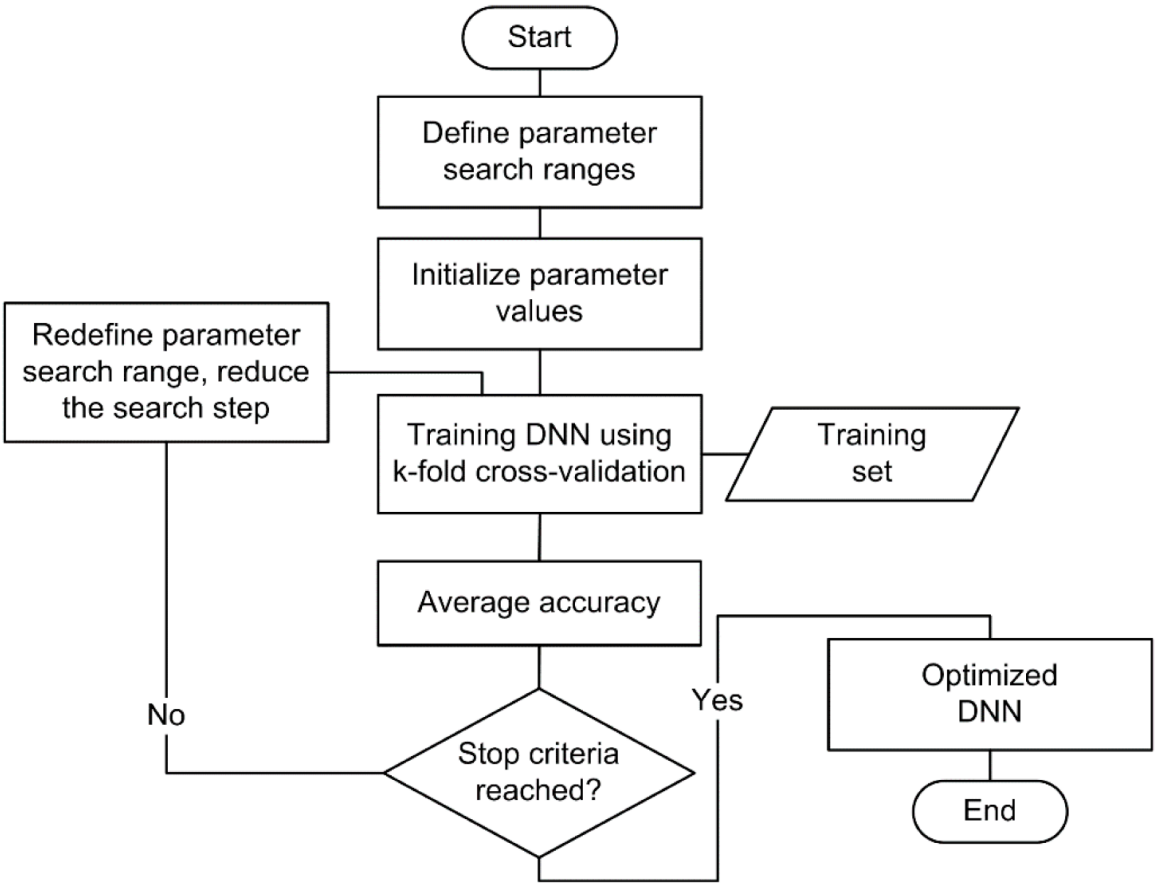


Fig 4.3. Flow Chart for Diabetes Prediction Using Neural Networks

• Here Flow chart it's show how to predict diabetes using neural networks.

• From start to define parameters and initialize that values and training

• After Training it will predict average accuracy if person not having diabetes, it's say's no and if person having means it's say's yes and showing accuracy.

IMPLEMENTATION

12.1 Random Forest:

Random Forests, also referred to as Random Decision Forests, are an ensemble method that is widely used for predicting outcomes.

The Random Forest Classifier selects a portion of the training set at random to generate a collection of decision trees. The final class of the test item is then determined by combining the findings of each separate decision tree.

The hyperparameters of the Random Forests Classifier contain the total number of trees to be produced as well as factors specific to decision trees, including the minimum split and split criteria.

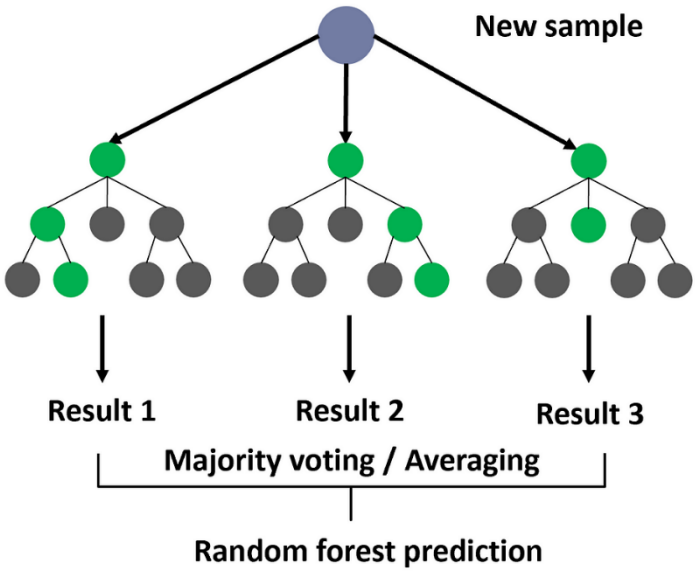
Random forest, KNN, gradient boost, decision tree, and deep learning models are a few of the well-liked machine learning algorithms employed to forecast diabetes. These algorithms use a combination of statistical and mathematical techniques to analyze large datasets and make accurate predictions.

An ensemble learning system called random forest builds several decision trees and then combines the results to provide a final forecast. It is a strong and adaptable algorithm that can manage a variety of input variables and generate precise outputs.

Explaination:

This code initializes 300 decision trees make up a random forest classifier model, which assigns it to the variable rf_model. Then, the fit method is called on the rf_model object with X_train and y_train as its input parameters. This fits to discover the patterns and connections between them, the model is trained using the training data X train (input features) and Y train (target variable). After After trained, the model may be used to forecast the desired variable for new input data using the predict method.

Sample Diagram

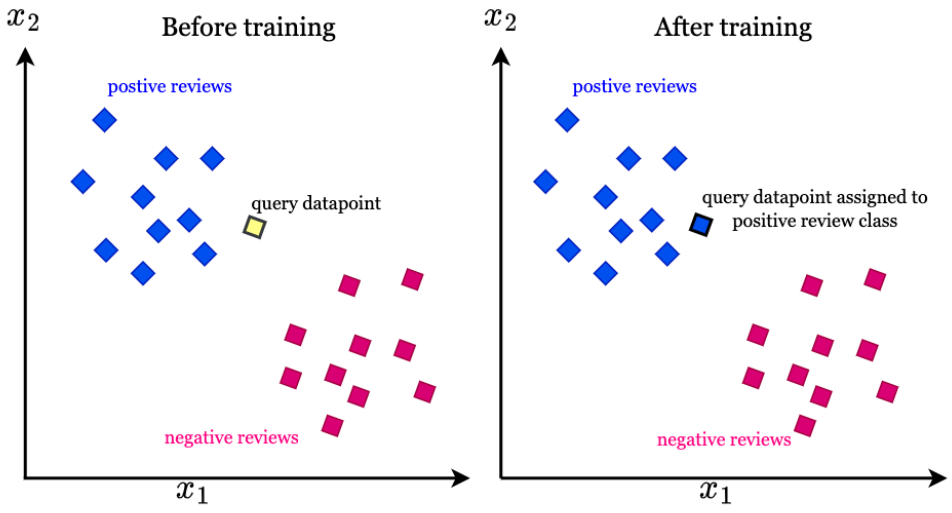


12.2 K-Nearest neighbour:

An effective approach for classification issues is the Classifier for K-Nearest Neighbors.

Nearest K Neighbor method evaluates distances such as Euclidean distance, hamming distance, cosine distance, etc. and predicts the class of the data point according to the majority of votes collected from the nearby places. Based on the votes, the new data point that has to be anticipated is assigned a label.

KNN, or A non-parametric approach called k-nearest neighbours makes predictions about a sample's class based on the classes of its k closest neighbours. It is a straightforward algorithm that works well can handle both binary and multi-class classification problems.



Explaination:

Here, we first load the diabetes dataset, which contains several features (such as glucose level, BMI, etc.) and a target variable indicating whether the patient has diabetes or not. The train test split tool from Scikit-Learn was then used to divide the dataset into training and testing sets.

With n neighbors set to 3, we design a k-NN classifier and use the fit technique to fit it to the training set of data. The predict method is then used to create predictions on the testing data, and the scikit-learn accuracy score function is used to assess the classifier's accuracy.

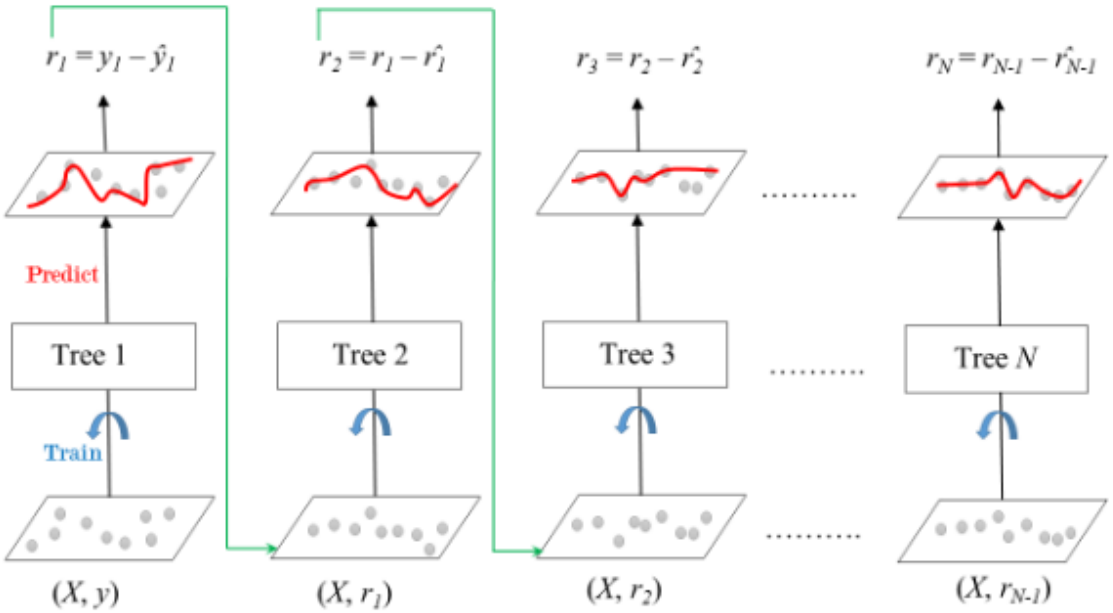
There are other methods to enhance the performance of the k-NN classifier; keep in mind that this is only a simple example of code, such as optimizing the value of k, using feature selection techniques, etc.

12.3. Classifier using Gradient Boosting:

Gradient boosting is a machine learning technique that builds a prediction model for regression and classification problems out of a collection of weak prediction models, frequently decision trees.

It builds the model in steps, just like earlier boosting approaches, but it generalises them by allowing the optimisation of any differentiable loss function.

Gradient boost is another ensemble learning algorithm that creates a strong predictor by combining a number of weak learners. It is a popular algorithm for regression and classification problems and is known for its high accuracy and robustness.



Explanation:

We create a Gradient Boosting Classifier with default hyperparameters, and use the fit technique to fit it to the training data. The predict method is then used to create predictions on the testing data, and the scikit-learn accuracy score function is used to assess the classifier's accuracy.

. Initially, a dictionary of parameters is used to define the Gradient Boosting Classifier's hyperparameters. The hyperparameters specified include: max_depth which sets the maximum depth of the decision trees, subsample which defines the number of samples that will be utilised for each tree, learning_rate which controls the step size at each iteration, both random state and min samples leaf provide the minimal number of samples that must be present at a leaf node, which sets the seed value for random number generation.

Next, a Gradient Boosting Classifier is created with n_estimators set to 290 and the lexicon of the params dictionary's hyperparameters. Following that, the fit technique enables the classifier to be fitted to the training set of data.

The predict method is used to make predictions on the testing data, and the accuracy score function is used to determine how accurate the classifier from scikit-learn.

Finally, the train predict function is called to On both the training and test sets of data, assess the classifier's performance. With the classifier, training, and testing data, this function outputs the training and testing accuracy scores, as well as the training time and prediction time. The results are then printed to the console.

Overall, this code demonstrates how to use the Gradient Boosting Classifier algorithm to predict diabetes, as well as how to assess the classifier's performance using various metrics.

The model's F1 score on the training set is shown in the second line of the output for the training set, which reads: 1.0000. The accuracy and recall of the classifier are both considered when calculating the F1 score. For the training set, an F1 score of 1.0 denotes flawless recall and accuracy.

The model's F1 score on the testing set is shown in the third line of the output, which reads: 0.6753. An F1 score of 0.68 indicates that the model performed well on the testing set in terms of recall and precision, but not as well as it did on the training set.

In summary, the model achieved a decent accuracy Despite having a strong F1 score on the training set and a low F1 score on the testing set, there appears to be some overfitting in the testing set. This suggests that the model might not translate well to novel, unforeseen facts.

12.4. Deep Neural Networks:

Deep learning models, such as artificial neural networks, are a powerful class of algorithms that can learn complex patterns and relationships in data. They are particularly useful for image and speech recognition tasks but can also be applied to numerical data for classification and regression problems.

To evaluate the effectiveness of these algorithms may be measured using a variety of measures, including accuracy, precision, recall, F1-score, and the confusion matrix. The number of true positives, true negatives, false positives, and false negatives in a classification issue are displayed in a table called the confusion matrix. It is a helpful tool for assessing a model's performance and pinpointing its weak points.
A computer model based on the biological neural network, is known as an artificial neural network (ANN), which is made up of different neurons. ANN seeks to imitate the network of neurons so that it may programmatically learn patterns and make better judgements.
The input layer, output layer, and hidden layers make up an ANN. They are given weights, which are then updated in accordance with the backward propagation procedure.

Key Concepts and Terminologies for Understanding the Neural Network Model:

Activation Function:

In neural network models, activation functions are crucial because they introduce non-linearity. The main goal is to change an ANN node's input signal from a signal's path from an input to an output. This output signal is now used as an input by the following rung of the stack.
To create the output layer of an ANN, we sum up all of the inputs and their corresponding weights, apply the activation function f(x), and then feed the result as an input to the subsequent layer. Various activation mechanisms take on various shapes. I'll just talk about the project's most often utilised activation routines.

Role of the sigmoid Activator:
Formula f(x)=1/1+exp describes it as a function of activation (-x). Range includes 0 and 1. useful for tasks involving binary categorization.
Tanh Activation function:
It may be calculated using the formula f(x)=1-exp(-2x)/1+exp (-2x) (-2x). The output is zero-centered due to its range being between -1 and 1. In this situation, optimisation is simpler and typically chosen than sigmoid function.
RELU Activation Function:
It is sometimes referred to as a linear rectified unit in common usage. If the input value is less than zero, it returns the value 0.0 instead of the value given as input explicitly.
G(z) = max(0,z) describes the function.

Explaination:

This code is used to normalize the feature values of the use the StandardScaler function from scikit-learn for both the training and test sets.
If normalizer = StandardScaler, a StandardScaler object is first created (). By scaling the supplied data, the StandardScaler function creates features with a mean of 0 and a standard deviation of 1, for each feature. Then, the fit transform method is used on the training data (X train) to normalise its values. The scaling transformation is applied to the feature values, and the mean and standard deviation of each feature in the training set are calculated.
The transform method is used to apply the same scaling transformation that was used on the training data to normalise the values of the testing data (X test).
Finally, the normalized training and testing sets are printed to the console using the print function.

Overall, normalizing the feature values can improve the performance of some machine learning algorithms, such as those based on distance metrics. This is because normalizing the values can help to reduce the influence of features with large ranges and ensure that all features are on a similar scale.

First, the StandardScaler function is used to normalize the feature values of the training set (X_train) using the fit_transform method. The result is an array of normalized feature values, which are printed to the console as X_train_normalized. Similarly, the transform method is used to normalize the feature values of the testing set (X_test). The result is an array of normalized feature values, which are printed to the console as X_test_normalized.

The output shows that the feature values in the training and testing sets have been normalized to 0 and 1 respectively for the mean and standard deviation. The array's rows represent instances (such as patients), even if its columns are featured (medical measurement). The normalized feature values are represented as floating-point numbers.

Overall, normalization is an important preprocessing step to ensure that all features are on a similar scale and can help improve the performance of some machine learning algorithms.

6.4.1. Developing Neural Networks With 1 Hidden Layer:

Here, we have defined a model having 3 dense layers.
First is the input layer wherein we have defined an input shape which is 8 and also defined activation function. I have used RELU activation function and as 12 neurons.
Second layer consists of a hidden 8 neurons are present in layer, and RELU activity is present.
The output layer, which makes up the third layer, has just one neuron and uses the Sigmoid activation function since the output might be between zero and one. Sigmoid functions are useful for binary classification problems. We have used Sequential class from Keras library wherein you can build a model, define the layers, input and output.
There are 4 major steps involved in model building:

- ➢ Establish a model
- ➢ Compile the model in
- ➢ Adjust the model to the data.
- ➢ Assume the result.

Explaination:

We begin by importing the required modules from the Keras library.
Then, We use the Sequential class to define the neural network's design. The relu activation function and a dense layer with 10 hidden units are added. The number of features in our dataset (i.e., X train.shape[1]) is the input dimension for this layer. Lastly, we incorporate a single-unit output layer with the sigmoid activation function.
The model is then assembled using the accuracy metric, the loss function and the Adam optimizer binary crossentropy.
Using the fit approach, we train the model on the training set. We define the batch size (i.e., the no. of samples) and the number of epochs (i.e., the number of times the complete training set is processed). processed before updating the model), and the verbosity level (i.e., how much information is displayed during training).
Lastly, we use the evaluate method to assess the model's performance on the testing set. We keep track of accuracy and loss values in the loss and accuracy variables, respectively. We print out the accuracy to the console.

Understanding The Model Compile Method:

The model compile () method is an important step in building a neural network as it specifies the optimizer and loss function to use during training. In this project, Stochastic Gradient Descent (SGD) 38

is the optimizer that we have chosen since it aids in minimising the loss function by finding a gradient at each iteration and updating the weights in the model using the backward propagation algorithm. Other options for optimizers include Adam and RMSProp.

For the loss function, we have used as this is a binary classification problem, binary cross-entropy was applied. Each vector component class is independent for binary cross-entropy loss, which computes the loss for each output vector component without taking other component values into account. To calculate the loss for binary classification tasks, it combines a sigmoid function with a cross-entropy loss.

Learning rate is another hyperparameter that determines how much we modify our network's weights in relation to the loss gradient. The gradient will be sluggish if the learning rate is too low, and it may overshoot the minimum and fail to converge if the learning rate is too high. To find the optimal value of the learning rate, we need to test and check the model's performance by trying different values. In this project, we have tested the model's performance using different learning rates.

6.4.2. Neural Network with 5 Hidden Layers:

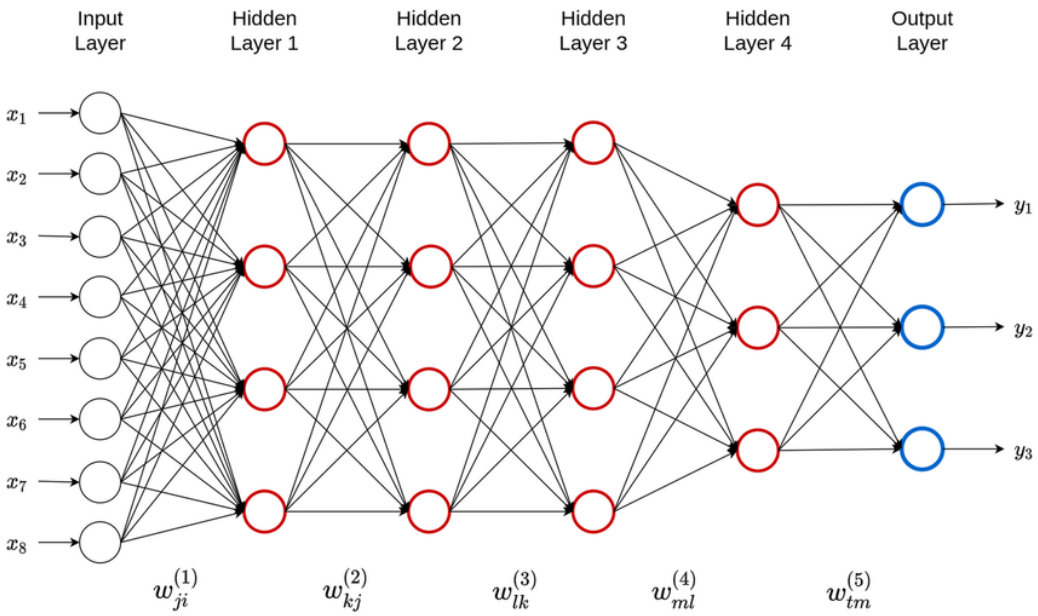
Certainly! Here's an explanation of the five hidden layers of the neural network for diabetes prediction:

Load diabetes dataset using a data loading library such as pandas.

Divide the dataset into the target variables and the features.

Use a technique like train test split from the sklearn.model selection module to divide the dataset into training and testing sets.

Standardize the features using a scaler such as StandardScaler from the sklearn.preprocessing



RESULTS AND SCREENSHOTS

Importing Libraries:

[7]:
```
#importing sklearn and traditional Python machine learning libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_auc_score, roc_curve, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier

# importing keras, tensorflow and related modules
import tensorflow as tf
from tensorflow.keras import layers
import keras
from keras.models import Sequential # intitialize the ANN
from keras.layers import Dense, Dropout      # create layers
from keras.optimizers import Adam, SGD, RMSprop
```

Dataset View:

[13]:
```
#Loading the dataset
diabetes_data = pd.read_csv("diabetes.csv")

#Print the first 5 rows of the dataframe.
diabetes_data.head()
```

[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[4]:
```
diabetes_df.tail()
```

[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.34	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

The dataset has 768 rows it’s showing Head and tail

Basic EDA and statistical analysis

[19]:
```
## gives information about the data types,columns, null value counts, memory usage etc
## function reference : https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.info.html


diabetes_data.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Correlation Matrix:

[9]:
```
# checking whether there are any missing values in the dataset.
diabetes_data.isnull().values.any()
```

[9]: False

##Pandas dataframe.corr() is used to find the pairwise correlation of all columns in the Pandas Dataframe in Python. Any NaN values are automatically excluded. Any non-numeric data type or columns in the Dataframe, it is ignored.

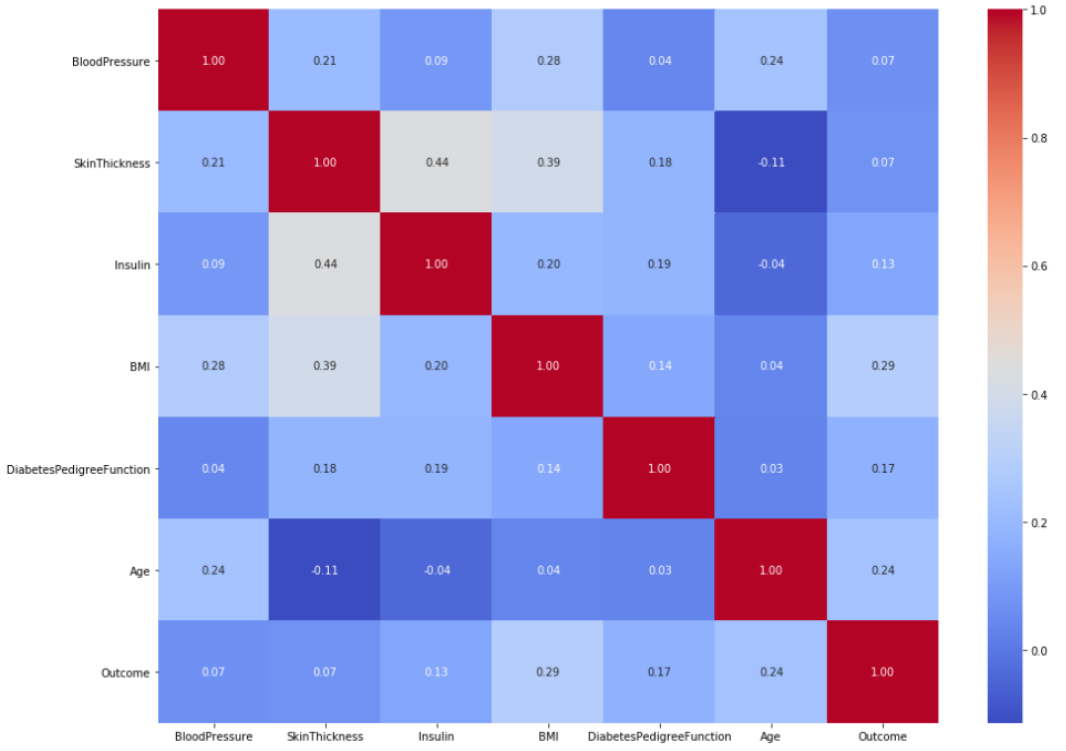
[10]:
```
diabetes_data.corr()
```

[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	

Heatmap of Correlation:

```
In [520]:  plt.figure(figsize=(16,12))
           sns.heatmap(data=diabetes_df.iloc[:,2:].corr(),annot=True,fmt='.2f',cmap='coolwarm')
           plt.show()
```



jupyter Diabetes Last Checkpoint: last year

File Edit View Run Kernel Settings Help

Not Trusted

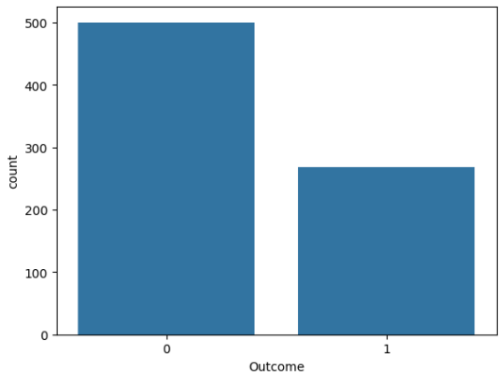
Code

JupyterLab Python 3 (ipykernel)

Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695

[11]:
```
sns.countplot(x = 'Outcome' , data = diabetes_data)
```

[11]:
```
<Axes: xlabel='Outcome', ylabel='count'>
```



jupyter Diabetes Last Checkpoint: last year

File Edit View Run Kernel Settings Help

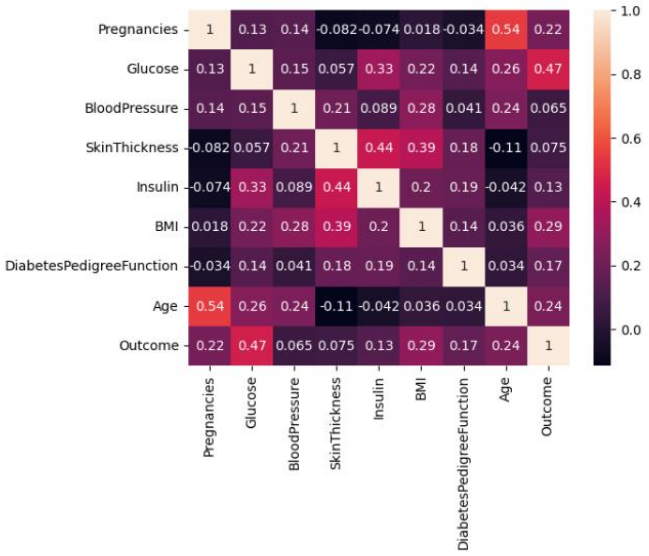
Not Trusted

Code

JupyterLab Python 3 (ipykernel)

The resulting heatmap shows a color-coded matrix where each cell represents the correlation between two features. Positive correlations are shown in red, while negative correlations are shown in blue. The strength of the correlation is indicated by the intensity of the color. The values inside each cell indicate the actual correlation coefficient, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation). A value of 0 indicates no correlation between the two features.

[16]:
```
corr_mat = diabetes_data.corr()
sns.heatmap(corr_mat , annot=True)
plt.show()
```



Random Forest Classifier:

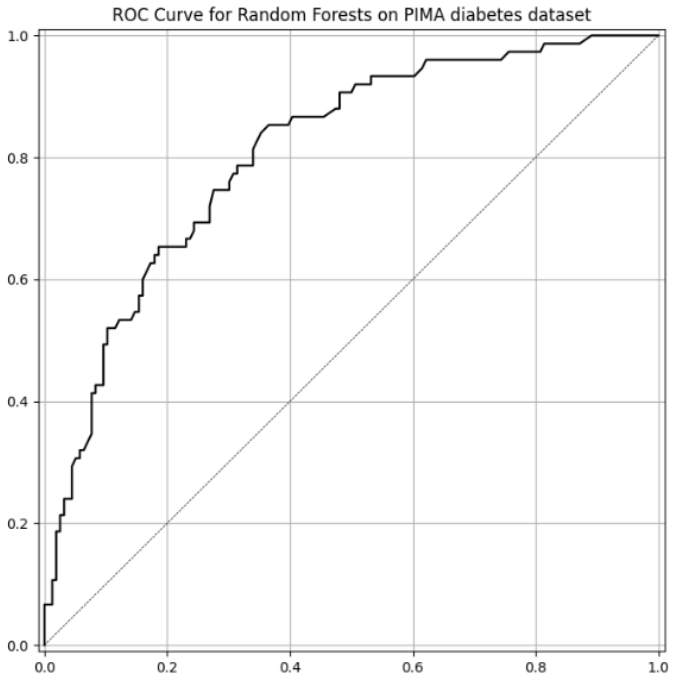
```
[21]: RandomForestClassifier
RandomForestClassifier(n_estimators=300)

[22]: # Make predictions on the test set - both "hard" predictions, and the scores (percent of trees voting yes)
y_pred_class_rf = rf_model.predict(X_test)
y_pred_prob_rf = rf_model.predict_proba(X_test)

print('Accuracy is {:.3f}'.format(accuracy_score(y_test,y_pred_class_rf)))
print('ROC-AUC is {:.3f}'.format(roc_auc_score(y_test,y_pred_prob_rf[:,1])))

Accuracy is 0.749
ROC-AUC is 0.808

[23]: plot_roc(y_test, y_pred_prob_rf[:, 1], 'Random Forests')
```



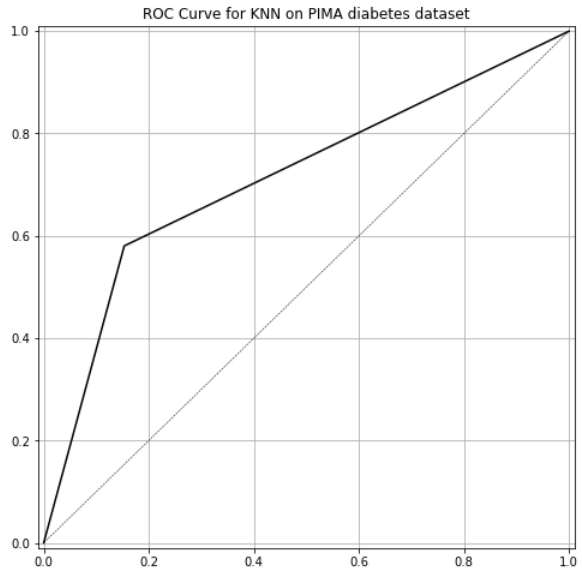
Result: Accuracy obtained is 72.3 % and AUC-ROC value is 0.81 by employing Random Forest Model.

K-Nearest Neighbour Accuracy:

```
# creating the confusion matrix
evaluate_cm = confusion_matrix(y_test, y_pred)
print (evaluate_cm)
print('F1 score is ' ,f1_score(y_test, y_pred))
print('Accuracy is' , accuracy_score(y_test, y_pred))
```

```
[[127  23]
 [ 34  47]]
F1 score is  0.6225165562913907
Accuracy is 0.7532467532467533
```

```
plot_roc(y_test, y_pred, 'KNN')
```



Accuracy score obtained from KNN method = 75.3%

Result:

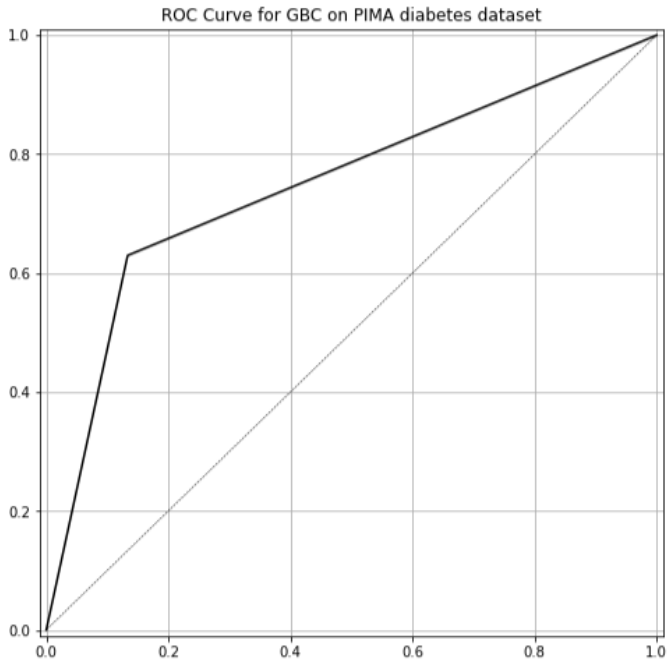
The Accuracy Score Obtained from K-Nearest Neighbour method is 75.3%

Gradient Boost Classifier Accuracy:

```
params = {'max_depth':9, 'subsample':0.5, 'learning_rate':0.01, 'min_samples_leaf':1, 'random_state':42}
gbc = GradientBoostingClassifier(n_estimators=290, **params)
clf_ = gbc.fit(X_train, y_train)
y_pred = clf_.predict(X_test)
print('Accuracy is {}'.format(accuracy_score(y_test,y_pred )))
train_predict(gbc, X_train, y_train, X_test, y_test)
```

```
Accuracy is 0.7835497835497836
F1 score for training set is: 1.0000
F1 score for testing set is: 0.6711
```

```
plot_roc(y_test, y_pred, 'GBC')
```



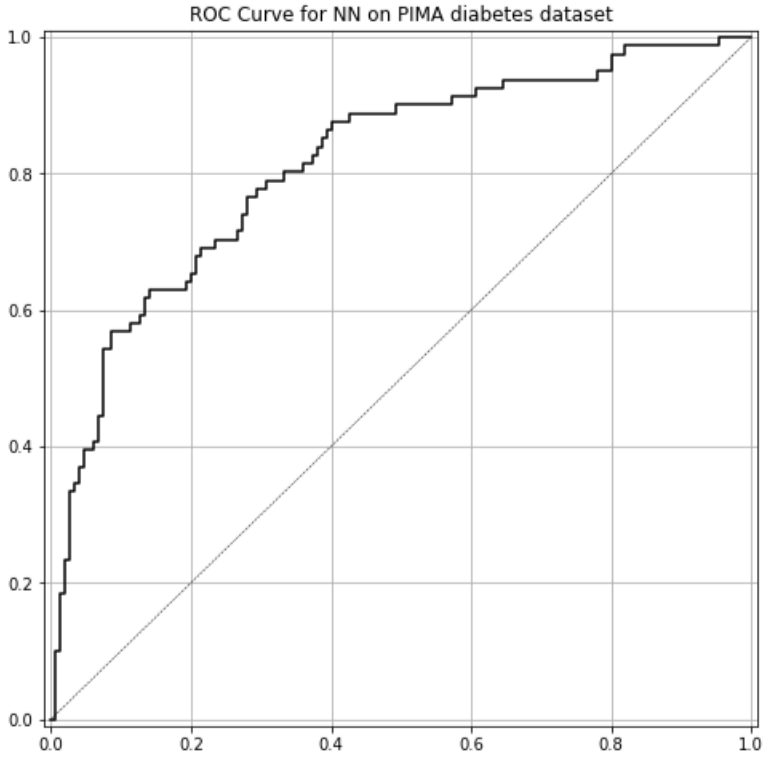
Result: Accuracy Of Gradient Boosting Classifier is 78.3%

Neural networks with 1 hidden layer accuracy:

```
# Print model performance and plot the roc curve
print('Accuracy is {:.3f}'.format(accuracy_score(y_test,y_pred_class_nn_1)))
print('ROC-AUC is {:.3f}'.format(roc_auc_score(y_test,y_pred_prob_nn_1)))

plot_roc(y_test, y_pred_prob_nn_1, 'NN')
```

```
Accuracy is 0.775
ROC-AUC is 0.814
```



Accuracy obtained from this model is slightly better 77.5% and ROC-AUC value is 0.814

Result:

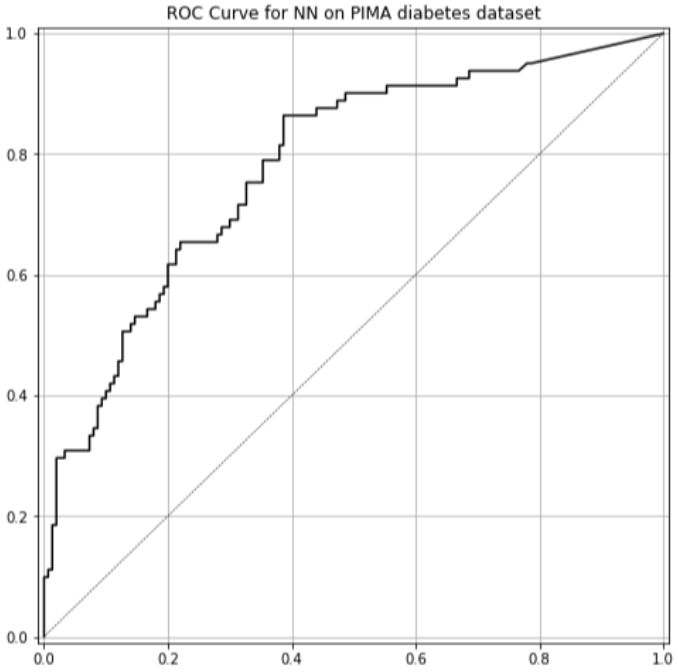
The Accuracy obtained from this model is slightly better 77.5% and ROC-AUC Value is 0.814.

Neural Networks With more hidden layers accuracy:

```
# Print model performance and plot the roc curve
print('accuracy is {:.3f}'.format(accuracy_score(y_test,y_pred_class_nn_mod2)))
print('roc-auc is {:.3f}'.format(roc_auc_score(y_test,y_pred_prob_nn_mod2)))

plot_roc(y_test, y_pred_prob_nn_mod2, 'NN')
```

```
accuracy is 0.723
roc-auc is 0.781
```



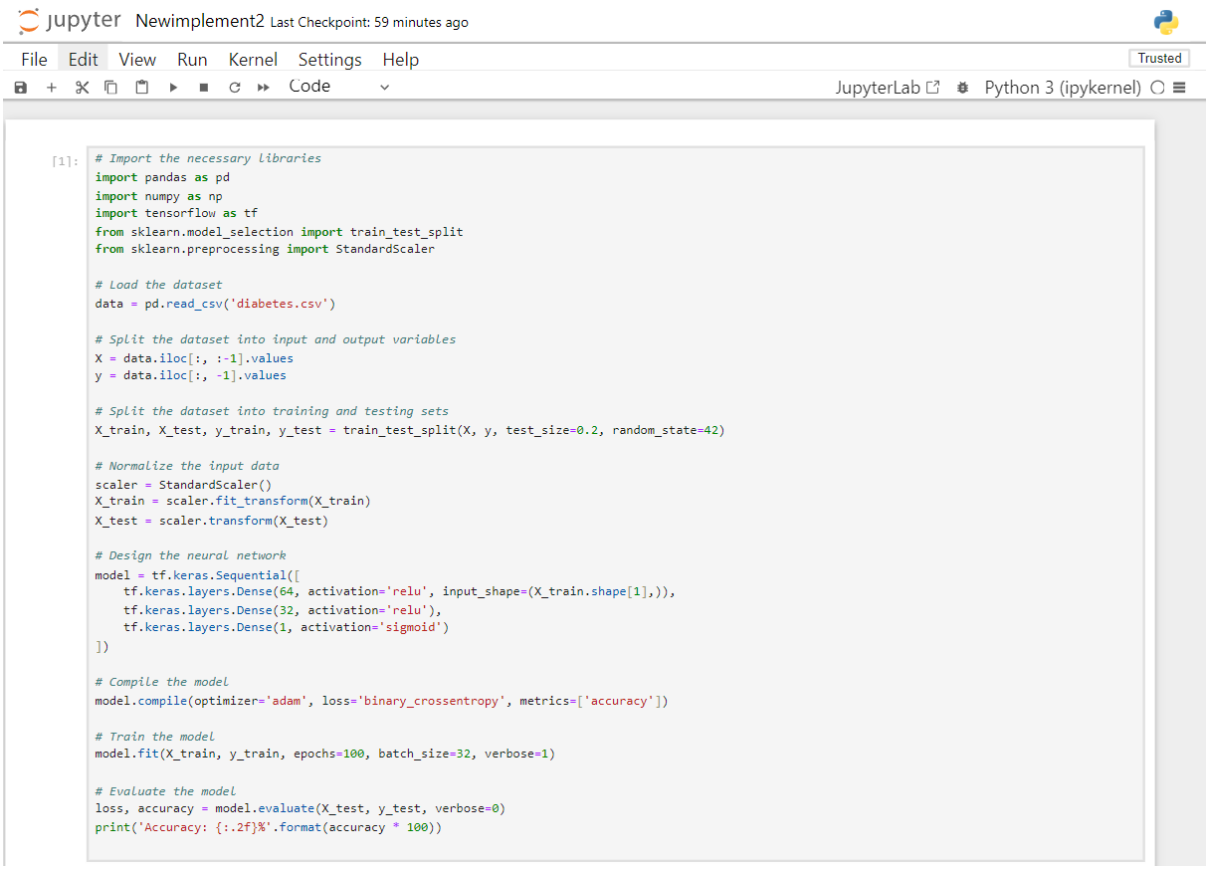
ccuracy obtained from this model is 72.3% for epochs=200 , learning rate=1e-3.

ow let's try for 400 epochs.

Result:

The Accuracy obtained from this model is 72.3% for epochs=200, learning rate = 1e-3

Accuracy of neural networks:




```
Epoch 92/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2616 - accuracy: 0.8925
Epoch 93/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2610 - accuracy: 0.8893
Epoch 94/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2598 - accuracy: 0.8909
Epoch 95/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2571 - accuracy: 0.8958
Epoch 96/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2581 - accuracy: 0.8925
Epoch 97/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2543 - accuracy: 0.9023
Epoch 98/100
20/20 [==============================] - 0s 1ms/step - loss: 0.2522 - accuracy: 0.9072
Epoch 99/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2522 - accuracy: 0.9039
Epoch 100/100
20/20 [==============================] - 0s 2ms/step - loss: 0.2520 - accuracy: 0.8990
Accuracy: 72.73%
```

Result:

72.73% of the neural network's accuracy is achieved.

A person has diabetes or not using neural networks algorithm with the mainly diabetes **8 attributes** that is

number of births, Blood pressure, Glucose levels, and Skin Thickness, BMI, insulin level

Type 2 diabetes family history developing code as below

Age

```
# Split the dataset into input and output variables
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Normalize the input data
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Design the neural network
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=100, batch_size=32, verbose=0)

# Ask user for data and predict diabetes
pregnancies = int(input("Enter number of pregnancies: "))
glucose = float(input("Enter glucose level: "))
blood_pressure = float(input("Enter blood pressure: "))
skin_thickness = float(input("Enter skin thickness: "))
insulin = float(input("Enter insulin level: "))
bmi = float(input("Enter BMI: "))
diabetes_pedigree_function = float(input("Enter diabetes pedigree function: "))
age = int(input("Enter age: "))

# Create a numpy array with the user input data
user_data = np.array([[pregnancies, glucose, blood_pressure, skin_thickness, insulin, bmi, diabetes_pedigree_function, age]])

# Normalize the user input data
user_data = scaler.transform(user_data)

# Make a prediction
prediction = model.predict(user_data)

if prediction[0][0] > 0.5:
    print("The patient is predicted to have diabetes.")
else:
    print("The patient is predicted to not have diabetes.")
```

```
Enter number of pregnancies: 2
Enter glucose level: 197
Enter blood pressure: 70
Enter skin thickness: 45
Enter insulin level: 543
Enter BMI: 30.5
Enter diabetes pedigree function: 0.158
Enter age: 53
1/1 [==============================] - 0s 69ms/step
The patient is predicted to have diabetes.
```

In the above code snippet, the architecture of a neural network with 5 layers for diabetes prediction is shown. The purpose of this neural network is to determine a person's likelihood of having diabetes using 8 different attributes. These attributes include blood pressure (B.P.), skin thickness, body mass index (BMI), glucose level, and others.

Kaggle provided the dataset that was utilised to train the neural network. This dataset was used to create the neural network method, which was then applied to predict whether or not a person has diabetes based on the provided attributes.

During In order to minimise the loss function during training, the neural network modifies its parameters (weights and biases). The neural network's ability to correctly anticipate the output given an input is measured by the loss function. Once the neural network has been trained, it can be used to predict the output (whether a person has diabetes or not) for new inputs (i.e., new values for the 8 attributes).

Overall, the purpose of this neural network is to provide a tool for predicting whether a person is at risk for diabetes based on their health attributes. By using a neural network, we can take advantage of its ability to learn complex patterns in the data and make accurate predictions based on those

```
# Load the dataset
data = pd.read_csv('diabetes.csv')

# Split the dataset into input and output variables
P = data.iloc[:, :-1].values
q = data.iloc[:, -1].values

# Normalize the input data
scaler = StandardScaler()
P = scaler.fit_transform(P)

# Reshape input data for use in Convolutional Layer
P = P.reshape(P.shape[0], P.shape[1], 1)

# Design the neural network
model = tf.keras.Sequential([
    tf.keras.layers.Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(P.shape[1], 1)),
    tf.keras.layers.MaxPooling1D(pool_size=2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(P, q, epochs=100, batch_size=32, verbose=0)

# Ask user for data and predict diabetes
pregnancies = int(input("Enter number of pregnancies: "))
glucose = float(input("Enter glucose level: "))
blood_pressure = float(input("Enter blood pressure: "))
skin_thickness = float(input("Enter skin thickness: "))
insulin = float(input("Enter insulin level: "))
bmi = float(input("Enter BMI: "))
diabetes_pedigree_function = float(input("Enter diabetes pedigree function: "))
age = int(input("Enter age: "))

# Create a numpy array with the user input data
user_data = np.array([[pregnancies, glucose, blood_pressure, skin_thickness, insulin, bmi, diabetes_pedigree_function, age]])

# Normalize the user input data
user_data = scaler.transform(user_data)

# Reshape user input data for use in Convolutional Layer
user_data = user_data.reshape(user_data.shape[0], user_data.shape[1], 1)

# Make a prediction
prediction = model.predict(user_data)

if prediction[0][0] > 0.5:
    print("The patient is predicted to have diabetes.")
else:
    print("The patient is predicted to not have diabetes.")

# Evaluate the model's accuracy on the training data
_, accuracy = model.evaluate(P, q, verbose=0)
print("Model accuracy on training data:", accuracy)
```

```
Enter number of pregnancies: 0
Enter glucose level: 156
Enter blood pressure: 152
Enter skin thickness: 38
Enter insulin level: 548
Enter BMI: 35
Enter diabetes pedigree function: 0.128
Enter age: 29
1/1 [==============================] - 0s 57ms/step
The patient is predicted to have diabetes.
Model accuracy on training data: 0.953125
```

Explanation:

Conv1D layer: On the input data, this layer applies one-dimensional convolution. It incorporates the following variables:

filters: The output space's dimensionality (i.e., the number of output filters in the convolution).
kernel size: The 1D convolution window's size.
activation: The appropriate activation method.
input shape: The input data's shape.

MaxPooling1D layer: This layer down samples the input along the time dimension (i.e., the first dimension) by taking the maximum value over a fixed window. It takes in the following parameter:

pool size: The size of the range that the maximum value should be taken.
The output from the preceding layer is flattened in this layer, which creates a one-dimensional array.

The Conv1D layer and MaxPooling1D layer were added to perform convolutional and pooling operations on the input data. The Flatten layer was added to convert the output of the Conv1D layer into a one-dimensional array, which can then be passed through the Dense layers.

Conclusion

Conclusion, deep learning techniques can be used to predict whether a person is at risk for diabetes based on their health attributes. By training a neural network with a dataset of relevant health attributes, we can develop an algorithm that accurately predicts whether a person has diabetes or not.

The neural network architecture used for diabetes prediction typically involves multiple hidden layers with non-linear activation functions. During training, the neural network adjusts its weights and biases to minimize the loss function and improve its accuracy in predicting the correct output.

In this project, a neural network with 5 hidden layers was developed and trained using a dataset from Kaggle. The model was being able to execute with great accuracy in predicting whether a person has diabetes or not based on their health attributes.

Overall, deep learning techniques offer a promising approach for predicting diabetes and other medical conditions based on patient data. With further development and optimization of neural network models, we can improve the accuracy of these predictions and provide better healthcare outcomes for patients.

I implemented two new layers were added to the code: one Conv1D layer and one MaxPooling1D layer. Additionally, the existing Dense layers were not removed, so the total number of layers in the neural network is now four.

Future Enhancement

The performance of neural network models could previously be enhanced by experimenting with models with various numbers of layers, learning rates, and L1 and L2 regularisation procedures. In order to further analyse and refine these models, they were used on additional healthcare-related datasets. Researchers were able to establish the neural network's ideal configuration by experimenting with different hyperparameters, which increased the neural network's accuracy in determining whether or not a person had diabetes. They were able to create more effective neural network models for datasets linked to healthcare with the use of prior knowledge and data. Additionally, these methods and models will keep developing in the future, resulting in even higher performance and more precise forecasts for datasets connected to healthcare.

REFERENCES

[1] https://www.geeksforgeeks.org/python-how-and-where-to-apply-feature-scaling/

[2] https://www.kaggle.com/adhishthite/pima-dataset-prediction-model-with-keras-80

[3] Harikrishna, N.B. (2019). Confusion matrix, accuracy, accuracy, recall, F1 score. [online]. 2019. Analysis Vidya. Available at: https://www.mdpi.com/2076-3417/11/9/4267.

[4] https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1

[5] Dilip Singh Sisodia and Deepti Sisodia (ICCIDS 2018) "Diabetes Prediction"

[6] Monisha. A, S. Nirmala Santiago and Shalin Christina (2018) "Decision Support System for Diabetes, a Chronic Illness"

[7] S. Gothai Nachiyar, K. Senthamarai Kannan, and S. Selvakumar (2017) "Diabetes Diagnose Prediction Using Classification-Based Data Mining Methods" published in International Journal of Statistics and Systems ISSN 0973-2675, Volume 12, No 2, pp. 183-188

[8] Veena (2015) "Decision Support Systems for Diabetic Mellitus Prediction: A Review" by Veena, Vijayan.V., and Anjali.C. at the World Conference on Communication Technologies (GCCT)

[9] Dong-Ping Rao, Qiug Liu, Yi-Xiang Huang, and Xue-Hui Meng (2016) "Evaluation of Three Data Mining Methods for Predicting Diabetes or Prediabetes on the Basis of Risk Variables" published in the Korean Journal of Medical Science, vol. 31, no. 2, pp. 229-237.