

MACHINE LEARNING TOOLS FOR TYPE-2 DIABETES RISK PREDICTION

A PROJECT REPORT ON **MACHINE LEARNING TOOLS FOR TYPE-2 DIABETES RISK PREDICTION**

Submitted for partial fulfillment of
Master of Computer Applications

2023

By

M. Bhargavi (4511-21-862-027)

Under the guidance of

Mrs. D. SANDHYA RANI.

(Assistant Professor)



Department of Computer Science and Informatics
UNIVERSITY COLLEGE OF ENGINEERING AND TECHNOLOGY,
MAHATMA GANDHI UNIVERSITY
NALGONDA-508001.

MACHINE LEARNING TOOLS FOR TYPE-2 DIABETES RISK PREDICTION

UNIVERSITY COLLEGE OF ENGINEERING AND TECHNOLOGY,
MAHATMA GANDHI UNIVERSITY
NALGONDA



DECLARATION BY THE CANDIDATE

This is **M.Bhargavi**, here by certify that the Project Report entitled

MACHINE LEARNING TOOLS FOR TYPE-2 DIABETES RISK PREDICTION

is submitted in partial fulfillment of **Masters of computer Applications (IVSEM)**.

This is a record of bonafide work carried out by me under the guidance of **Mrs. D. SANDHYA RANI**, Assistant Professor, University college of Engineering and Technology Mahatma Gandhi University, Nalgonda. The results embodied in this Project Report has not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the Award of any other degree or diploma.

M.BHARGAVI
(4511-21-862-027)

UNIVERSITY COLLEGE OF ENGINEERING AND TECHNOLOGY
MAHATMA GANDHI UNIVERSITY
NALGONDA – 508 001.
COMPUTER SCIENCE & INFORMATICS DEPARTMENT



CERTIFICATE FROM PROJECT GUIDE

This is to certify that the project report entitled “**MACHINE LEARNING TOOLS FOR TYPE-2 DIABETES RISK PREDICTION**” is submitted **M.BHARGAVI** partial fulfillment of **Masters of computer Applications (IV SEM)** is a bonafide work carried by them during the academic year 2023.

The results of the investigations enclosed in this report have been verified and found satisfactory.

Mrs. D. SANDHYA RANI.
(Assistant Professor)

UNIVERSITY COLLEGE OF ENGINEERING AND TECHNOLOGY
MAHATMA GANDHI UNIVERSITY
NALGONDA – 508 001.
COMPUTER SCIENCE & INFORMATICS DEPARTMENT



CERTIFICATE FROM HOD

This is to certify that the project work entitled **MACHINE LEARNING TOOLS FOR TYPE-2 DIABETES RISK PREDICTION** being submitted by M.BHARGAVI to University College of Engineering and Technology MGU, in partial fulfillment of the of **Masters of Computer Applications(IV SEM)**, is a record of bonafide work carried by them.

External Examiner

Head of the Department,

Department of CS&I

ACKNOWLEDGEMENT

I express my sincere thanks to **Mrs.Ch.Sudha Rani**, Head of the Dept. of MCA, for giving me opportunity to do project, without his blessings it would not have been possible for me to carry put this treatise work.

I express my deep sense of gratitude to **Mrs. D. SANDHYA RANI**, Department of Computer Science and Informatics Internal guide for his help through provoking discussions, invigorating suggestions extended to me with immense care,zeal throughout my work.

Finally, it is our pleasure to thank all the faculty members of department for their cooperation in completes this project.

M.BHARGAVI
(4511-21-862-027)

MACHINE LEARNING TOOLS FOR TYPE-2 DIABETES RISK PREDICTION

ABSTRACT

This project uses today's 5G technology to monitor the condition of diabetic patients with a low cost. Nowadays many peoples are suffering with diabetic disease due to work stress or unhealthy lifestyles and people will not know about the current health condition till symptoms appear or diagnosed osis through medical check-ups and the condition of the disease will be severe by that time and there is no possible way to get thatintimation prior. Diabetes will be of two type's diabetes-1 and diabetes-2. Diabetes-2 require hospitalization and in diabetes-1 condition, we can monitor patient and alert himor doctors about his current condition.

LIST OF CONTENTS

ABSTRACT

LIST OF FIGURES

CHAPTER 1: INTRODUCTION.....1

CHAPTER 2: LITERATURE SURVEY.....3

CHAPTER 3: EXISTING SYSTEM.....5

3.1 RANDOM FOREST ALGORITHM.....5

3.2 DRAWBACKS OF EXISTING SYSTEM.....8

CHAPTER 4: PROPOSED SYSTEM.....9

4.1 DATASET.....10

4.2 INTRODUCTION TO DECISION TREE ALGORITHM.....11

4.3 SUPPORT VECTOR MACHINE ALGORITHM.....13

4.4 ARTIFICIAL NEURAL NETWORKS.....19

4.5 K-NEAREST NEIGHBOR.....21

4.6 ADVANTAGES OF THE PROPOSED SYSTEM.....25

CHAPTER 5: UML DIAGRAMS.....26

5.1 USE CASE DIAGRAM.....26

5.2 SEQUENCE DIAGRAM.....27

5.3 CLASS DIAGRAM.....29

CHAPTER 6: MACHINE LEARNING.....30

6.1 WHAT IS MACHINE LEARNING30

6.2 CATEGORIES OF MACHINE LEARNING30

6.3 NEED FOR MACHINE LEARNING	31
6.4 CHALLENGES IN MACHINE LEARNING.....	31
6.5 APPLICATIONS OF MACHINE LEARNING.....	32
6.6 ADVANTAGES OF MACHINE LEARNING.....	34
6.7 DISADVANTAGES OF MACHINE LEARNING.....	35
 CHAPTER 7: SOFTWARE ENVIRONMENT	36
7.1 INTRODUCTION TO PYTHON.....	36
7.2 PYTHON DEVELOPMENT STEPS.....	40
7.3 MODULES USED IN PYTHON	41
 CHAPTER 8 : SYSTEM REQUIREMENTS.....	50
8.1 SOFTWARE REQUIREMENTS.....	50
8.2 HARDWARE REQUIREMENTS.....	50
 CHAPTER 9: FUNCTIONAL REQUIREMENTS	51
9.1 OUTPUT DESIGN	51
9.2 INPUT DESIGN	51
9.3 ERROR AVOIDANCE	53
9.4 USER INTERFACE DESIGN.....	53
 CHAPTER 10: SOURCE CODE	56
 CHAPTER 11: RESULTS.....	60
 CHAPTER 12: CONCLUSION & FUTURE SCOPE	66
12.1 CONCLUSION	66
12.2 FUTURE SCOPE	66
12.3 REFERENCES.....	67

LIST OF FIGURES

Figure 3.1: Random forest Algorithm	5
Figure 3.2: Random Forest Classifier Analysis	7
Figure 3.3: Boosting Random Forest Classifier	8
Figure 4.1(a): Block Diagram of Cloud Application	9
Figure 4.1(b): Block Diagram of User Application	10
Figure 4.2.1: Decision tree algorithm	12
Figure 4.2.2: Decision tree Working	13
Figure 4.3.1: Support Vector Machine Hyperplane	14
Figure 4.3.2: Support Vector Machine	15
Figure 4.3.3: SVM Hyperplane working	17
Figure 4.4.1: Artificial Neural Networks	20
Figure 4.4.2: ANN Working	20
Figure 4.5.1: KNN Classifier	21
Figure 4.5.2: KNN Working Structure	22
Figure 4.5.3: KNN Ecludian Distance	23
Figure 4.5.4: KNN Working on New Data Point	24
Figure 5.1: Use Case Diagram	27
Figure 5.2: Sequence Diagram	28
Figure 5.3: Sequence Diagram	z 28

CHAPTER 1

INTRODUCTION

Diabetes is an extremely common chronic disease from which nearly 8.5 percent of the world population suffers; 422 million people worldwide must struggle with diabetes. It is crucial to note that type 2 diabetes mellitus makes up about 90 percent of the cases [1]. More critically, the situation will be worse, as reported in, with more teenagers and youth becoming susceptible to diabetes as well. Because diabetes has a huge impact on global well-being and economy, it is urgent to improve methods for the prevention and treatment of diabetes. Furthermore, various factors can cause the disease, such as improper and unhealthy lifestyle, vulnerable emotion status, along with accumulated stress from society and work. However, the existing diabetes detection system faces the following problems:

- The system is uncomfortable, and real-time data collection is difficult. Furthermore, it lacks continuous monitoring of multidimensional physiological indicators of patients suffering from diabetes.
- The diabetes detection model lacks a data sharing mechanism and personalized analysis of big data from different sources including lifestyle, sports, diet, and so on [2].
- There are no continuous suggestions for the prevention and treatment of diabetes and corresponding supervision strategies.

To solve the above problems, in this article, we first propose a next generation diabetes solution called the 5G-Smart Diabetes system, which integrates novel technologies including fifth generation (5G) mobile networks, machine learning, medical big data, social networking, smart clothing, and so on. Then we present the data sharing mechanism and personalized data analysis model for 5G-Smart Diabetes. Finally, based on the smart clothing, smartphone, and big data healthcare clouds, we build a 5G-Smart Diabetes testbed and give the experiment results.

Furthermore, the “5G” in 5G-Smart Diabetes has a two-fold meaning. On one hand, it refers to the 5G technology that will be adopted as the communication infrastructure to realize high-quality and continuous monitoring of the physiological states of patients with diabetes and to provide treatment services for such patients without restraining their freedom. On the other hand, “5G” refers to the following “5 goals”: cost effectiveness, comfortability, personalization, sustainability, and smartness.

Cost Effectiveness: It is achieved from two aspects. First, 5G-Smart Diabetes keeps users in a healthy lifestyle to prevent users from getting the disease in the early stage. The reduction of

disease risk would lead to decreasing the cost of diabetes treatment. Second, 5G-Smart Diabetes facilitates out-of-hospital treatment, thus reducing the cost compared to on-the-spot treatment, especially long-term hospitalization of the patient.

Comfortability: To achieve comfort for patients, it is required that 5G-Smart Diabetes does not disturb the patients' daily activities as much as possible. Thus, 5G-Smart Diabetes integrates smart clothing [3], mobile phones, and portable blood glucose monitoring devices to easily monitor patients' blood glucose and other physiological indicators.

Personalization: 5G-Smart Diabetes utilizes various machine learning and cognitive computing algorithms to establish personalized diabetes diagnosis for the prevention and treatment of diabetes. Based on the collected blood glucose data and individualized physiological indicators, 5G-Smart Diabetes produces personalized treatment solutions for patients.

Sustainability: By continuously collecting, storing, and analyzing information on personal diabetes, 5G-Smart Diabetes adjusts the treatment strategy in time based on the changes of patients' status. Furthermore, to be sustainable for data-driven diabetes diagnosis and treatment, 5G-Smart Diabetes establishes effective information sharing among patients, relatives, friends, personal health advisors, and doctors.

With the help of social networking, the patient's mood can be better improved so that he or she is more self-motivated to perform a treatment plan in time.

Smartness: With cognitive intelligence toward patients' status and network resources, 5G-Smart Diabetes achieves early detection and prevention of diabetes and provides personalized treatment to patients. The remaining part of the article is organized as follows. We first present the system architecture of 5G-Smart Diabetes. Then we explain the data sharing mechanism and propose the personalized data analysis model. Furthermore, we introduce the 5G-Smart Diabetes testbed.

CHAPTER 2

LITERATURE SURVEY

Chen et al. proposed the 5G-Smart Diabetes system, which combined the state-of-the-art technologies such as wearable 2.0, machine learning, and big data to generate comprehensive sensing and analysis for patients suffering from diabetes. Then this work presented the data sharing mechanism and personalized data analysis model for 5G-Smart Diabetes. Finally, this work builds a 5G-Smart Diabetes testbed that includes smart clothing, smartphone, and big data clouds. The experimental results showed that the system can effectively provide personalized diagnosis and treatment suggestions to patients.

Rghioui et al. presented an intelligent architecture for monitoring diabetic patients by using machine learning algorithms. The architecture elements included smart devices, sensors, and smartphones to collect measurements from the body. The intelligent system collected the data received from the patient and performed data classification using machine learning to make a diagnosis. The proposed prediction system was evaluated by several machine learning algorithms, and the simulation results demonstrated that the sequential minimal optimization (SMO) algorithm gives superior classification accuracy, sensitivity, and precision compared to other algorithms.

Venkatachalam et al. motivated to develop a diabetes monitoring system for patients using IoT device in their body which monitors their blood sugar level, blood pressure, sport activities, diet plan, oxygen level, ECG data. The data are processed using feature selection algorithm called as particle swarm optimization and transmitted to nearest edge node for processing in 5G networks. Secondly, data are processed using DBN Layer. Thirdly, this work shared the diagnosed data output through the wireless communication such as LTE/5G to the patients connected through the edge nodes for further medical assistance. The patient wearable devices are connected to the social network. The Result of this proposed system is evaluated with some existing system. Time and Performance outperform than other techniques.

Prakash et al. introduced a neural network-based ensemble voting classifier to predict accurately the diabetes in the patients via online monitoring. The study consists of Internet of Things (IoT) devices to monitor the instances of the patients. While monitoring, the data are transferred from IoT devices to smartphones and then to the cloud, where the process of classification takes place. The simulation is conducted on the collected samples using the python tool. The results of the simulation show that the proposed method achieves a higher accuracy rate, higher precision, recall, and f-measure than existing state-of-art ensemble models.

Tsoulchas et al. proposed a model to monitor the health of people with diabetes melitus, a disease with high incident rates mainly at the elderly but also in younger people. Specifically, a study about the existing medically approved technologies for continuous measurement of diabetes is described. Subsequently, the model for monitoring patient's blood glucose levels is described. Whenever a patient's blood glucose levels are Low or High, the model triggers an alarm to a Cloud infrastructure in order remote medical staff to provide immediate cure to the patient. Furthermore, to assure the immediate response of the remote medical staff, the proposed model is deployed upon a 5G wireless network architecture.

Huang et al. proposed a 5G-based Artificial Intelligence Diabetes Management architecture (AIDM), which can help physicians and patients to manage both acute complications and chronic complications. The AIDM contains five layers: the sensing layer, the transmission layer, the storage layer, the computing layer, and the application layer. We build a test bed for the transmission and application layers. Specifically, this work applied a delay-aware RA optimization based on a double-queue model to improve access efficiency in smart hospital wards in the transmission layer. In application layer, this work builds a prediction model using a deep forest algorithm.

CHAPTER 3

EXISTING SYSTEM

3.1 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

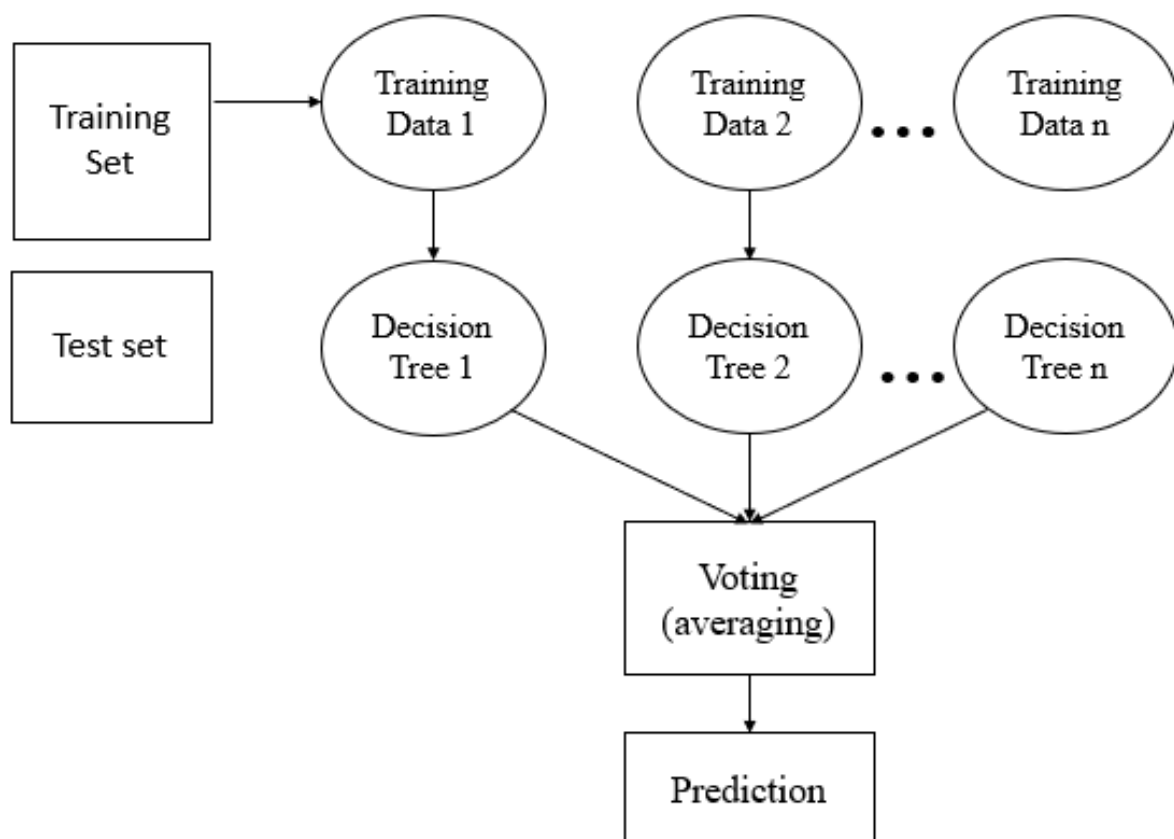


Fig. 3.1: Random Forest algorithm.

Random Forest algorithm

Step 1: In Random Forest n number of random records are taken from the data set having k

number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and Regression respectively.

Important Features of Random Forest

- **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **Immune to the curse of dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
- **Train-Test split**- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability**- Stability arises because the result is based on majority voting/ averaging.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

Types of Ensembles

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as

Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

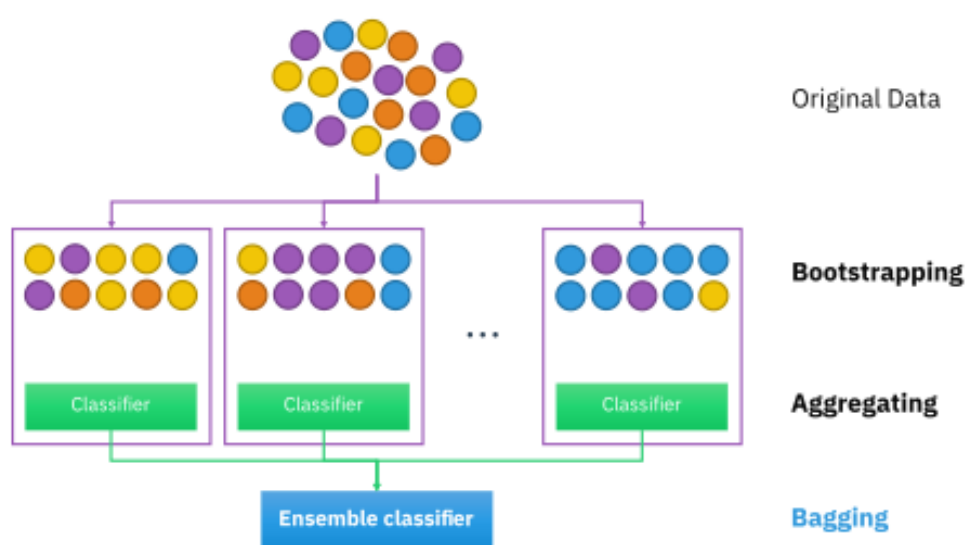


Fig. 3.2: RF Classifier analysis.

Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

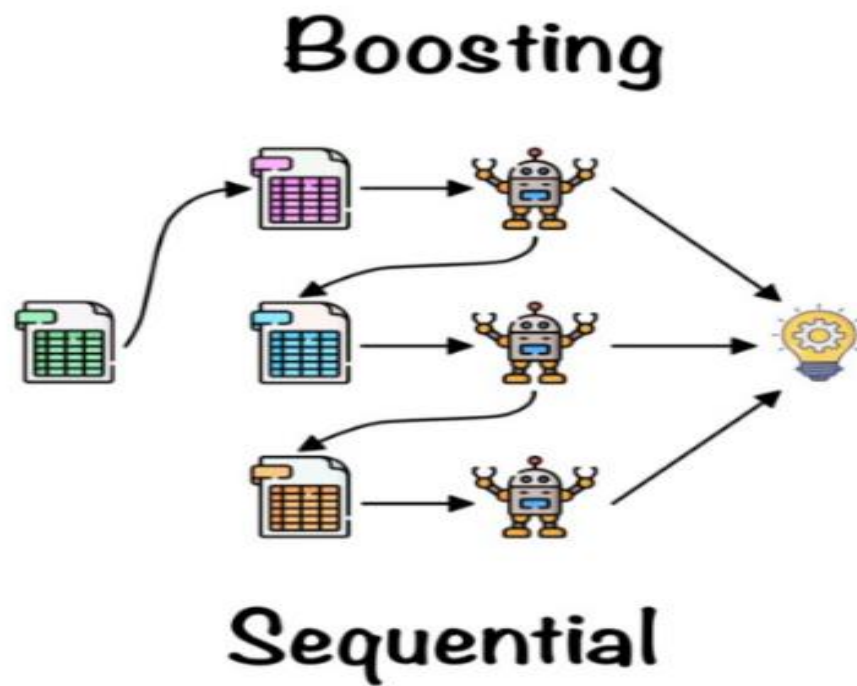


Fig. 3.3: Boosting RF Classifier.

3.2 Drawbacks of Existing system

- Unstable nature. One of the limitations of decision trees is that they are largely unstable compared to other decision predictors. ...
- Less effective in predicting the outcome of a continuous variable.

CHAPTER 4

PROPOSED SYSTEM

In proposed work, we are using Decision Tree, SVM, Artificial Neural Network algorithms from python to predict patient condition from his data. To train these algorithms we are using diabetes dataset. To predict data efficiently author is using Ensemble Algorithm which is combination of Decision Tree, SVM and ANN algorithm. Training model of all these three algorithms will be merging inside Ensemble Algorithm to get better accuracy and prediction.

- 1) Personalization: In this technique one patient can share his data with other patient based on distance between cloud servers they are using to store data. Here we are using dataset so sharing is not possible but i am making all predicted test data values to be open so all users can see or share it.
- 2) Smartness: this technique will be considered as smart as its require no human effort to inform patient about current condition.

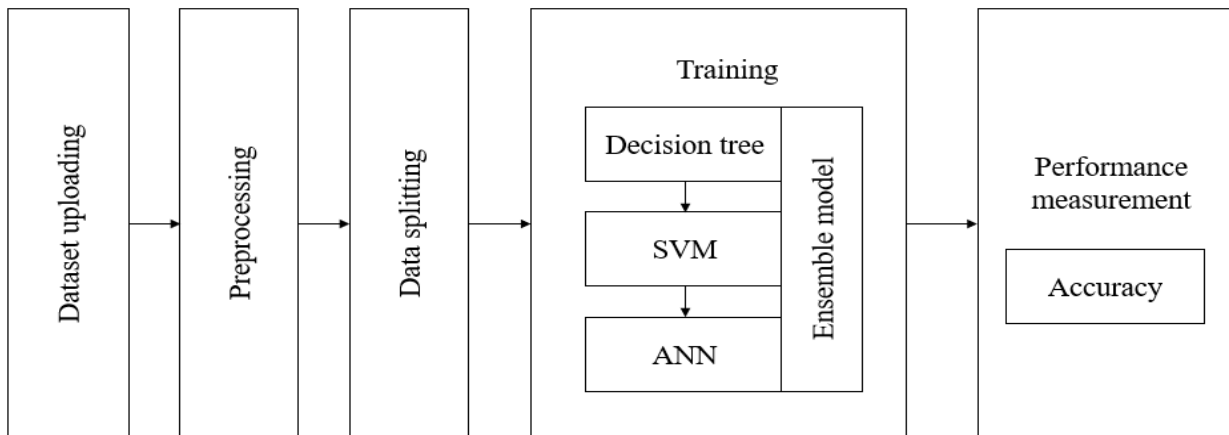


Fig. 4.1(a): Block diagram of cloud application.

Here we have designed two applications to implement above technique

- 1) Cloud Application: This application act like a cloud server and storage and train dataset model with various algorithms such as decision tree, SVM and ANN and Ensemble algorithms.
- 2) User Application: In this application we will upload some test data and will be consider as user sense data and this data will be sent to cloud server and cloud server will apply decision and SVM and ANN model on test data to predict patient condition and send resultant data to this application. As we don't have sensors to sense data, so we consider

uploaded test data as sense data. Here we don't have user details to share data so i am keeping all predicted data to be open so all users can see and share.

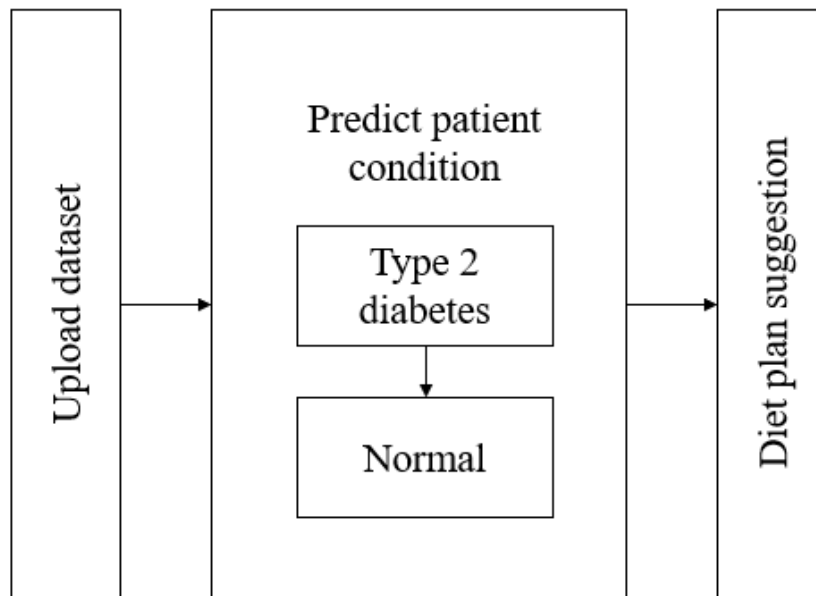


Fig. 4.1(b): Block diagram of user application.

4.1 Dataset

Pregnancies,Glucose,BloodPressure, SkinThickness, Insulin,BMI, DiabetesPedigreeFunction, Age, Outcome

6,148,72,35,0,33.6,0.627,50,1

1,85,66,29,0,26.6,0.351,31,0

8,183,64,0,0,23.3,0.672,32,1

1,89,66,23,94,28.1,0.167,21,0

In above dataset values first record contains dataset column names and other records are the dataset values. All dataset records in last column contains class values as 0 and 1. 1 value indicates patient values show diabetes 1 symptoms and 0 value indicates patient has normal values but indicates diabetes 1 symptoms. Above dataset is used for training and test data will have only patient data but no result values such as 0 or 1. This test data will be applied on train model to predict as 0 or 1.

Below are test values and these values are inside 'users.txt' file inside User/data folder

6,148,72,35,0,33.6,0.627,50

1,85,66,29,0,26.6,0.351,31

8,183,64,0,0,23.3,0.672,32

1,89,66,23,94,28.1,0.167,21

In above test records we can see there is no 0 and 1 values and cloud server will receive and predict values for above test records

4.2 Introduction of Decision Tree Algorithm

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed based on features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, like a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- To build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Features of Decision Tree Learning

- Method for approximating discrete-valued functions (including boolean)
- Learned functions are represented as decision trees (or if-then-else rules)
- Expressive hypotheses space, including disjunction
- The decision tree is robust to noisy data

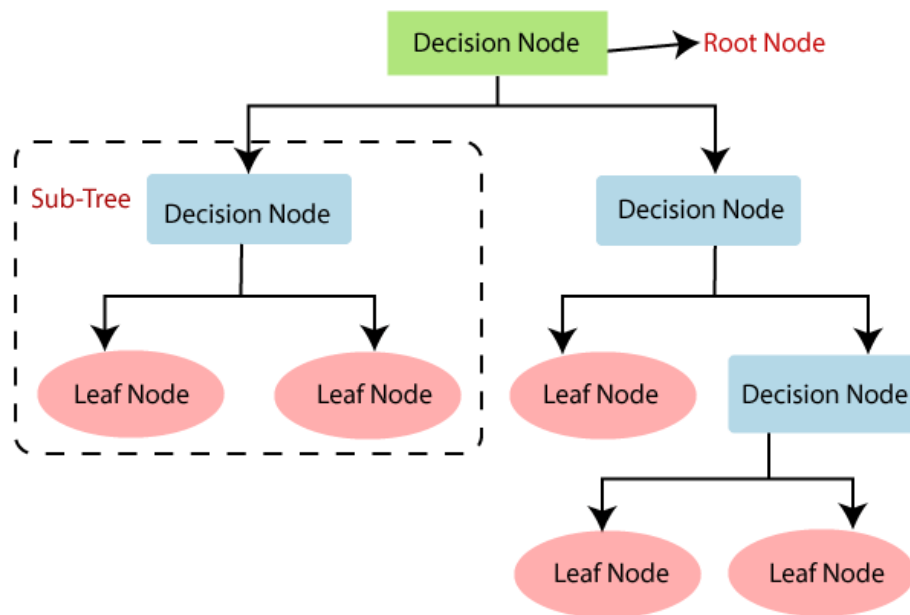


Fig 4.2.1 : Descion Tree structure

Decision Tree Representation

A decision tree is an arrangement of tests that provides an appropriate classification at every step in an analysis.

"In general, decision trees represent a disjunction of conjunctions of constraints on the attribute-values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions".

More specifically, decision trees classify instances by sorting them down the tree from the root node to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

An instance is classified by starting at the root node of the decision tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute. This process is then repeated at the node on this branch and so on until a leaf node is reached.

Diagram

- Each nonleaf node is connected to a test that splits its set of possible answers into subsets corresponding to different test results.
- Each branch carries a particular test result's subset to another node.
- Each node is connected to a set of possible answers.

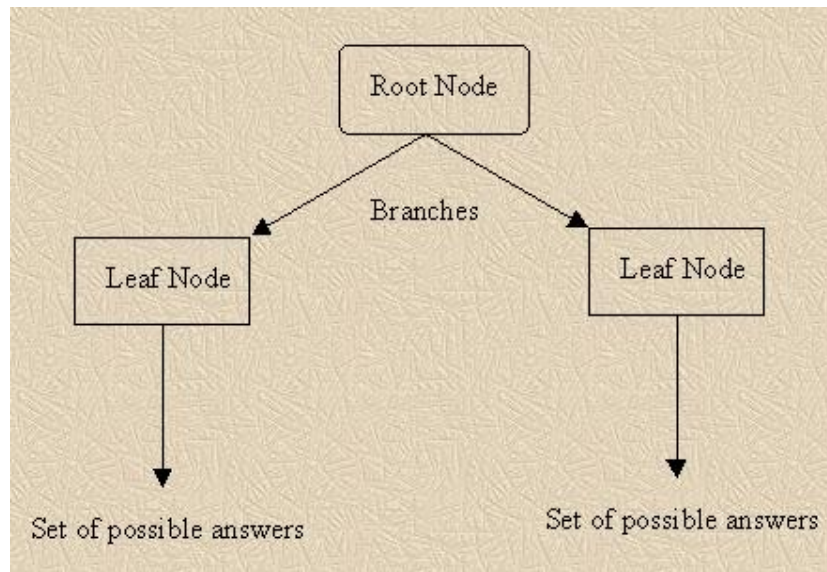


Fig 4.2.2: Decision tree Working

4.3 Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

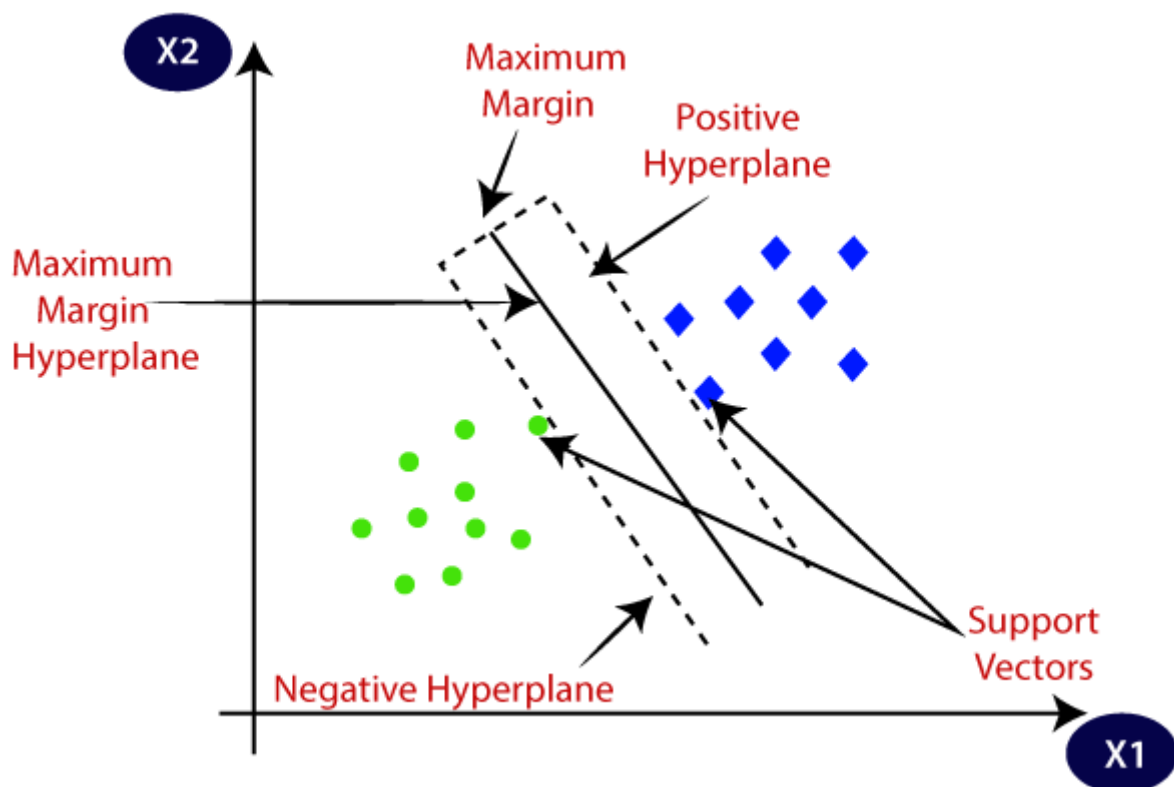


Fig 4.3.1: SVM hyperplane

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

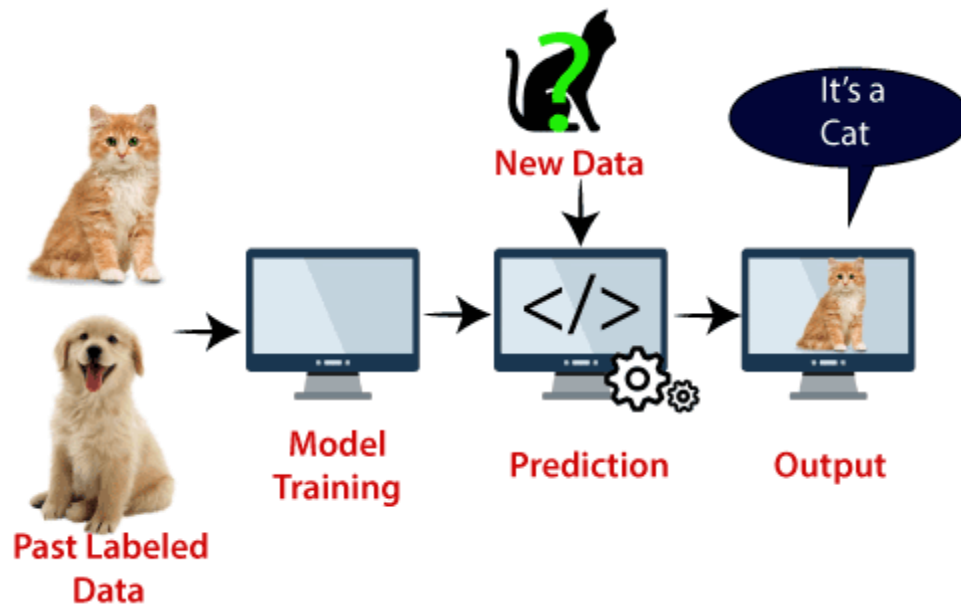


Fig 4.3.2 : SVM working

SVM algorithm can be used for **Face detection, image classification, text categorization**, etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

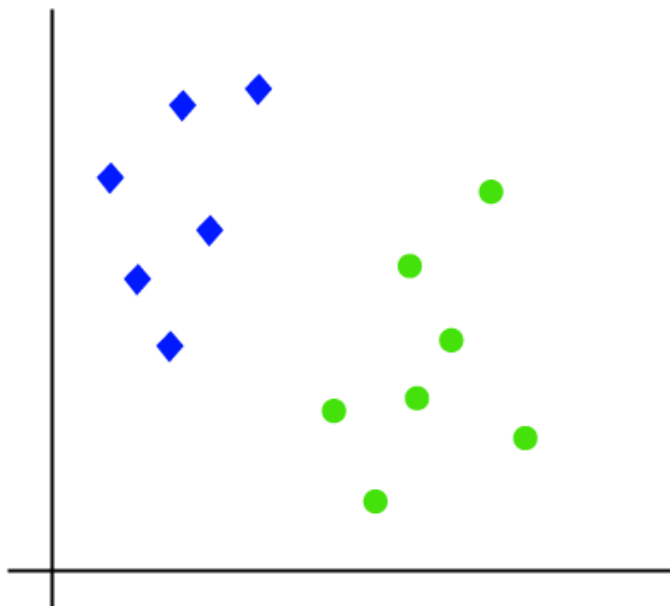
Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

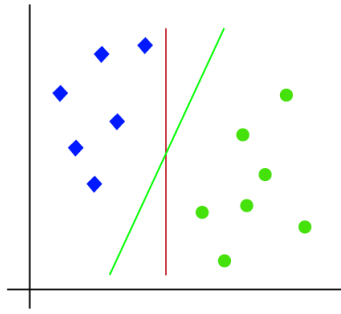
How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair(x_1 , x_2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

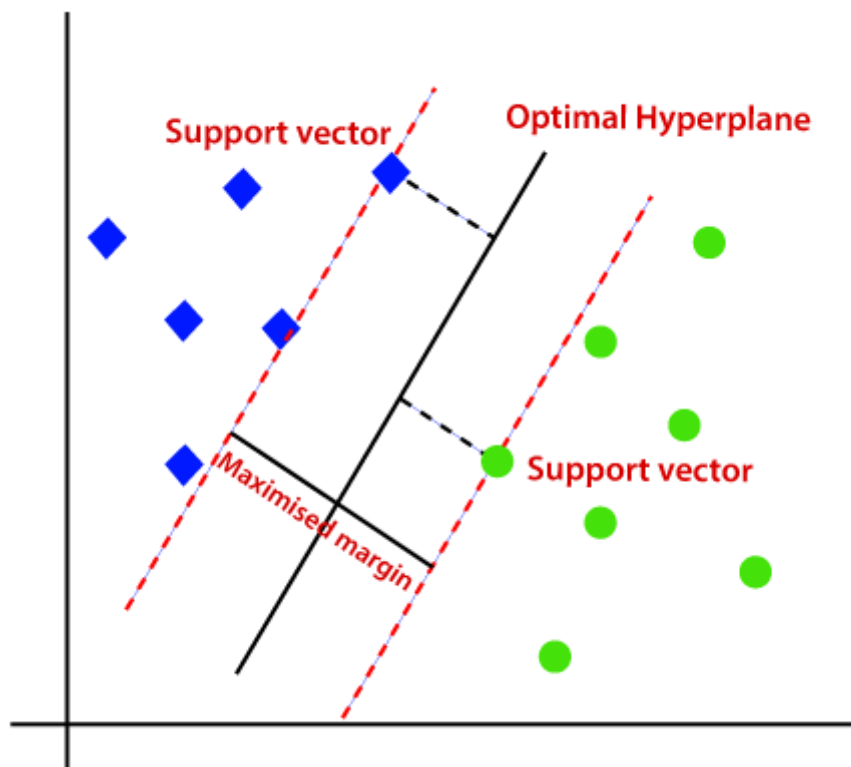
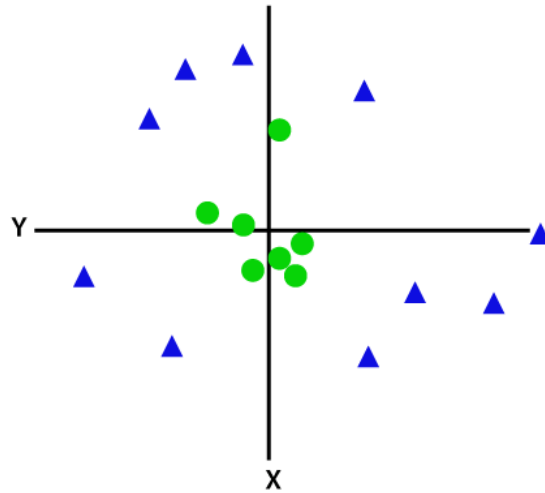


Fig 4.3.3 : SVM Hyperplane Working

Non-Linear SVM:

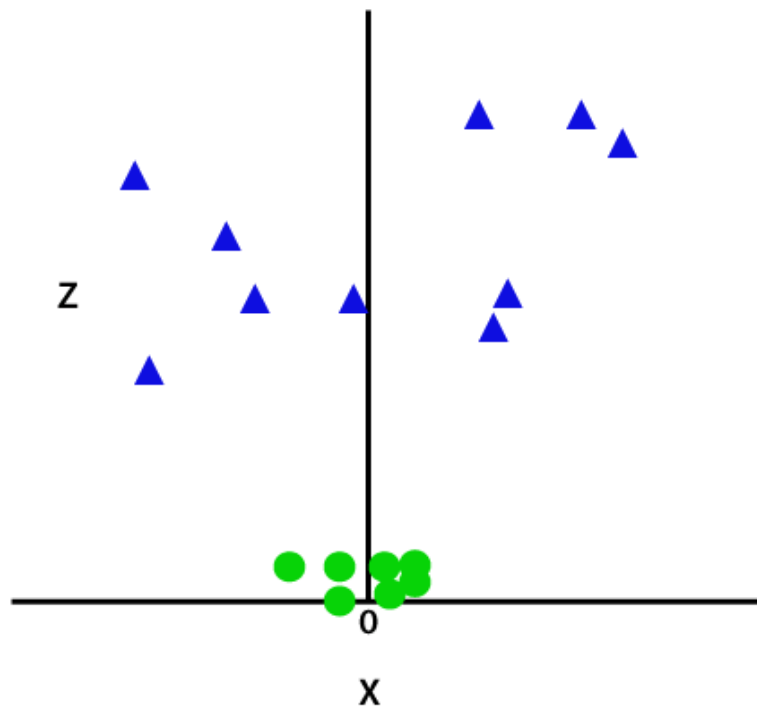
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



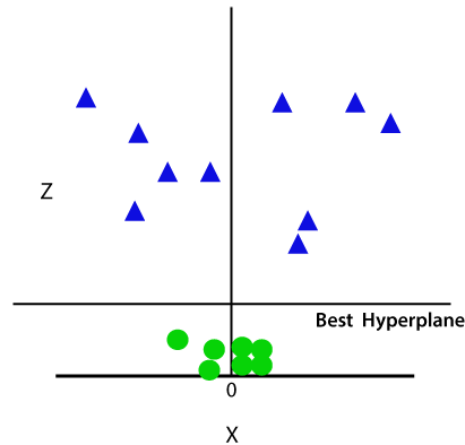
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y , so for non-linear data, we will add a third dimension z . It can be calculated as:

$$z = x^2 + y^2$$

By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with $z=1$, then it will become as:

Applications

- Face recognition
- Weather prediction
- Medical diagnosis
- Spam detection
- Age/gender identification
- Language identification
- Sentimental analysis
- Authorship identification
- News classification

4.4 Artificial Neural Network (ANN)

The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

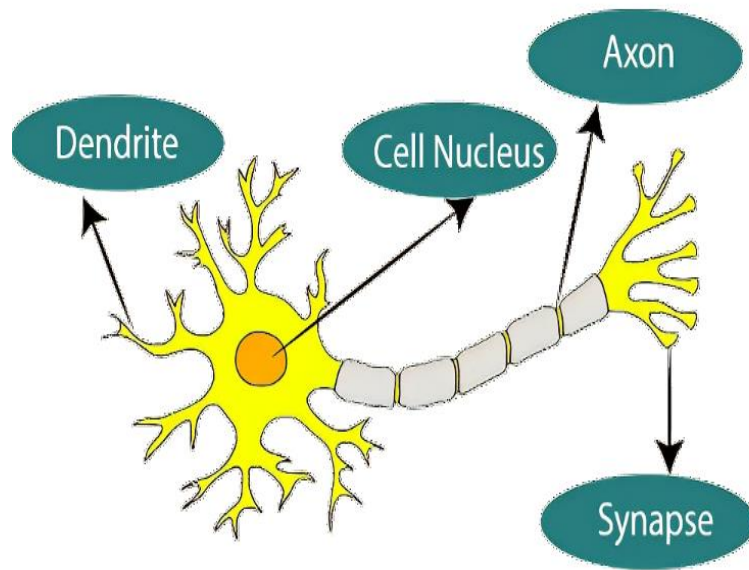


Fig 4.4.1: ANN Structure

The given figure illustrates the typical diagram of Biological Neural Network.
The typical Artificial Neural Network looks something like the given figure.

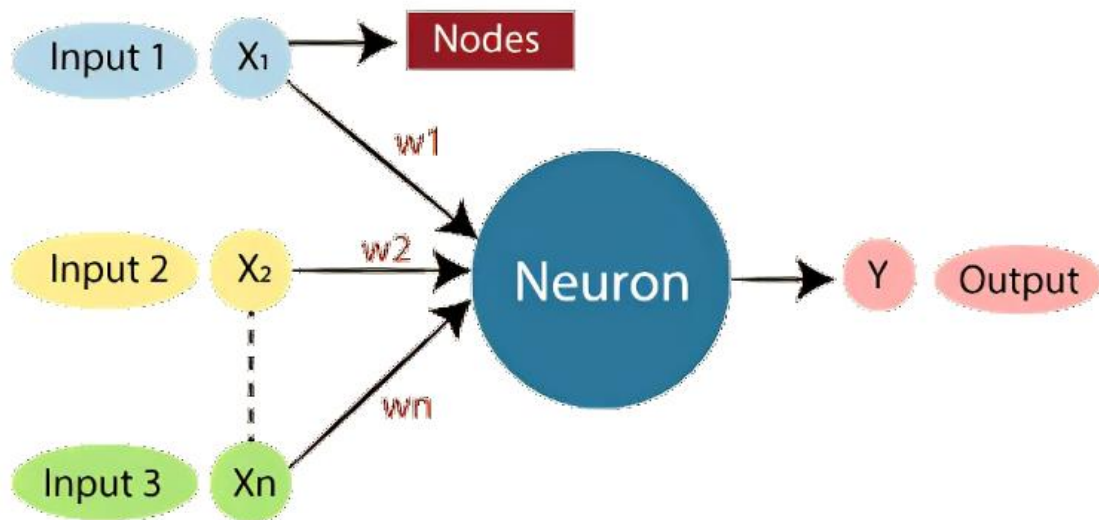


Fig 4.4.2: ANN Working Structure

Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

4.5 K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Fig 4.5.1 : KNN Classifier

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x_1 , so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

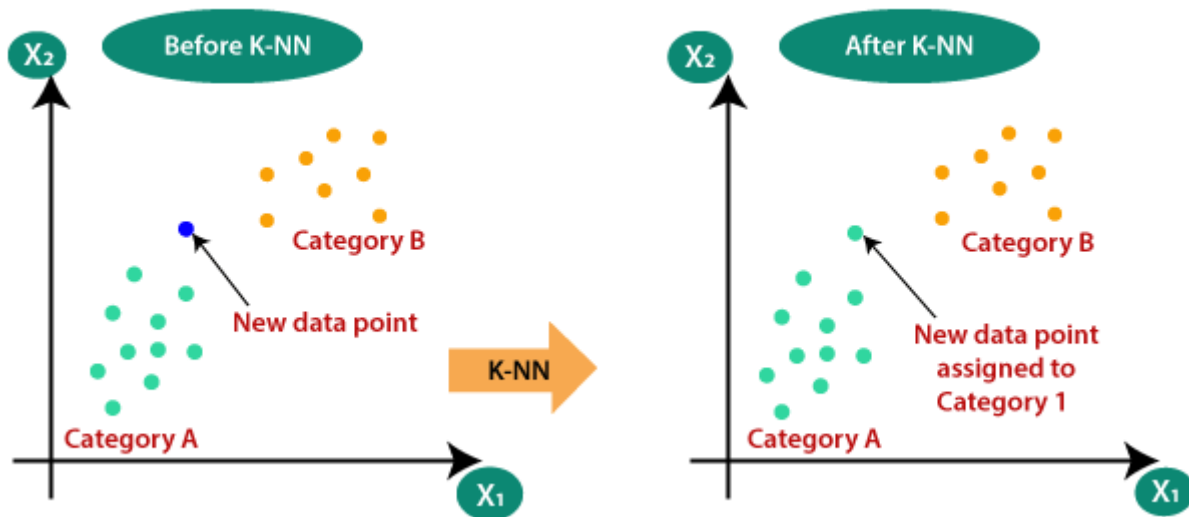


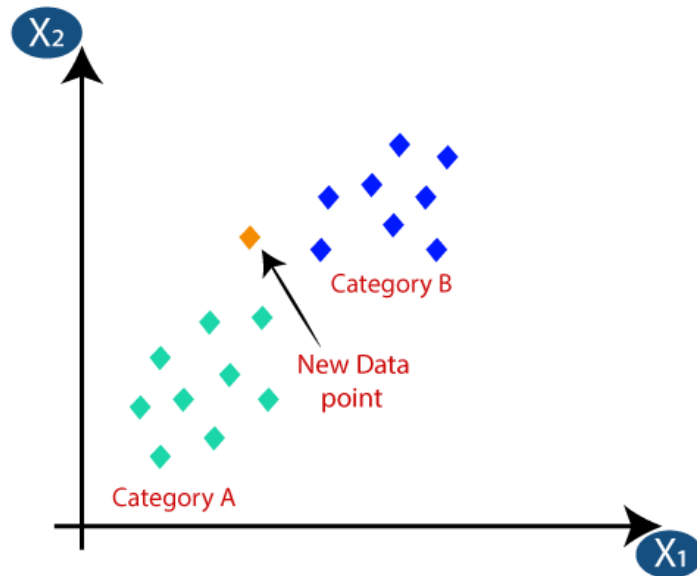
Fig 4.5.2: KNN Working

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the $k=5$.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

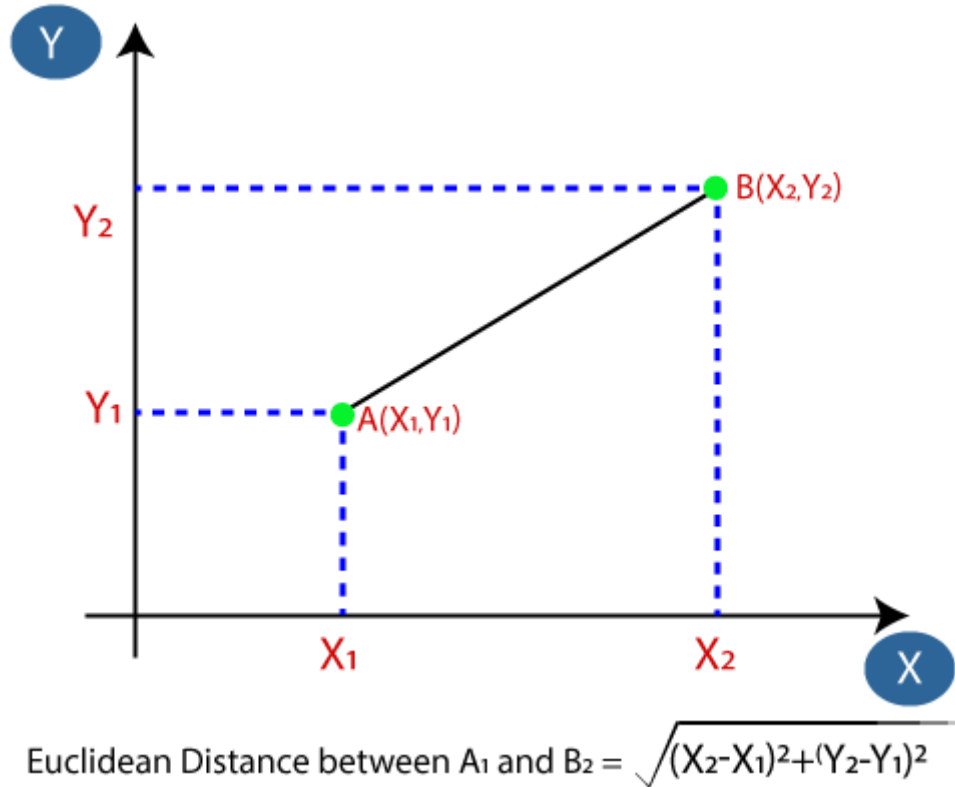


Fig 4.5.3 : KNN Distance from the nearest point

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

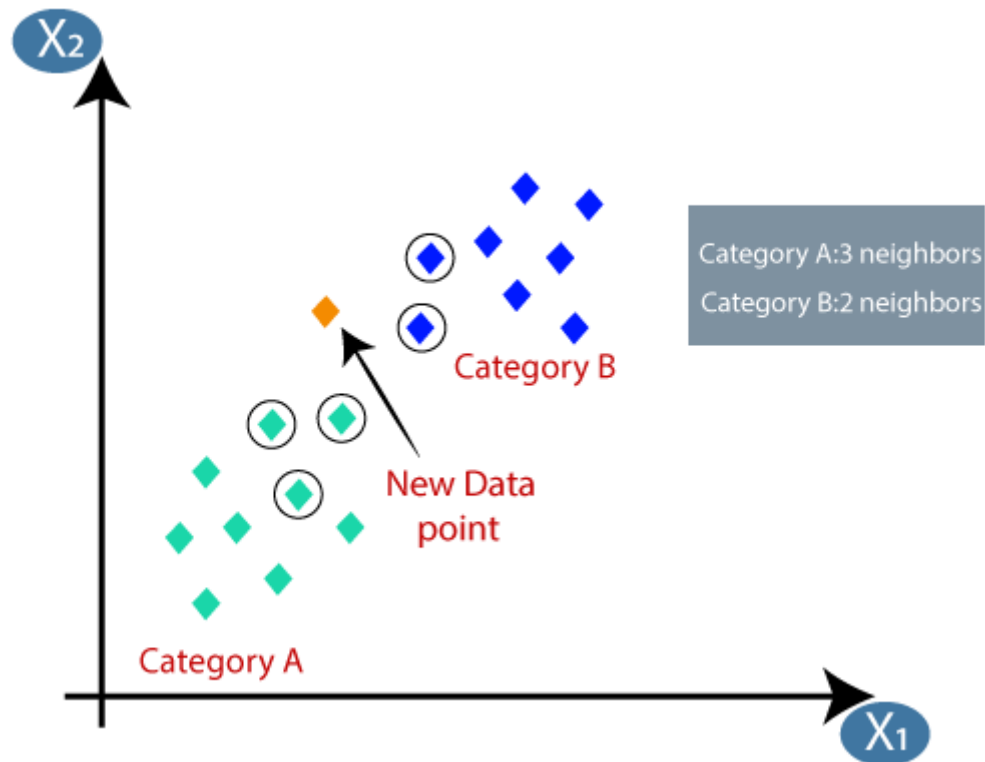


Fig 4.5.4 : KNN working on New Data Point

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm

Below are some points to remember while selecting the value of K in the K-NN algorithm:

Backward Skip 10sPlay VideoForward Skip 10s

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

4.6 Advantages of proposed system

- SVM works relatively well when there is a clear margin of separation between classes.
- SVM is more effective in high dimensional spaces.
- Decision tree is Simple to Understand for Coders.
- Missing Values Aren't an Issue
- KNN is simple to implement
- KNN is robust to the noisy training data
- It can be more effective if the training data is large.

CHAPTER 5

UML DAIGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS: The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

5.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

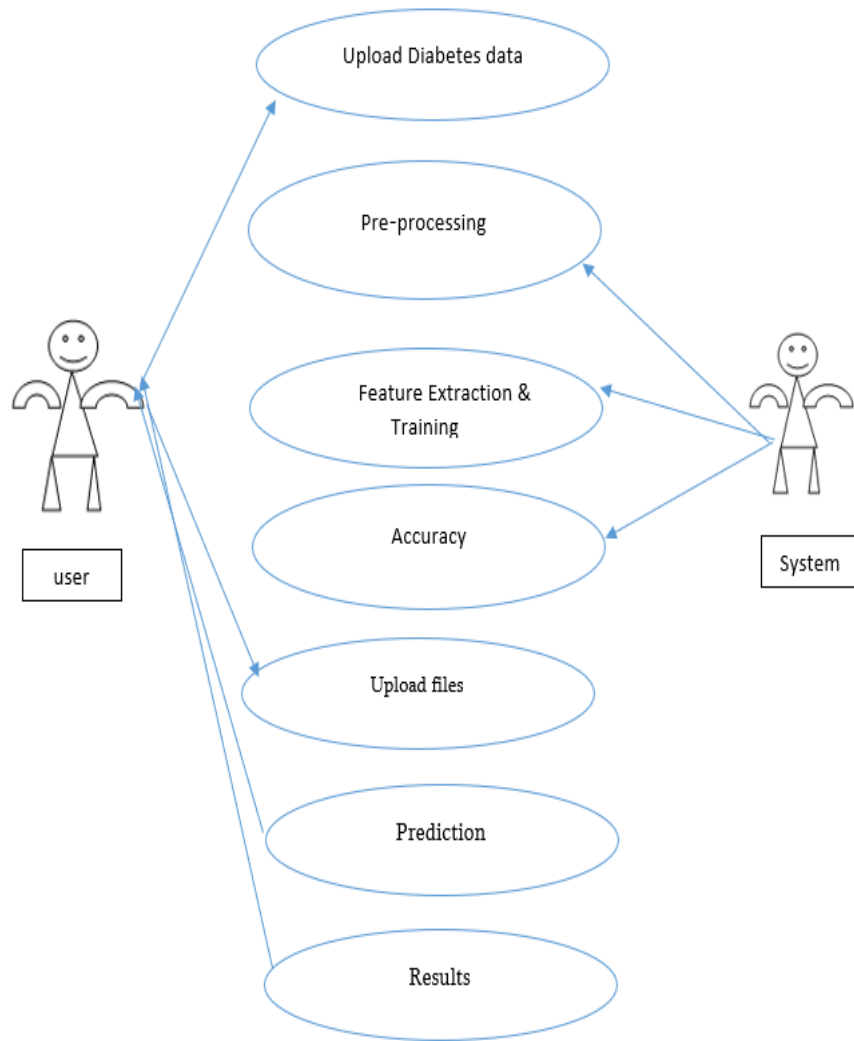


Fig 5.1 : USE Case Diagram

5.2 Sequence Diagram

Represent the objects participating in the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

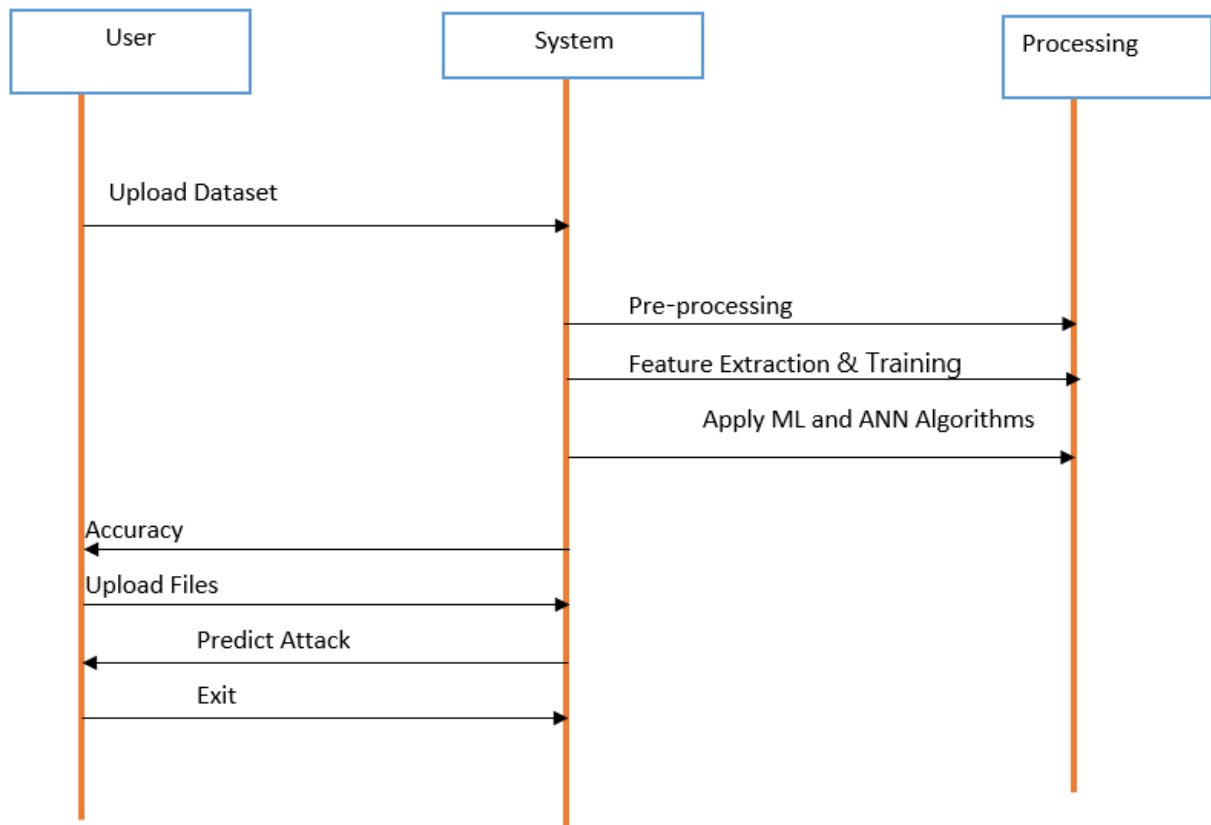


Fig 5.2 : Sequence Diagram

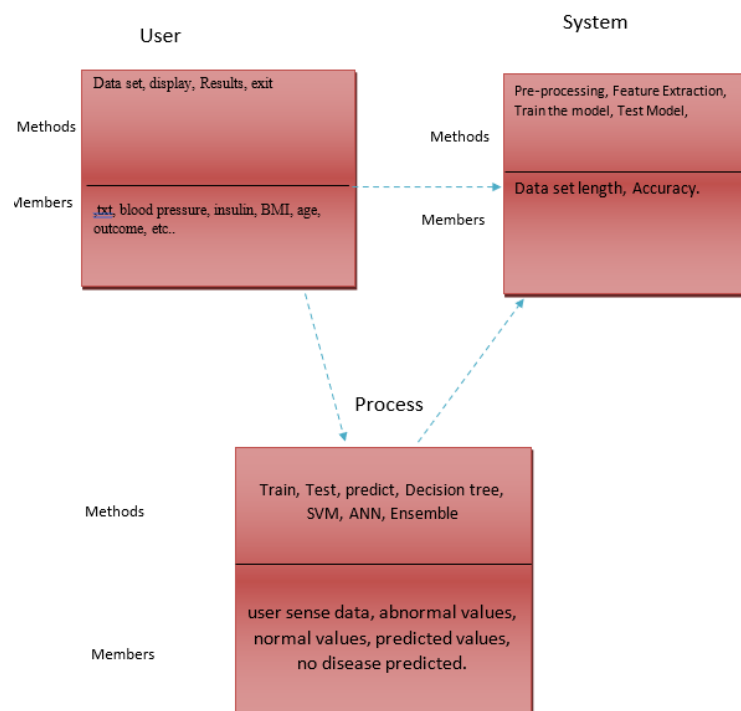


Fig 5.3: Class Diagram

5.3 Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

CHAPTER 6

MACHINE LEARNING

6.1 What is Machine Learning

Before we look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

6.2 Categories of Machine Learning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

6.3 Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and have not surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

6.4 Challenges in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

1. Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
5. Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.
6. Curse of dimensionality – Another challenge ML model faces is too many features of data points. This can be a real hindrance.
7. Difficulty in deployment – Complexity of the ML model makes it quite difficult to be deployed in real life.

6.5 Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of \$146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and

Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature** – A feature is an individual measurable property of the data. A set of numeric

features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning** – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning** – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

6.6 Advantages of Machine learning

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving

machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

6.7 Disadvantages of Machine Learning

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

CHAPTER 7

SOFTWARE ENVIRONMENT

7.1 Introduction to Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free,

but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often

results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later

on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

7.2 Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 3:

Print is now a function.

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges

throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

7.3 Modules Used in Project

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little 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.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you

must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4ae77b9ab010f09be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583da01a442cba0ce08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d83999573a2c56b2ac56cade0b477cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c3cfd9ec0bfabe63184a40728a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76d9db30c3a383e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c60f8bd73aebf53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3bd18841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc802942a54446a3d845147e394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

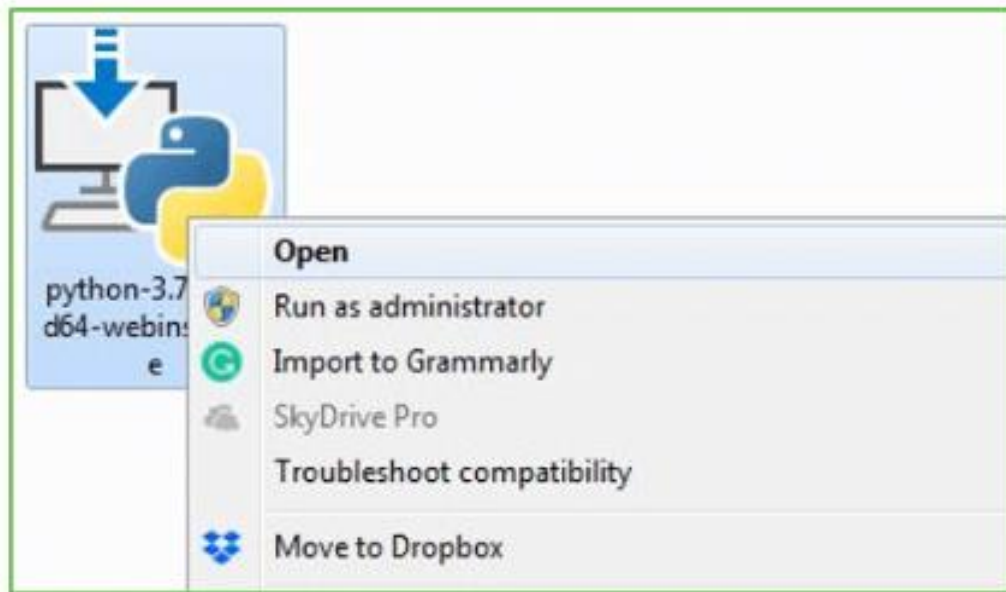
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.



With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

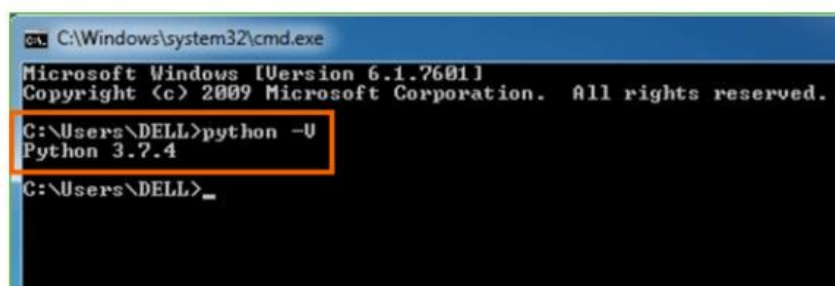
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python -V and press Enter.



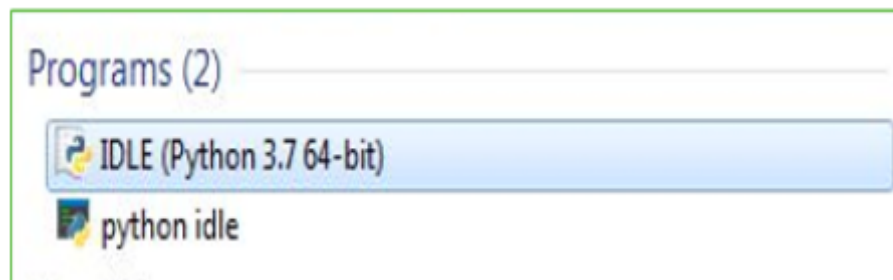
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

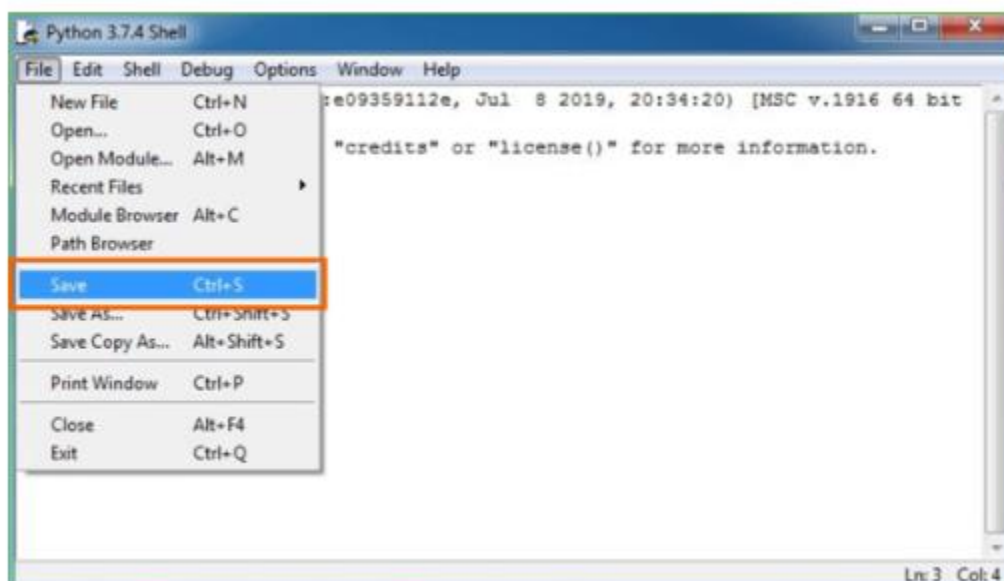
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



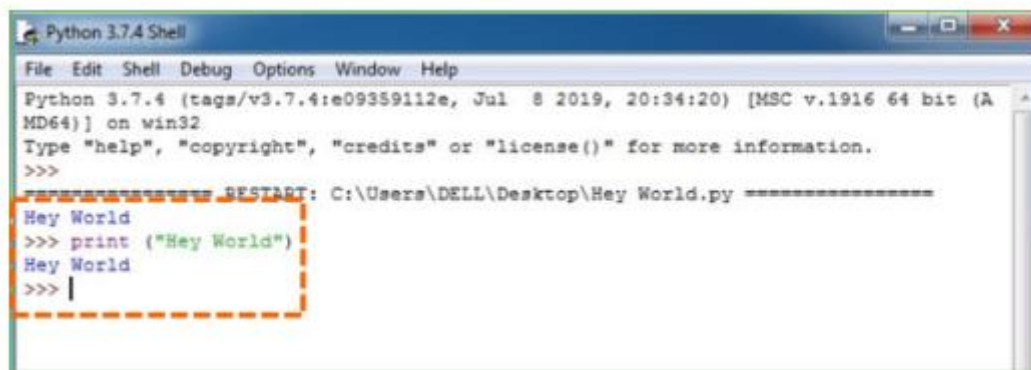
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

A screenshot of a Windows application window titled "Python 3.7.4 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following content: "Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32.", "Type 'help', 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>>". Below the prompt, a dashed orange box highlights the following lines: "Hey World", ">>> print ('Hey World')", "Hey World", and ">>> |". Above this box, a line of text reads "***** RESTART: C:\Users\DELL\Desktop\Hey World.py *****".

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32.
Type "help", "copyright", "credits" or "license()" for more information.
>>>
***** RESTART: C:\Users\DELL\Desktop\Hey World.py *****
Hey World
>>> print ('Hey World')
Hey World
>>> |
```

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

CHAPTER 8

SYSTEM REQUIREMENTS

8.1 SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google collab

8.2 HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- Operating system : Windows, Linux
- Processor : minimum intel i3
- Ram : minimum 4 GB
- Hard disk : minimum 250GB

CHAPTER 9

FUNCTIONAL REQUIREMENTS

9.1 OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

9.2 INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

INPUT STAGES

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

INPUT TYPES

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

INPUT MEDIA

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

9.3 ERROR AVOIDANCE

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

ERROR DETECTION

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

DATA VALIDATION

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

9.4 USER INTERFACE DESIGN

It is essential to consult the system users and discuss their needs while designing the user interface:

USER INTERFACE SYSTEMS CAN BE BROADLY CLASSIFIED AS:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays

further information.

USER INITIATED INTERFACES

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

COMPUTER-INITIATED INTERFACES

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

ERROR MESSAGE DESIGN

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

SOURCE CODE

```
#!/usr/bin/env python

# coding: utf-8

# In[1]:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# In[2]: # In[3]:
variable=pd.read_csv("C:\\Users\\bhargavi\\OneDrive\\Desktop\\16. Prediction of Type-2 Diabetes\\diabetes.csv")

# In[4]: #Describe
data=variable.describe()

# In[5]: #information of dataset
variable.info()

# In[6]:
#Check for all null values
variable.isnull().values.any()

# In[7]:
#histogram
variable.hist(bins=10, figsize=(10,10)) plt.show()

# In[8]: #Correlation
sns.heatmap(variable.corr())

# we see that skin thickness, age, insulin and pregnancies are fully independent on each other
#age and pregnanacies have negative correlation

# In[9]:

#lets count total outcome in each target 0 1
#0 means no diabeted #1 means
patient with diabtes
sns.countplot(y=variable['OUTCOME'],palette='Set1')

# In[10]:
sns.set(style="ticks")
sns.pairplot(variable,
hue="OUTCOME")

# In[11]:
#box plot for outlier visualisation
sns.set(style="whitegrid")
variable.boxplot(figsize=(15,6))

# In[12]: #box plot
sns.set(style="whitegrid")
sns.set(rc={'figure.figsize':(8,4)})
sns.boxplot(x=variable['INSULIN'])
plt.show()
sns.boxplot(x=variable['BLOOD PRESSURE'])
plt.show()
sns.boxplot(x=variable['DIABETES PEDIGREE FUNCTION'])
plt.show()

# In[13]:
#outlier remove
Q1=variable.quantile(0.25)
Q3=variable.quantile(0.75)
IQR=Q3-Q1
print("---Q1-
-- \n",Q1)
print("\n---Q3---
\n",Q3)
print("\n---IQR---\n",IQR)
#print((df < (Q1 - 1.5 * IQR))|(df > (Q3 + 1.5 * IQR)))

# In[14]: #outlier remove
variable_out = variable[~((variable < (Q1 - 1.5 *
```

```

IQR)) |(variable > (Q3 + 1.5 *
IQR))).any(axis=1)]
variable.shape,variable_out.shape #more than 80
records deleted # In[15]:

```

```

#Scatter matrix after removing outlier sns.set(style="ticks") sns.pairplot(variable_out,
hue="OUTCOME") plt.show() # In[16]:

```

```

#lets extract features and targets

```

```

X=variable_out.drop(columns=['OUTCOME']) y=variable_out['OUTCOME']

```

```

# In[17]:

```

```

#Splitting train test data 80 20 ratio from

```

```

sklearn.model_selection import train_test_split

```

```

train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2
) # In[18]:

```

```

train_X.shape,test_X.shape,train_y.shape,test_y.shape #
In[19]:

```

```

from sklearn.metrics import confusion_matrix,accuracy_score,make_scorer
from sklearn.model_selection import cross_validate def tn(y_true, y_pred):
return confusion_matrix(y_true, y_pred)[0, 0] def fp(y_true, y_pred): return
confusion_matrix(y_true, y_pred)[0, 1] def fn(y_true, y_pred): return
confusion_matrix(y_true, y_pred)[1, 0] def tp(y_true, y_pred):
return confusion_matrix(y_true, y_pred)[1, 1] #cross validation
purpose scoring = {'accuracy': make_scorer(accuracy_score),'prec':
'precision'} scoring = {'tp': make_scorer(tp), 'tn': make_scorer(tn),
'fp': make_scorer(fp), 'fn': make_scorer(fn)} def
display_result(result):

```

```

    print("TP: ",result['test_tp'])
print("TN: ",result['test_tn'])
print("FN: ",result['test_fn'])
print("FP: ",result['test_fp']) #

```

```

In[20]: #Logistic Regression from
sklearn.linear_model import
LogisticRegression

```

```

from sklearn.metrics import roc_auc_score acc=[]
roc=[] clf=LogisticRegression()
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X) #find
accuracy
ac=accuracy_score(test_y,y_pred) acc.append(ac)

```

```

#find the ROC_AOC curve

```

```

rc=roc_auc_score(test_y,y_pred) roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))

```

```

#cross val score

```

```

result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10) display_result(result)

```

```

#display predicted values uncomment below line

```

```

#pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head() # In[21]:
#Support Vector Machine from
sklearn.svm import SVC
clf=SVC(kernel='linear')
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X) #find
accuracy
ac=accuracy_score(test_y,y_pred)
acc.append(ac) #find the
ROC_AOC curve
rc=roc_auc_score(test_y,y_pred) roc.append(rc) print("\nAccuracy
{0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10) display_result(result)
#display predicted values uncomment below line
#pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head() # In[22]:

#KNN
from sklearn.neighbors import KNeighborsClassifier clf=KNeighborsClassifier(n_neighbors=3)
clf.fit(train_X,train_y) y_pred=clf.predict(test_X)
#find accuracy ac=accuracy_score(test_y,y_pred)
acc.append(ac) #find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred) roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))
#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10) display_result(result)
#display predicted values uncomment below line
#pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head() # In[23]:
#Random forest from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier() clf.fit(train_X,train_y)
y_pred=clf.predict(test_X) #find accuracy ac=accuracy_score(test_y,y_pred)
acc.append(ac)
#find the ROC_AOC curve rc=roc_auc_score(test_y,y_pred) roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac,rc))
#cross val score

result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10) display_result(result)
#display predicted values uncomment below line
#pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head() # In[24]:

#Naive Bayes Theorem #import library
from sklearn.naive_bayes import
GaussianNB clf=GaussianNB()
clf.fit(train_X,train_y)
y_pred=clf.predict(test_X) #find accuracy

```

```

ac=accuracy_score(test_y,y_pred)
acc.append(ac) #find the ROC_AOC curve
rc=roc_auc_score(test_y,y_pred)
roc.append(rc) print("\nAccuracy {0} ROC
{1}".format(ac,rc))
#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10) display_result(result)
#display predicted values uncomment below line
#pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head() # In[25]:
#Gradient Boosting Classifier from sklearn.ensemble
import GradientBoostingClassifier
clf=GradientBoostingClassifier(n_estimators=50,learning_rate=0.2) clf.fit(train_X,train_y)
y_pred=clf.predict(test_X)

#find accuracy

ac=accuracy_score(test_y,y_pred)
acc.append(ac) #find the
ROC_AOC curve

rc=roc_auc_score(test_y,y_pred)
roc.append(rc) print("\nAccuracy
{0} ROC {1}".format(ac,rc))

#cross val score
result=cross_validate(clf,train_X,train_y,scoring=scoring,cv=10)
display_result(result)
#display predicted values uncomment below line
#pd.DataFrame(data={'Actual':test_y,'Predicted':y_pred}).head() # In[26]:

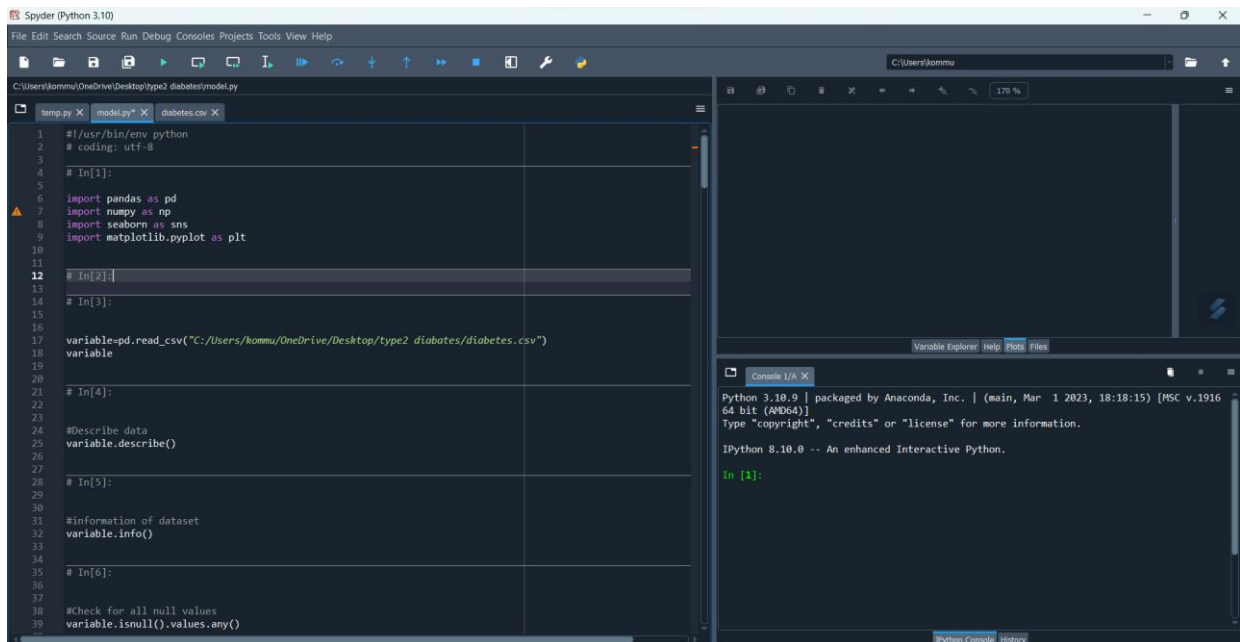
#lets plot the bar graph ax=plt.figure(figsize=(9,4))

plt.bar(['Logistic Regression','SVM','KNN','Random Forest','Naivye Bayes','Gradient
Boosting'],acc,label='Accuracy')
plt.ylabel('Accuracy Score')
plt.xlabel('Algortihms') plt.show()
ax=plt.figure(figsize=(9,4)) plt.bar(['Logistic
Regression','SVM','KNN','Random Forest','Naivye Bayes','Gradient
Boosting'],roc,label='ROC AUC') plt.ylabel('ROC AUC')
plt.xlabel('Algortihms') plt.show()

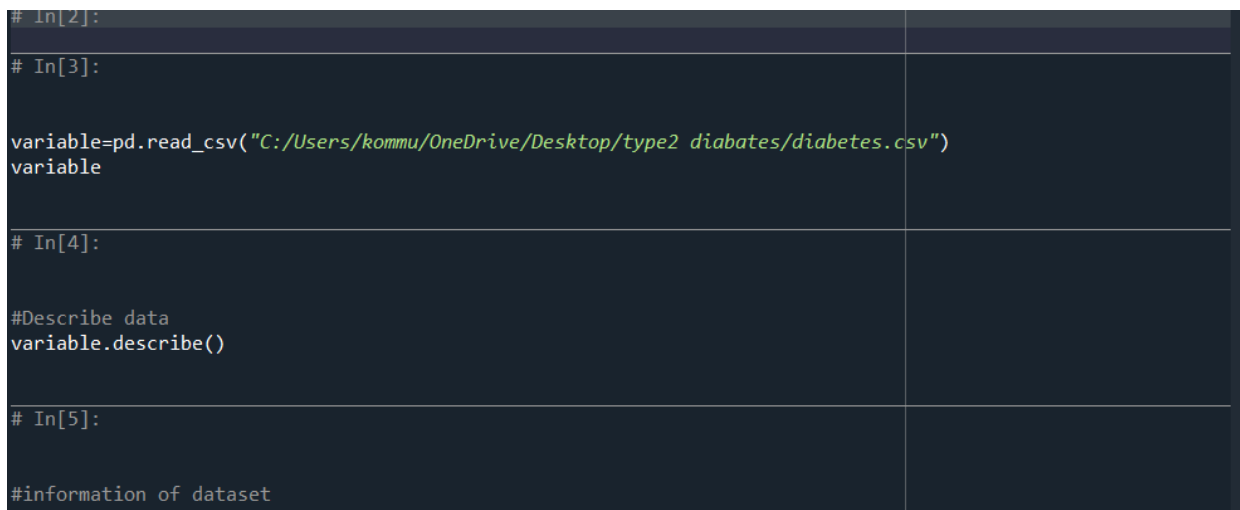
```

CHAPTER 11

RESULTS



Open Spyder and Enter the code



Now upload the data set in [3] . copy the path of the data set in the Folder and insert the path in the Code

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6  import pandas as pd
7  import numpy as np
8  import seaborn as sns
9  import matplotlib.pyplot as plt
10
11
12 # In[2]:
13
14 # In[3]:
15
16
17 variable=pd.read_csv("C:/Users/kommu/OneDrive/Desktop/type2 diabates/diabetes.csv")
18 variable
19
20
21 # In[4]:
22
23
24 #Describe data
25 variable.describe()
26
27
28 # In[5]:
29
30
31 #information of dataset
32 variable.info()
33
34
35 # In[6]:
36
37
38 #Check for all null values
39 variable.isnull().values.any()
40

```

In the above screen run the file in the Spyder

```

In [1]: runfile('C:/Users/kommu/OneDrive/Desktop/type2 diabates/model.py', wdir='C:/Users/kommu/OneDrive/Desktop/type2 diabates')
<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   BLOOD PRESSURE         768 non-null   int64
3   SKIN THICKNESS         768 non-null   int64
4   INSULIN                768 non-null   int64
5   BMI                   768 non-null   float64
6   DIABETES PEDIGREE FUNCTION 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

```

In above screen after pre-process total dataset records are 768. Now all the algorithms started building the Confusion matrix


```
# In[13]:

#outlier remove
Q1=variable.quantile(0.25)
Q3=variable.quantile(0.75)
IQR=Q3-Q1
print("---Q1--- \n",Q1)
print("\n---Q3--- \n",Q3)
print("\n---IQR--- \n",IQR)

#print((df < (Q1 - 1.5 * IQR))|(df > (Q3 + 1.5 * IQR)))

# In[14]:

#outlier remove
variable_out = variable[~((variable < (Q1 - 1.5 * IQR)) |(variable > (Q3 + 1.5 * IQR))).any(axis=1)]
variable.shape,variable_out.shape
#more than 80 records deleted
```

We have now got the outlier Remover as per the code in line 13

```
---Q1---
PREGNANCIES          1.00000
GLUCOSE              99.00000
BLOOD PRESSURE       62.00000
SKIN THICKNESS        0.00000
INSULIN              0.00000
BMI                  27.30000
DIABETES PEDIGREE FUNCTION 0.24375
AGE                  24.00000
OUTCOME              0.00000
Name: 0.25, dtype: float64

---Q3---
PREGNANCIES          6.00000
GLUCOSE             140.25000
BLOOD PRESSURE       80.00000
SKIN THICKNESS       32.00000
INSULIN             127.25000
BMI                  36.60000
DIABETES PEDIGREE FUNCTION 0.62625
AGE                  41.00000
OUTCOME              1.00000
Name: 0.75, dtype: float64

---IQR---
PREGNANCIES          5.00000
GLUCOSE              41.25000
BLOOD PRESSURE       18.00000
SKIN THICKNESS       32.00000
INSULIN             127.25000
BMI                   9.30000
DIABETES PEDIGREE FUNCTION 0.3825
AGE                  17.00000
```

```
# In[19]:
from sklearn.metrics import confusion_matrix, accuracy_score, make_scorer
from sklearn.model_selection import cross_validate

def tn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 0]
def fp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 1]
def fn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 0]
def tp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 1]

#cross validation purpose
scoring = {'accuracy': make_scorer(accuracy_score), 'prec': 'precision'}
scoring = {'tp': make_scorer(tp), 'tn': make_scorer(tn),
           'fp': make_scorer(fp), 'fn': make_scorer(fn)}
def display_result(result):
    print("TP: ", result['test_tp'])
    print("TN: ", result['test_tn'])
    print("FN: ", result['test_fn'])
    print("FP: ", result['test_fp'])

# In[20]:
#Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

acc=[]
roc=[]

clf=LogisticRegression()
clf.fit(train_X, train_y)
y_pred=clf.predict(test_X)
#find accuracy
ac=accuracy_score(test_y, y_pred)
```

```
TP: [ 5 12  5  5  7 10 11  7 11  7]
TN: [33 33 31 32 31 32 30 28 35 30]
FN: [11  4 11 11  9  6  5  9  5  9]
FP: [ 3  2  4  3  4  3  5  7  0  5]

Accuracy 0.78125 ROC 0.6909090909090909
TP: [ 5 12  9  6  7 10 11  6 12  7]
TN: [33 34 32 32 31 32 29 29 34 29]
FN: [11  4  7 10  9  6  5 10  4  9]
FP: [ 3  1  3  4  3  6  6  1  6]

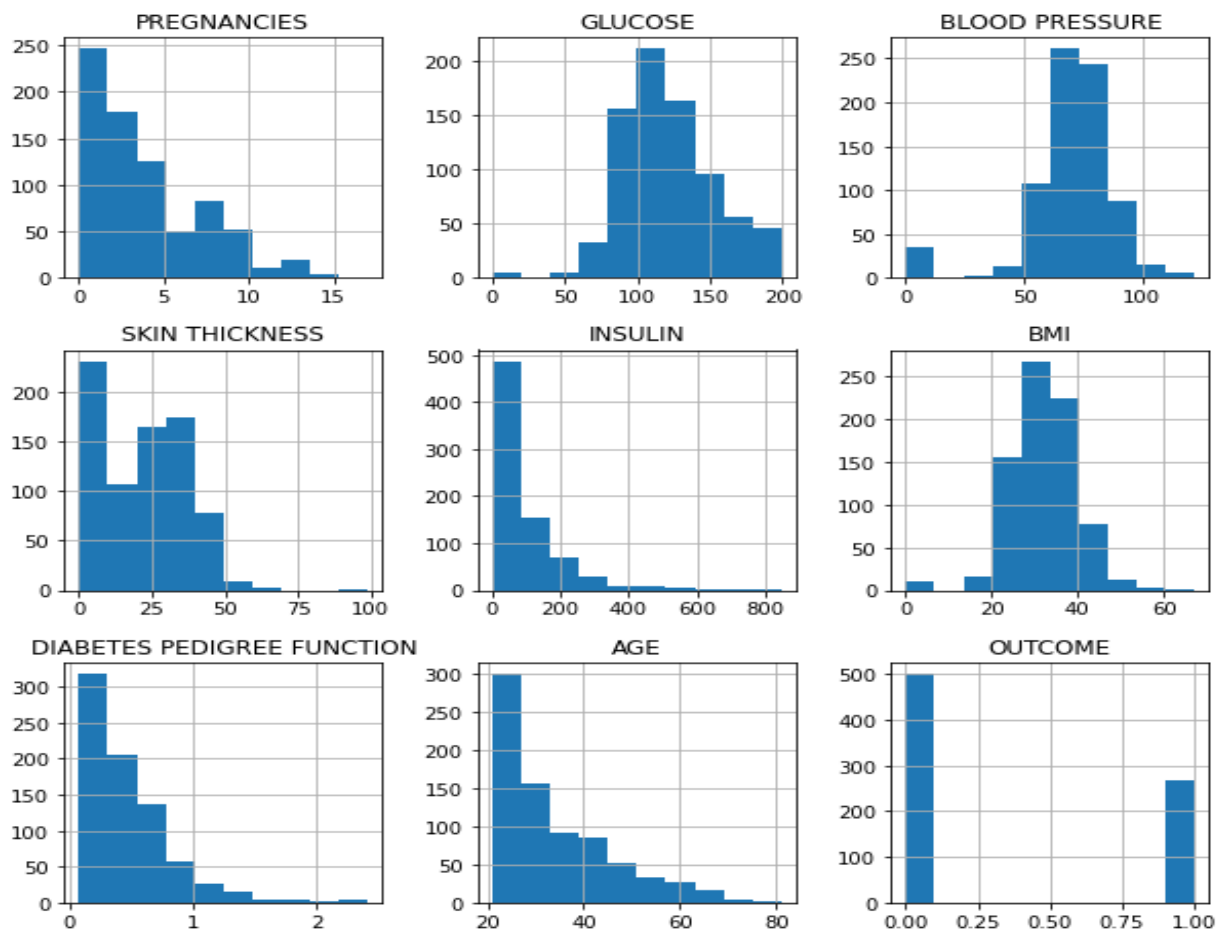
Accuracy 0.734375 ROC 0.6568181818181818
TP: [11 10  9  9 12  9 10  9  5  5]
TN: [31 28 27 28 26 28 26 31 30 28]
FN: [ 5  6  7  7  4  7  6  7 11 11]
FP: [ 5  7  8  7  9  7  9  4  5  7]

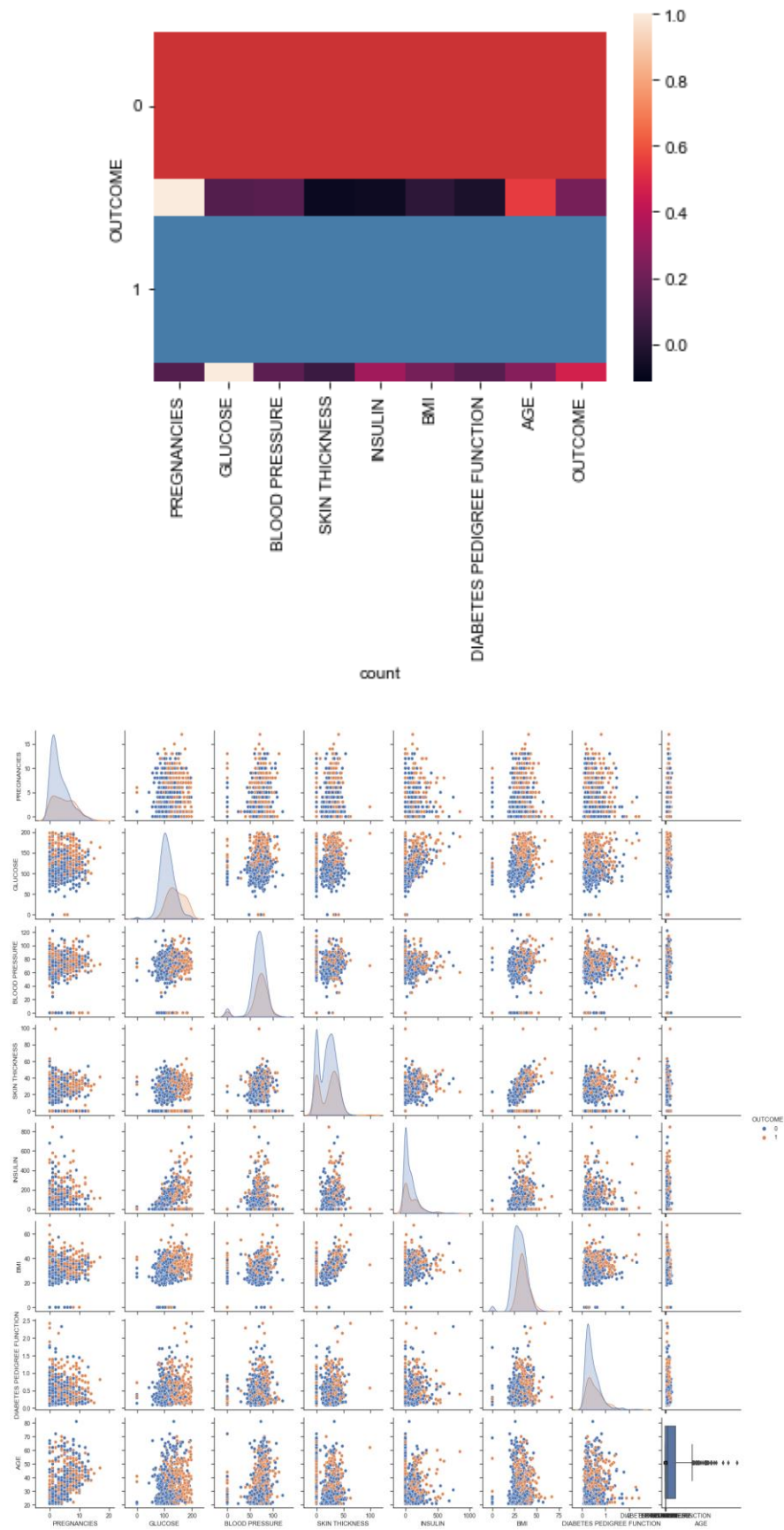
Accuracy 0.7734375 ROC 0.6920454545454546
TP: [ 8 11  6  6  8  8 12  7 10  8]
TN: [33 33 30 32 30 28 28 30 33 28]
FN: [ 8  5 10 10  8  8  4  9  6  8]
FP: [ 3  2  5  3  5  7  7  5  2  7]

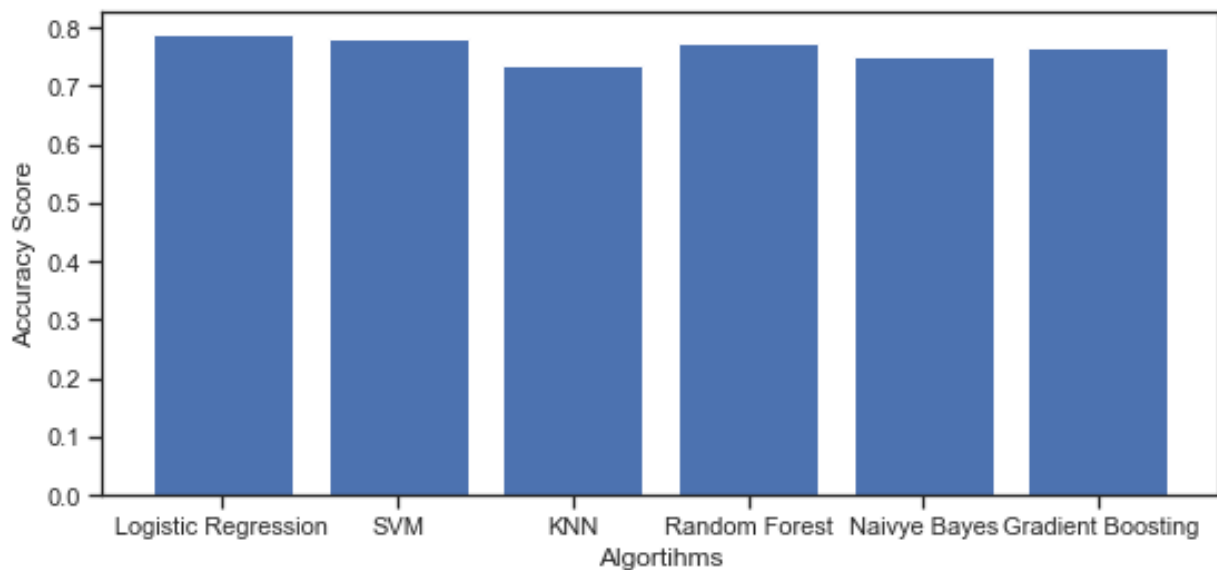
Accuracy 0.75 ROC 0.7090909090909091
TP: [ 7 12  8  7 10  9 13  7 12  8]
TN: [31 30 30 31 29 29 26 26 34 28]
FN: [ 9  4  8  9  6  7  3  9  4  8]
FP: [ 5  5  4  6  6  9  9  1  7]

Accuracy 0.765625 ROC 0.6931818181818182
TP: [ 8 12  8  7  8  7 11  8  9  8]
TN: [31 34 28 32 30 27 29 28 31 28]
FN: [ 8  4  8  9  8  9  5  8  7  8]
```

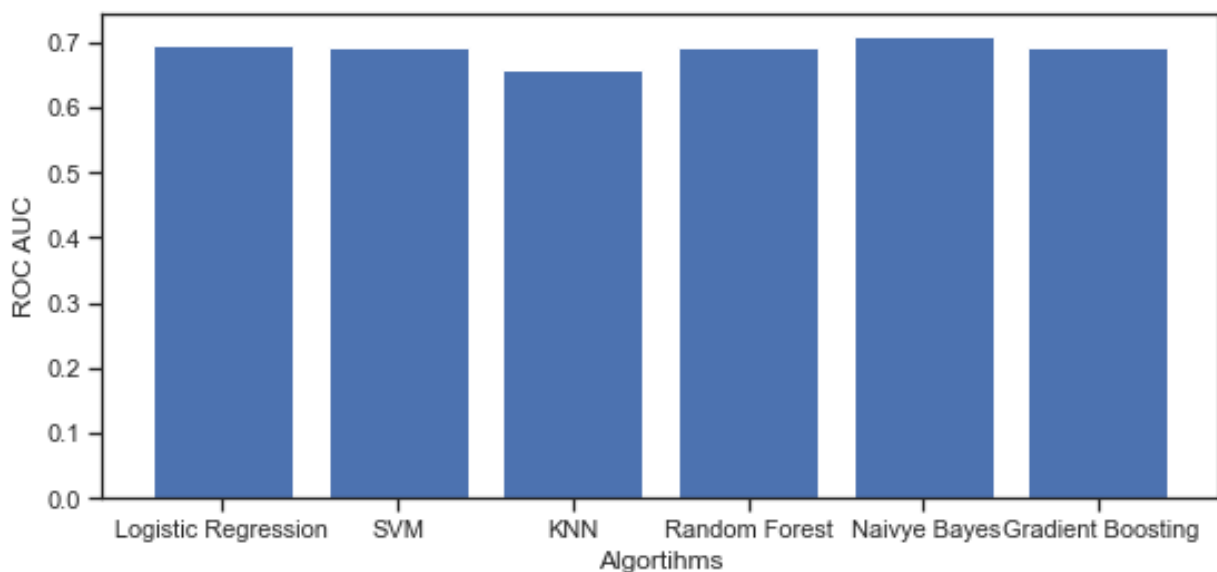
As per the line[19] we have got the confusion matrix and the Accuracy of each algorithms







In above screen we got accuracy for all algorithms, now click on 'Accuracy Graph' button to get accuracy of all algorithms



In above screen graph x-axis represents algorithm name and y-axis represents accuracy values.

Now click on 'Start Cloud Server' button to start server and this server will receive data from user and predict disease details. and now double clicks on 'run.bat' file from User folder to start User sensing application and click on 'Upload Files' button to upload test file and to predict patient condition. After uploading users data will get below prediction results

In above screen for each user data, we predicted 0 and 1 values and also indicates patient values as normal or abnormal. Further, it also suggests the diet plan.

CHAPTER 12

CONCLUSION AND FUTURE SCOPE

Conclusion

In this article, we at first propose a 5G-Smart Diabetes structure that joins a distinguishing layer, an altered end layer, and a data sharing layer. Appeared differently in relation to Diabetes 1.0 and Diabetes 2.0, this system can achieve supportable, viable, and understanding diabetes assurance. By then we propose a very financially savvy data sharing framework in social space and data space. Besides, using AI procedures, we present a tweaked data examination model for 5G- Smart Diabetes. Finally, considering the shrewd dress, wireless and server ranch, we gather a 5G-Smart Diabetes testbed. The preliminary outcomes exhibit that our structure can give tweaked finding and treatment proposals to patients.

Future Scope

This work extended with an intelligent architecture for monitoring diabetic patients by using machine learning algorithms. The architecture elements included smart devices, sensors, and smartphones to collect measurements from the body. The intelligent system collected the data received from the patient, and performed data classification using machine learning in order to make a diagnosis. The proposed prediction system was evaluated by several machine learning algorithms, and the simulation results demonstrated that the sequential minimal optimization (SMO) algorithm gives superior classification accuracy, sensitivity, and precision compared to other algorithms.

REFERENCES

- [1] S. Mendis, “Global Status Report on Noncommunicable Diseases 2014,” WHO, tech. rep.; <http://www.who.int/nmh/publications/ncd-status-report-2014/en/>, accessed Jan. 2015.
- [2] B. Lee, J. Kim, “Identification of Type 2 Diabetes Risk Factors Using Phenotypes Consisting of Anthropometry and Triglycerides Based on Machine Learning,” *IEEE J. Biomed. Health Info.*, vol. 20, no. 1, Jan. 2016, pp. 39--46.
- [3] M. Chen et al., “Disease Prediction by Machine Learning over Big Healthcare Data,” *IEEE Access*, vol. 5, June 2017, pp. 8869—79
- [4] M. Chen, J. Yang, J. Zhou, Y. Hao, J. Zhang and C. -H. Youn, "5G-Smart Diabetes: Toward Personalized Diabetes Diagnosis with Healthcare Big Data Clouds," in *IEEE Communications Magazine*, vol. 56, no. 4, pp. 16-23, April 2018, doi: 10.1109/MCOM.2018.1700788.
- [5] Rghioui A, Lloret J, Sendra S, Oumnad A. A Smart Architecture for Diabetic Patient Monitoring Using Machine Learning Algorithms. *Healthcare*. 2020; 8(3):348. <https://doi.org/10.3390/healthcare8030348>
- [6] Venkatachalam, K., Prabu, P., Alluhaidan, A.S. et al. Deep Belief Neural Network for 5G Diabetes Monitoring in Big Data on Edge IoT. *Mobile Netw Appl* 27, 1060–1069 (2022). <https://doi.org/10.1007/s11036-021-01861-y>.
- [7] E P, Prakash et al. “Implementation of Artificial Neural Network to Predict Diabetes with High-Quality Health System.” *Computational intelligence and neuroscience* vol. 2022 1174173. 30 May. 2022, doi:10.1155/2022/1174173.
- [8] V. Tsoulchas, N. Tsoilis, E. Zoumi, E. Skondras and D. D. Vergados, "Health Monitoring of People with Diabetes using IoT and 5G Wireless Network Infrastructures," 2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA, 2020, pp. 1-6, doi: 10.1109/IISA50023.2020.9284388.

- [9] R. Huang, W. Feng, S. Lu, T. shan, C. Zhang, Y. Liu, "An artificial intelligence diabetes management architecture based on 5G", *Digital Communications and Networks*, 2022, ISSN 2352-8648, <https://doi.org/10.1016/j.dcan.2022.09.004>.
- [10] Suvilesh K. N, Sadasivam V. K, Srivatsa S. K. (2019). Predicting Type 2 Diabetes Mellitus using Machine Learning Techniques: A Literature Survey. *International Journal of Engineering and Advanced Technology (IJEAT)*, 9(2), 1307-1312.
- [11] Navaneethakrishnan, N., & Kumar, S. V. (2019). A Review on Machine Learning Techniques for Prediction of Diabetes. 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, pp. 975-978.
- [12] Chaudhary, P., & Varma, V. (2018). Machine Learning Approaches for Diabetes Prediction: A Systematic Review. *Journal of Medical Systems*, 42(7), 1-12.
- [13] Mehmood, Z., & Yasmin, M. (2020). A Comprehensive Review of Machine Learning Approaches for Diabetes Prediction. *Health and Technology*, 10(2), 383-394.
- [14] Mramba, L. K., Fokoué, E., & Zhao, Q. (2020). A Survey on Machine Learning Techniques for Diabetes Prediction. *International Journal of Machine Learning and Cybernetics*, 11(2), 305-336.
- [15] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.
- [16] Luo, J., Wu, M., Gopukumar, D., Zhao, Y. (2018). Big data application in biomedical research and health care: A literature review. *Biomedical Informatics Insights*, 10, 1-10.
- [17] Razavian, N., Blecker, S., Schmidt, A. M., & Smith-McLallen, A. (2018). Predicting Type 2 Diabetes Mellitus Risk Using Health Data. *Journal of Medical Internet Research*, 20(3), e111.
- [18] Capozzoli, M., L'abbate, L., & Parikh, S. V. (2020). Machine Learning Models for Predicting the Onset of Type 2 Diabetes Mellitus: A Systematic Review. *Journal of Diabetes Science and Technology*, 14(4), 741-756.