##### A Project Report on

**DIABETES DISEASE PREDICTION USING MACHINE**

**LEARNING ALGORITHMS**

###### A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

###### **Bachelor of Technology**

**in**

**Computer Science and Engineering**

Submitted by

G HARSHITHA

(19H51A0509)

G VINAY

(19H51A05D4)

J VISHNUKANTH

(20H55A0511)

Under the esteemed guidance of

MR. B. SIVAIAH

(Associate Professor)



**Department of Computer Science and Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

(UGC Autonomous)

\*Approved by AICTE \*Affiliated to JNTUH \*NAAC Accredited with A+ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

#### 2019- 2023

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



#### CERTIFICATE

This is to certify that the Major Project report entitled **"DIABETES DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS"** being submitted by G.HARSHITHA(19H51A0509), G.VINAY(19H51A05D4), J.VISHNUKANTH(20H55A0511)in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

###### The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

**Mr. B. Sivaiah Dr. Siva Skandha Sanagala**

**Associate Professor Associate Professor and HOD**

**Dept. of CSE Dept. of CSE**

#### ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **Mr. B. Sivaiah, Associate Professor**, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Vijaya Kumar Koppula**, Dean-Academic, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major** **Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

G. HARSHITHA 19H51A0509

G. VINAY 19H51A05D4

J. VISHNUKANTH 20H55A0511

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO.**

**NO.**

LIST OF FIGURES iii

LIST OF TABLES v

ABSTRACT vi

**1** **INTRODUCTION** 1

1.1 Problem Statement 4

1.2 Objectives 5

1.3 Project Scope and Limitations 5

**2** **BACKGROUND WORK** 6

2.1. An Intelligent System for Diabetes Prediction. 7

2.1.1. Introduction 7

2.1.2. Merits and Demerits 7

2.1.3. Implementation of An Intelligent System for

Diabetes Prediction. 7

2.2. Prediction of Diabetes using Classification Algorithms 10

2.2.1. Introduction 10

2.2.2. Merits and Demerits 10

2.2.3. Implementation of Prediction of Diabetes using

Classification Algorithms 11

2.3. Diabetes disease prediction using machine learning on

big data of healthcare. 15

2.3.1. Introduction 15

2.3.2. Merits and Demerits 15

2.3.3. Implementation of Diabetes disease prediction using

machine learning on big data of healthcare. 15

**3 PROPOSED SYSTEM** 19

3.1. Objective of Proposed Model 20

3.2. Algorithms Used for Proposed Model 20

3.3. Designing 35

3.3.1.UML Diagram 35

3.4. Stepwise Implementation and Code 41

**4 RESULTS AND DISCUSSION** 48 4.1. Performance metrics 53

**5** **CONCLUSION** 54

5.1 Conclusion and Future Enhancement 55

**6** **REFERENCES** 56

6.1 References 57

6.2 GitHub Link 58

**List of Figures**

**FIGURE TITLE PAGE NO.**

**NO.**

1 Number of diabetic patients estimated with respect to year 2

2.1.3.2 Support Vector Machine Diagram 9

2.1.3.3 Naive Bayes Model Diagram 10

2.2.3.4 Proposed Model Diagram 12

2.2.3.9 Classified Instances 14

2.3.3.2 Flow chart 17

3.2.1 Decision Tree 21

3.2.2 Decision Tree example 22

3.2.3 KNN Algorithm example 24

3.2.4 KNN algorithm graphs(a) 24

3.2.5 KNN algorithm graphs(b) 25

3.2.6 Logistic regression algorithm graph 26

3.2.7 Random Forest algorithm 29

3.2.8 Random Forest algorithm example 30

3.2.9 Support Vector Machine Algorithm 31

3.2.10 Support Vector Machine Algorithm Example 32

3.2.11 Support Vector Machine Algorithm graphs(a) 33

3.2.12 Support Vector Machine Algorithm graphs(b) 34

3.2.13 Support Vector Machine Algorithm graphs(c) 34

3.3.1.1 Class Diagram 35

3.3.1.2 Use Case Diagram 36

3.3.1.3 Data Flow Diagram 37

3.3.1.4 Sequence Diagram 38

3.3.1.5 Flow Chart: Remote User 39

3.3.1.6 Flow Chart: Service Provider 40

4.1.1 Random User Registration Page 49

4.1.2 Random User 49

4.1.3 Upload Diabetes Dataset 49

4.1.4 Search and Predict Diabetes Page 50

4.1.5 View User Profile Page 50

4.1.6 Service Provider Login Page 50

4.1.7 View All Remote Users Page 51

4.1.8 View Trained and Tested Accuracy in Bar Chart Page 51

4.1.9 Find Diabetic Ratio on Datasets Page 51

4.1.10 Pie Chart and Line Chart Page 52

4.1.11 Accuracy of Algorithms 52

4.1.12 Random User Output 52

**List of Tables**

**FIGURE**

**NO. TITLE PAGE NO.**

2.1.3.1 The ranges of the values of all attributes are given in Table 8

2.2.3.1 Confusion Matrix of Naive Bayes 11

2.2.3.2 Confusion Matrix of Decision Tree 12

2.2.3.3 Confusion Matrix of SVM 12

2.2.3.5 Dataset Description 13

2.2.3.6 Accuracy Description 13

2.2.3.7 Accuracy Measures 14

2.2.3.8 Comparative Performance of Classification Algorithms on

Various Measures 14

2.3.3.1 Dataset Attributes 16

2.3.3.3 Performance metrics 18

2.3.3.4 Major Accuracy Measure Values 18

4.2.1 Performance Metrics 53

**ABSTRACT**

Diabetes is an illness caused because of high glucose level in a human body. Diabetes should not be ignored if it is untreated then Diabetes may cause some major issues in a person like: heart related problems, kidney problem, blood pressure, eye damage and it can also affects other organs of human body. The goal this project is to predict the Diabetes in a human body or a patient for a higher accuracy through applying, Various Machine Learning Techniques. Machine learning techniques Provide better result for prediction by constructing models from datasets collected from patients. In this work we will use Machine Learning Classification and ensemble techniques on a dataset to predict diabetes.

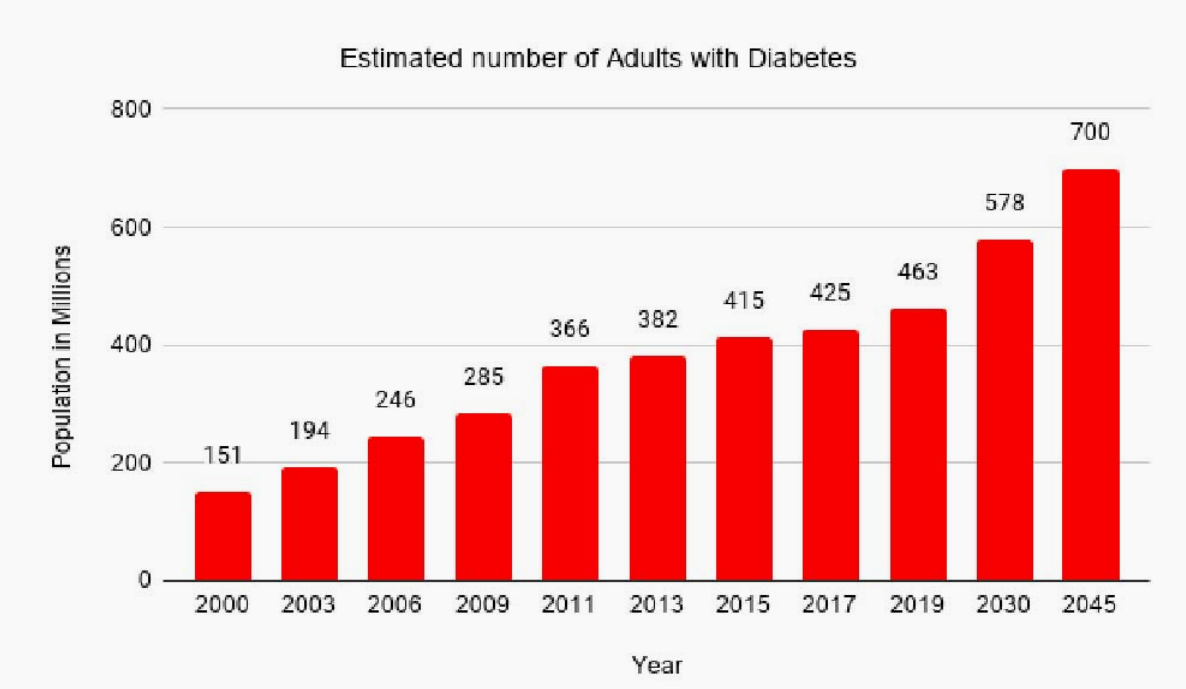
**CHAPTER 1**

**INTRODUCTION**

**CHAPTER 1**

**INTRODUCTION**

In this day and age, one of the most notorious diseases to have taken the world by storm is Diabetes, which is a disease which causes an increase in blood glucose levels as a result of the absence or low levels of insulin. Due to the many criterion to be taken into consideration for an individual to harbour this disease, it’s detection and prediction might be tedious or sometimes inconclusive. Nevertheless, it isn’t impossible to detect it, even at an early stage.



**Figure.1. Number of diabetic patients estimated with respect to year**

In the year 2019, approx. 463 million adults between the age of 20-79 years had diabetes (International Diabetes Federation- IDF). 79% of the adult population were living in the countries with the low and middle-income groups. It is estimated that by the year 2045 approx. 700 million people will have diabetes (IDF). Diabetes is increasing day by day in the world because of environmental, genetic factors. The numbers are rising rapidly due to several factors which include unhealthy foods, physical inactivity and many more. Diabetes is a hormonal disorder in which the inability of the body to produce insulin causes the metabolism of sugar in the body to be abnormal, thereby, raising the blood glucose levels in the body of a particular individual. Intense hunger, thirst and frequent urination are some of the observable characteristics.

**Types of Diabetes**

1) Type one diabetes outcomes due to the failure of the pancreas to supply enough hypoglycemic agent. This type was spoken as "insulin-dependent polygenic disease mellitus" (IDDM) or "juvenile diabetes". The reason is unidentified. The type one polygenic disease found in children beneath twenty years old. People suffer throughout their life because of the type one diabetic and rest on insulin vaccinations. The diabetic patients must often follow workouts and fit regime which are recommended by doctors

2) The type two diabetes starts with hypoglycemic agent resistance, a situation in which cells fail to respond the hypoglycemic agents efficiently. The sickness develops due to the absence of hypoglycemic agent that additionally built. This type was spoken as "non-insulin-dependent polygenic disease mellitus". The usual cause is extreme weight. The quantity of people affected by type two will be enlarged by 2025. The existence of diabetes mellitus is condensed by 3% in rural zone as compared to the urban zone. The prehypertension is joined with bulkiness, obesity and diabetes mellitus. The study found that an individual United Nations agency has traditional vital sign.

3) Type 3 Gestational diabetes occurs when a woman is pregnant and develops high blood sugar levels without a previous history of diabetes. Therefore, it is found that in total 18% of women in pregnancy have diabetes. So, in the older age there is a risk of emerging gestational diabetes in pregnancy. Obesity is one of the main reasons for type-2 diabetes. The type-2 polygenic disease is under control by proper workout and taking appropriate regime. When the aldohexose level isn't reduced by the higher strategies then medications are often recommended. The polygenic disease static report says that 29.1 million people of the United States inhabitants has diabetes.

Certain risk factors such as age, BMI, Glucose Levels, Blood Pressure, etc., play an important role to the contribution of the disease. In the Fig. 1 we can see that the number of cases is rising every year and there is not slowing down in the active cases. It is a very crucial thing to worry as diabetes has become one of the most dangerous and fastest diseases to take the lives of many individuals around the globe. Machine Learning is very popular these days as it is used everywhere, where a large amount of data is present, and we need some knowledge from it. Generally, we can categorise the Machine Learning algorithms in two types but not limited to-

*•* **Unsupervised Learning:** In unsupervised learning, the information is not labelled and also not trained. Here, we just put the data in action to find some patterns if possible.

*•* **Supervised Learning:** In supervised learning, we train the model based on the labels attached to the information and based on that we classify or test the new data with labels.

With the rise of Machine Learning and its relative algorithms, it has come to light that the significant problems and hindrances in its detection faced earlier, can now be eased with much simplicity, yet, giving a detailed and accurate outcome. As of the modern-day, it is comprehended that Machine Learning has become even more effective and helpful in collaboration with the domain of Medicine. Early determination of a disease can be made possible through machine learning by studying the characteristics of an individual. Such early tries can lead to the inhibition of disease as well as obstruction of permitting the disease to reach a critical degree. The work which will be described in this paper is to perform the diabetes disease prediction using machine learning algorithms for early care of an individual.

* 1. **PROBLEM STATEMENT:**
     + Diabetes is a most common disease caused by a group of metabolic disorders. It is also known as Diabetic mellitus. It affects the organs of the human body. It can be controlled by predicting this disease earlier. If diabetic’s patient is untreated for a long time, it may lead to increase blood sugar.
     + Now a days, Healthcare industries generating large volume of data. Machine Learning algorithms and statistics are used to predict the disease with the help of current and past data. Machine learning techniques help the doctors to predict early stage for diabetics. Diabetes patient medical record and different types of algorithms are added in dataset for experimental analysis.
  2. **OBJECTIVES:**
* This research work aims to analyse the Diabetes dataset, design, and implement a Diabetes prediction and recommendation system utilizing machine learning classification techniques. The specific objectives of this project work are:

1. To review existing literature along the area of diabetes diagnosis and prediction.
2. Design and develop a model using machine learning techniques.
3. To analyse the Diabetes dataset and use various machine learning algorithms to develop a prediction model.
4. To identify and discuss the benefits of the designed system along with effective applications.

**1.3.** **PROJECT SCOPE AND LIMITATIONS:**

* Diabetes prediction is one of the most important tasks in today's world, especially in light of its serious complications. Diabetes is the leading cause of death in the world. The system model is primarily focused on detecting diabetes using a few parameters.
* Physicians can utilize the system to anticipate diabetes in the early stages. So that patients can receive traditional treatments and remedies. For the prediction, the system employed some approaches such as machine learning (ML) in order to provide more precise findings. For hospitals and clinicians, developing a diabetes illness prediction system is beneficial.
* The system predicts disease at an early stage, allowing clinicians to better treat patients. The proposed model is a real-time application that can be used in various hospitals and predicts disease in a shorter amount of time. We will receive more accurate and efficient findings when we employ machine learning algorithms for disease prediction.

**CHAPTER 2**

**BACKGROUND WORK**

**2. BACKGROUND WORK**

* 1. **An Intelligent System for Diabetes Prediction**

**2.1.1 Introduction:**

**AUTHORS**: Z. Tafa, N. Pervetica, and B. Karahoda “An Intelligent System for Diabetes Prediction” presents a joint Matlab implementation of the SVM and Naïve Bayes methods in a new dataset acquired from the patients examined for diabetes in Kosovo. The developed diagnostic tool enables the intelligent computer-based prediction on diabetes, based on the previously acquired values. The statistical analysis shows the high accuracy of data classification. Also, the proposed joint implementation of two algorithms aims to improve the reliability of the decision by using the power of both algorithms in minimizing their individual weakness.

**2.1.2 Merits and Demerits:**

* Ensemble of naive bayes and support vector machine has achieved the accuracy of 97.6%.
* When run alone on the dataset naive bayes achieving an accuracy of 94.52 and support vector machine achieving 95.52%.
* The authors have not mentioned any preprocessing technique to filter out any unwanted values from the dataset.
  + 1. **Implementation of An Intelligent System for Diabetes Prediction:**

**Dataset description:**

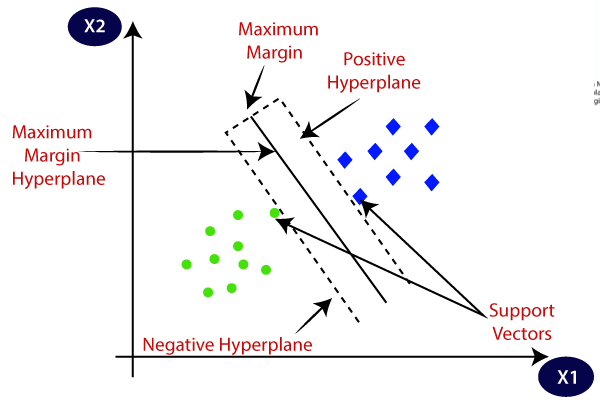
Dataset consists of 402 instances taken from three different locations in Kosovo. During the data acquisition process, the appropriate importance is given to the patient’s data privacy and anonymity. The attributes of the database are: BMI (Body Mass Index), glucose level before meal and after meal, the systolic and diastolic blood pressure, the hereditarily factor, the regular diet, and daily physical activities. The last two attributes are evaluated as follows. Regarding the issue of regular diet, while relying on inputs from the medical clinicians, patients were asked if they took their meals in approximately same equidistant daily intervals at least three times a day and also if their meals were not voluminous. With these answers being positive, we consider that a patient is having the regular diet. On the other hand, according to the U.S. Center for Disease Control and Prevention (CDCP), the adult person is considered to be physically active if he/she conducts the 150-200 min of physical activities a week. With the answers on the family history questions, and in accordance to the above given thresholds, the answers of the examinee regarding the last three questions are mapped into two values:1 and 0.

**Table.2.1.3.1. The ranges of the values of all attributes are given in Table.**

|  |  |  |
| --- | --- | --- |
| **ATTRIBUTE** | **Value Range** | |
| **From** | **To** |
| BMI | 15 | 40 |
| Pre meal glucose | 3.5 | 19 |
| Post meal glucose | 4.9 | 22.8 |
| Diastolic blood pressure | 55 | 110 |
| Systolic blood pressure | 90 | 200 |
| Family history of diabetes | No(0) | Yes(1) |
| Regular diet | No(0) | Yes(1) |
| Physical Activites | No(0) | Yes(1) |

**Support Vector Machine(SVM):**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. 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 .The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three. 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.



**Figure.2.1.3.2. Support Vector Machine Diagram**

**Naive Bayes:**

The Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions**. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**. Some popular examples of Naïve Bayes Algorithm are **Spam filtration, Sentimental analysis, and classifying articles.**

Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:

IMG_256

**Were,**

**P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.

**P(B|A) is Likelihood probability**: Probability of the evidence given that the

probability of a hypothesis is true.

**P(A) is Prior Probability:** Probability of hypothesis before observing the evidence.

**P(B) is Marginal Probability:** Probability of Evidence.

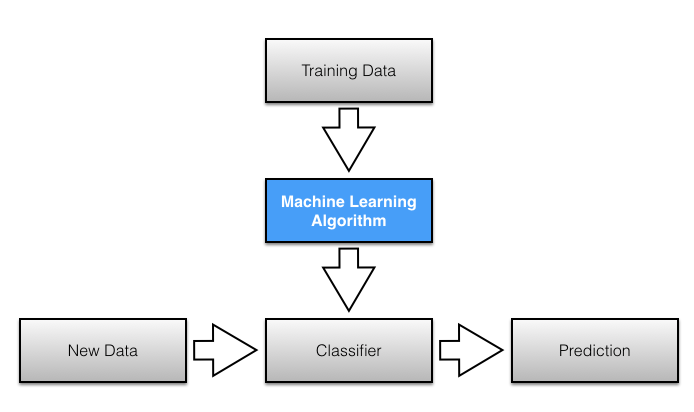
The Naïve Bayes working can be explained on the basis of the below algorithm:

**STEP 1:** Convert the given dataset into frequency tables.

**STEP 2:** Generate a Likelihood table by finding the probabilities of given features.

**STEP 3:** Now, use the Bayes theorem to calculate the posterior probability.

Gaussian Naive Bayes:

Real-valued attributes are estimated by assuming a Gaussian distribution. Easiest to work with, only need mean and std from training data and calculate mean and std of input values(X) for each class to summarize the distr.

**Figure.2.1.3.3. Naive Bayes Model Diagram**

**2.2. Prediction of Diabetes using Classification Algorithms.**

**2.2.1. Introduction:**

* In this model the dataset is collected from Pima Indian Diabetes Dataset.
* The model was proposed using a combination of Naive Bayes, Decision Tree and support vector machine algorithms for diabetes prediction.
* Eight attributes were present inside the dataset, and it consisted of 768 patient’s data.
* When run alone on the dataset, that is, Naïve Bayes achieving an accuracy of 76.30%Support vector machine achieving 65.10%Decision Tree 73.82%.

**2.2.2. Merits and Demerits:**

* Experimental results determine the adequacy of the designed system with an achieved accuracy of 76.30 % using the Naive Bayes classification algorithm.
* They did not go with the cross-validation step as it is imperative to get the optimal and accurate results as well.

**2.2.3. Implementation of Prediction of Diabetes using Classification Algorithms:**

**Naive Bayes Classifier:**

Naive Bayes is a classification technique with a notion which defines all features are independent and unrelated to each other. It defines that status of a specific feature in a class does not affect the status of another feature. Since it is based on conditional probability it is considered as a powerful algorithm employed for classification purpose. It works well for the data with unbalancing problems and missing values. Naive Bayes [24] is a machine learning classifier which employs the Bayes Theorem. Using Bayes theorem posterior probability *P*(*C*|*X*) can be calculated from P(C),P(X) and *P*(*X*|*C*) [23].

Therefore, *P*(*C*|*X*) = (*P*(*X*|*C*) P(C))/*P*(*X*)

Where, *P*(*C*|*X*) = target class’s posterior probability.

*P*(*X*|*C*) = predictor class’s probability.

P(C) = class C’s probability being true.

P(X) = predictor’s prior probability. The evaluated performance of Naive Bayes algorithm using Confusion Matrix is as follows:

**Table.2.2.3.1. Confusion Matrix of Naive Bayes**

|  |  |  |
| --- | --- | --- |
|  | **A** | **B** |
| A-Tested Negative | 422 | 78 |
| B-Tested Positive | 104 | 164 |

**Decision Tree Classifier:**

Decision Tree is a supervised machine learning algorithm used to solve classification problems. The main objective of using Decision Tree in this research work is the prediction of target class using decision rule taken from prior data. It uses nodes and internodes for the prediction and classification. Root nodes classify the instances with different features. Root nodes can have two or more branches while the leaf nodes represent classification. In every stage, Decision tree chooses each node by evaluating the highest information gain among all the attributes. The evaluated performance of Decision Tree technique using Confusion Matrix is as follows:

|  |  |  |
| --- | --- | --- |
|  | **A** | **B** |
| A-Tested Negative | 407 | 93 |
| B-Tested Positive | 108 | 164 |

**Table 2.2.3.2. Confusion Matrix of Decision Tree**

**Support Vector Machine (SVM):**

SVM is one of the standard sets of supervised machine learning model employed in classification. Given a two-class training sample the aim of a support vector machine is to find the best highest-margin separating hyperplane between the two classes The Accuracy of the experiment is evaluated using WEKA interface. The SVM finds the optimal separating hyperplane by maximizing the distance between the two decision boundaries. Mathematically, we will maximize the distance between the hyperplane which is defined by  *x* + *b* = −1 and the hyperplane defined by  *x* + *b* = 1

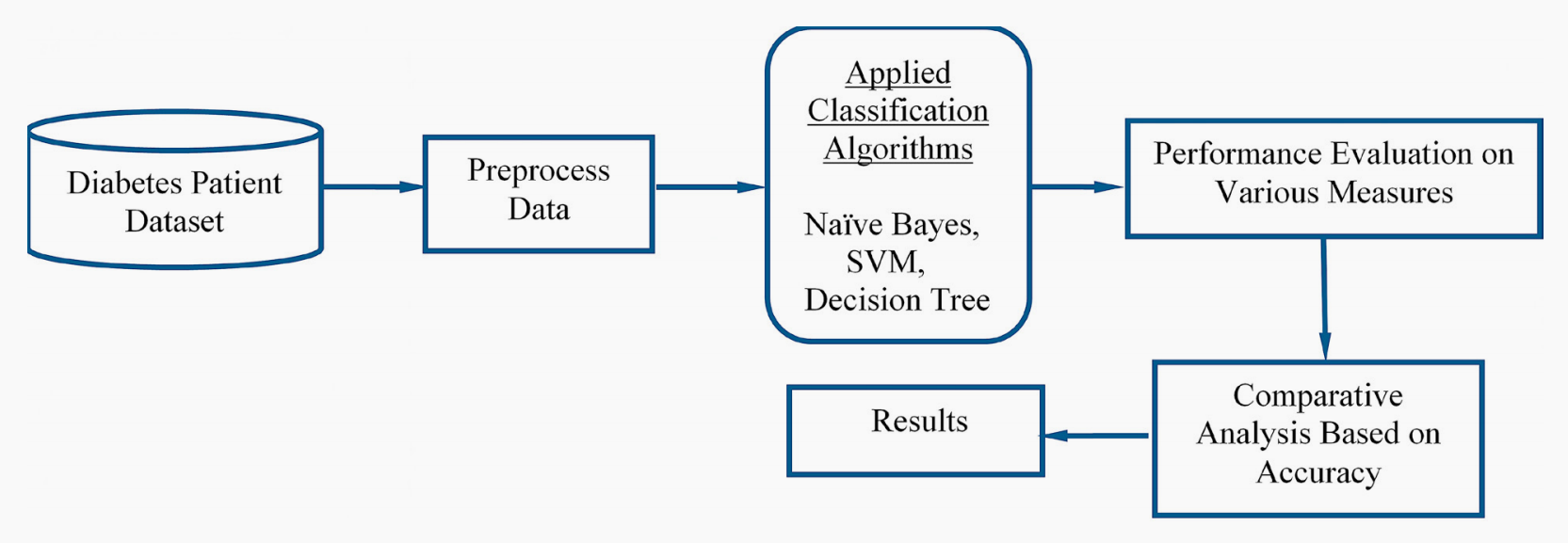
This distance is equal to . This means we want to solve max Equivalently we want min . The SVM should also correctly classify all x(i), which means (+ *b*) >= 1, ∀*i* ∈ {1, ¢¢, *N*}. The evaluated performance of SVM algorithm for prediction of Diabetes [16], [30] using Confusion Matrix is as follows:

**Table.2.2.3.3. Confusion Matrix of SVM**

|  |  |  |
| --- | --- | --- |
|  | **A** | **B** |
| A-Tested Negative | 500 | 0 |
| B-Tested Positive | 268 | 0 |

Model Diagram:

Proposed procedure is summarized in figure-1 below in the form of model diagram. The figure shows the flow of the research conducted in constructing the model.

****

**Figure.2.2.3.4. Proposed Model Diagram**

**Dataset Used:**

In this work WEKA tool is used for performing the experiment. WEKA is a software which is designed in the country New Zealand by University of Waikato, which includes a collection of various machine learning methods for data classification, clustering, regression, visualization etc. One of the biggest advantages of using WEKA is that it can be personalized according to the requirements. The main aim of this study is the prediction of the patient affected by diabetes using the WEKA tool by using the medical database PIDD. Table-4 shows a brief description of the dataset.

**Table.2.2.3.5. Dataset Description**

|  |  |  |
| --- | --- | --- |
| **Database** | **No. of Attributes** | **No. of Instances** |
| PIDD | 8 | 768 |

PIDD-Pima Indians Diabetes Dataset:

The proposed methodology is evaluated on Diabetes Dataset namely (PIDD), which is taken from UCI Repository. This dataset comprises of medical detail of 768 instances which are female patients. The dataset also comprises numeric-valued 8 attributes where value of one class ’0’ treated as tested negative for diabetes and value of another class ’1’ is treated as tested positive for diabetes.

Accuracy Measures: Naive Bayes, SVM and Decision Tree algorithms are used in this research work.

**Table.2.2.3.6. Accuracy Description**

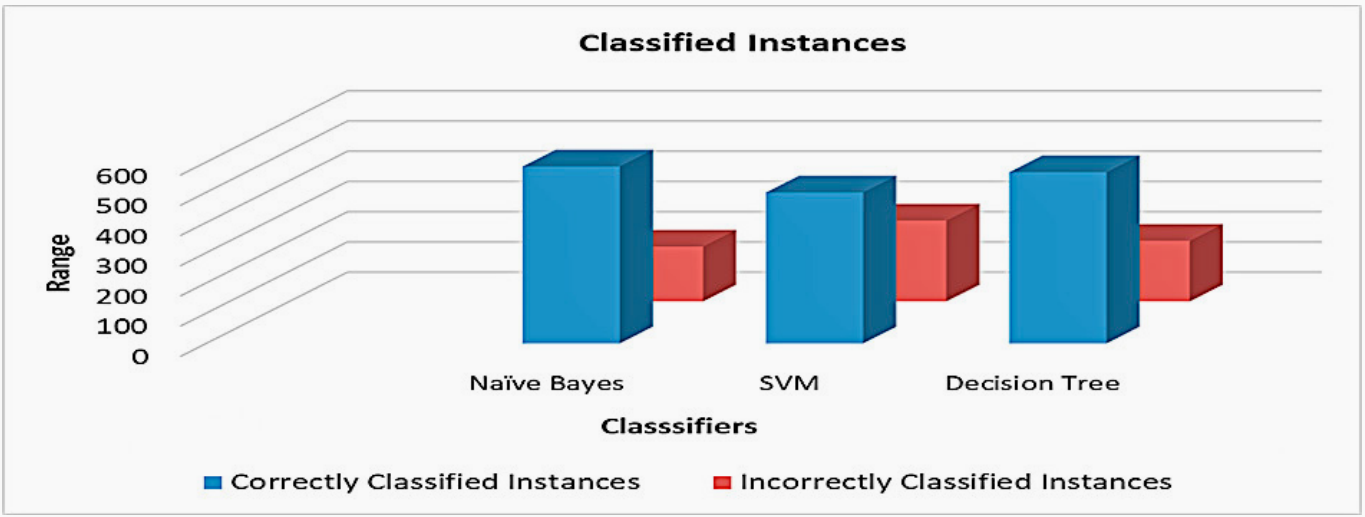
|  |  |  |
| --- | --- | --- |
| **S.No** | **Attribute** | **Abbreviation of Attributes** |
| 1 | Number of times pregnant | Pr |
| 2 | Plasma glucose concentration | Pl |
| 3 | Diastolic blood pressure (mm Hg) | Pr |
| 4 | Skin fold thickness (mm) | Sk |
| 5 | 2-Hour serum insulin (mu U/ml) | In |
| 6 | BMI (weight in kg/) | Ma |
| 7 | Diabetes pedigree function | Pe |
| 8 | Age in years | Ag |
| 9 | Class ’0’ or ’1’ | cl |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Measures** | **Definitions** | **Formula** |
| 1 | Accuracy (A) | Accuracy determines the accuracy of the algorithm in predicting instances. | A=(TP+TN) / (Total no of samples) |
| 2 | Precision (P) | Classifiers correctness/accuracy is measured by Precision. | P = TP / (TP+ FP) |
| 3 | Recall (R) | To measure the classifiers completeness or sensitivity, Recall is used. | R =TP / (TP+FN) |
| 4 | F-Measure | F-Measure is the weighted average of precision and recall. | F=2\*(P\*R) / (P+R) |
| 5 | ROC | ROC(Receiver Operating Curve) curves are used to compare the usefulness of tests. |  |

**Table.2.2.3.7. Accuracy Measures**

**Table.2.2.3.8.Comparative Performance of Classification Algorithms on Various Measures**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification Algorithms** | **Precision** | **Recall** | **F-Measure** | **Accuracy %** | **ROC** |
| Naive Bayes | 0.759 | 0.763 | 0.760 | 76.30 | 0.819 |
| SVM | 0.424 | 0.651 | 0.513 | 65.10 | 0.500 |
| Decision Tree | 0.735 | 0.738 | 0.736 | 73.82 | 0.751 |



**Figure.2.2.3.9.Classified Instances**

* 1. **Diabetes disease prediction using machine learning on big data of healthcare**
     1. **Introduction:**

Healthcare domain is a very prominent research field with rapid technological advancement and increasing data day by day. In order to deal with large volume of healthcare data we need Big Data Analytics which is an emerging approach in Healthcare domain. Millions of patients seek treatments around the globe with various procedure. Analyzing the trends in treatment of patients for diagnosis of a particular disease will help in making informed and efficient decisions to improve the overall quality of healthcare. Machine Learning is a very promising approach which helps in early diagnosis of disease and might help the practitioners in decision making for diagnosis. This paper aims at building a classifier model using WEKA tool to predict diabetes disease by employing Naive Bayes, Support Vector Machine, Random Forest and Simple CART algorithm. The research hopes to recommend the best algorithm based on efficient performance result for the prediction of diabetes disease. Experimental results of each algorithm used on the dataset was evaluated. It is observed that Support Vector Machine performed best in prediction of the disease having maximum accuracy.

* + 1. **Merits and demerits:**
* The experimental has performed by using Naive Bayes, Support Vector Machine, Random Forest and Simple CART classifiers on the diabetes patient dataset.
* Selected the best algorithm after performing the performance evaluation of all the four classifies.
* WEKA is only suitable for small datasets.
  + 1. **Implementation of Diabetes disease prediction using machine learning on big data of healthcare:**

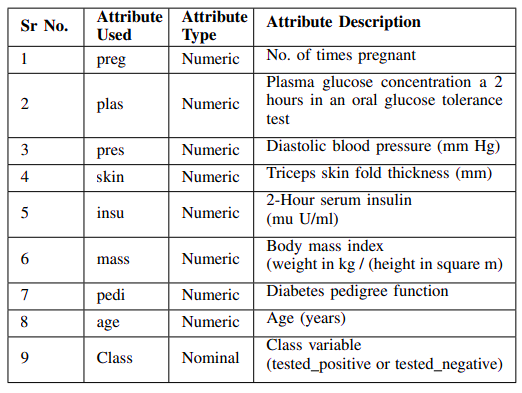
WEKA Tool Description**:** WEKA [Waikata Enviroment for Knowledge Analysis]

* It is a very popular machine learning and data mining toolkit for conducting data driven researches.
* Developed in New Zealand at the University of Waikato
* The collection of machine learning and data mining algorithms present are written in Java
* The version of WEKA used for experimentation in this paper is WEKA Version 3.82 the research made use of WEKA tool as it helps in performance evaluation and performing comparison of various machine learning techniques conveniently on real time data.

**Diabetes Disease Dataset:**

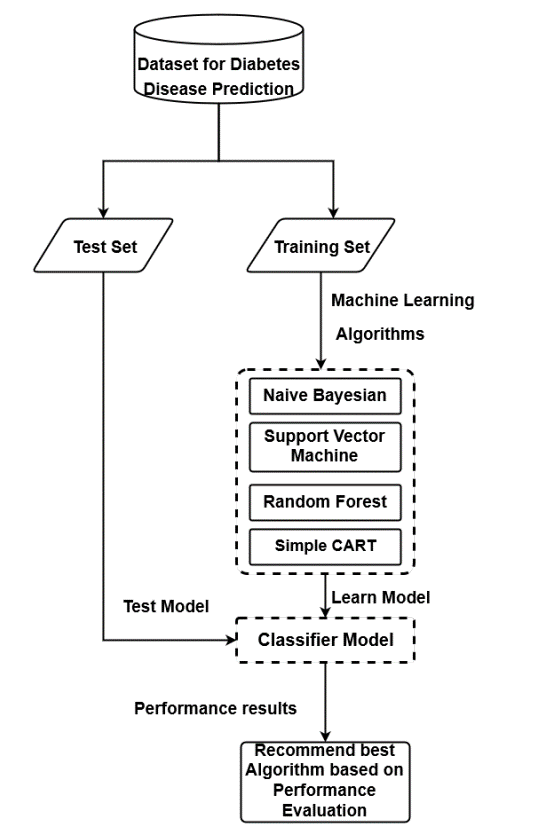
The name of the dataset that has been considered is Pima Indians Diabetes Database which is collected from National Institute of Diabetes and Digestive and Kidney Diseases. The total No. of Instances are 768 and the size is 37 KB. The total no. of attributes is 9 including the target class attribute. The name of two target classes is tested positive and tested negative. The no. of instances for tested positive are 268 and the no. of instances for tested negative are 500. The data pre-processing is automatically performed by WEKA tool.

**Table.2.3.3.1. Dataset Attributes**

****

In this model specifically considered diabetes disease and takes input of the dataset is processed using four machine learning algorithms that are Naive Bayes, SVM, Random Forest, Simple CART and for each algorithm respective classifier model is trained and tested and the results are gathered. Based on the experimental results the best performing algorithm can be determined which will help in accurate prediction of the disease.

Flow chart:



**Figure.2.3.3.2. Flow chart**

**Procedure:**

* Step 1: - Preprocess the input dataset for diabetes disease in WEKA tool.
* Step 2: - Perform percentage split of 70% to divide dataset as Training set and Test set.
* Step 3: - Select the machine learning algorithm i.e., Naive Bayes, Support Vector Machine, Random Forest and Simple CART algorithm.
* Step 4: - Build the classifier model for the mentioned machine learning algorithm based on training set.
* Step 5: - Test the Classifier model for the mentioned machine learning algorithm based on test set.
* Step 6: - Perform Comparison Evaluation of the experimental performance results obtained for each classifier.
* Step 7: - After analyzing based on various measure conclude the best performing algorithm.

**Performance metrics:**

**Classification Accuracy Results:**

The following table represents the experimental classification accuracy results of Naive Bayes, Support Vector Machine, Random Forest and Simple CART algorithm. The table displays the Training time, testing time and Accuracy value of each algorithm.

**Table.2.3.3.3. Performance metrics**

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Training Time** | **Testing Time** | **Accuracy Value** |
| Naïve Bayes | 0.03 sec | 0.02 sec | 0.77 |
| Support Vector Machine | 0.14 sec | 0.03 sec | 0.7913 |
| Random Forest | 0.67 sec | 0.06 sec | 0.765 |
| Simple Cart | 1.38 sec | 0.02 sec | 0.765 |

**Table.2.3.3.4. Major Accuracy Measure Values**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **TP Rate** | **FP Rate** | **Precision** | **Recall** | **F-Measure** |
| Naïve Bayes | 0.770 | 0.317 | 0.767 | 0.770 | 0.768 |
| Support Vector Machine | 0.791 | 0.345 | 0.784 | 0.791 | 0.782 |
| Random Forest | 0.765 | 0.326 | 0.756 | 0.765 | 0.758 |
| Simple Cart | 0.765 | 0.364 | 0.762 | 0.763 | 0.446 |

**CHAPTER 3**

**PROPOSED SYSTEM**

**CHAPTER 3**

**PROPOSED SYSTEM**

**3.1. Objective of Proposed Model**

The primary aim of this project is to analyze the Diabetes Dataset and use Decision Tree, Random Forest, K-Nearest Neighbors, Logistic regression, XGBoost algorithms for prediction and to develop a prediction engine. The secondary aim is to develop a web application with following feature. Allow users to predict diabetes utilizing the prediction engine. The objective is set to achieve the aims of the project through research on statistical models in machine learning and to understand how the algorithms works.

**3.2. Algorithms Used for Proposed Model**

**Decision Tree Classifier:**

* 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 on the basis of 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, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
* In order 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:



**Figure.3.2.1. Decision Tree**

**Why use Decision Trees?**

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

* Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
* The logic behind the decision tree can be easily understood because it shows a tree-like structure.

## Decision Tree Terminologies

* + - * **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
      * **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
      * **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
      * **Branch/Sub Tree:** A tree formed by splitting the tree.
      * **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
      * **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

**How does the Decision Tree algorithm Work?**

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

* **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
* **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
* **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
* **Step-4:** Generate the decision tree node, which contains the best attribute.
* **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



**Fig.3.2.2. Decision Tree example**

**Example:** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:

**K-Nearest Neighbors (KNN)**

* K-Nearest Neighbor 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.



**Figure.3.2.3. KNN algorithm example**

**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 x1, 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:



**Figure.3.2.4. KNN algorithm graphs(a)**

**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:

**Figure.3.2.5. KNN algorithm graphs(b)**

* 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:
* 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:
* 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:

* 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.

## Advantages of KNN Algorithm:

* It is simple to implement.
* It is robust to the noisy training data
* It can be more effective if the training data is large.

**Disadvantages of KNN Algorithm:**

* Always needs to determine the value of K which may be complex some time.
* The computation cost is high because of calculating the distance between the data points for all the training samples.

**Logistic Regression Classifier**

* Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
* Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1.**
* Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems.**
* In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
* The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
* Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



**Figure.3.2.6. Logistic regression algorithm graph**

**Logistic Function (Sigmoid Function):**

* The sigmoid function is a mathematical function used to map the predicted values to probabilities.
* It maps any real value into another value within a range of 0 and 1.
* The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
* In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

* The dependent variable must be categorical in nature.
* The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

Logistic Regression in Machine Learning

* In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

Logistic Regression in Machine Learning

* But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

Logistic Regression in Machine Learning

The above equation is the final equation for Logistic Regression.

**Type of Logistic Regression:**

On the basis of the categories, Logistic Regression can be classified into three types:

* **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
* **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
* **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

**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.**

The below diagram explains the working of the Random Forest algorithm:



**Figure.3.2.7. Random Forest algorithm**

**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.

Why use Random Forest?

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.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

**Step-4:** Repeat Step 1 & 2.

**Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

**Example:** Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random Forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



**Figure.3.2.8. Random Forest algorithm example**

## Applications of Random Forest

There are mainly four sectors where Random Forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

* Random Forest is capable of performing both Classification and Regression tasks.
* It is capable of handling large datasets with high dimensionality.
* It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

* Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

**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:



**Figure.3.2.9. Support Vector Machine Algorithm**

**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:



**Figure.3.2.10. Support Vector Machine Algorithm example**

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 x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:

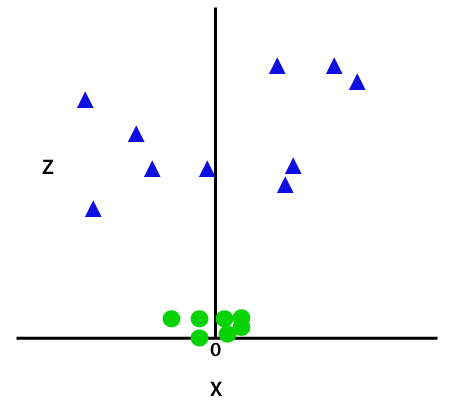


**Figure.3.2.11. Support Vector Machine Algorithm graphs(a)**

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**.

**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:

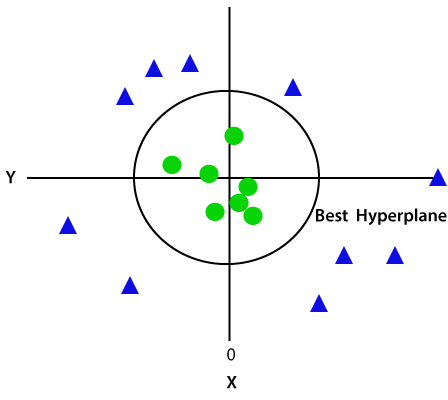
**Figure.3.2.12. Support Vector Machine Algorithm graphs(b)**

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=x2 +y2

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:

**Figure.3.2.13. Support Vector Machine Algorithm graphs(c)**

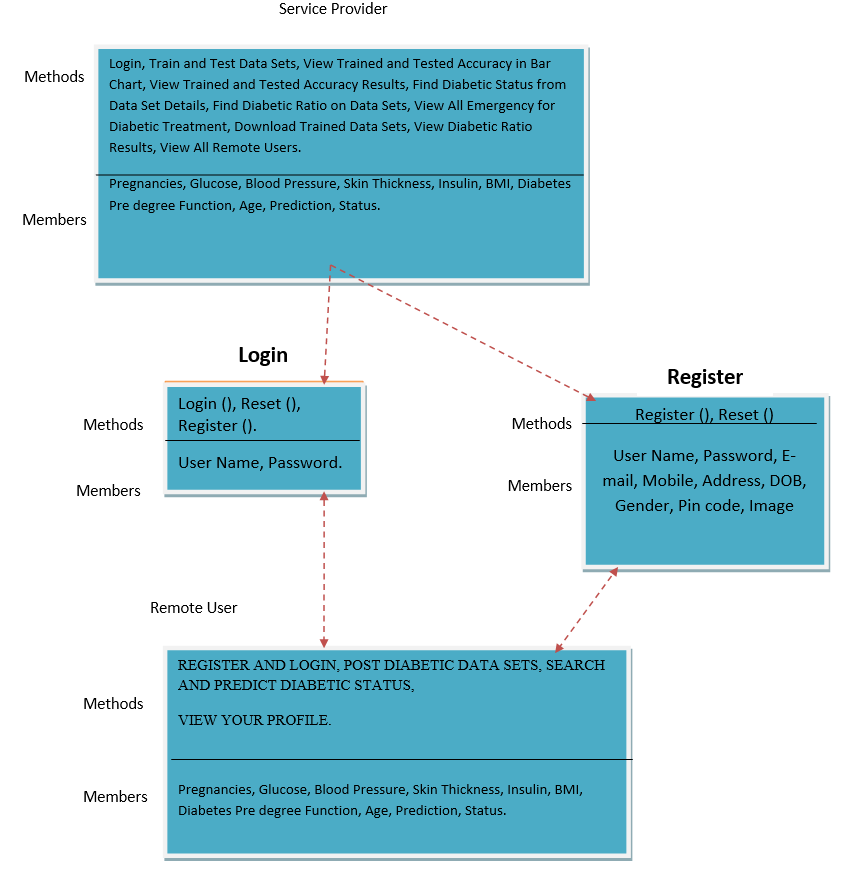
Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

Hence, we get a circumference of radius 1 in case of non-linear data.

**3.3. Designing**

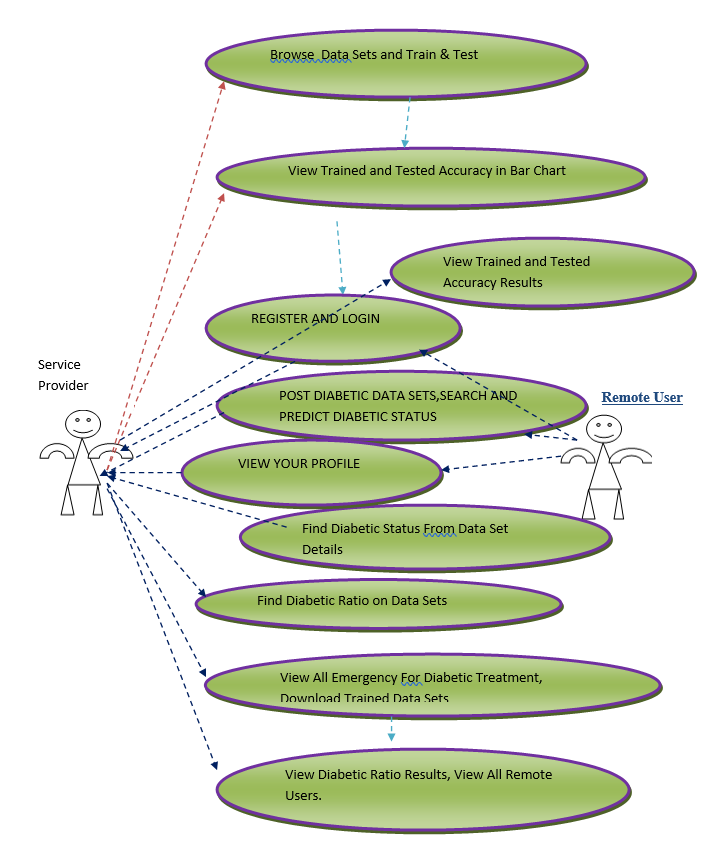
**3.3.1. UML Diagram**

**Class Diagram:**

****

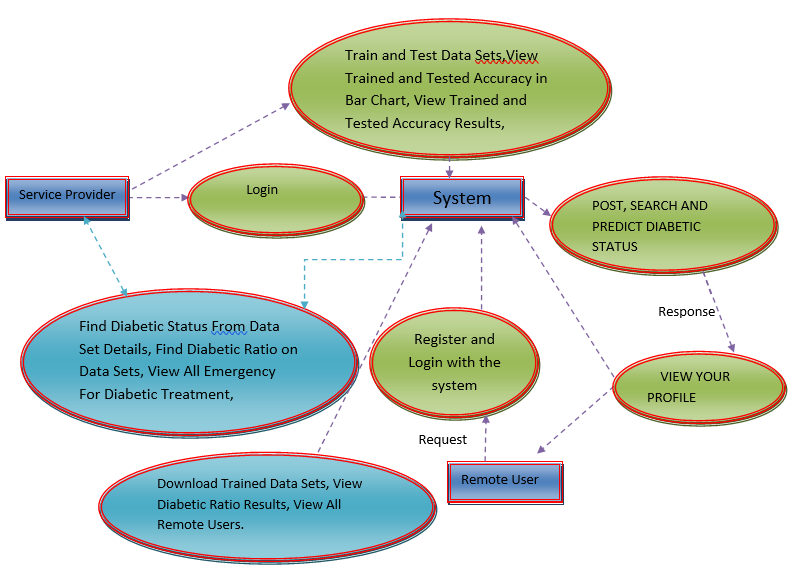
**Figure.3.3.1.1. Class Diagram**

**Use case Diagram:**

****

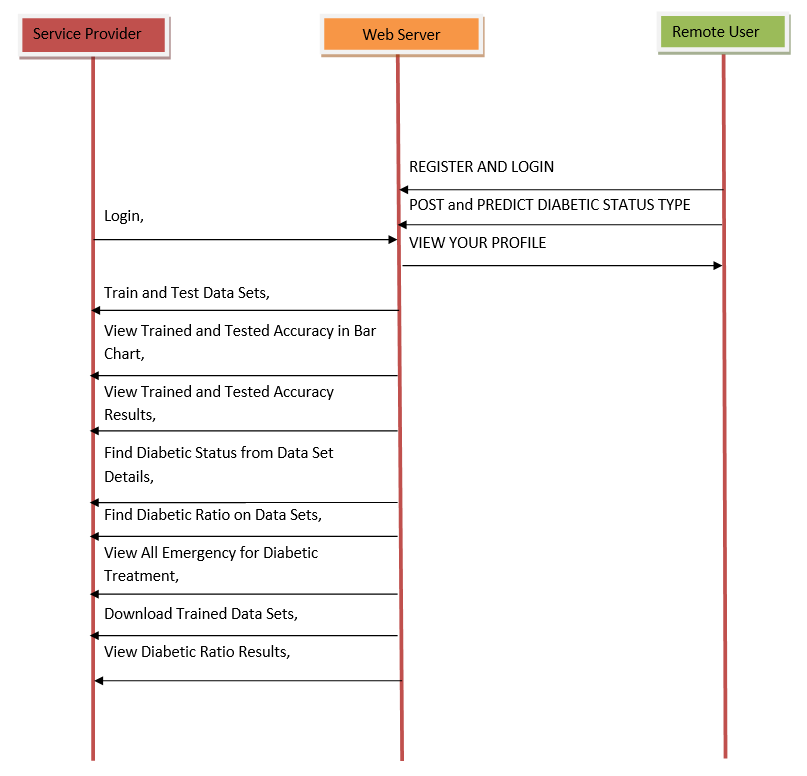
**Figure.3.3.1.2. Use Case Diagram**

**Data Flow Diagram:**

****

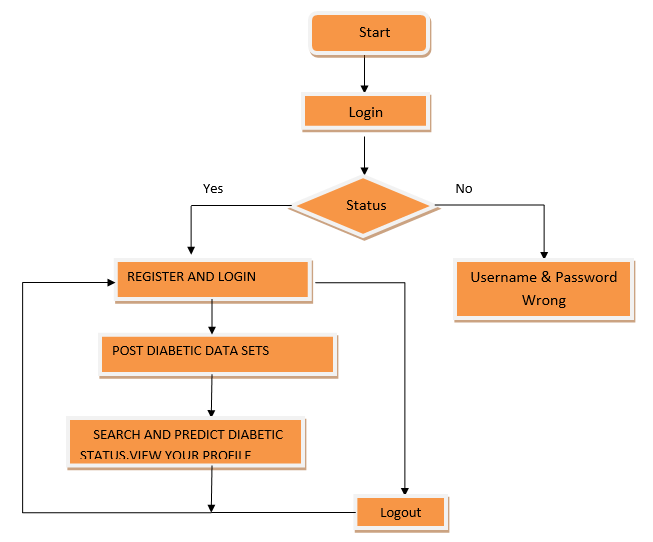
**Figure.3.3.1.3. Data Flow Diagram**

**Sequence Diagram:**

****

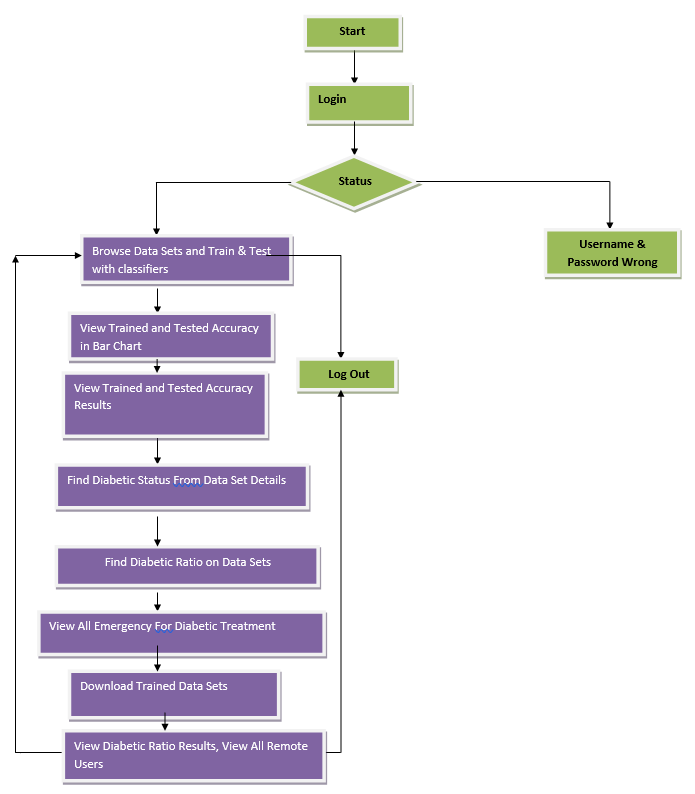
**Figure.3.3.1.4. Sequence Diagram**

**Flow Chart: Remote User**

****

**Figure.3.3.1.5. Flow Chart: Remote User**

**Flow Chart: Service Provider**

****

**Figure.3.3.1.6. Flow Chart: Service Provider**

**3.4. Stepwise implementation and Code**

**Importing Libraries**

3.3.Stepwise implementation and Code

Importing Libaries

#!pip install numpy==1.16.4

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib.pyplot import figure

from sklearn.metrics import confusion\_matrix

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

import pickle

import sklearn

import scipy

import seaborn as sns

sns.set()

Data

data = pd.read\_csv(‘diabetes.csv')

data.head()

data.shape

(768, 9)

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

Pregnancies 768 non-null int64

Glucose 768 non-null int64

BloodPressure 768 non-null int64

SkinThickness 768 non-null int64

Insulin 768 non-null int64

BMI 768 non-null float64

DiabetesPedigreeFunction 768 non-null float64

Age 768 non-null int64

Outcome 768 non-null int64

dtypes: float64(2), int64(7)

memory usage: 54.1 KB

EDA

data.describe().T

Histogram Plot

data\_feature = data.columns

for feature in data\_feature:

p = sns.distplot(a = data[feature])

plt.show()

Removal of Zeros

Since there are many zeros in data and values of 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI' cannot be zero, Therefore, Converriting Zeros into NaN value

data\_zeros = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

data[data\_zeros] = np.where((data[data\_zeros] == 0), np.nan, data[data\_zeros])

data.isnull().sum()

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

# for feature in data\_feature:

# plt.hist(data[feature])

# plt.show()

p = data.hist(figsize = (20,20))

Handling Missing Values

Filling NaN values with suitable mean and median values

data.describe().T

data['Glucose'] = data['Glucose'].fillna(data['Glucose'].mean())

# data.isnull().sum()

data['BloodPressure'] = data['BloodPressure'].fillna(data['BloodPressure'].mean())

# data.isnull().sum()

sns.boxplot(y = 'SkinThickness', data = data)

<matplotlib.axes.\_subplots.AxesSubplot at 0x14221e5f9e8>

data['SkinThickness'].mean(), data['SkinThickness'].median()

(29.153419593345657, 29.0)

data['SkinThickness'] = data['SkinThickness'].fillna(data['SkinThickness'].median())

# data.isnull().sum()

data['Insulin'].mean(), data['Insulin'].median()

(155.5482233502538, 125.0)

data['Insulin'] = data['Insulin'].fillna(data['Insulin'].median())

# data.isnull().sum()

data['BMI'].mean(), data['BMI'].median()

(32.45746367239099, 32.3)

data['BMI'] = data['BMI'].fillna(data['BMI'].median())

# data.isnull().sum()

for i in range(9):

print(data.columns[i])

Pregnancies

Glucose

BloodPressure

SkinThickness

Insulin

BMI

DiabetesPedigreeFunction

Age

Outcome

# for feature in data.columns:

# plt.hist(data[feature])

# plt.title(feature)

# plt.show()

p = data.hist(figsize = (20,20))

Pair Plot to see Distribution of all data at a time and dependencies

sns.pairplot(data =data, hue = 'Outcome')

plt.show()

FAC1 = 2\*(np.pi\*bw/RANGE)\*\*2

Heat Map

Gives Relation of different attribute with each other

plt.figure(figsize=(12,10))

sns.heatmap(data.corr(), annot = True, cmap = "YlGnBu")

plt.show()

from scipy import stats

for feature in data.columns:

stats.probplot(data[feature], plot = plt)

plt.title(feature)

plt.show()

Standardizing Data

from sklearn.preprocessing import StandardScaler

scale = StandardScaler()

data.head()

X = data.iloc[:, :-1]

y = data.iloc[:, -1]

X.head()

y.head()

0 1

1 0

2 1

3 0

4 1

Name: Outcome, dtype: int64

# X[:] = scale.fit\_transform(X[:])

[34]

X.head()

Splitting data into train and test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

Fitting data in various models

def svm\_classifier(X\_train, X\_test, y\_train, y\_test):

classifier\_svm = SVC(kernel = 'rbf', random\_state = 0)

classifier\_svm.fit(X\_train, y\_train)

y\_pred = classifier\_svm.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

return print(f"Train score : {classifier\_svm.score(X\_train, y\_train)}\nTest score : {classifier\_svm.score(X\_test, y\_test)}")

# print("-"\*100)

# print(cm)

def knn\_classifier(X\_train, X\_test, y\_train, y\_test):

classifier\_knn = KNeighborsClassifier(metric = 'minkowski', p = 2)

classifier\_knn.fit(X\_train, y\_train)

y\_pred = classifier\_knn.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

return print(f"Train score : {classifier\_knn.score(X\_train, y\_train)}\nTest score : {classifier\_knn.score(X\_test, y\_test)}")

# print("-"\*100)

# print(cm)

def naive\_classifier(X\_train, X\_test, y\_train, y\_test):

classifier\_naive = GaussianNB()

classifier\_naive.fit(X\_train, y\_train)

y\_pred = classifier\_naive.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

return print(f"Train score : {classifier\_naive.score(X\_train, y\_train)}\nTest score : {classifier\_naive.score(X\_test, y\_test)}")

# print("-"\*100)

# print(cm)

def tree\_classifier(X\_train, X\_test, y\_train, y\_test):

classifier\_tree = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

classifier\_tree.fit(X\_train, y\_train)

y\_pred = classifier\_tree.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

return print(f"Train score : {classifier\_tree.score(X\_train, y\_train)}\nTest score : {classifier\_tree.score(X\_test, y\_test)}")

# print("-"\*100) # print(cm)

def forest\_classifier(X\_train, X\_test, y\_train, y\_test):

classifier\_forest = RandomForestClassifier(criterion = 'entropy', random\_state = 0)

classifier\_forest.fit(X\_train, y\_train)

y\_pred = classifier\_forest.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

return print(f"Train score : {classifier\_forest.score(X\_train, y\_train)}\nTest score : {classifier\_forest.score(X\_test, y\_test)}")

# print("-"\*100) # print(cm)

def print\_score(X\_train, X\_test, y\_train, y\_test):

print("SVM:\n")

svm\_classifier(X\_train, X\_test, y\_train, y\_test)

print("-"\*100)

print()

print("KNN:\n")

knn\_classifier(X\_train, X\_test, y\_train, y\_test)

print("-"\*100)

print()

print("Naive:\n")

naive\_classifier(X\_train, X\_test, y\_train, y\_test)

print("-"\*100)

print()

print("Decision Tree:\n")

tree\_classifier(X\_train, X\_test, y\_train, y\_test)

print("-"\*100)

print()

print("Random Forest:\n")

forest\_classifier(X\_train, X\_test, y\_train, y\_test)

print\_score(X\_train, X\_test, y\_train, y\_test)

SVM:

Train score : 1.0

Test score : 0.6948051948051948

----------------------------------------------------------------------------------------------------

KNN:

Train score : 0.8013029315960912

Test score : 0.7662337662337663

----------------------------------------------------------------------------------------------------

Naive:

Train score : 0.745928338762215

Test score : 0.7857142857142857

----------------------------------------------------------------------------------------------------

Decision Tree:

Train score : 1.0

Test score : 0.6883116883116883

----------------------------------------------------------------------------------------------------

Random Forest:

Train score : 0.988599348534202

Test score : 0.7792207792207793

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Performance Metrics

[43]

classifier\_forest = RandomForestClassifier(criterion = 'entropy')

classifier\_forest.fit(X\_train, y\_train)

y\_pred = classifier\_forest.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

cm

array([[94, 13],

[23, 24]], dtype=int64)

# classifier\_svm = SVC(kernel = 'rbf', random\_state = 0, probability=True)

# classifier\_svm.fit(X\_train, y\_train)

# y\_pred = classifier\_svm.predict(X\_test)

# cm = confusion\_matrix(y\_test, y\_pred)

# cm

pd.crosstab(y\_test, y\_pred, rownames=['True'], colnames=['Predicted'], margins=True)

Checking data is balanced or not

data['Outcome'].value\_counts()

0 500

1 268

Name: Outcome, dtype: int64

Classification Report (Accuracy, Precision, Recall, F1-score)

from sklearn.metrics import roc\_auc\_score, roc\_curve, classification\_report

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.80 0.88 0.84 107

1 0.65 0.51 0.57 47

avg / total 0.76 0.77 0.76 154

Getting probability instead of A/B test

y\_pred\_prob = classifier\_forest.predict\_proba(X\_test)[:,1]

y\_pred\_prob

array([1. , 0.2, 0. , 0.3, 0.1, 0. , 1. , 0.7, 0.6, 0.5, 0.8, 0.8, 0.1,

0.1, 0.1, 0.6, 0.8, 0. , 0.9, 0.1, 0.6, 0.1, 0. , 0.3, 0. , 0.2,

0. , 0.8, 0. , 0.1, 0.4, 0.3, 0.1, 0.4, 0. , 0.9, 0.4, 0. , 0.2,

1. , 0.1, 0.2, 0.1, 1. , 0.6, 0.2, 0. , 0.2, 0.5, 0.1, 0.4, 0.2,

0.8, 0.4, 0. , 0. , 0. , 0.4, 0.2, 0.5, 0.9, 0.5, 0. , 0.4, 1. ,

0.7, 0.6, 0.4, 0.5, 0.1, 0.1, 0.4, 0.1, 0.7, 1. , 0.5, 0.4, 0.5,

0.2, 0.1, 0.5, 0.2, 0.3, 0. , 0. , 0. , 0. , 0.4, 1. , 0.2, 0.5,

0.1, 0.2, 0. , 0.6, 0. , 0.1, 0.2, 0.3, 0.4, 0.3, 0. , 0.2, 0. ,

0.5, 0.6, 0. , 0.5, 0. , 0.7, 0. , 0.6, 0.5, 0.4, 0.7, 0.5, 0.3,

0.5, 0. , 0.9, 0.7, 0.8, 0.2, 0.3, 0. , 0. , 0.4, 0.2, 0.5, 0.6,

0.6, 0.6, 0. , 0.5, 0. , 0.8, 0.4, 0.5, 0.3, 0.1, 0. , 0.6, 0. ,

0.2, 0.7, 0. , 0. , 0.1, 0.1, 0.3, 0. , 0.4, 0. , 0.3])

Evaluating FPR, TPR, Threshold

fpr, tpr, threshold = roc\_curve(y\_test, y\_pred\_prob)

print("FPR:\n\n", fpr)

print("-"\*100)

print("TPR:\n\n", tpr)

FPR:

[0. 0.00934579 0.01869159 0.05607477 0.12149533 0.22429907

0.31775701 0.38317757 0.51401869 0.6728972 1. ]

----------------------------------------------------------------------------------------------------

TPR:

[0. 0.12765957 0.19148936 0.40425532 0.5106383 0.63829787

0.76595745 0.85106383 0.91489362 0.9787234 1. ]

Plotting ROC Curve

plt.plot([0, 1], [0, 1], "k--", label = '50% AUC')

plt.plot(fpr, tpr, label = "Random Forest")

plt.xlabel("FPR")

plt.ylabel("TPR")

plt.title("ROC Curve - Random Forest")

plt.show()

ROC Score

roc\_auc\_score(y\_test,y\_pred\_prob)

0.8103002585006959

Hyperparameter Tunning

from sklearn.model\_selection import cross\_val\_score

accuracies = cross\_val\_score(estimator = classifier\_forest, X = X\_train, y = y\_train, cv = 10)

print(accuracies.mean(), accuracies.std())

0.7476077157462667 0.053927192330761375

from sklearn.model\_selection import GridSearchCV

parameters = {

'n\_estimators': [25, 50, 200, 300],

'criterion': ['gini', 'entropy'],

'max\_depth': [14, 20, 25, 30] }

grid\_search = GridSearchCV(estimator = classifier\_forest,

param\_grid = parameters,

scoring = 'accuracy',

cv = 10,

n\_jobs = -1)

grid\_search = grid\_search.fit(X\_train, y\_train)

print('best\_accuracy = ',grid\_search.best\_score\_)

print('best\_parameters = ', grid\_search.best\_params\_)

best\_accuracy = 0.760586319218241

best\_parameters = {'criterion': 'gini', 'max\_depth': 20, 'n\_estimators': 300}

classifier\_forest = RandomForestClassifier(criterion = 'gini', max\_depth = 25, n\_estimators = 200, random\_state = 0)

classifier\_forest.fit(X\_train, y\_train)

y\_pred = classifier\_forest.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

cm

array([[94, 13],

[13, 34]], dtype=int64)

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.88 0.88 0.88 107

1 0.72 0.72 0.72 47

avg / total 0.83 0.83 0.83 154

Saving model using pickle

filename = 'diabetes\_model.pkl'

pickle.dump(classifier\_forest, open(filename, 'wb'))

model = open('diabetes\_model.pkl','rb')

forest = pickle.load(model)

y\_pred = forest.predict(X\_test)

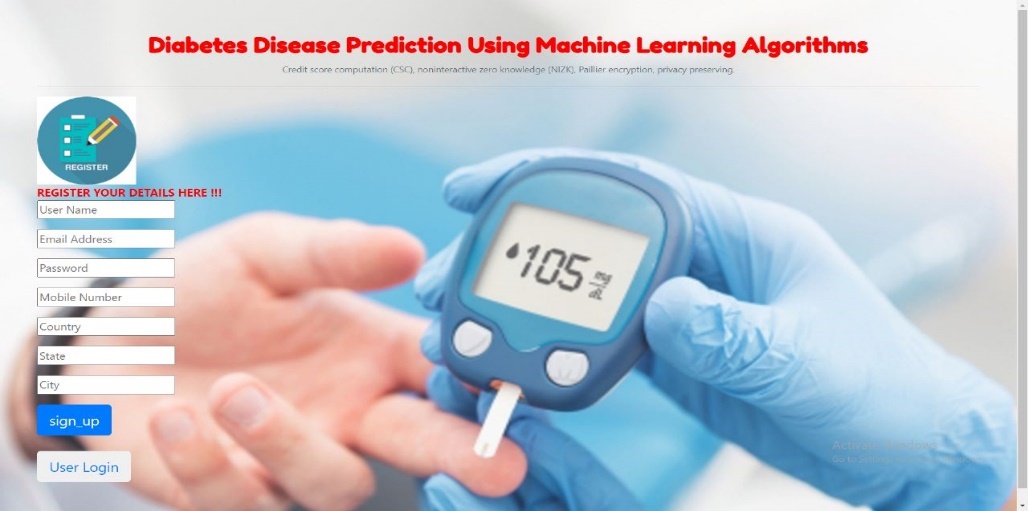
confusion\_matrix(y\_test, y\_pred)

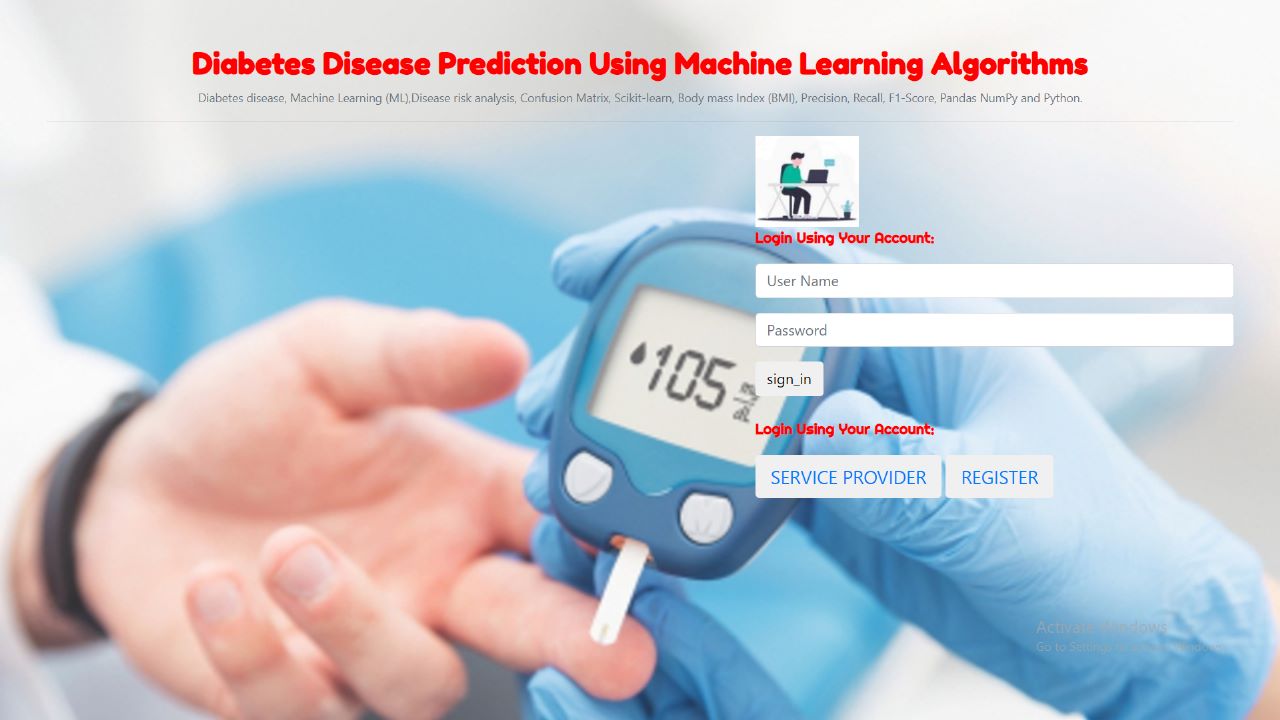
array([[94, 13],

[13, 34]], dtype=int64)

**CHAPTER 4**

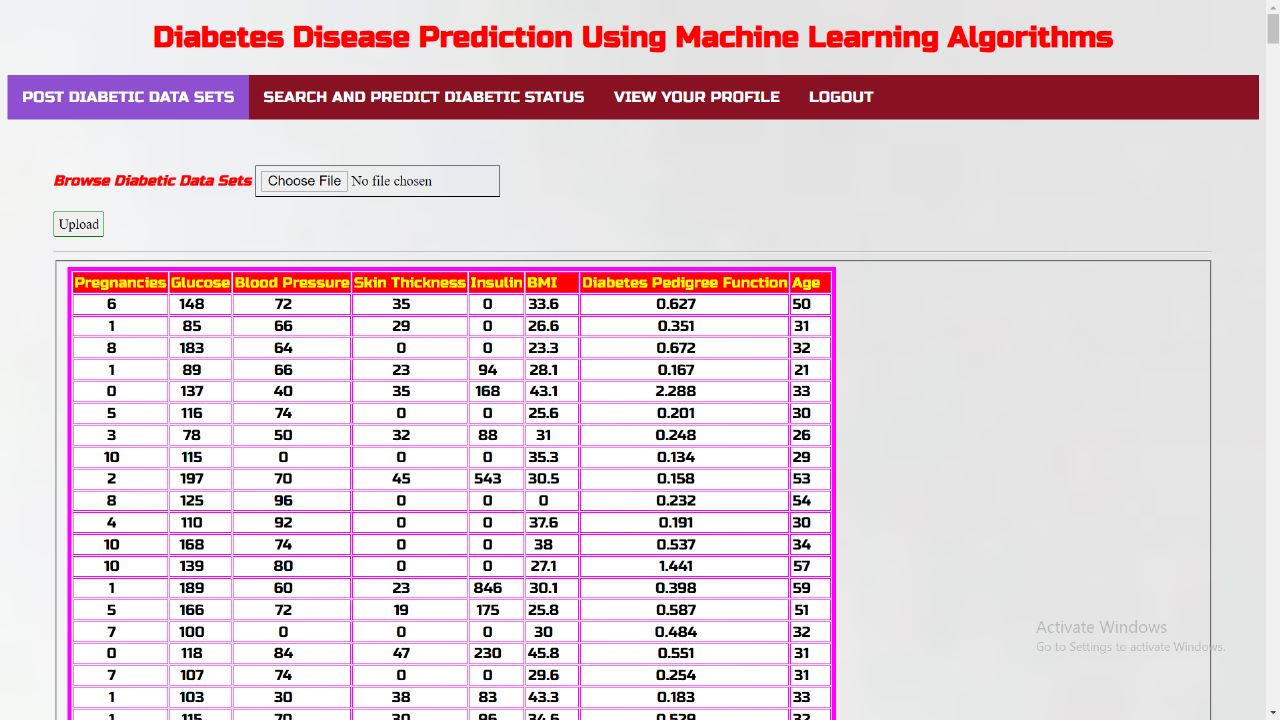
**RESULTS AND DISCUSSION**

**4.1. RESULTS AND DISCUSSION**

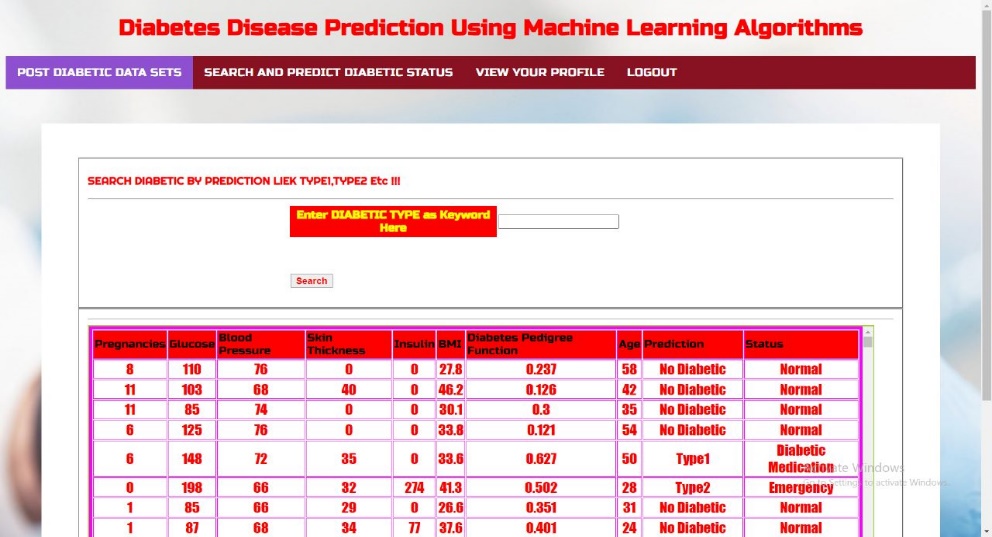


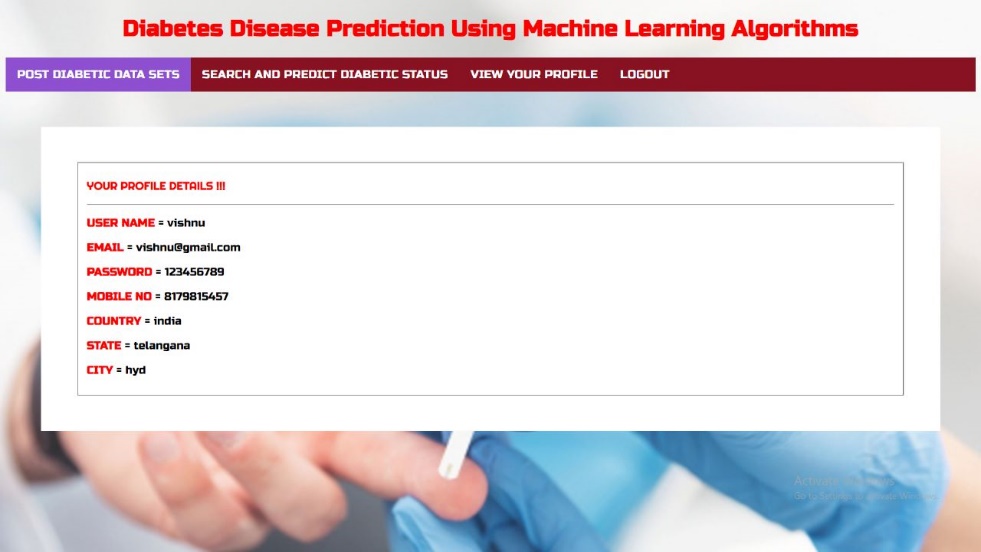
**Figure.4.1.1. Random user registration page**

**Figure.4.1.2. Random User login page**

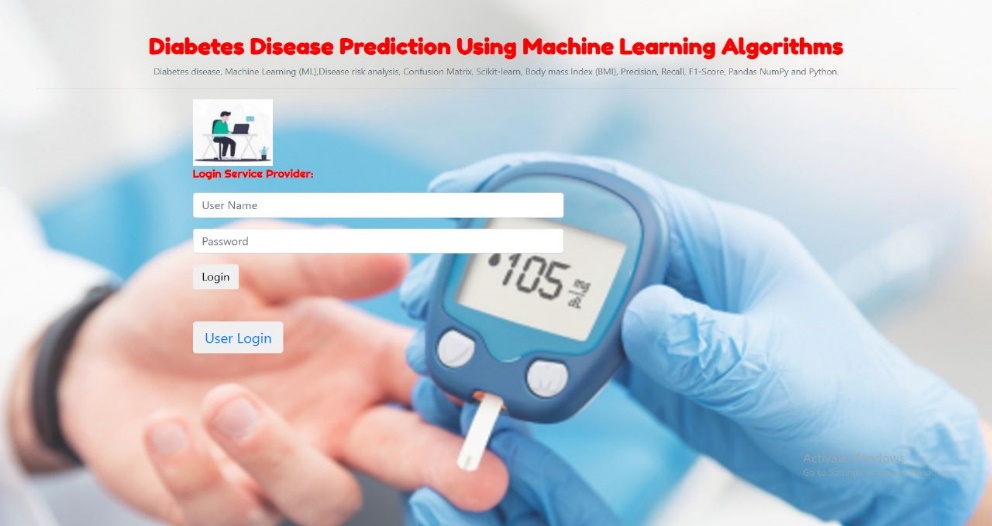


**Figure.4.1.3. Upload Diabetes Dataset page**



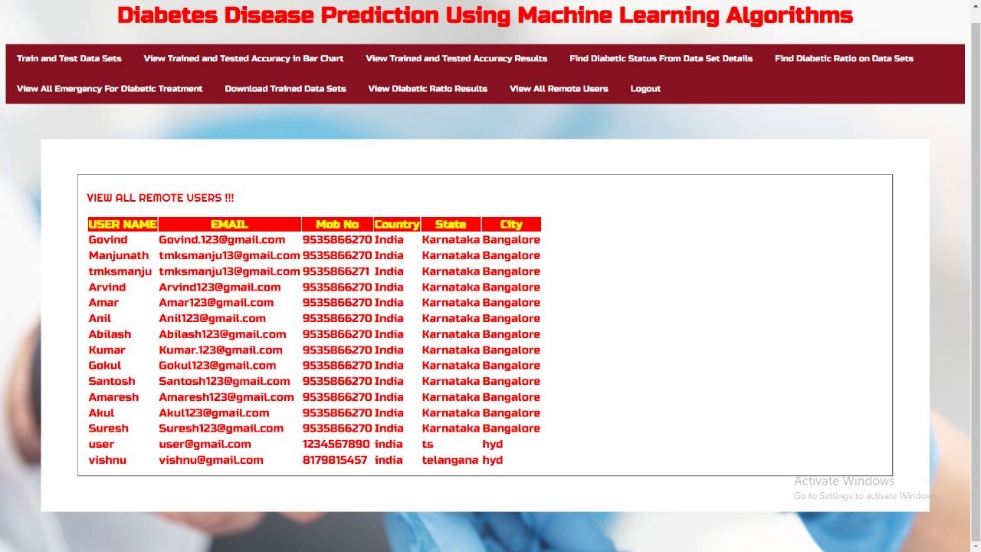


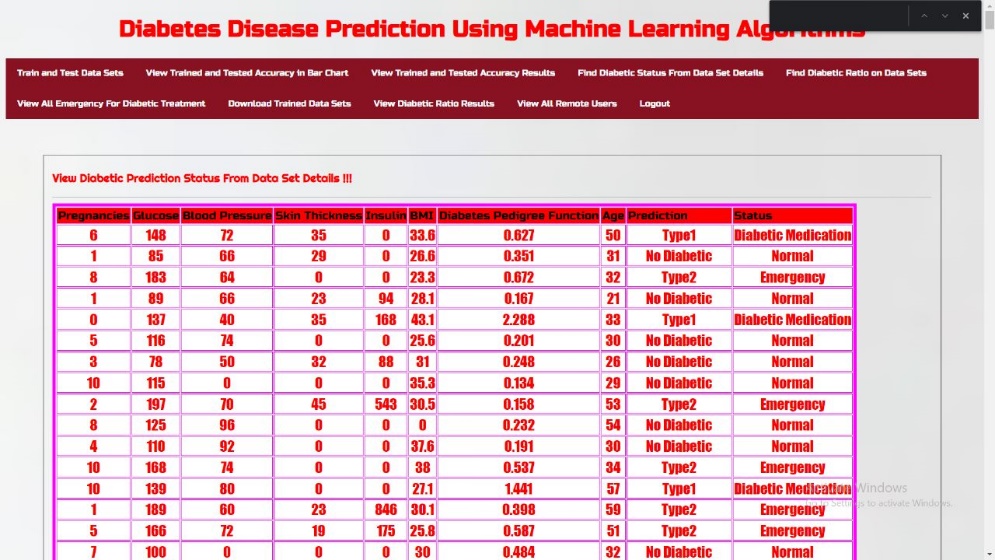
**Figure.4.1.4. Search and Predict Diabetes Page**



**Figure.4.1.5. View User Profile Page**

**Figure.4.1.6. Service provider login page**





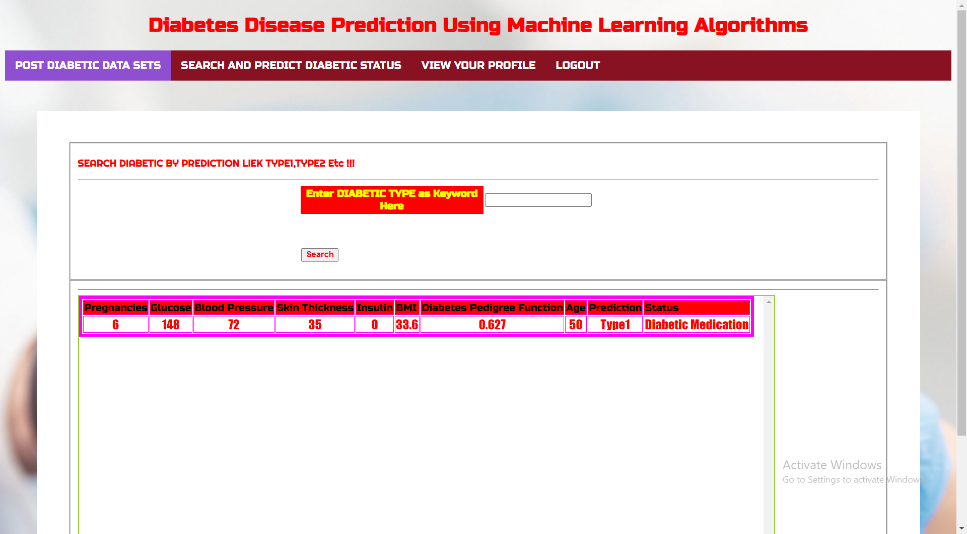
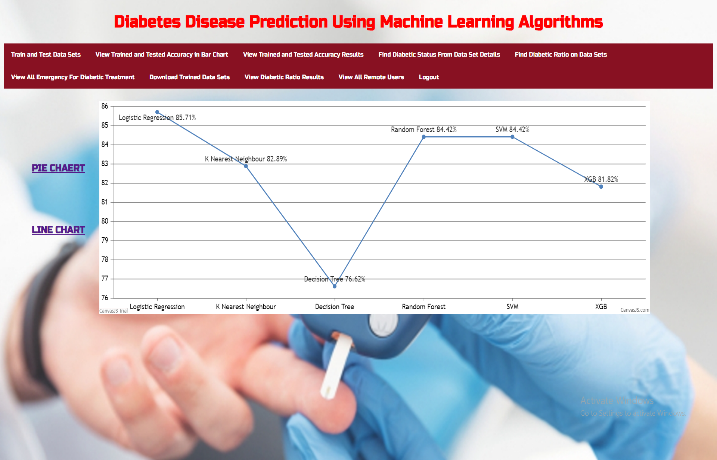
**Figure.4.1.7. View all remote users page**



**Figure.4.1.8. View trained and tested accuracy in bar chart page**

­­--

**Figure.4.1.9. Find diabetic ratio on datasets page**

****

**Figure.4.1.10. Pie chart and Line chart page**

**Figure.4.1.11. Accuracy of algorithms**

**Figure.4.1.12. Random User output**

* 1. **PERFORMANCE METRICS**

**Table.4.2.1. Performance Metrics**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classifier | P | R | F1 | A (10 - fold) | Accuracy |
| KNN | 0.76 | 0.73 | 0.75 | 0.76 | 82.89 |
| SVM | 0.73 | 0.74 | 0.73 | 0.75 | 84.42 |
| DT | 0.72 | 0.71 | 0.71 | 0.71 | 76.62 |
| RF | 0.70 | 0.71 | 0.71 | 0.71 | 84.42 |
| LR | 0.78 | 0.76 | 0.77 | 0.76 | 85.71 |
| XGB | 0.81 | 0.82 | 0.81 | 0.82 | 81.82 |

**CHAPTER 5**

**CONCLUSION**

**5.1 CONCLUSION AND FUTURE ENHANCEMENT**

Machine learning can help doctors identify and cure diabetes. In this report we have discussed

existing solutions for prediction of diabetes. We will conclude that improving the accuracy of the classification will help the machine learning models perform better. The performance analysis is in terms of accuracy rate among all the classification techniques such as K-nearest neighbors, Naive Bayes, Decision Tree Classifier, Support Vector Machine, random forest. Machine learning has the great ability to revolutionize the diabetes risk prediction with the help of advanced computational methods and availability of large amount of epidemiological and genetic diabetes risk dataset. The main aim is to design and implement diabetes prediction using machine learning methods and performance analysis of that method. The proposed method approach uses SVM, KNN, logistic regression, and random forest. The technique may also help researchers to develop an accurate and effective tool that will reach at the table of clinicians to help them make better decision about the disease status.

In future, we will try to create a diabetes dataset in collaboration with a hospital or a medical institute and will try to achieve better results. We will be incorporating more Machine Learning and Deep learning models for achieving better results as well.

**CHAPTER 6**

**REFERENCES**

## 6.1.REFERENCES

1. Z. Tafa, N. Pervetica, and B. Karahoda, “An Intelligent System for Diabetes Prediction,” IEEE Explore, in Proceedings of the 4th Mediterranean Conference on Embedded Computing (MECO), pp. 378–382, Budva, Montenegro, June 2015.
2. D. Sisodia and D. S. Sisodia, “Prediction of diabetes using classification algorithms,” Procedia Computer Science, vol. 132, pp. 1578 – 1585, 2018, international Conference on Computational Intelligence and Data Science.[Online].

Available: <http://www.sciencedirect.com/science/article/pii/S1877050918308548>

1. Mir and S. N. Dhage, “Diabetes disease prediction using machine learning on big data of healthcare,” in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1–6.
2. P. S. Kohli and S. Arora, “Application of machine learning in disease prediction,” in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 2018, pp. 1–4.
3. Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, St´efan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.
4. C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R’ıo, M. Wiebe, P. Peterson, P. G’erard-Marchant, K.Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2
5. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and ´ Edouard Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, p. 28252830, 2011.
6. J. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes,“Using the adap learning algorithm to forcast the onset of diabetesmellitus,” Proceedings - Annual Symposium on Computer Applicationsin Medical Care, vol. 10, 11 1988.

**6.2. GitHub Link:**

[**https://github.com/HarshithaGajendrula/Diabetes-Disease-Prediction-Using-Machine-Learning-Algorithms.github.io**](https://github.com/HarshithaGajendrula/Diabetes-Disease-Prediction-Using-Machine-Learning-Algorithms.github.io)