

“END TO END DIABETES PREDICTION SYSTEM”

A PROJECT REPORT

Submitted by

Modi Krushil A. (2021095900003707)

Patel Utkarsh S. (2021095900003681)

Patel Nisarg R. (2021095900003603)

In fulfilment of the subject Major Project (1ET1030801)

Of

B.E. Semester VIII

In

COMPUTER ENGINEERING & INFORMATION TECHNOLOGY



Sankalchand Patel University, Visnagar

May 2024



Sankalchand Patel College of Engineering, Visnagar

At & Post: Visnagar, Gujarat - 384315

CERTIFICATE

This is to certify that the project report submitted along with the project entitled “**END TO END DIABETES PREDICTION SYSTEM**” has been carried out by **Modi Krushil Akshaykumar(2021095900003707)** under my guidance in partial fulfillment of the subject Major Project (1ET1030801) in **Computer Engineering**, 8th Semester of Sankalchand Patel University, Visnagar during the academic year 2023-24.

Prof. Jayesh M Mevada
Internal Guide

Dr. Kirit Modi
Head of the Department



Sankalchand Patel College of Engineering, Visnagar

At & Post: Visnagar, Gujarat - 384315

CERTIFICATE

This is to certify that the project report submitted along with the project entitled “**END TO END DIABETES PREDICTION SYSTEM**” has been carried out by **Patel Utkarsh Sanjaybhai(2021095900003681)** under my guidance in partial fulfillment of the subject Major Project (1ET1030801) in **Computer Engineering**, 8th Semester of Sankalchand Patel University, Visnagar during the academic year 2023-24.

Prof. Jayesh M Mevada
Internal Guide

Dr. Kirit Modi
Head of the Department



Sankalchand Patel College of Engineering, Visnagar

At & Post: Visnagar, Gujarat - 384315

CERTIFICATE

This is to certify that the project report submitted along with the project entitled “**END TO END DIABETES PREDICTION SYSTEM**” has been carried out by **Patel Nisarg Rakeshkumar(2021095900003603)** under my guidance in partial fulfillment of the subject Major Project (1ET1030801) in **Computer Engineering**, 8th Semester of Sankalchand Patel University, Visnagar during the academic year 2023-24.

Prof. Jayesh M Mevada
Internal Guide

Dr. Kirit Modi
Head of the Department

ACKNOWLEDGMENT

With a sense of gratitude and respect, we would like to extend our heartiest thanks to all those who provided help and guidance in making this project a success. The successful completion of any task is accompanied by a deep emotion of fulfilment and satisfaction. It was a pleasant and highly educative experience throughout the development of the project.

Sincerely, we thank our Head of the Department of College **Dr. Kirit J. Modi**, who gave us the opportunity to undertake such kind of challenging and innovative work and our internal guide **Prof. Jayesh M Mevada** who gave us guidance, help and motivation throughout the entire project. Without Sir's support, these activities would have been tougher. We are grateful to our guide for the guidance and constructive suggestions that helped us in the preparation of this project.

- 1) Krushil Modi
- 2) Utkarsh Patel
- 3) Nisarg Patel

ABSTRACT

- Diabetes is one of the chronic diseases that causes blood sugar levels to rise. If diabetes is left untreated and undiagnosed, it can lead to complications. The time-consuming identification process leads to a patient's referral to a diagnostic Centre and consultation with a doctor. Predictive analytics in healthcare is a difficult challenge, but it can eventually assist physicians in making timely decisions about a patient's health and condition based on data. The emergence of machine learning methods solves this crucial issue.
- The aim of this project is to create a model that can reliably predict the accuracy of diabetes in patients. Dataset splits into three then classification techniques are implemented. Training Dataset, Dataset sample that is used to fit the model. Validation Dataset, Dataset sample that is used for hyper tuning the parameters, and comparing the accuracy and error rates of the model performance between using the training dataset and the validation dataset. Testing Dataset, Dataset sample that is used to test the model performance (predictive power).
- To detect diabetes at an early stage, this project employs machine learning classification algorithms:, SVM, Decision tree and Random Forest are implemented. The Pima Indians Diabetes Database (PIDD) is used in the experiments. The National Institute of Diabetes and Digestive and Kidney Diseases provided the results. The dataset's purpose is to diagnose whether a patient has diabetes using diagnostic measures included in the dataset. Various measures like Precision, Accuracy, Specificity, and Recall are measured over classified instances using Confusion Matrix.
- The accuracy of the algorithms used are compared and discussed. The study's comparison of the various machine learning techniques shows which algorithm is better suited for diabetes prediction. Using machine learning methods, this project aims to assist doctors and physicians in the early detection of diabetes

TABLE OF CONTENTS

Acknowledgement	3
Abstract	4
Chapter 1 Introduction	5
1.1 Introduction.....	5
1.2 Objectives	6
1.3 Motivation.....	6
1.4 Overview of the Project	7
1.5 Chapter wise Summary	7
Chapter 2 System Environment	8
2.1 Hardware configuration	8
2.2 Software configuration.....	8
2.3 Technology	8
2.3.1.1 Python	9
2.3.1.2 Machine Learning	10
2.3.1.3 Flask.....	11
Chapter 3 data analysis.....	9
3.1 Structure of Data	9
3.2 Parameters Implemented.....	11
3.3 Exploratory Data analysis	12
3.4 Histogram.....	15
3.5 Correlation	16
3.6 SNS Pair plot.....	17
Chapter 4 System design.....	21
4.1 Use case diagram	21
4.2 Activity diagram	22
4.3 Data flow diagram.....	24

Chapter 5 Implementation	25
5.1 Hypothesis Testing.....	25
5.2 Splitting of Dataset	26
5.3 Feature Scaling.....	27
5.4 Implementing Machine Learning Algorithms	28
5.4.1 Decision Tree Model.....	29
5.4.2 Random Forest Model.....	30
5.4. 3 Support Vector Machine Model.....	31
5.4. 4 Bagging Classifier.....	32
5.4. 5 Ada Boost Classifier	32
 Chapter 6 Test Results/ experiments/verification.....	 33
Testing	33
Comparative Analysis	35
Chapter 7 Snapshots	36
Screenshot	38
Chapter 8 Conclusions and Further Scope.....	39
Chapter 9 Reference	40

1. INTRODUCTION

1.1 INTRODUCTION:-

- Various classification strategies are used in the medical field for classifying data into different classes. Diabetes is a condition that affects the body's ability to produce the hormone insulin, which causes carbohydrate metabolism to become irregular and blood glucose levels to increase. High blood sugar is a common symptom of diabetes. If diabetes is not treated, it can lead to a variety of complications. Diabetic ketoacidosis and nonketotic hyperosmolar coma are two significant complications. Diabetes is considered a severe health problem in which the amount of sugar in the blood cannot be regulated. Diabetes is influenced by a variety of factors such as height, weight, genetic factors, and insulin, but the most important factor to remember is sugar concentration. The only way to avoid problems is to identify the problem early. This dataset comes from the 'National Institute of Diabetes and Digestive Diseases' Pima Indians Diabetes Database (PIDDD). Several constraints were taken from the massive database.
- The dataset is divided into three sections, after which classification techniques are used. The training dataset is a sample of the dataset that is used to match the model. Validation Dataset, a dataset sample used for fine-tuning parameters and comparing model output accuracy and error rates between the training and validation datasets. Testing Dataset is a sample of a dataset that is used to assess the model's output.

1.2 OBJECTIVES:-

- Since a decade, the number of people diagnosed with diabetes has risen significantly. The current human lifestyle is the primary cause of diabetes rise.
- Main objective of this project is to analyze the data, and see if it is possible to gleam any furthe information from the data to determine correlation between parameters and diabetes.
- The second is to attempt to get the best accuracy score using various supervised learning machine learning algorithms. To find out which algorithm is able to best predict whether a person has diabetes or not based on this dataset.
- The accuracy of the algorithms used are compared and discussed. The study's comparison of the various machine learning techniques shows which algorithm is better suited for diabetes prediction. Using machine learning methods, this project aims to assist doctors and physicians for predicting whether a person has diabetes or not.

1.3 MOTIVATION:-

The current human lifestyle is the primary cause of increasing diabetes. The three types of errors that may occur in today's medical diagnosis method:

1. The false-negative form, in which a patient is diabetic in fact but test results show that he or she does not have diabetes.
- 2.The false-positive type. In this type, a patient in reality is not a diabetic patient but test reports say that he/she is a diabetic patient.
3. The third type is an unclassifiable type in which a system cannot diagnose a given case. This happens because of insufficient knowledge extraction from past data, a given patient may get predicted in an unclassified type. However, in fact, the patient must predict whether he or she will be diabetic or non-diabetic. Such diagnostic errors can result in unnecessary treatments or no treatments at all when they are needed. To prevent or mitigate the magnitude of such an effect, a machine learning algorithm must be used to build a framework that provides reliable results while reducing human effort.

1.4 OVERVIEW OF PROJECT:-

- Machine learning has the great ability to revolutionize the diabetes risk prediction with the help of advanced computational methods and availability of a large amount of epidemiological and genetic diabetes risk dataset. Detection of diabetes in its early stages is the key for treatment. This work has described a machine learning approach to predicting diabetes or not. 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 decisions about disease status.

1.5 CHAPTERWISE SUMMARY:-

- The first chapter is an introductory chapter, which gives an overview of the project. It includes four divisions - introduction, objectives, motivation, overview and chapter wise summary. The second chapter is data analysis, where the dataset is analyzed and studied for further classifications. Third chapter deals with the different machine learning models used. To detect diabetes at an early stage, this project employs machine learning classification algorithms: SVM, Decision tree and Random Forest tree are implemented. The last chapter gives an elaborate idea about the results of different models. Let's get to know more about the dataset in the upcoming chapter.

2. SYSTEM ENVIRONMENT

2.1 HARDWARE CONFIGURATION

1. Pentium IV Processor
2. 4 GB RAM
3. 40GB HDD
4. 1024*768 Resolution colour Monitor

Note: This is not the “System Requirements”.

2.2 SOFTWARE CONFIGURATION

1. OS: Windows 7 OR More
2. Vs code - 1.88
3. Terminal - 1.19.10821.0

2.3 Technology

- Python 3.6 & related libraries
- Machine Learning
- Flask 3.0.2
- Html 5 & CSS 3

2.3.1.1 Python

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability and a syntax that allows programmers to express concepts in fewer lines of code compared to languages like C++ or Java.

Key features of Python include:

1. Simple and Easy to Learn: Python's syntax is straightforward and easy to understand, making it an ideal language for beginners. Its readability reduces the cost of program maintenance and development.
2. Interpreted: Python code is executed line by line by the Python interpreter, which means you don't need to compile your code before running it. This makes development faster and more interactive.
3. High-level Language: Python abstracts many complex programming concepts, allowing developers to focus more on solving problems rather than worrying about low-level details.

4. **Dynamic Typing:** Python is dynamically typed, meaning you don't need to specify the data type of variables explicitly. This feature simplifies coding and makes the language flexible.
5. **Rich Standard Library:** Python comes with a vast standard library that provides modules and functions for various tasks, such as file I/O, networking, web development, and more. This extensive library reduces the need for external dependencies in many cases.
6. **Cross-platform:** Python is available on multiple platforms, including Windows, macOS, and various Unix-based operating systems, making it highly portable.
7. **Community Support:** Python has a large and active community of developers who contribute to its growth. This community provides libraries, frameworks, and resources to extend Python's capabilities.

Usage

Python is a versatile programming language with a wide range of applications across various domains. Some of the common uses of Python include:

1. **Web Development:** Python is widely used for web development, both on the server-side and client-side. Frameworks like Django, Flask, and Pyramid are popular choices for building robust web applications and APIs.
2. **Data Science and Machine Learning:** Python is the preferred language for data science, machine learning, and artificial intelligence (AI) projects. Libraries like NumPy, Pandas, Matplotlib, scikit-learn, TensorFlow, and PyTorch provide powerful tools for data analysis, visualization, and building machine learning models.
3. **Automation and Scripting:** Python is excellent for automating repetitive tasks and writing scripts for various purposes. It's commonly used for system administration, file manipulation, web scraping, and task scheduling.
4. **Scientific Computing:** Python is widely used in scientific computing and computational science for numerical simulations, data analysis, and visualization. Libraries like SciPy and SymPy provide tools for scientific computing, while Jupyter Notebook is popular for interactive computing and data exploration.
5. **Game Development:** Python is used in game development, both for creating games from scratch and for scripting game engines like Unity and Godot. Libraries like Pygame provide tools for developing 2D games, while Panda3D and Pyglet are used for 3D game development.
6. **Desktop GUI Applications:** Python can be used to develop desktop graphical user interface (GUI) applications using libraries like Tkinter, PyQt, and wxPython. These libraries provide tools for creating windows, buttons, menus, and other GUI components.
7. **Web Scraping:** Python's simplicity and powerful libraries like BeautifulSoup and Scrapy make it an excellent choice for web scraping tasks. Developers use Python to extract data from websites and APIs for various purposes, such as data collection, market research, and content aggregation.

2.3.1.2 Machine Learning

Machine learning (ML) is a subset of artificial intelligence (AI) that focuses on the development of algorithms and models that allow computers to learn from and make predictions or decisions based on data, without being explicitly programmed to perform specific tasks. In Python, machine learning is facilitated by various libraries and frameworks that provide tools for building, training, and deploying machine learning models.

1. **Data Preparation:** Machine learning models require data to learn patterns and make predictions. Python offers libraries like NumPy, Pandas, and scikit-learn to load, preprocess, and manipulate datasets. These libraries provide functions for tasks such as data cleaning, feature engineering, and data splitting.
2. **Choosing a Model:** Python provides a wide range of machine learning algorithms and models through libraries like scikit-learn, TensorFlow, and PyTorch. Depending on the nature of the problem (e.g., classification, regression, clustering), developers can choose the appropriate algorithm to train their model.
3. **Training the Model:** Once the dataset and model are prepared, developers can train the model using the training data. During the training process, the model adjusts its parameters to minimize the difference between its predictions and the actual target values. Python libraries like scikit-learn provide simple interfaces for training models with just a few lines of code.
4. **Evaluation:** After training the model, it's essential to evaluate its performance to assess how well it generalizes to new, unseen data. Python libraries offer functions for calculating various metrics such as accuracy, precision, recall, and F1-score for classification tasks, as well as mean squared error, mean absolute error, and R-squared for regression tasks.
5. **Hyperparameter Tuning:** Many machine learning algorithms have parameters that need to be tuned to achieve optimal performance. Python libraries like scikit-learn provide tools for hyperparameter tuning, such as GridSearchCV and RandomizedSearchCV, which automatically search through a specified set of hyperparameters to find the best combination.
6. **Deployment:** Once the model is trained and evaluated, it can be deployed into production environments to make predictions on new data. Python libraries like Flask and Django are commonly used for building web services and APIs to serve machine learning models.

2.3.1.3 Flask

Flask is a lightweight and flexible web framework for Python. It's designed to make it easy to build web applications quickly and with minimal boilerplate code. Flask is known for its simplicity, ease of use, and extensibility, making it a popular choice among developers for building web applications, APIs, and microservices.

Key features of Flask include:

- 1.Minimalistic: Flask is minimalist by design, providing only the essential components needed for web development. This simplicity makes it easy to understand and use, especially for beginners.
- 2.Flexible: Flask allows developers to structure their applications as they see fit. It doesn't impose any rigid patterns or conventions, giving developers the freedom to organize their code in a way that suits their project.
- 3.Extensibl: Flask is highly extensible, with a rich ecosystem of extensions that add additional functionality to the framework. These extensions cover a wide range of features such as database integration, authentication, form validation, and more.
- 4.inja2 Templating: Flask uses the Jinja2 templating engine, which allows developers to build dynamic HTML pages by embedding Python code directly into HTML templates. This makes it easy to generate HTML content dynamically based on data from the application.
- 5.Built-in Development Serve: Flask comes with a built-in development server, making it easy to get started with web development without the need for additional setup. The development server is suitable for testing and debugging applications during development.
- 6.RESTful Support: Flask provides built-in support for building RESTful APIs, allowing developers to create web services for serving JSON data to client applications.
- 7.Werkzeug Integration: Flask is built on top of the Werkzeug WSGI toolkit, which provides low-level utilities for handling HTTP requests and responses. This integration allows Flask to handle HTTP routing, request parsing, and response generation efficiently.
- 8.Lightweight: Flask has minimal dependencies and a small codebase, making it lightweight and fast. This makes it suitable for building small to medium-sized web applications and APIs.

3. DATA ANALYSIS

3.1 STRUCTURE OF DATA :

- The dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. The datasets consist of several medical predictor variables and one target variable, Outcome. Predictor variables include the number of patient has, their BMI, insulin level, age etc.

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn import metrics
from sklearn.metrics import roc_curve, auc

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

Fig3.1.1 Importing Libraries

Fig3.1.1, Importing libraries to implement various machine learning for classification techniques.


```
df = pd.read_csv("diabetes.csv")  
df.head()
```

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

Fig3.1.2 Loading Dataset

Fig3.1.2, Loading the dataset to understand data structure.

```
data.shape
```

```
(768, 9)
```

Fig 3.1.3 Shape of dataset

Fig 3.1.3, represent total number of rows and columns in Dataset

3.2 PARAMETERS IMPLEMENTED:-

- ❖ Glucose: Plasma glucose concentration for 2 hours in an oral glucose tolerance test.
- ❖ Blood Pressure: Diastolic blood pressure (mm Hg). It is the bottom number in blood pressure tests, and is the pressure in the arteries when the heart rests between beats. A normal diastolic blood pressure is < 80 mm HG.
- ❖ Skin Thickness: Triceps skin fold thickness (mm). Studies have been conducted, with conclusions that there are associations between people with thicker skin and diabetes.
- ❖ Insulin: 2-Hour serum insulin (μ U/ml). Insulin is a hormone made by the pancreas that allows your body to use sugar (glucose) from carbohydrates in the food that you eat for energy or to store glucose for future use. A high insulin level is associated with diabetes.
- ❖ BMI: Body mass index (weight in kg/ (height in m) ^2)
 - Range of BMI: BMI < 18.5 - underweight
 - $18.5 < \text{BMI} < 24.9$ - ideal weight
 - $25 < \text{BMI} < 29.9$ – overweight
 - $29.9 < \text{BMI}$ - obese
- ❖ Diabetes Pedigree Function: It is a synthesis of the diabetes mellitus history in relatives and the genetic relationship of those relatives to the subject.
- ❖ Results show that a person with a higher pedigree function tested positive and those who had a lower pedigree function tested negative.
- ❖ Age: Age of the patient in years
- ❖ Outcome: The target column which we are interested in finding out. 1 - diabetic, 0 - non-diabetic.

3.3 EXPLORATORY DATA ANALYSIS:-

```
[ ] df.isnull().sum()

Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DiabetesPedigreeFunction  0
Age              0
Outcome          0
dtype: int64
```

Fig3.3.1 Exploratory Data Analysis

Fig 3.3.1, is analyzing the dataset and checking any missing values.

```
[ ] df.info()

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

Fig 3.3.2 Dataset Information

Fig 3.3.2 Dataset information's are checked.

```
[ ] df.describe()
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Fig 3.3.3 Calculating Mean, Count, Min, Max and Standard Deviation.

```
[ ] df_0 = df.copy()

cols_0 = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

for i in cols_0:
    df_0[i].replace(0, df_0[i].mean(), inplace=True)

df_0.head()
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	148.0	72.0	35.000000	79.799479	33.6	0.627	50	1
1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	0
2	183.0	64.0	20.536458	79.799479	23.3	0.672	32	1
3	89.0	66.0	23.000000	94.000000	28.1	0.167	21	0
4	137.0	40.0	35.000000	168.000000	43.1	2.288	33	1

Fig 3.3.4 Based on the understanding of the parameters, it seems highly unlikely that glucose, blood pressure, skin thickness, insulin and BMI levels are 0.

```
[ ] df_0.describe()
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	121.681605	72.254807	26.606479	118.660163	32.450805	0.471876	33.240885	0.348958
std	30.436016	12.115932	9.631241	93.080358	6.875374	0.331329	11.760232	0.476951
min	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750	24.000000	0.000000
50%	117.000000	72.000000	23.000000	79.799479	32.000000	0.372500	29.000000	0.000000
75%	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Fig 3.3.5 Creating a copy of the original dataset and replace the 0 values of the impacted columns with the mean values Now that the 0 values are accounted for, we can proceed with the rest of the Exploratory Data Analysis.

3.4 HISTOGRAM PLOT OF DATA :-

```
col = list(df_0.columns)
df_0[col].hist(stacked=True, bins=20, figsize=(10,10), layout=(3,3))
plt.tight_layout()
```

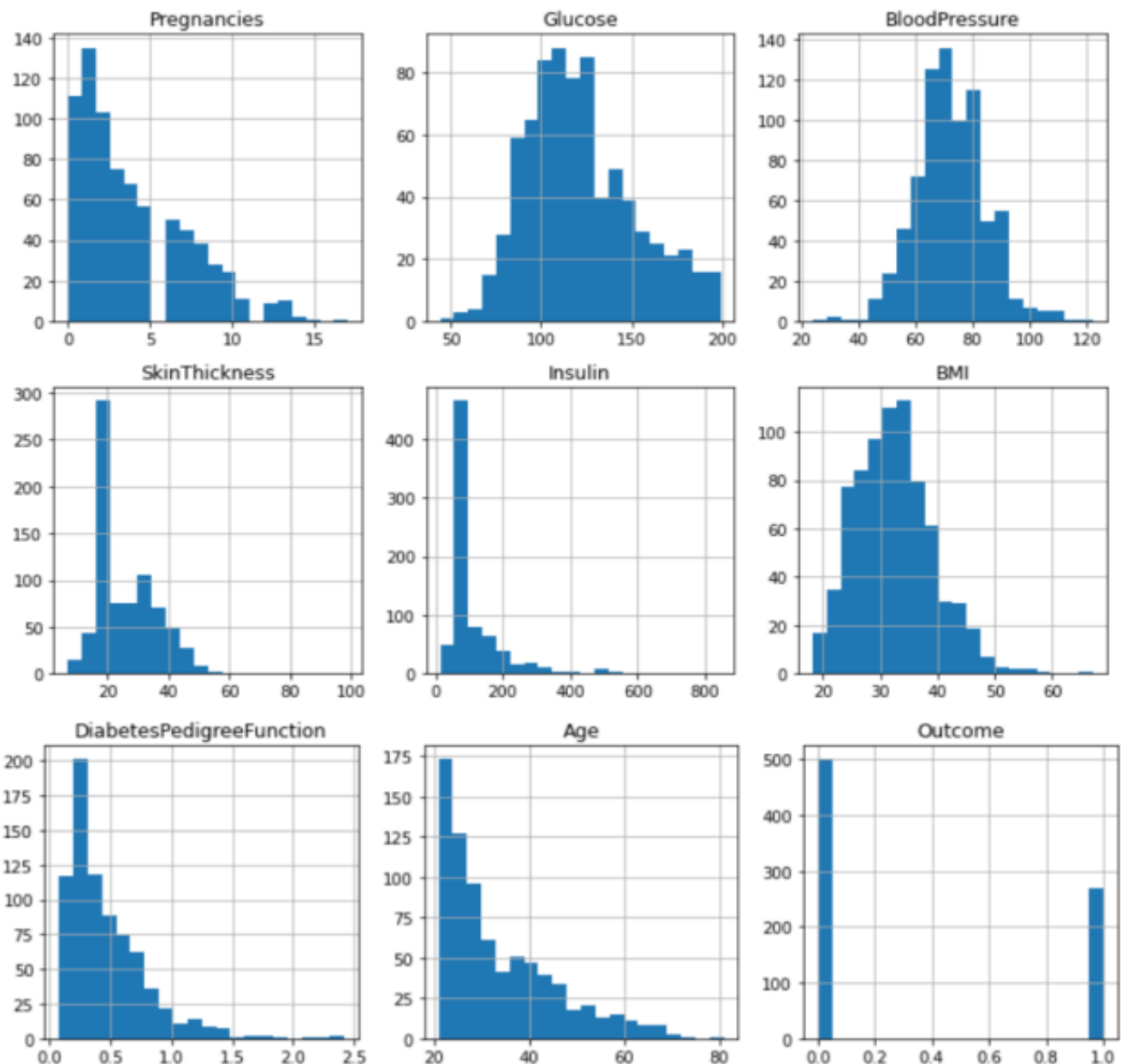


Fig 3.4.1 Histogram

The above histogram plots give a high-level view of the bucket distribution of the dataset parameters. At first glance, most of them appear to be positively skewed, with Glucose and Blood Pressure with the closest distribution to a normal distribution.

3.5 CORRELATION OF DATA:-

```
df_0.corr()
```

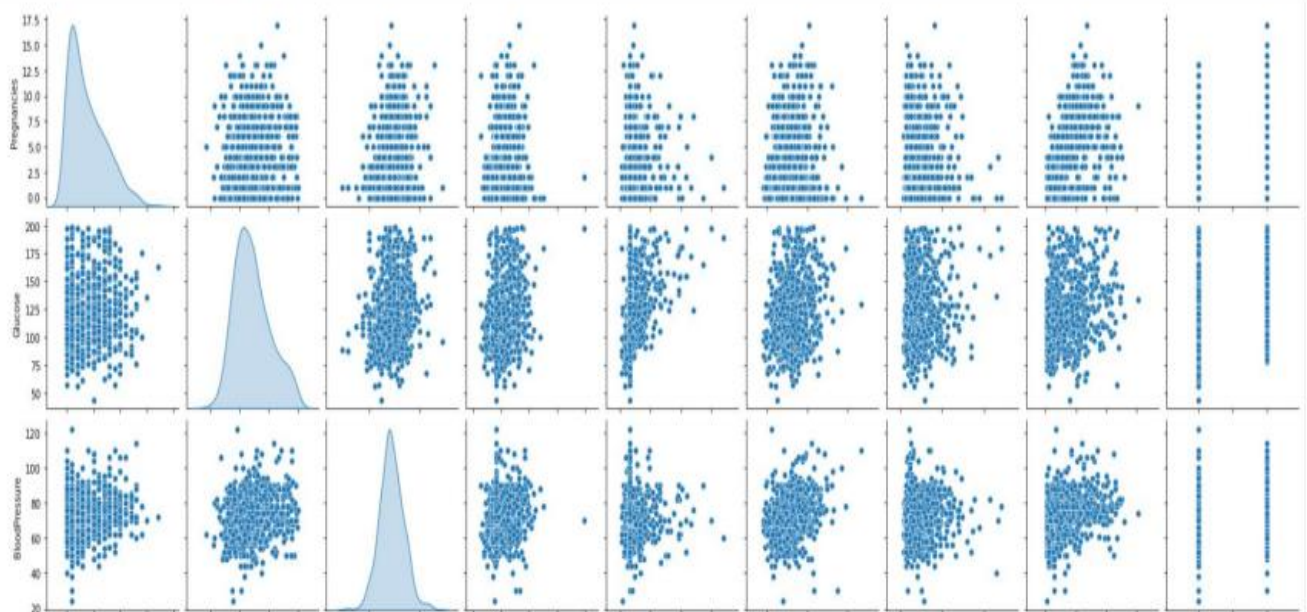
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.127964	0.208984	0.013376	-0.018082	0.021546	-0.033523	0.544341	0.221898
Glucose	0.127964	1.000000	0.219666	0.160766	0.396597	0.231478	0.137106	0.266600	0.492908
BloodPressure	0.208984	0.219666	1.000000	0.134155	0.010926	0.281231	0.000371	0.326740	0.162986
SkinThickness	0.013376	0.160766	0.134155	1.000000	0.240361	0.535703	0.154961	0.026423	0.175026
Insulin	-0.018082	0.396597	0.010926	0.240361	1.000000	0.189856	0.157806	0.038652	0.179185
BMI	0.021546	0.231478	0.281231	0.535703	0.189856	1.000000	0.153508	0.025748	0.312254
DiabetesPedigreeFunction	-0.033523	0.137106	0.000371	0.154961	0.157806	0.153508	1.000000	0.033561	0.173844
Age	0.544341	0.266600	0.326740	0.026423	0.038652	0.025748	0.033561	1.000000	0.238356
Outcome	0.221898	0.492908	0.162986	0.175026	0.179185	0.312254	0.173844	0.238356	1.000000

Fig 3.5.1 Correlation of Data

The parameter with the highest positive correlation to each other is BMI and Skin Thickness. This is further confirmed by the SNS pair plot. The rest do not have strong multi-collinearity to each other.

3.6 SNS PAIR PLOT:-

```
[ ] sns.pairplot(df_0,diag_kind='kde');
```



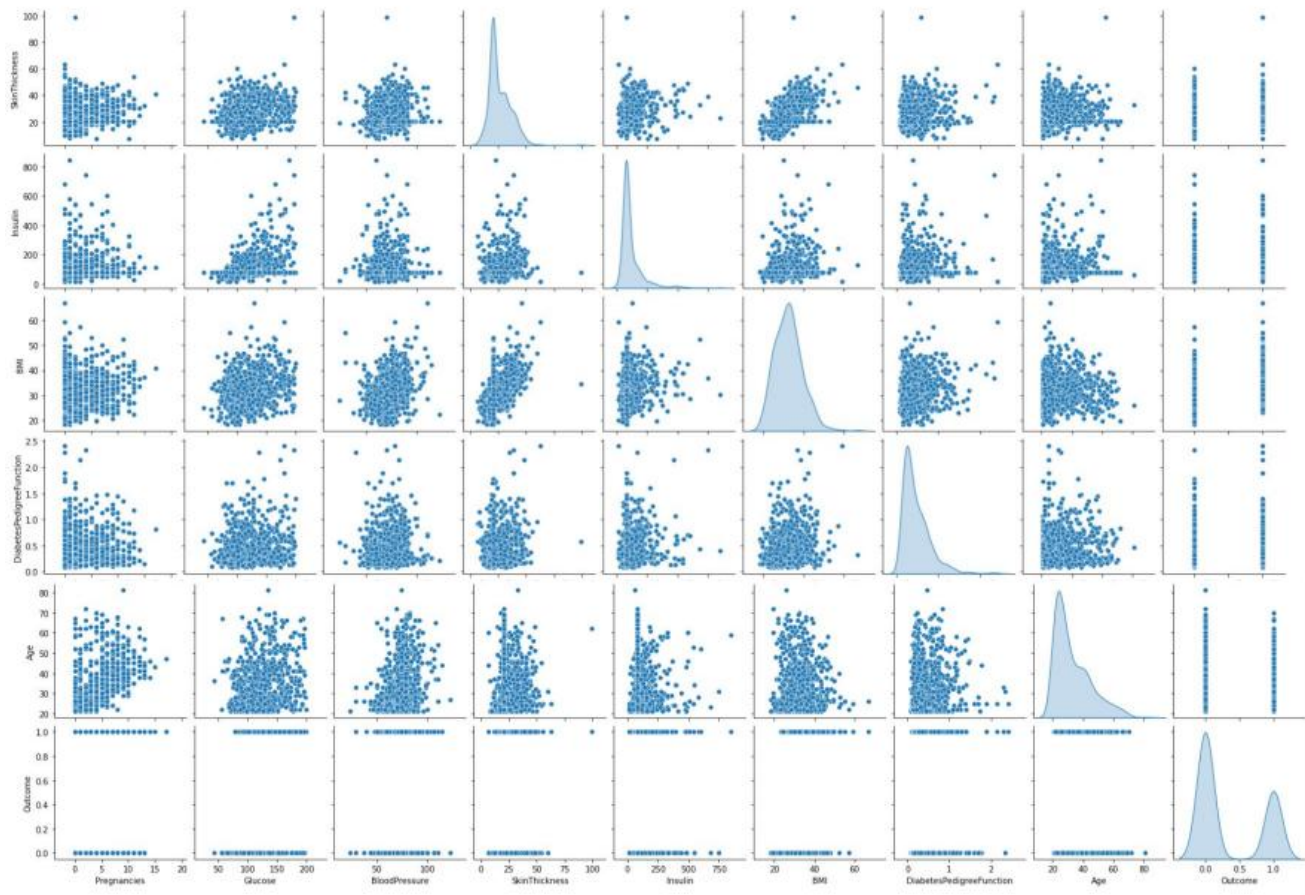


Fig 3.6.1 SNS Pair plot

From the plots, we can see that in histogram plot distribution, that most of the parameters are positively skewed, with outcome having a bimodal distribution, which is to be expected. Glucose and Blood Pressure are the only parameters which most resemble a normal distribution. Plot a pair plot to see which parameters might have a stronger correlation with either outcomes of diabetic patient and non-diabetic patient.

```
df_t = df_0.copy()

df_t['Outcome'].astype('category')
df_t['Outcome'].replace(0,"non-diabetic",inplace=True)
df_t['Outcome'].replace(1,"diabetic",inplace=True)

sns.pairplot(df_t,hue='Outcome',diag_kind='kde');
```


4.SYSTEM DESIGN

USE CASE DIAGRAM:

- Use Case Diagram Displays the relationship among actors and use cases.
- A use case diagram shows a set of use cases and actors (a special kind of class) and their relationships.
- Use case diagrams address the static use case view of a system.
- These diagrams are especially important in organizing

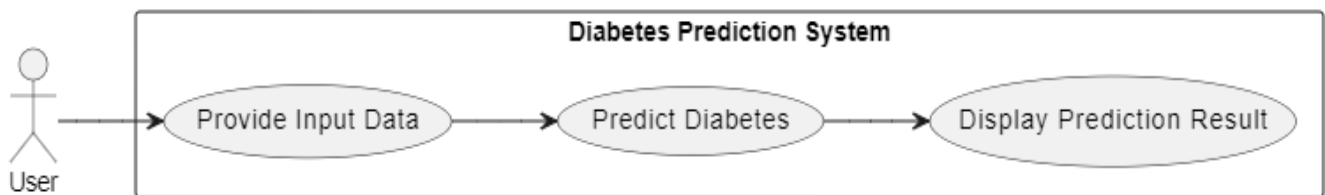


Fig 4.1 Use Case Diagram

Activity diagram:

Activity diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of systems. An activity diagram is essentially a flowchart, showing flow of control from activity to activity. You use activity diagrams to model the dynamic aspects of a system. For the most part, this involves modeling the sequential (and possibly concurrent) steps in a computational process. With an activity diagram, you can also model the flow of an object as it moves from state to state at different points in the flow of control. Activity diagrams may stand alone to visualize, specify, construct, and document the dynamics of a society of objects, or they may be used to model the flow of control of an operation. Whereas interaction diagrams emphasize the flow of control from object to object, activity diagrams emphasize the flow of control from activity to activity.

An activity is an ongoing non atomic execution within a state machine. Activities ultimately result in some action, which is made up of executable atomic computations those results in a change in state of the system or the return of a value. Activity diagrams are not only important for modeling the dynamic aspects of a system, but also for constructing executable systems through forward and reverse engineering.

Action states and activity states are just special kinds of states in a state machine. When you enter an action or activity state, you simply perform the action or the activity; when you finish, control passes to the next action or activity. Activity states are somewhat of shorthand, therefore. An activity state is semantically equivalent to Expanding its activity graph (and transitively so) in place until you only see actions. Nonetheless, activity states are important because they help you break complex computations into parts, in the same manner as you use operations to group and reuse expressions.

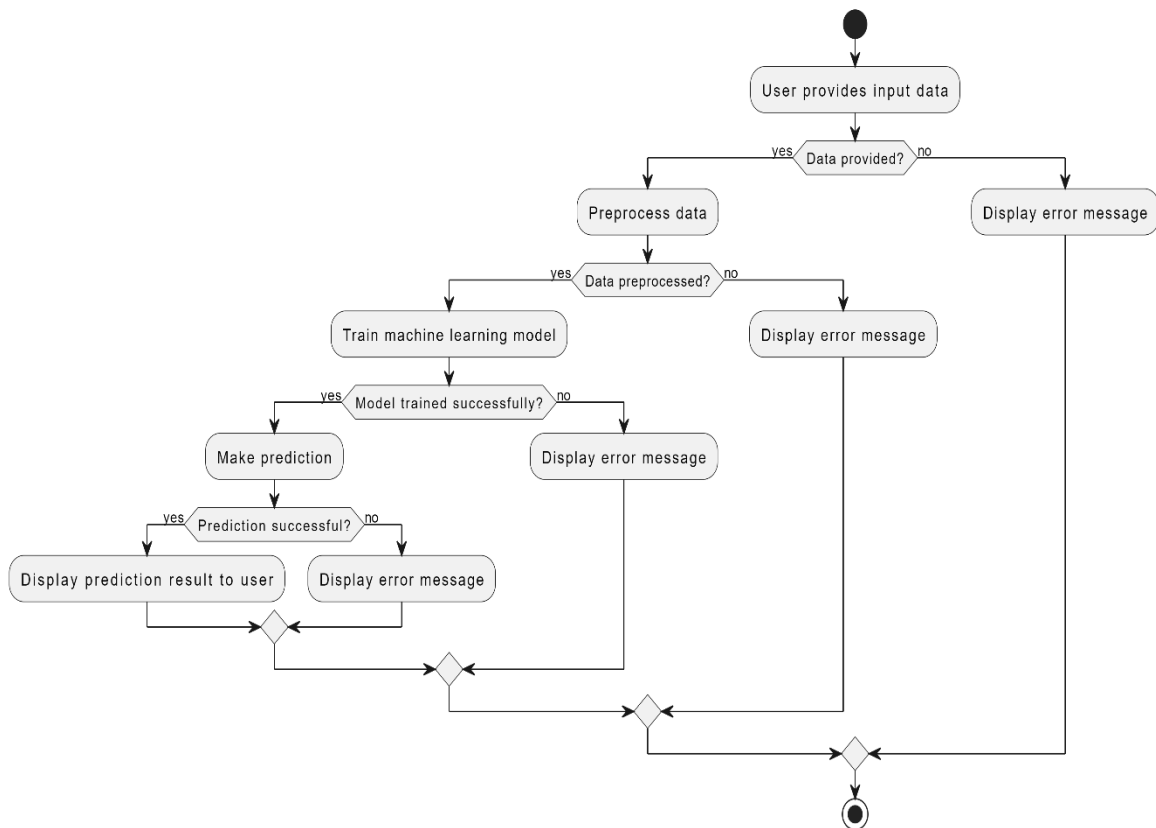


Fig 4.2 Activity Diagram

Data flow diagram:

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an Information System. A data flow diagram can also be used for the visualization of Data Processing.

It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then "exploded" to show more detail of the system being modeled.

A DFD represents flow of data through a system. Data flow diagrams are commonly used during problem analysis. It views a system as a function that transforms the input into desired output. A DFD shows movement of data through the different transformations or processes in the system.

Dataflow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to restock how any system is developed can be determined through a dataflow diagram. The appropriate register saved in database and maintained by appropriate authorizes.

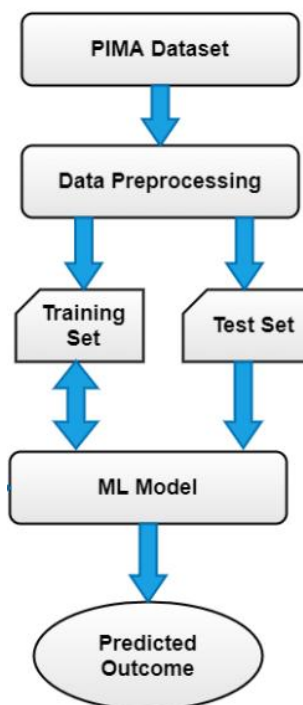


Fig 4.3 Data Flow Diagram

5.IMPLEMENTATION

5.1 HYPOTHESIS TESTING :-

- First hypothesis test would be to try and confirm my suspicion if Glucose data has a normal distribution.
- Next hypothesis test would be that based on the above hypothesis test, I would test the correlation between Glucose and the target outcome.

```
#Assuming a 5% significance level for all the tests
alpha = 0.05

def hypothesis_result(p_value):
    p_value = round(p_value,4)
    if p_value < alpha:
        print ("As p-value is ",p_value," , which is lower than the significance level, we reject the null hypothesis.")
    else:
        print ("As p-value is ",p_value," , which is higher than the significance level, we do not reject the null hypothesis.")
```

Fig 5.1.1 Hypothesis Testing

- First Hypothesis Test
- Null Hypothesis: The sample comes from a normal distribution. Alternative Hypothesis: The sample does not come from a normal distribution.

```
[ ] from scipy import stats

s2, p2 = stats.normaltest(df_0['Glucose'])

hypothesis_result(p2)
```

As p-value is 0.0 , which is lower than the significance level, we reject the null hypothesis.

```
[ ] from scipy import stats

s2, p2 = stats.normaltest(df_0['Glucose'])

hypothesis_result(p2)
```

As p-value is 0.0 , which is lower than the significance level, we reject the null hypothesis.

5.2 SPLITTING OF DATASET (TRAINING/VALIDATION/TESTING):-

- The splitting of the dataset for validation and testing. Training Dataset: Dataset sample that is used to fit the model. Validation Dataset: Dataset sample that is used for hyper tuning the parameters, and comparing the accuracy and error rates of the model performance between using the training dataset and the validation dataset. Testing Dataset: Dataset sample that is used to test the model performance (predictive power).

```
# Splitting the data set into training and test set
```

```
X = df_0.drop(['Outcome'],axis=1)  
Y = df_0['Outcome']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
```

```
# Print the distribution of labels between the training and testing dataset
```

```
print("Ratio of Diabetes to Non-Diabetic Labels in training dataset is: {}".format(round(y_train.value_counts()[0] \
```

```
len(y_train),2)))  
print("Ratio of Diabetes to Non-Diabetic Labels in testing dataset is: {}".format(round(y_test.value_counts()[0] \
```

```
len(y_test),2)))  
Ratio of Diabetes to Non-Diabetic Labels in training dataset is: 0.66
```

```
Ratio of Diabetes to Non-Diabetic Labels in testing dataset is: 0.63
```

5.3 Feature Scaling:-

- Here StandardScaler() is used to perform feature scaling. This will retain the mean and the standard deviation of the sample distribution of the data set, and reuse it to transform the X_train and X_test subsequently. I try to reuse the mean and standard deviation obtained from the training set and apply it to the testing set as well. Standardizing data after data splitting is to prevent data leakage from test dataset into train dataset.

```
# Scaling the x training and testing dataset
```

```
scaler = preprocessing.StandardScaler().fit(X_train)
```

```
X_train_scaled = scaler.transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
```

5.4 Implementing Machine Learning Algorithms:-

- Different machine learning algorithms to try and classify the pima Indian diabetes dataset. First a confusion matrix function is formed.
- a. Accuracy: $(TP+TN)/All$
- b. Recall: $TP/(TP+FN)$
- c. Precision: $TP/(TP+FP)$
- d. Specificity: $TN/(TN+FP)$

```
# Confusion Matrix function

def conf_mtx(y_act,y_pred):
    cm=metrics.confusion_matrix(y_act, y_pred, labels=[1, 0])
    df_cm = pd.DataFrame(cm, index = [i for i in ["Diabetic","Non-Diabetic"]],
                        columns = [i for i in ["Predict Diabetic","Predict Non-Diabetic"]])
    plt.figure(figsize = (6,6))
    plt.title("Confusion Matrix")
    sns.heatmap(df_cm, annot=True ,fmt='g')

    Score_Accuracy = "%.2f%%" %(metrics.accuracy_score(y_act,y_pred)*100)
    Score_Recall = "%.2f%%" %(metrics.recall_score(y_act,y_pred)*100)
    Score_Precision = "%.2f%%" %(metrics.precision_score(y_act,y_pred)*100)

    print("Model Accuracy Score: " + Score_Accuracy)
    print("Model Recall Score: " + Score_Recall)
    print("Model Precision Score: " + Score_Precision)

    return Score_Accuracy,Score_Recall,Score_Precision
```

```
# Prepare an empty summary dataframe to append the data of the various models for comparison
summary = pd.DataFrame(columns=('Model', 'Training Accuracy', 'Test Accuracy Score', 'Test Recall Score', \
                                'Test Precision Score', 'AUC'))
```

```
[ ] def ML_test(Mdl,Param_grid):
    if bool(Param_grid):
        Mdl = GridSearchCV(Mdl,Param_grid,cv=10)
        Mdl.fit(X_train_sm,y_train_sm)
        Mdl_params = Mdl.best_params_
        Mdl_train_sc = Mdl.cv_results_['mean_test_score'].mean()
        Mdl_test_sc = Mdl.score(X_test_scaled,y_test)
        probas = Mdl.predict_proba(X_test_scaled)

        print("Best fit parameter is: " + str(Mdl_params))

    else:
        Mdl = Mdl
        Mdl.fit(X_train_sm,y_train_sm)
        Mdl_train_sc = round(Mdl.score(X_train_sm,y_train_sm),4)
        Mdl_test_sc = round(Mdl.score(X_test_scaled,y_test),4)
        probas = Mdl.predict_proba(X_test_scaled)

    y_pred = Mdl.predict(X_test_scaled)

    print("Training score is: " + str(Mdl_train_sc))
    print("Test Mean score is: " + str(Mdl_test_sc))

    Score_Accuracy,Score_Recall,Score_Precision = conf_mtx(y_test,y_pred)
    Mdl_train_sc = "%.2f%%" % (Mdl_train_sc*100)

    # Calculating AUC
    fpr, tpr, thresholds = roc_curve(y_test, probas[:, 1])
    roc_auc = round(auc(fpr, tpr),4)
    print("Area under the ROC curve : " + str(roc_auc))

    return Mdl_train_sc, Score_Accuracy, Score_Recall, Score_Precision, roc_auc
```

Fig5.4.0 Building a function for performing ML algos testing

5.4.1 Support Vector Machine Model:-

```
Mdl = SVC(probability=True)

model_name = "Support Vector Machine"
Param_grid_SVC = {'C': np.linspace(0.1,1.1,10), 'kernel': ['linear','poly','rbf',]}

Mdl_train_sc, Score_Accuracy, Score_Recall, Score_Precision, roc_auc = ML_test(Mdl,Param_grid_SVC)

summary = summary.append({'Model' : model_name, 'Training Accuracy' : Mdl_train_sc, 'Test Accuracy Score' : Score_Accuracy,\
                          'Test Recall Score' : Score_Recall, 'Test Precision Score' : Score_Precision, 'AUC': roc_auc}, \
                          ignore_index=True)
```

Fig 5.4.4.1 Support Vector Machine Code

5.4.2 Decision Tree Model:-

```
Mdl = DecisionTreeClassifier(random_state=1)

model_name = "DecisionTreeClassifier"
Param_grid_dt = {'criterion':['gini','entropy'],'max_depth': [3, 4, 5, 6, 7, 8],\
                  'min_impurity_decrease': [0.0001, 0.0003, 0.0005, 0.0007, 0.009]}

Mdl_train_sc, Score_Accuracy, Score_Recall, Score_Precision, roc_auc = ML_test(Mdl,Param_grid_dt)

summary = summary.append({'Model' : model_name, 'Training Accuracy' : Mdl_train_sc, 'Test Accuracy Score' : Score_Accuracy,\
                          'Test Recall Score' : Score_Recall, 'Test Precision Score' : Score_Precision, 'AUC': roc_auc}, \
                          ignore_index=True)
```

Fig 5.4.2.1 Decision Tree Code

5.4.3 Random Forest Model

```
Mdl = RandomForestClassifier(random_state=1,n_estimators=100)

model_name = "RandomForestClassifier"
Param_grid_rf = {'criterion':['gini','entropy'],'max_depth': [3, 4, 5, 6, 7, 8],\
                  'min_impurity_decrease': [0.0001, 0.0003, 0.0005, 0.0007, 0.009]}

Mdl_train_sc, Score_Accuracy, Score_Recall, Score_Precision, roc_auc = ML_test(Mdl,Param_grid_rf)

summary = summary.append({'Model' : model_name, 'Training Accuracy' : Mdl_train_sc, 'Test Accuracy Score' : Score_Accuracy,\
                          'Test Recall Score' : Score_Recall, 'Test Precision Score' : Score_Precision, 'AUC': roc_auc}, \
                          ignore_index=True)
```

Fig 5.4.3 Random Forest Code

5.4.4 Bagging Classifier :-

```
# Bagging Classifier

Mdl = BaggingClassifier(n_estimators=100, bootstrap=True)

model_name = "BaggingClassifier"
Param_grid_bc = {'max_samples': list(np.arange(0.1,1.1,0.1))}

Mdl_train_sc, Score_Accuracy, Score_Recall, Score_Precision, roc_auc = ML_test(Mdl,Param_grid_bc)

summary = summary.append({'Model' : model_name, 'Training Accuracy' : Mdl_train_sc, 'Test Accuracy Score' : Score_Accuracy, \
                          'Test Recall Score' : Score_Recall, 'Test Precision Score' : Score_Precision, 'AUC': roc_auc}, \
                          ignore_index=True)
```

Fig 5.4.4 Bagging Classifier Code

5.4.5 AdaBoost Classifier :-

```
Mdl = AdaBoostClassifier( n_estimators= 100)

model_name = "AdaBoostClassifier"
Param_grid_abc = {'learning_rate': list(np.arange(0.1,1.1,0.1))}

Mdl_train_sc, Score_Accuracy, Score_Recall, Score_Precision, roc_auc = ML_test(Mdl,Param_grid_abc)

summary = summary.append({'Model' : model_name, 'Training Accuracy' : Mdl_train_sc, 'Test Accuracy Score' : Score_Accuracy, \
                          'Test Recall Score' : Score_Recall, 'Test Precision Score' : Score_Precision, 'AUC': roc_auc}, \
                          ignore_index=True)
```

Fig 5.4.5Ada Boost Classifier Code

6. TEST RESULTS

6.1 RESULTS :-

6.1.1 Support Vector Machine Model :-

Best fit parameter is: {'C': 0.9888888888888888, 'kernel': 'rbf'}
Training score is: 0.739758551307847
Test Mean score is: 0.8095238095238095
Model Accuracy Score: 80.95%
Model Recall Score: 77.65%
Model Precision Score: 72.53%
Area under the ROC curve : 0.8674

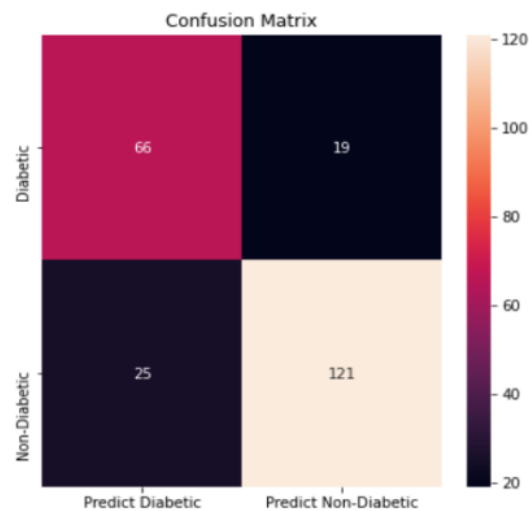


Fig 6.1.1

6.1.2 Decision Tree Model:-

Best fit parameter is: {'criterion': 'entropy', 'max_depth': 8, 'min_impurity_decrease': 0.009}
Training score is: 0.7519604292421196
Test Mean score is: 0.7662337662337663
Model Accuracy Score: 76.62%
Model Recall Score: 83.53%
Model Precision Score: 63.96%
Area under the ROC curve : 0.8587

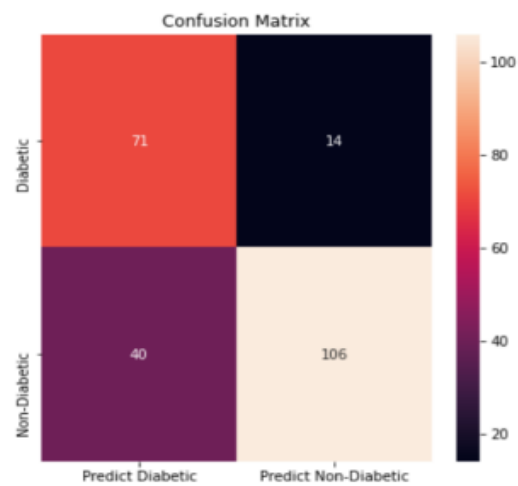


Fig 6.1.2

6.1.3 Random Forest Model:-

Best fit parameter is: {'criterion': 'entropy', 'max_depth': 8, 'min_impurity_decrease': 0.0007}
Training score is: 0.7882669349429912
Test Mean score is: 0.8095238095238095
Model Accuracy Score: 80.95%
Model Recall Score: 83.53%
Model Precision Score: 70.30%
Area under the ROC curve : 0.879

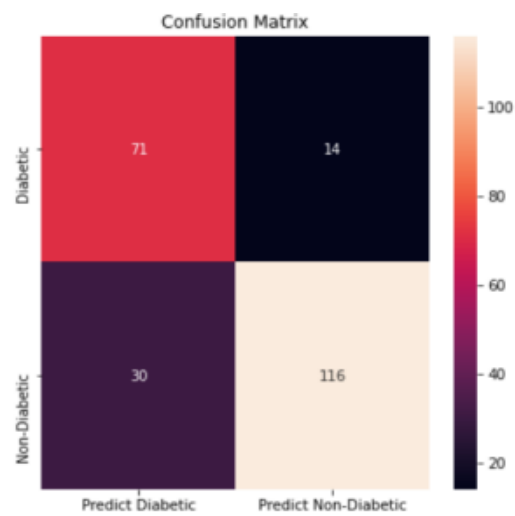


Fig 6.1.3

6.1.4 Bagging Classifier:-

Best fit parameter is: {'learning_rate': 0.30000000000000004}
Training score is: 0.7807183098591549
Test Mean score is: 0.7835497835497836
Model Accuracy Score: 78.35%
Model Recall Score: 82.35%
Model Precision Score: 66.67%
Area under the ROC curve : 0.8504

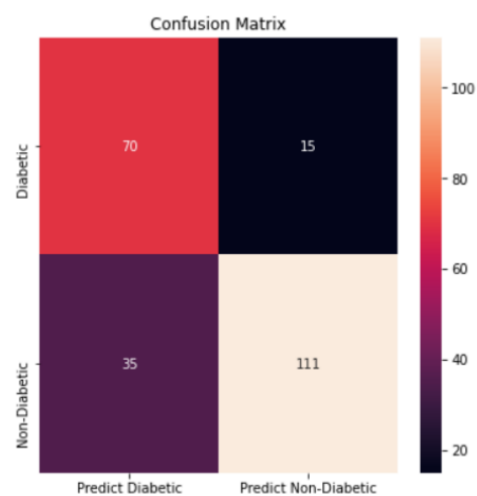


Fig 6.1.4

6.1.5 AdaBoost Classifier:-

Best fit parameter is: {'learning_rate': 0.30000000000000004}
Training score is: 0.7807183098591549
Test Mean score is: 0.7835497835497836
Model Accuracy Score: 78.35%
Model Recall Score: 82.35%
Model Precision Score: 66.67%
Area under the ROC curve : 0.8504

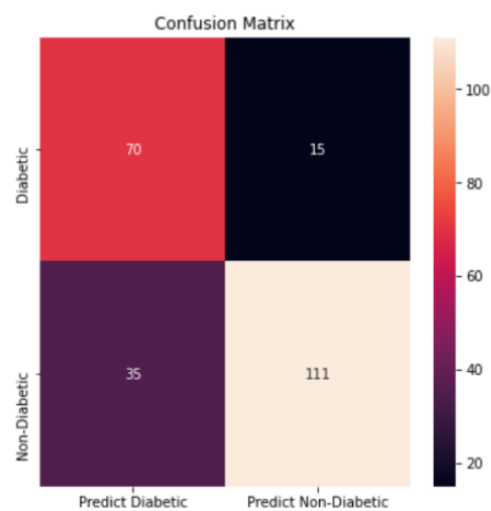


Fig 6.1.5

6.2 COMPARATIVE ANALYSIS:-

summary

	Model	Training Accuracy	Test Accuracy Score	Test Recall Score	Test Precision Score	AUC
0	LogisticRegression	72.53%	79.22%	77.65%	69.47%	0.8763
1	GaussianNB	71.89%	75.76%	71.76%	65.59%	0.8558
2	k-Nearest Neighbours	75.77%	73.59%	74.12%	61.76%	0.7884
3	Support Vector Machine	73.98%	80.95%	77.65%	72.53%	0.8674
4	DecisionTreeClassifier	75.20%	76.62%	83.53%	63.96%	0.8587
5	RandomForestClassifier	78.83%	80.95%	83.53%	70.30%	0.8790
6	BaggingClassifier	81.51%	80.09%	76.47%	71.43%	0.8663
7	AdaBoostClassifier	78.07%	78.35%	82.35%	66.67%	0.8504
8	GradientBoostingClassifier	82.23%	79.22%	71.76%	71.76%	0.8496

Table 6.2.1 Comparative Analysis Table

6.3 TEST RESULTS ANALYSIS:-

Finally, we have trained our models, and summarized table of the metrics of the various models. Objectives were,

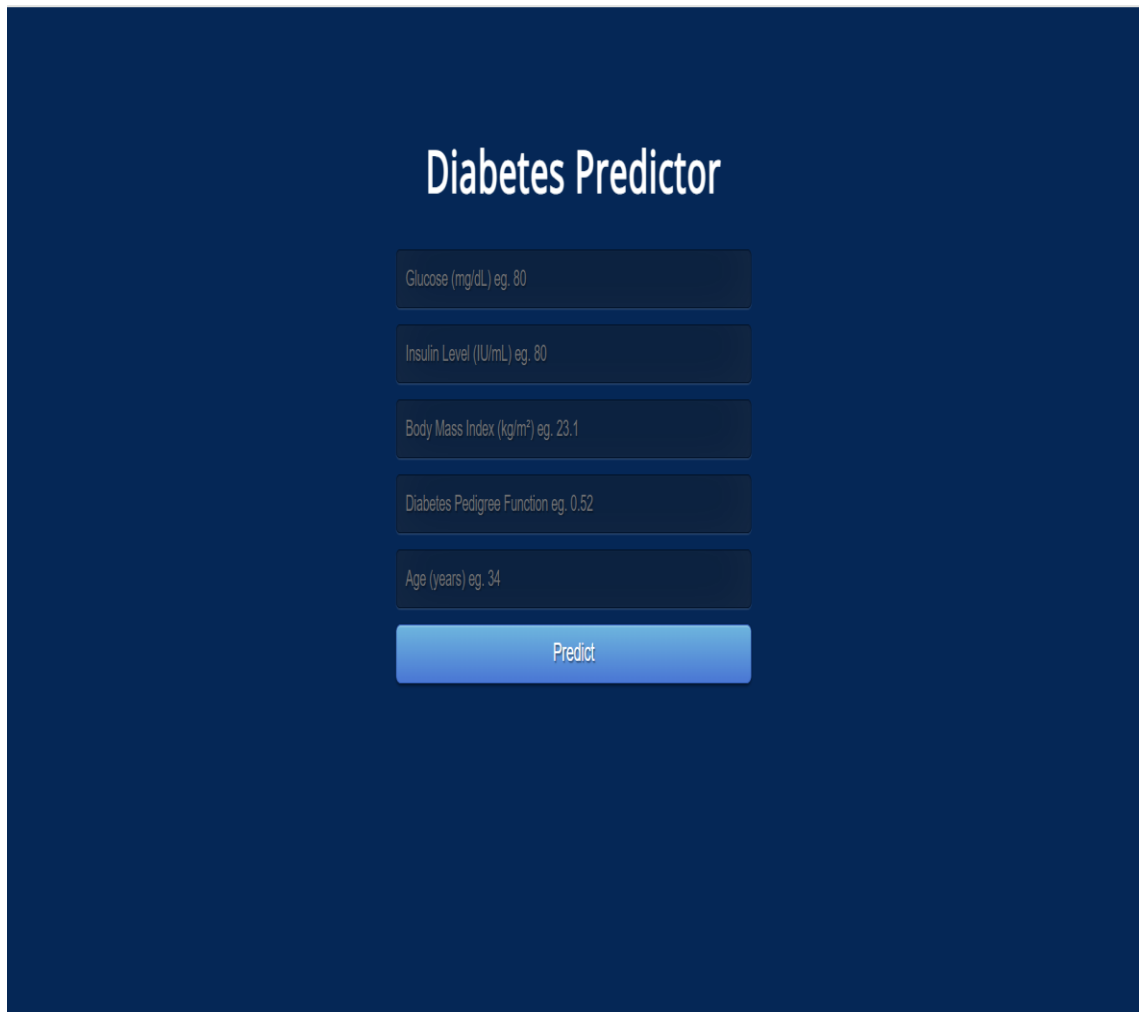
- 1) To attempt to see if it is possible to glean any further information from the data to determine correlation between parameters and diabetes.
- 2) To attempt to get the best accuracy score using various supervised learning machine learning algorithms.
 - i. For the first objective, based on the hypothesis test, we can tell that glucose levels are positively correlated to a person having diabetes, but we are not able to confirm if there is causality. For the second objective, based on the comparison between the various algorithms used, Random Forest seems to produce the best results to me.
 - ii. The aim of this project is to create a model that can reliably predict the accuracy of diabetes in patients. The main aim of this project is to design and implement Diabetes Prediction Using Machine Learning Methods and Performance Analysis of that methods and it has been achieved successfully.
 - iii. The proposed approach uses various classification and ensemble learning method in which SVM, Random Forest, Decision Tree and Gradient Boosting classifiers are used. A machine learning algorithm must be used to build a framework that provides reliable results while reducing human effort.
 - iv. The test accuracy of the various models is generally within the same range, from approximately 73% to 81%. Based on Accuracy and Recall score, overall the Random Forest Classifier produced the best result

7.1 SNAPSHOT:-

1. Logo Page



2.Input Or parameters Page



The screenshot shows a web application titled "Diabetes Predictor" on a dark blue background. The title is centered at the top in a large, white, sans-serif font. Below the title, there are five input fields, each with a light gray border and a small example value in a lighter gray font. The fields are stacked vertically and centered. At the bottom of the input section is a prominent blue button with the word "Predict" in white, centered text.

Diabetes Predictor

Glucose (mg/dL) eg. 80

Insulin Level (IU/mL) eg. 80

Body Mass Index (kg/m²) eg. 23.1

Diabetes Pedigree Function eg. 0.52

Age (years) eg. 34

Predict

3.Result Page

Prediction: Opps! You have DIABETES.

Prediction: Wow ! You DON'T have diabetes.

8 CONCLUSION AND FUTURE SCOPES

- learning has the great ability to revolutionize the diabetes prediction with the help of advanced computational methods.
- Detection of Diabetes in its early stage is the key for treatment.
- The technique may also help researchers to develop an accurate and efficient tool that will reach at the table of clinicians to help them make better decisions about the disease.
- More parameters and factors would be involved in the future scope of this project.
- The accuracy will increase even more when the parameters increase Using traditional techniques and algorithms, we can enhance the accuracy by improving the data.

9 REFERENCE

- [Pima Indians Diabetes Database \(kaggle.com\)](https://kaggle.com/datasets/stone-island/pima-indians-diabetes-database)
- [Diabetes Prediction using Machine Learning Algorithms - ScienceDirect](#)
- [Predicting Diabetes Mellitus With Machine Learning Techniques - PMC \(nih.gov\)](#)
- [Diabetes Prediction using Machine Learning Algorithms \(researchgate.net\)](#)
- Debadri Dutta, Debpriyo Paul, Parthajeet Ghosh, "Analyzing Feature Importance's for Diabetes Prediction using Machine Learning". IEEE, pp 942-928, 2018.
- K.VijiyaKumar, Blavanya, Nirmala, S.Sofia Caroline, "Random Forest Algorithm for the diabetes prediction.