

DIABETES PREDICTION MODEL

A

Project Report

Submitted In Partial Fulfillment of the Requirements

For the Award Of
Bachelor of Technology

Under Guidance Of

ANASUA BANDYOPADHYAY
MICROSOFT & HPE CERTIFIED TECHNICAL TRAINER

Project Carried Out
At



Ardent Computech Pvt. Ltd.

(An ISO 9001:2015 Certified)

SDF Building, Module No: 132,

Ground Floor Sector V,

GP Block, Kolkata- 700091

Submitted By-

FAIZAN MAHFOOZ (25500121110)

NOOR AFSHA (25500121030)

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. We take this sincere opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project work.

Our heartfelt thanks to **Jit Dutta** for providing us the opportunity to develop the project at **Ardent Computech Pvt. Ltd.**

We would like to show our greatest appreciation to **Anasua Bandyopadhyay, Technical Trainer at Ardent**, Durgapur. We always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized. We also want to thank them for sharing their pearls of wisdom with us during the course of this project.

We are also immensely grateful to **Anasua mam** and **Jit Sir** for their comments on earlier versions of the manuscripts, although any errors are our own and should not tarnish the reputations of these esteemed professionals.

Words are inadequate in offering our thanks to the other trainees, **project assistants** and other members at **Ardent Computech Pvt. Ltd.** for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

1. Title of the Project : **Diabetes Prediction Model**
2. Name of the Guide : **Ms. Anasua Bandyopadhyay**

Ph.D.(Reg) M-Tech MCA M.Sc. MBA B-Tech

3. Educational Qualification of the Guide(s) ☐ ☐ ☐ Y ☐ ☐ ☐

4. Working / Teaching experience of the Guide: **1 Year**

5. Software used in the Project

- Python 3.8
- Anaconda 3.5
- MS- Office
- Draw.io

Signature of the Guide

Anasua Bandyopadhyay
.....

For Office Use Only

Approved

Not Approved

**Signature, Designation Stamp
of the Project Proposal
Evaluator**

Date:...05/03/2023**.....**

Self Certificate

This is to certify that the dissertation/project proposal entitled “**Diabetes Prediction Model**” is done by Faizan Mahfooz , Noor Afsha. It is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** under the guidance of **Ms.Anasua Bandyopadhyay**. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Name of the Students:

1. Faizan Mahfooz

: Faizan Mahfooz

2. Noor Afsha

: Noor Afsha

Certificate by Guide

This is to certify that this project entitled “**Diabetes Prediction Model**” submitted in partial fulfillment of the degree of **Bachelor of Technology (B.Tech)** by **Dr. Sudhir Chandra Sur Institute of Technology and Sports Complex** done by **Faizan Mahfoooz , Noor Afsha** is an authentic work carried out under my guidance & best of our knowledge and belief.

Faizan Mahfoooz Noor Afsha

Signature of Student

Date:05/03/2023

Anasua Bandyopadhyay

Signature of the Guide

Date:05/03/2023

Certificate of Approval

This is to certify that this documentation of **Beyond Curriculum Trainig 2023**, entitled “**Diabetes Prediction Model**” is a record of bona-fide work, carried out by **Faizan Mahfooz , Noor Afsha** under my supervision and guidance.

In my opinion, the report in its present form is in fulfillment of all the requirements, as specified by the **Dr. Sudhir Chandra Sur Institute Of Technology** and as per regulations of the **Ardent Computech Pvt. Ltd.** In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for **Bachelor of Technology**.

Anasua Bandyopadhyay

Mr. Anasua Bandyopadhyay
Microsoft & HP Certified Technical
Trainer Ardent Computech Pvt Ltd
(An ISO 9001:2015 Certified)
(Approved by NCVT & Ministry of HRD, Government of India)

TABLE OF CONTENTS

S.No.	Name of the Topic	Page No.
1.	Company Profile	1
2.	Abstract	2
3.	Introduction	2
4.	Algorithms	3-6
	KNN	3
	Decision Tree Classifier	3
	Random Forest Classifier	4
	Logistic Regression	5
	Various Libraries Used	6
5.	Problem Statement	7
6.	Background Dataset	7
7.	What Is Artificial Intelligence and Machine Learning	8-10
8.	System Analysis	11-13
8.1	Identification of Need	11

8.2	Feasibility Study	12
8.3	Work Flow	13
8.4	Hardware and Software Requirements	13
9.	System Design	14-20
9.1	Gantt Chart	14
9.2	Pert Chart	15
9.3	Sequence Diagram	16
9.4	Activity Diagram	18
9.5	Use Case Diagram	20
10.	User Interface Design	21-29
11.	Accuracy Comparison Table	30
12.	Visualization Graph	31-34
13.	Implementation and Testing	35-40
13.1	Introduction	35
13.2	Objective of Testing	35
13.3	Process Overview	36
13.4	Test Cases	36
13.5	Testing Step	36
13.6	Validation	37
13.7	White Box Testing	39
13.8	Black Box Testing	39

13.9	System Testing	39
13.10	Output Testing	39
13.11	User Acceptance Testing	40
13.12	Integration Testing	40
13.13	Functional Testing	40
14.	System Security Measures	41
14.1	Database Security	41
14.2	System Security	41
15.	Cost Estimation	41-46
16.	Future Scope & Further Enhancements	47
17.	Conclusions	47
18.	Bibliography	47

1.ARDENT COMPUTECH PVT.LTD.

Ardent Computech Private Limited is an ISO 9001-2008 certified Software Development Company in India. It has been operating independently since 2003. It was recently merged with ARDENT TECHNOLOGIES.

Ardent Technologies

ARDENT TECHNOLOGIES is a Company successfully providing its services currently in UK, USA, Canada and India. The core line of activity at ARDENT TECHNOLOGIES is to develop customized application software covering the entire responsibility of performing the initial system study, design, development, implementation and training. It also deals with consultancy services and Electronic Security systems. Its primary clientele includes educational institutes, entertainment industries, resorts, theme parks, service industry, telecom operators, media and other business houses working in various capacities.

Ardent Collaborations

ARDENT COLLABORATIONS, the Research Training and Development Department of ARDENT COMPUTECH PVT LTD is a professional training Company offering IT enabled services & industrial trainings for B-Tech, MCA, BCA, MSc and MBA fresher's and experienced developers/programmers in various platforms. Summer Training / Winter Training / Industrial training will be provided for the students of B.TECH, M.TECH, MBA, MCA and BCA only. Deserving candidates may be awarded stipends, scholarships and other benefits, depending on their performance and recommendations of the mentors.

Associations

Ardent is an ISO 9001:2008 company.

It is affiliated to National Council of Vocational Training (NCVT), Directorate General of Employment & Training (DGET), Ministry of Labor & Employment, and Government of India.

2. ABSTRACT

Prediction Models using various machine learning techniques can provide a head-start to the technical development required in the medical and diagnostic fields. These techniques can provide near to accurate results for prediction of early stage of a particular health issue, from datasets collected from the user through health-related interviews, medical reports and questionnaires. This project will make use of ML classification such as K-Nearest Neighbor, Decision Tree, Random Forest, and Logistic regression, and compare the accuracy of the different models. We attempt to build an accurate model to detect early onset of diabetes through our Diabetes Prediction Model by collecting a dataset containing 768 user inputs. Our project found that Random Forest and Logistic Regression achieved the highest accuracy amongst other.

3. INTRODUCTION

Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin, a hormone that regulates blood sugar, or when the body cannot effectively use the insulin, it produces. The two major types of diabetes are the Type 1 and the Type 2. The former (insulin-dependent, juvenile-onset) is characterized by deficient insulin production which requires a daily administration of insulin. The cause and its prevention are unknown. The latter (non-insulin-dependent, or adult-onset) results from the body's ineffective use of insulin. Most people with diabetes have type 2 diabetes. This type is largely the result of excess body weight and physical inactivity. Symptoms include excessive excretion of urine (polyuria), thirst (polydipsia), constant hunger, weight loss, vision changes, and fatigue. Over time, it can damage the heart, blood vessels, eyes, kidneys, and nerves. One can minimize the chances by maintaining a healthy body weight, eat a healthy diet, avoid sugar, saturated fats, and tobacco.

The Diabetes Prediction Model is a type of information filtering system which attempts to take in medical information of a user including age, pregnancies, glucose level, blood pressure, skin thickness, insulin, BMI, and diabetes pedigree function; and make a prediction for the diabetic outcome based on these reports. Often, these systems can collect information through datasets containing the sign and symptoms data of newly diabetic or would be diabetic patient procured from questionnaires and user medical reports, and then can use this information in better understanding of the cause and symptoms of Diabetes.

The main approaches that are widely used for Diabetes Prediction Model are KNN, Decision Tree, Random Forest, and Logistic Regression. Decision Tree can generate understandable rules, perform classification without requiring much computation and handle both continuous and categorical variables whereas Random Forest model grows and combines multiple decision trees that are merged for a more accurate prediction. Logistic regression is basically a **supervised classification algorithm**, i.e., in a classification problem, the target variable (or output) y , can take only discrete values for given set of features (or inputs) X . The accuracy is different for every model in comparison to others, finally showing which can predict diabetes effectively.

Machine Learning is used to train computers and/or machines to provide efficient result to collect knowledge by building various classification and ensemble models from a particular collected dataset. Such collected data can be useful to predict diabetes and other communicable/non-communicable disease and health conditions. This can lead to a break-through of technical diagnosis in medical field.

4. ALGORITHMS

➤ K-NEAREST NEIGHBOUR(KNN):

K-Nearest Neighbor is one of the **simplest Machine Learning algorithms** based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and puts the new case into the category that is most like the available categories. It stores all the available data and classifies a new data point based on the similarity. When new data appears thus 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 problems, but mostly it is used for Classification problems. It 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.

This 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 like the new data.

For example, if 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, for this identification, we can use the KNN algorithm, as it works on a similar measure. The KNN model will then 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.

- **KNeighboursClassifier is imported from sklearn.neighbors**

➤ 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 based on features of the given dataset. 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. A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.

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.

Entropy(s): is a metric to measure the impurity in a given attribute. It specifies randomness in data.

Information Gain: Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class.

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)]

A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

- **DecisionTreeClassifier is imported from sklearn.tree**

➤ RANDOM FOREST CLASSIFIER:

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. **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.**

Below are some points that explain why one should prefer using 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.
- ✓ Use of multiple trees reduce the risk of overfitting.

- **RandomForestClassifier is imported from sklearn.ensemble**

➤ LOGISTIC REGRESSION:

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.** It is actually much similar to Linear Regression except for how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving 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).

- **LogisticRegression is imported from sklearn.linear_model**

VARIOUS LIBRARIES USED:

- **NUMPY:** NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. It is used for numerical data processing.
- **PANDAS:** It is a data analysis library which provides data frames. It can select data in rows and columns. It labels the data into rows & columns.
- **Matplotlib:** It is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is a two-dimensional plotting library for creating graphs and plots (visualization of data). Matplotlib is basically a python package for 2D graph plotting.
- **Scikit-Learn:** It is the most useful **library for machine learning in Python**. It provides a selection of efficient tools for machine learning including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. Scikit-Learn is a higher-level library that includes implementations of several machine learning algorithms.
- **Seaborn:** Seaborn is a **data visualization library** built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

5. PROBLEM STATEMENT

In the first step, we collect the medical history of different people through questionnaire and interviews. These questionnaires' will contain questions related to their age, pregnancies, glucose level, blood pressure, skin thickness, insulin, BMI. The outcome will be the dependent variable while rest of the attributes will be independent. The proposed method aims to focus on selecting the attributes that aid in early detection of Diabetes Mellitus using Predictive analysis. Our challenge is to make this model as accurate and specific as possible, using KNN, Decision Tree, Random Forest and Logistic Regression.

6. BACKGROUND DATASET

- Medical information of 768 patients was collected.
- The dataset has been divided into 9 attributes which include pregnancies, glucose level, blood pressure, skin thickness, insulin, BMI, age.
- The 9th attribute is class variable of each data points. This class variable shows the outcome 0 and 1 for diabetics which indicates positive or negative for diabetics.
- Distribution of Diabetic patient- We made a model to predict diabetes however the dataset was slightly imbalanced having around 500 classes labeled as 0 means negative means no diabetes and 268 labeled as 1 means positive means diabetic.

7. WHAT IS ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING?

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving

The word Artificial Intelligence comprises of two words “Artificial” and “Intelligence”. Artificial refers to something which is made by human or non-natural thing and Intelligence means ability to understand or think. There is a misconception that Artificial Intelligence is a system, but it is not a system .AI is implemented in the system.

Artificial intelligence is based on the principle that human intelligence can be defined in a way that a machine can easily mimic it and execute tasks, from the simplest to those that are even more complex. The goals of artificial intelligence include mimicking human cognitive activity. Researchers and developers in the field are making surprisingly rapid strides in mimicking activities such as learning, reasoning, and perception, to the extent that these can be concretely defined. Some believe that innovators may soon be able to develop systems that exceed the capacity of humans to learn or reason out any subject. But others remain skeptical because all cognitive activity is laced with value judgments that are subject to human experience.

The ideal characteristic of artificial intelligence is its ability to rationalize and take actions that have the best chance of achieving a specific goal. A subset of artificial intelligence is machine learning, which refers to the concept that computer programs can automatically learn from and adapt to new data without being assisted by humans. Deep learning techniques enable this automatic learning through the absorption of huge amounts of unstructured data such as text, images, or video.

As technology advances, previous benchmarks that defined artificial intelligence become outdated. For example, machines that calculate basic functions or recognize text through optical character recognition are no longer considered to embody artificial intelligence, since this function is now taken for granted as an inherent computer function. AI is continuously evolving to benefit many different industries. Machines are wired using a cross-disciplinary approach based on mathematics, computer science, linguistics, psychology, and more.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The type of algorithm data scientists chooses to use depends on what type of data they want to predict. Supervised learning: In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

- **Unsupervised learning:** This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.
- **Semi-supervised learning:** This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.
- **Reinforcement learning:** Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

Supervised machine learning requires the supervised machine learning requires the data scientist to train the algorithm with both labelled inputs and desired outputs. Supervised learning algorithms are good for the following tasks:

- Binary classification: Dividing data into two categories.
- Multi-class classification: Choosing between more than two types of answers.
- Regression modelling: Predicting continuous values.
- Ensembling: Combining the predictions of multiple machine learning models to produce an accurate prediction to train the algorithm with both labelled inputs and desired outputs.

8.SYSTEM ANALYSIS

8.1. IDENTIFICATION OF NEED

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The System is viewed as a whole and the input to the system are identified. The outputs from the organization are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and Decisional variables, analysis and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like questionnaires and medical reports. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is a loop that ends as soon as the user is satisfied with the proposal.

8.2. FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose the organization for work

Effort and time spent on it: Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various area that were considered very carefully during the feasibility study of this project such as Technical, Economic, and operational feasibilities.

- **Technical Feasibility:** This project is technically feasible as all it has got to do to extract tweets, is to get the proper credentials from the Developer's console provided by Twitter. After the credentials are obtained, i.e., the Access Token Key, Access Token Secret Key, Consumer Key and Consumer Secret key, Twitter gives us access to its tweets. Hence, we get a sufficiently large dataset to conduct sentiment analysis. Also, the range of tweets obtained is limited to 300 tweets per page, which ensures that the results do not go out of bounds. Thus, this is technically feasible.
- **Economic Feasibility:** This project work is economically feasible as it does not consider any additional costs. Whatever data is extracted, it is done without any charges. Twitter provides free use of this data that is non-encrypted and publicly available for analysis purpose. Hence, this work is economically feasible as well.
- **Operational Feasibility:** This is operationally feasible as well. As already mentioned, it takes in 300 tweets per page as that is the limit set by Twitter. Therefore, it is operationally feasible as well. The system won't hang when getting the results.

83. WORK FLOW:

- First, we need to check the health factors based on user-user similarity and outcome-outcome similarity. For that, first we need to calculate the number of unique patients with their data of having diabetes or not.
- Next create a user-outcome matrix which can be used to calculate the similarity between users and outcomes.
- Now, we will check for the similarity.
- This gives us the outcome-outcome and user-user similarity in an array form. The next step is to make predictions based on these similarities
- Finally, we will make predictions based on user similarity and their outcomes similarity.

84. HARDWARE AND SOFTWARE REQUIREMENTS

8.4.1. Hardware Requirements

- Standard computer with at least i3 processor
- Standard computer with 2GB of RAM
- Standard computer with 100GB of free space

8.4.2. Software Requirements

- python 3.8
- Anaconda-3.5
- MS Office
- Draw.io

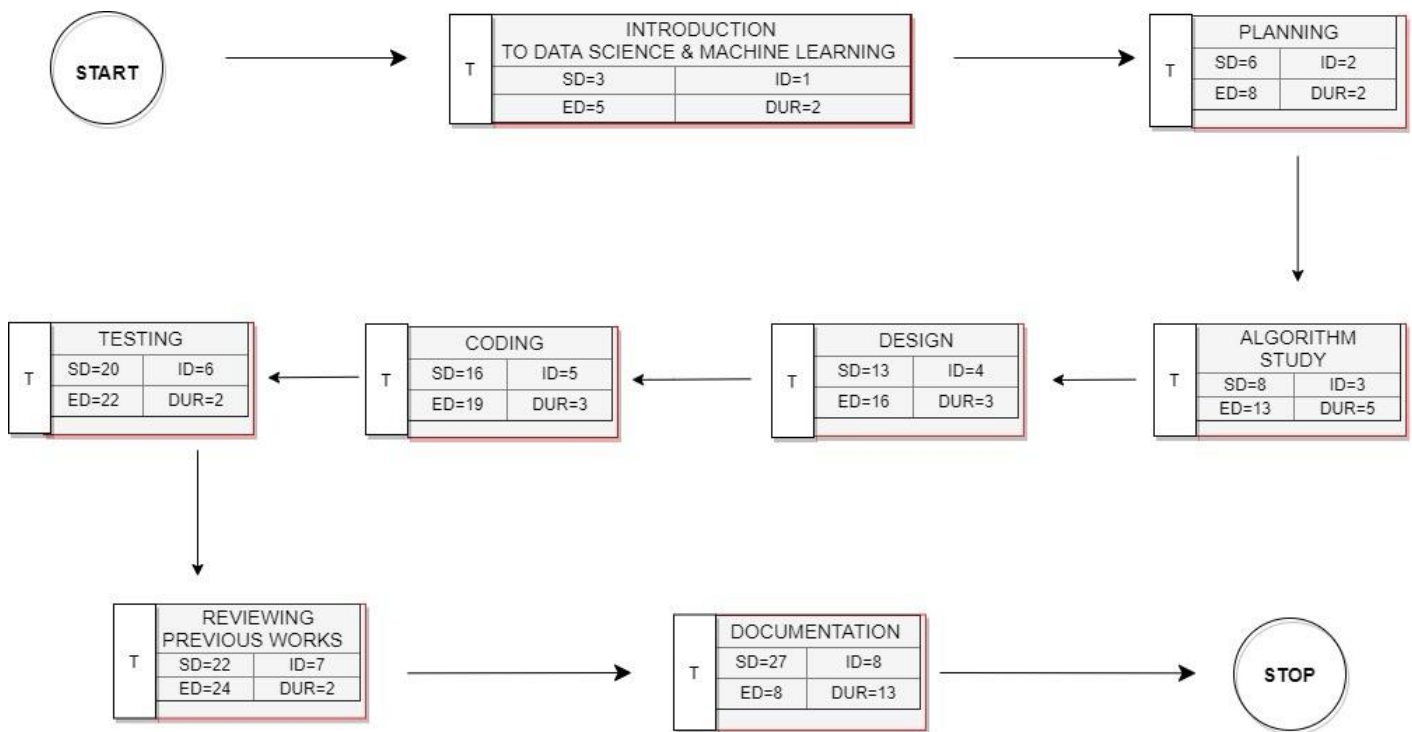
SYSTEM DESIGN

9.1. GANTT CHART

	TASK	START DATE	END DATE	DURATIO N
1	INTRODUCTION TO DATA SCIENCE & MACHINE LEARNING	27 Feb 23	28-Feb 23	2
2	PLANNING	1-mar-23	1-mar- 23	2
3	ALGORITHM STUDY	1-mar-23	1-mar-23	5
4	DESIGN	1-mar-23	1-mar-23	3
5	CODING	2-mar-23	2-mar-23	3
6	TESTING	3-mar-23	3-mar-23	2
7	REVIEWING PREVIOUS WORKS	3-mar-23	3-mar-23	2
8	DOCUMENTATION	4-mar-23	4-mar-23	13

9.2. PERT CHART:

	TASK	START DATE	END DATE	DURATION
1	INTRODUCTION TO DATA SCIENCE & MACHINE LEARNING	27 Feb 23	28-Feb 23	2
2	PLANNING	1-mar-23	1-mar- 23	2
3	ALGORITHM STUDY	1-mar-23	1-mar-23	5
4	DESIGN	1-mar-23	1-mar-23	3
5	CODING	2-mar-23	2-mar-23	3
6	TESTING	3-mar-23	3-mar-23	2
7	REVIEWING PREVIOUS WORKS	3-mar-23	3-mar-23	2
8	DOCUMENTATION	4-mar-23	4-mar-23	13



PERT CHART DIAGRAM

9.3. SEQUENCE DIAGRAM

Sequence diagrams can be useful reference diagrams for businesses and other organizations.

Try drawing a sequence diagram to:

- 9.3.1. Represent the details of a UML use case.
- 9.3.2. Model the logic of a sophisticated procedure, function, or operation.
- 9.3.3. See how tasks are moved between objects or components of a process.
- 9.3.4. Plan and understand the detailed functionality of an existing or future scenario.

9.3.1 Popular Sequence Diagram Uses:

Usage Scenario – A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.**Method Logic** - Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to Usage Scenario - A usage scenario is a diagram of how your system could potentially be used. It's a great explore the logic of any function, procedure, or complex process.

Service Logic - If you consider a service to be a high-level method used by different clients, a sequencediagram is an ideal way to map that out.

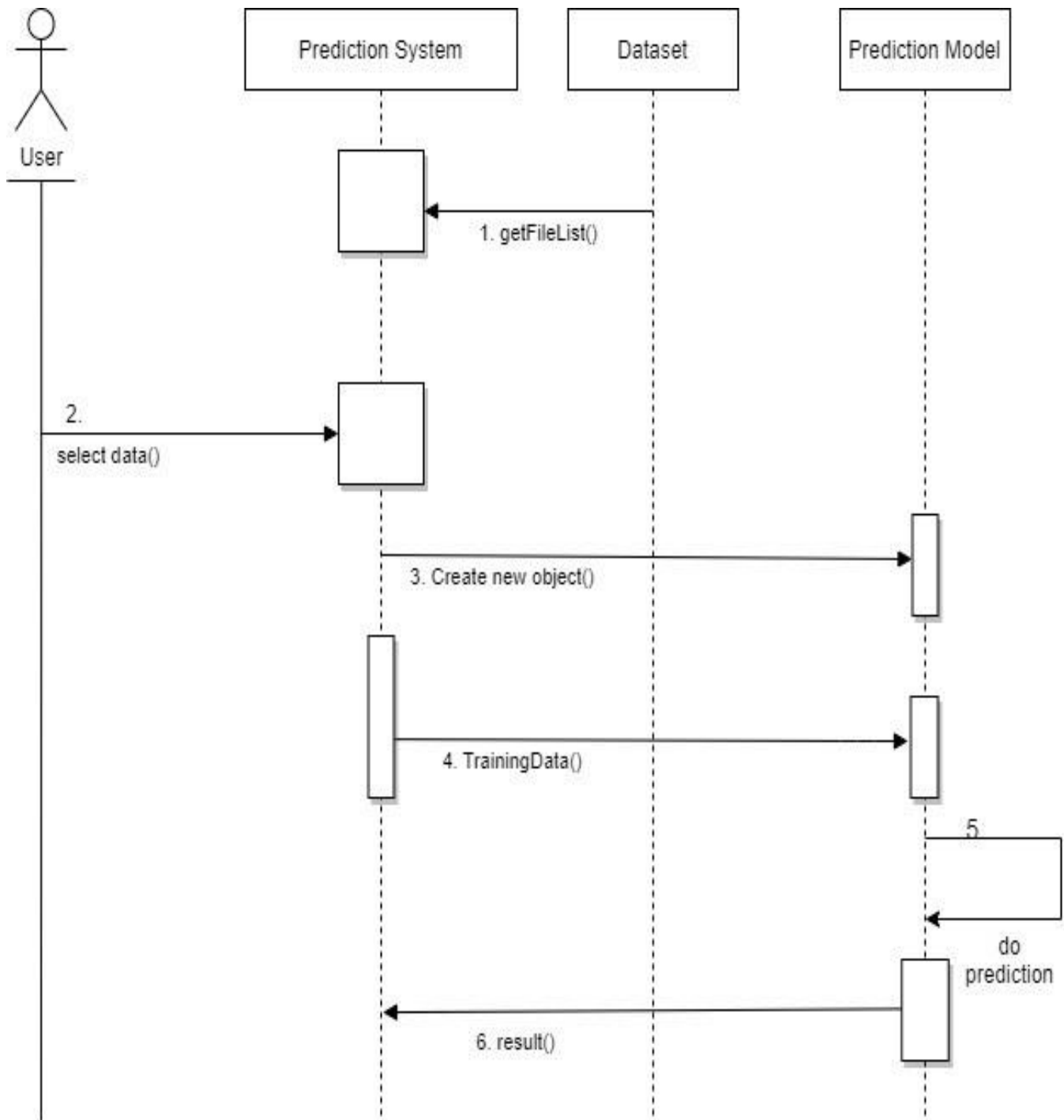


Fig: Sequence Diagram

9.4. ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Purpose of Activity Diagrams

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another, but activity diagram is used to show message flow from one activity to another.

Where to Use Activity Diagrams?

The basic usage of activity diagram is like other four UML diagrams. The specific usage is to model the control flow from one activity to another. This control flow does not include messages.

Activity diagram is suitable for modeling the activity flow of the system. An application can have multiple systems. Activity diagram also captures these systems and describes the flow from one system to another. This specific usage is not available in other diagrams. These systems can be database, external queues, or any other system.

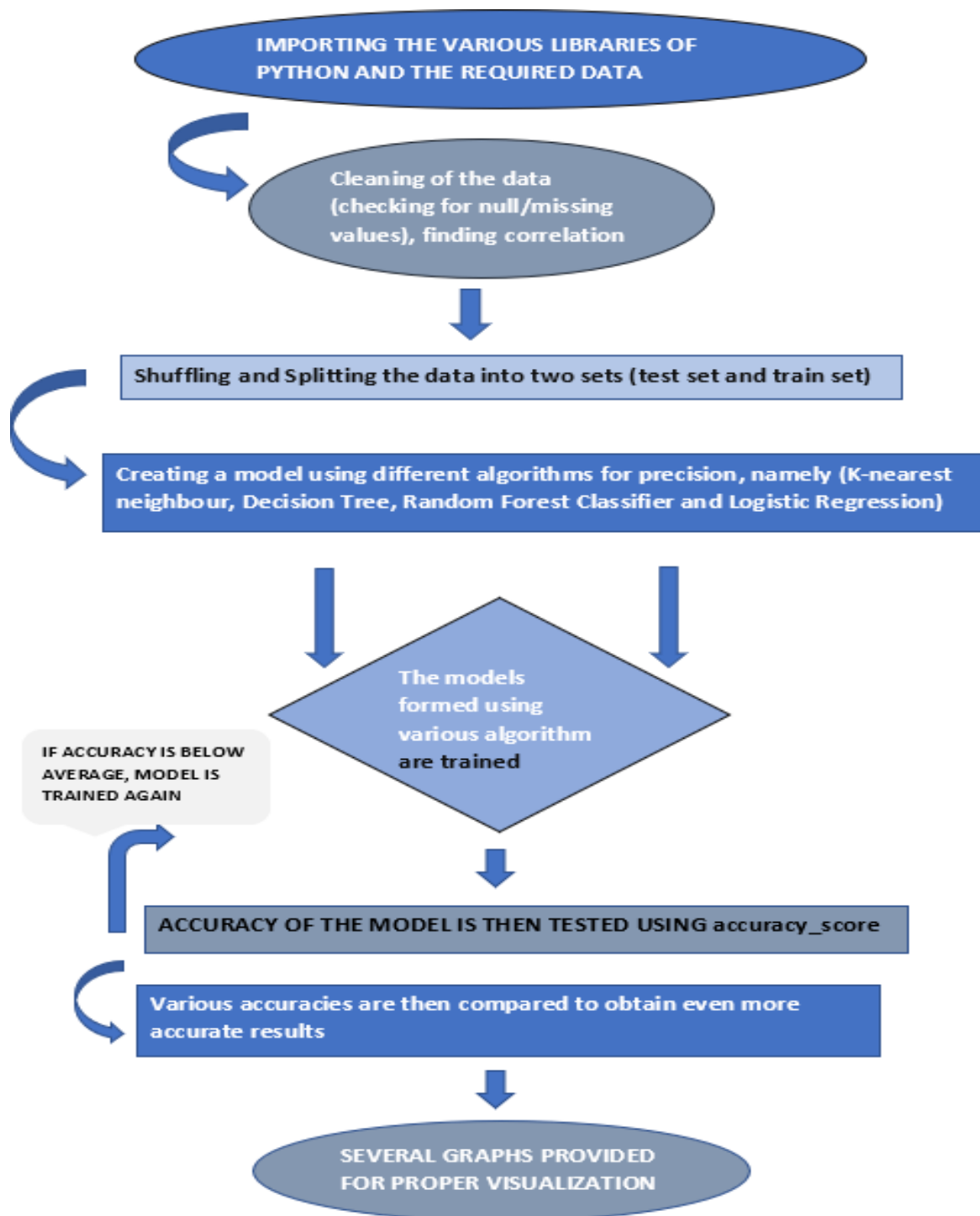


Fig: Activity Diagram

9.5 USE CASE DIAGRAM

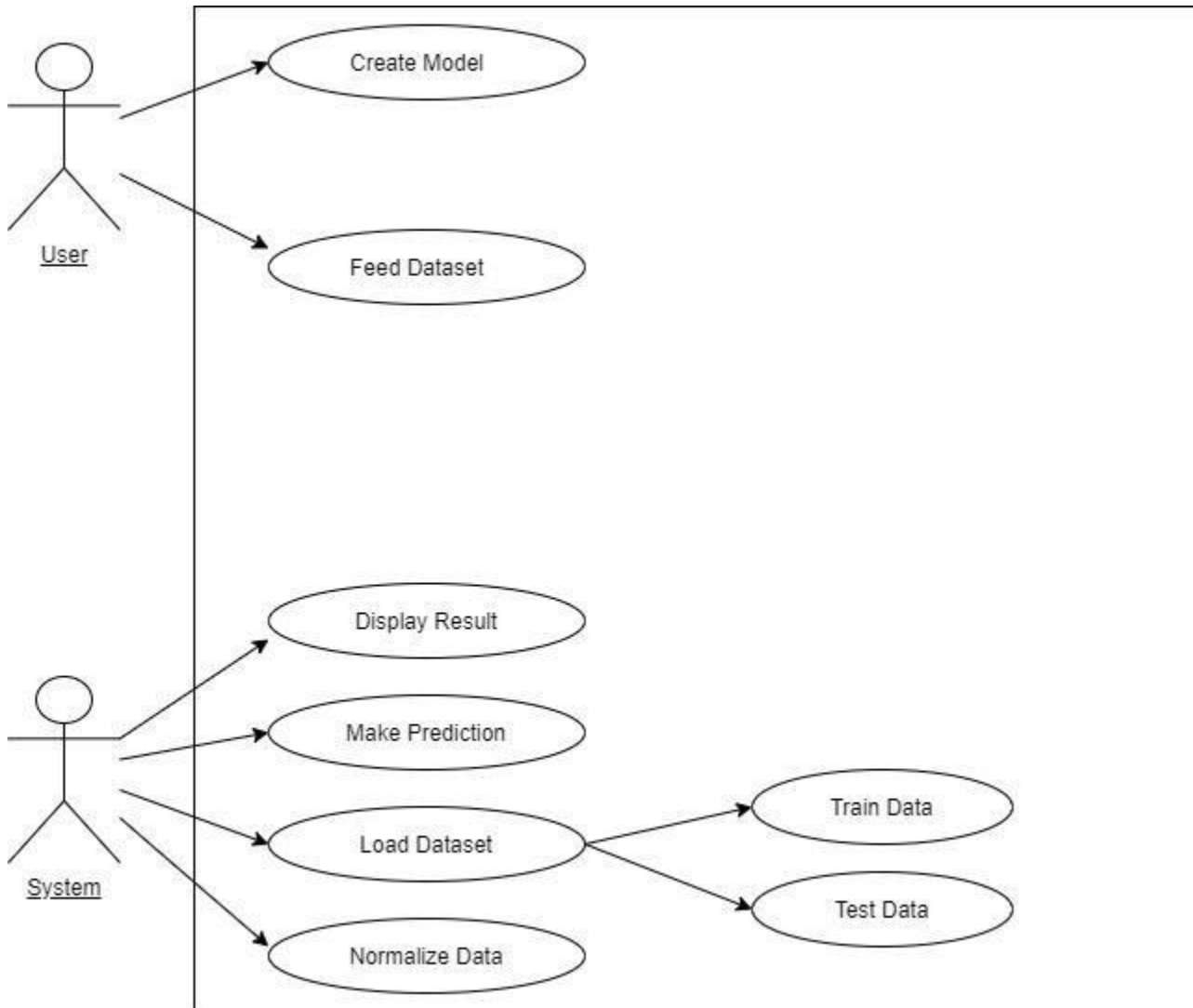


Fig: Use Case Diagram

10. USER INTERFACE DESIGN

1. The design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

2. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Snapshots:

jupyter Untitled19 Last Checkpoint 4 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

In [2]: df = pd.read_csv('diabetes.csv')

In [3]: df.head()

Out[3]:
```

	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

```
In [4]: df.tail()

Out[4]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

jupyter Untitled19 Last Checkpoint 5 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [5]: df.shape
Out[5]: (768, 9)

In [6]: df.size
Out[6]: 6912

In [7]: #checking for null values
df.isnull().sum()

Out[7]:
```

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

```
In [8]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   column              non-null count  dtype
0  Pregnancies         768 non-null  int64
1  Glucose              768 non-null  int64
2  BloodPressure        768 non-null  int64
3  SkinThickness        768 non-null  int64
4  Insulin              768 non-null  int64
```

Jupyter Untitled19 Last Checkpoint 5 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [8]: df.info()
```

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

```
In [9]: df.describe()
```

```
Out[9]:
```

	Pregnancies	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	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.478951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.800000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Jupyter Untitled19 Last Checkpoint 6 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [10]: #used to compute pairwise covariance of columns
df.cov()
```

```
Out[10]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	11.354056	13.947131	9.214538	-4.390041	-28.555231	0.469774	-0.037426	21.570620	0.356618
Glucose	13.947131	1022.248314	94.430956	29.239183	1220.935799	55.726987	1.454875	99.082805	7.115079
BloodPressure	9.214538	94.430956	374.647271	64.029396	190.378412	43.004695	0.264638	54.523453	0.600697
SkinThickness	-4.390041	29.239183	64.029396	254.473245	802.979941	49.373869	0.972136	-21.381023	0.568747
Insulin	-28.555231	1220.935799	190.378412	802.979941	13281.180078	179.775172	7.066681	-57.143290	7.175671
BMI	0.469774	55.726987	43.004695	49.373869	179.775172	62.155984	0.367405	3.360330	1.100638
DiabetesPedigreeFunction	-0.037426	1.454875	0.264638	0.972136	7.066681	0.367405	0.109779	0.130772	0.027472
Age	21.570620	99.082805	54.523453	-21.381023	-57.143290	3.360330	0.130772	138.303046	1.336953
Outcome	0.356618	7.115079	0.600697	0.568747	7.175671	1.100638	0.027472	1.336953	0.227483

```
In [11]: #used to find the pairwise correlation of all columns
df.corr()
```

```
Out[11]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

Jupyter Untitled19 Last Checkpoint 6 minutes ago (autosaved)

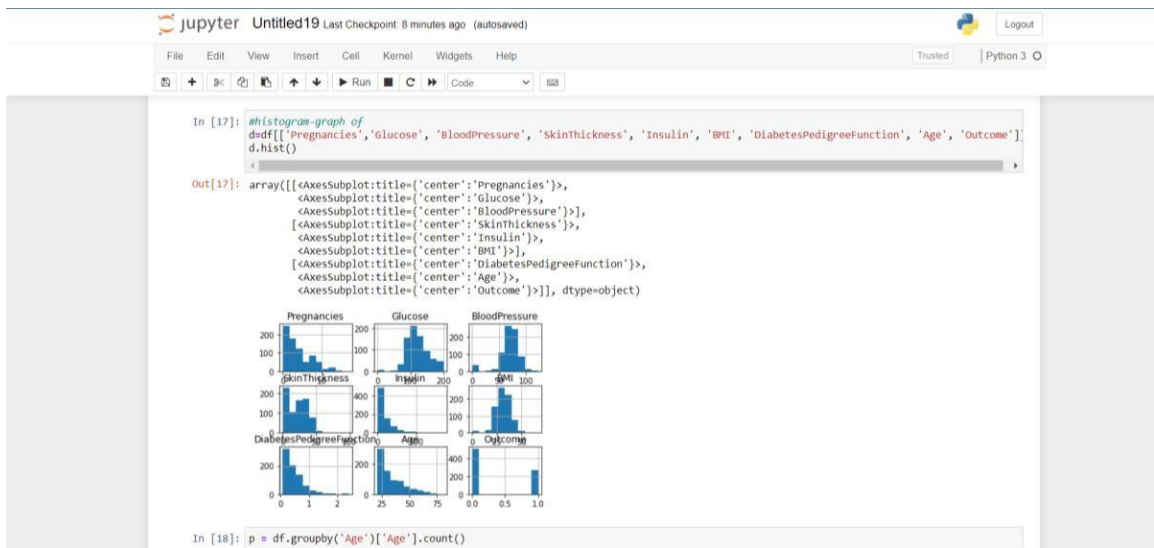
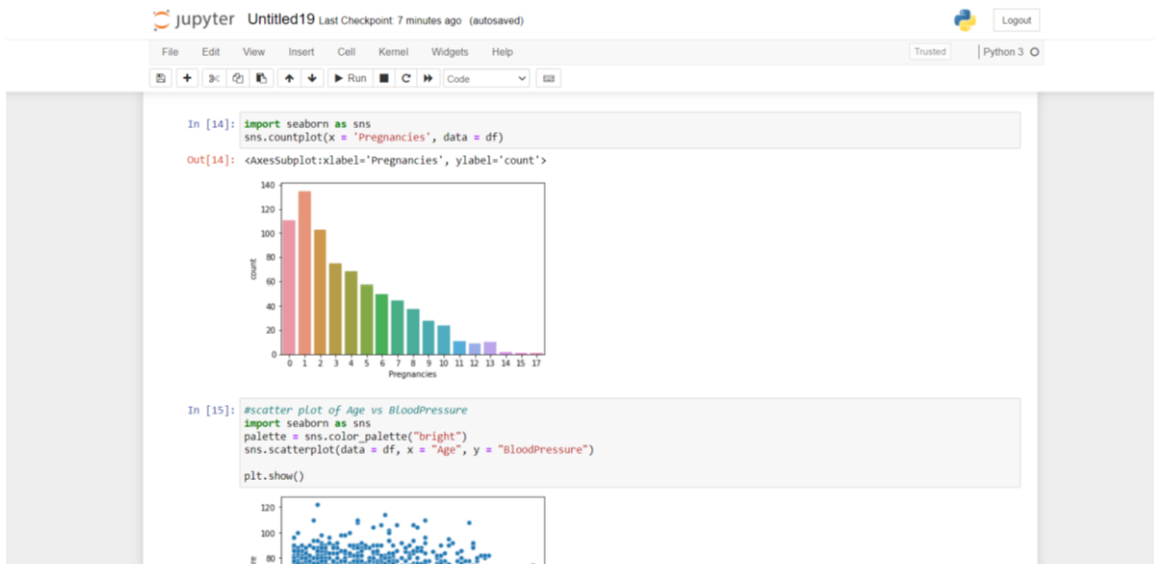
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

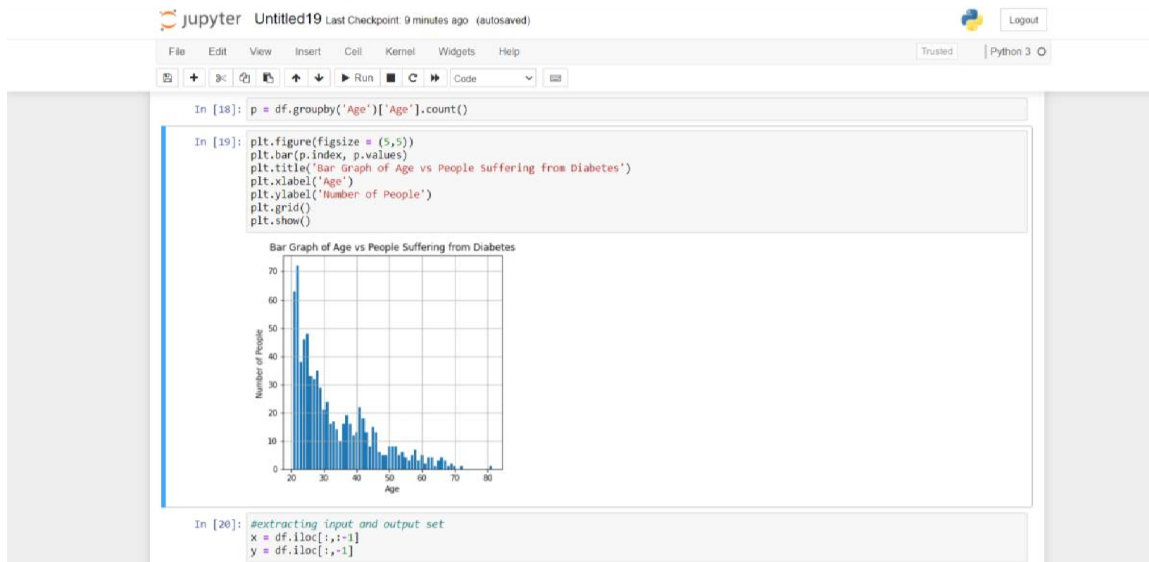
```
In [12]: #counting how many people are suffering and NOT suffering from Diabetes
s = df.groupby('Outcome')['Outcome'].count()
```

```
In [13]: plt.figure(figsize = (5,5))
plt.pie(s.values, labels = ['People NOT suffering from Diabetes', 'People suffering from Diabetes'], autopct = '%1.1f%%')
plt.title('Pie Chart for people suffering from Diabetes')
plt.show()
```

Pie Chart for people suffering from Diabetes

Category	Count	Percentage
People NOT suffering from Diabetes	509	65.1%
People suffering from Diabetes	259	34.9%





jupyter Untitled19 Last Checkpoint: 8 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [22]: #training and testing the dataset
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 = 42)

In [23]: #find the accuracy using DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

model = DecisionTreeClassifier()
model.fit(x_train, y_train)
prediction1 = model.predict(x_test)
accuracy1 = accuracy_score(y_test, prediction1)
accuracy1

Out[23]: 0.7727272727272727

In [24]: #comparing the predicted value using DecisionTree with the actual value
pd.DataFrame({'actual': y_test, 'prediction': prediction1})

Out[24]:
```

	actual	prediction
668	0	1
324	0	0
624	0	0
690	0	0
473	0	0
...
355	1	1
534	0	0
344	0	1

jupyter Untitled19 Last Checkpoint: 10 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [24]: #find the accuracy using RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

model2 = RandomForestClassifier()
model2.fit(x_train, y_train)
prediction2 = model2.predict(x_test)
accuracy2 = accuracy_score(y_test, prediction2)
accuracy2

Out[24]: 0.8116883116883117

In [25]: #comparing the predicted value using RandomForest with the actual value
pd.DataFrame({'actual': y_test, 'prediction': prediction2})

Out[25]:
```

	actual	prediction
576	0	0
257	0	0
0	1	1
49	0	0
108	0	0
...
603	1	1
543	0	0
350	0	0
230	1	1
72	1	1

```
jupyter Untitled19 Last Checkpoint: 10 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [26]: #find the accuracy using LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model3 = LogisticRegression()
model3.fit(x_train, y_train)
prediction3 = model3.predict(x_test)
accuracy3 = accuracy_score(y_test, prediction3)
accuracy3

D:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(

Out[26]: 0.8246753246753247

In [27]: #comparing the predicted value using logistic Regression with the actual value
pd.DataFrame({'actual': y_test, 'prediction': prediction3})

Out[27]:
```

	actual	prediction
576	0	0
257	0	0
0	1	1
49	0	0
108	0	0

```
jupyter Untitled19 Last Checkpoint: 10 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
603 1 1
543 0 0
390 0 0
230 1 1
72 1 1
154 rows x 2 columns

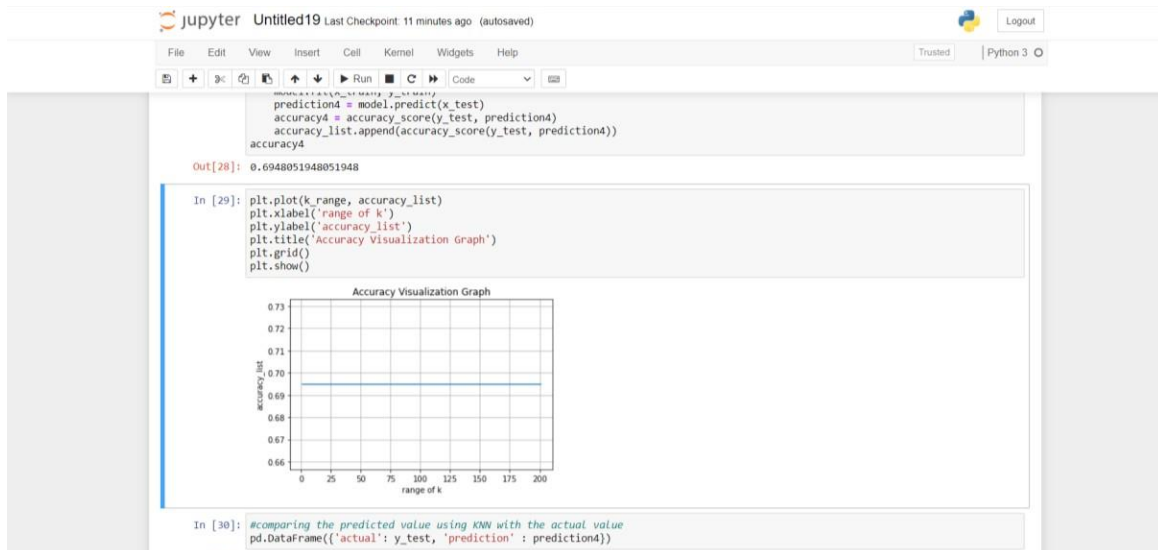
In [28]: #finding the accuracy using KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

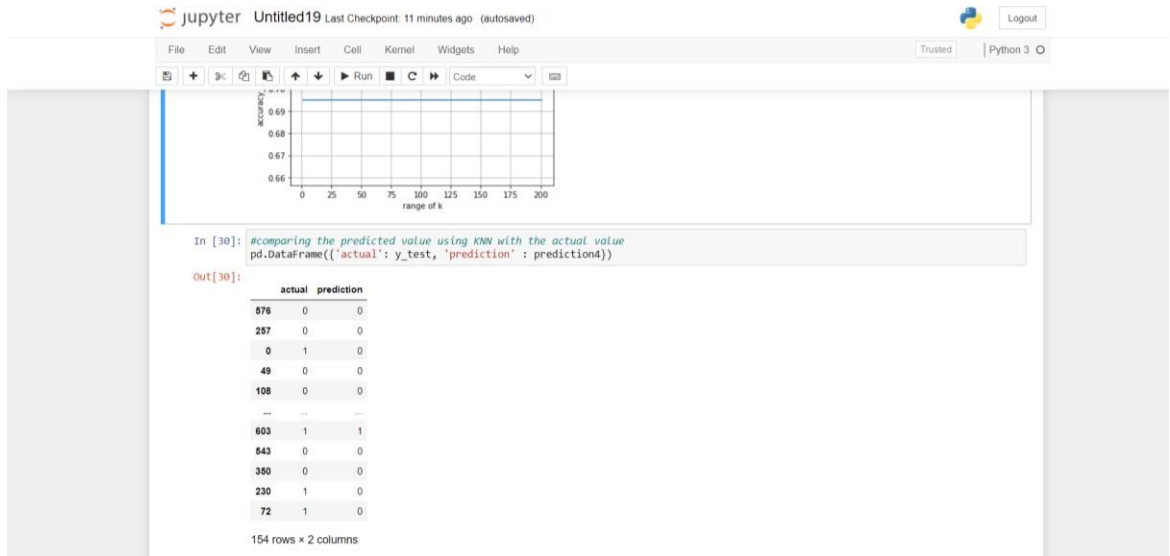
accuracy4 = {}
accuracy_list = []

k_range = (1,201)
for k in k_range:
    model = KNeighborsClassifier()
    model.fit(x_train, y_train)
    prediction4 = model.predict(x_test)
    accuracy4 = accuracy_score(y_test, prediction4)
    accuracy_list.append(accuracy_score(y_test, prediction4))
accuracy4

Out[28]: 0.6948051948051948

In [29]: plt.plot(k_range, accuracy_list)
plt.xlabel('range of k')
plt.ylabel('accuracy_list')
plt.title('Accuracy Visualization Graph')
plt.grid()
plt.show()
```



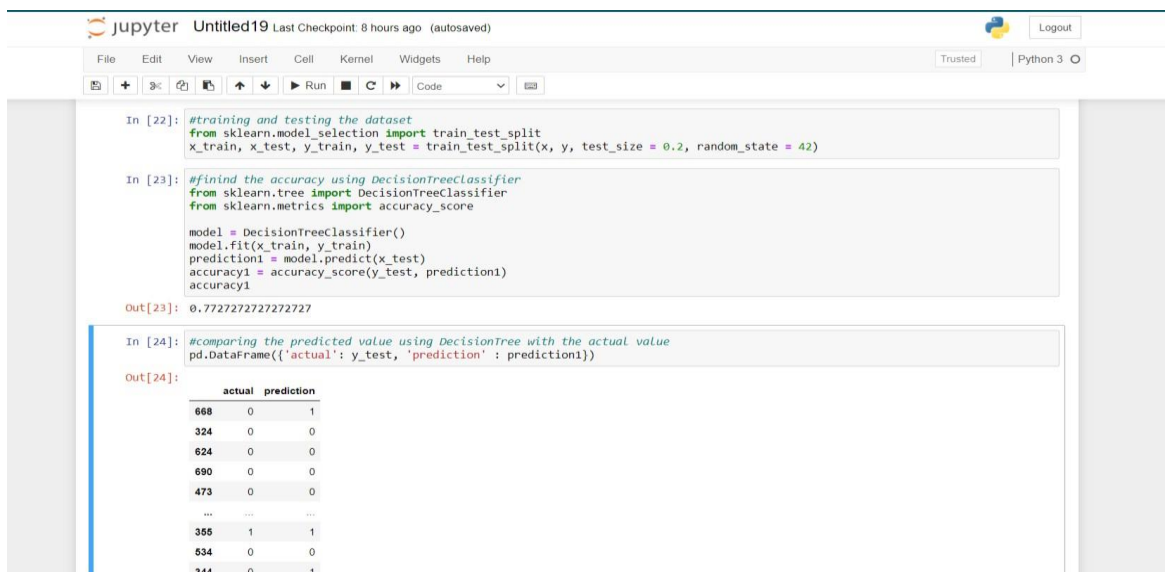


Accuracy Comparison:

To find the accuracy of the dataset, 4 algorithms are used –

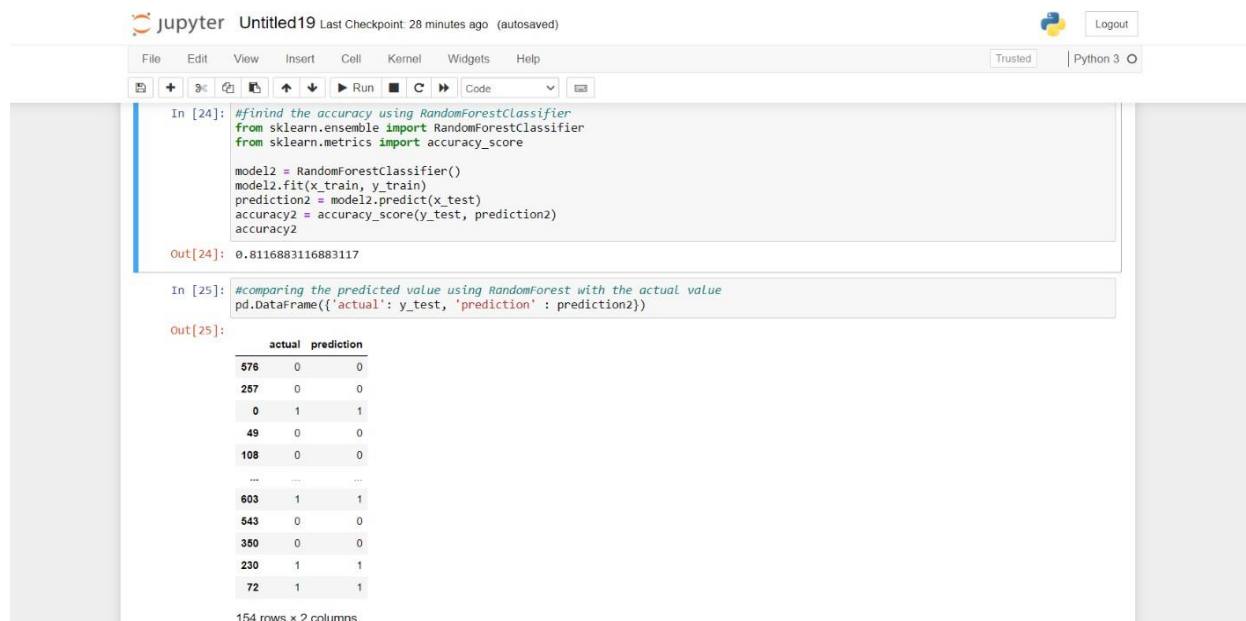
1. Decision Tree
2. Random Forest
3. Logistic Regression
4. KNN

1. Decision Tree:



Accuracy obtained using Decision Tree Classifier is 77.27%

2. Random Forest:



The Jupyter Notebook interface shows the following code and output for the Random Forest Classifier:

```
In [24]: #find the accuracy using RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

model2 = RandomForestClassifier()
model2.fit(x_train, y_train)
prediction2 = model2.predict(x_test)
accuracy2 = accuracy_score(y_test, prediction2)

Out[24]: 0.8116883116883117
```

```
In [25]: #comparing the predicted value using RandomForest with the actual value
pd.DataFrame({'actual': y_test, 'prediction': prediction2})

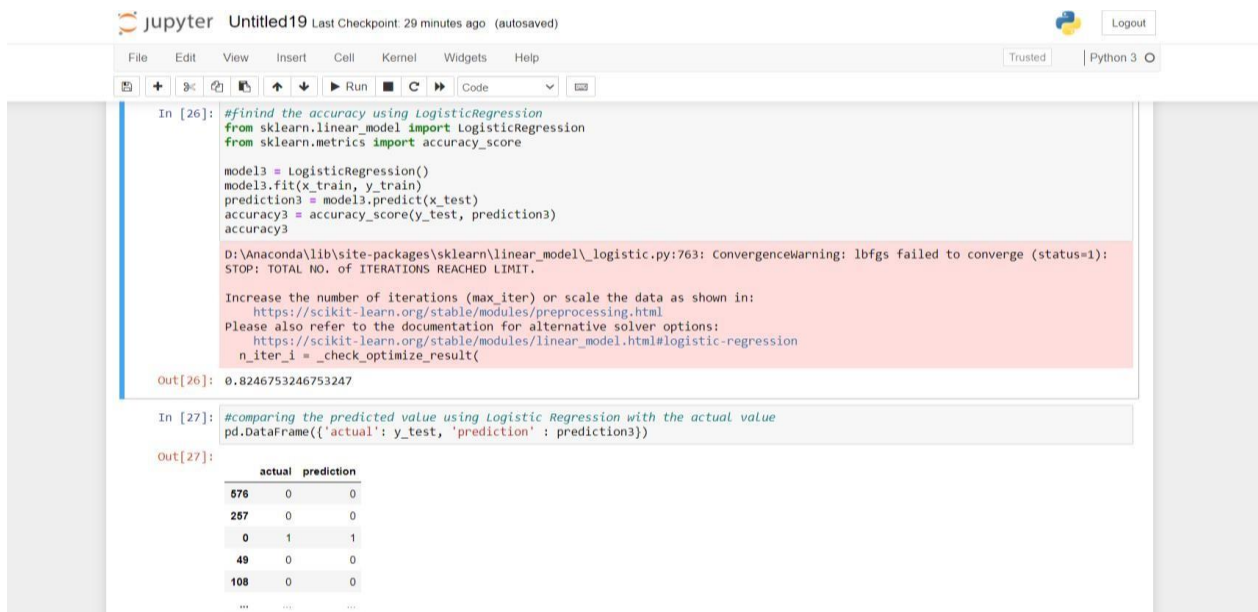
Out[25]:
```

	actual	prediction
576	0	0
257	0	0
0	1	1
49	0	0
108	0	0
...
603	1	1
543	0	0
350	0	0
230	1	1
72	1	1

154 rows x 2 columns

Accuracy obtained using Random Forest Classifier is 81.17%.

3. Logistic Regression:



The Jupyter Notebook interface shows the following code and output for the Logistic Regression model:

```
In [26]: #find the accuracy using LogisticRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model3 = LogisticRegression()
model3.fit(x_train, y_train)
prediction3 = model3.predict(x_test)
accuracy3 = accuracy_score(y_test, prediction3)

Out[26]: 0.8246753246753247
```

A warning message is displayed:

```
D:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
  https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
  https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
```

```
In [27]: #comparing the predicted value using Logistic Regression with the actual value
pd.DataFrame({'actual': y_test, 'prediction': prediction3})

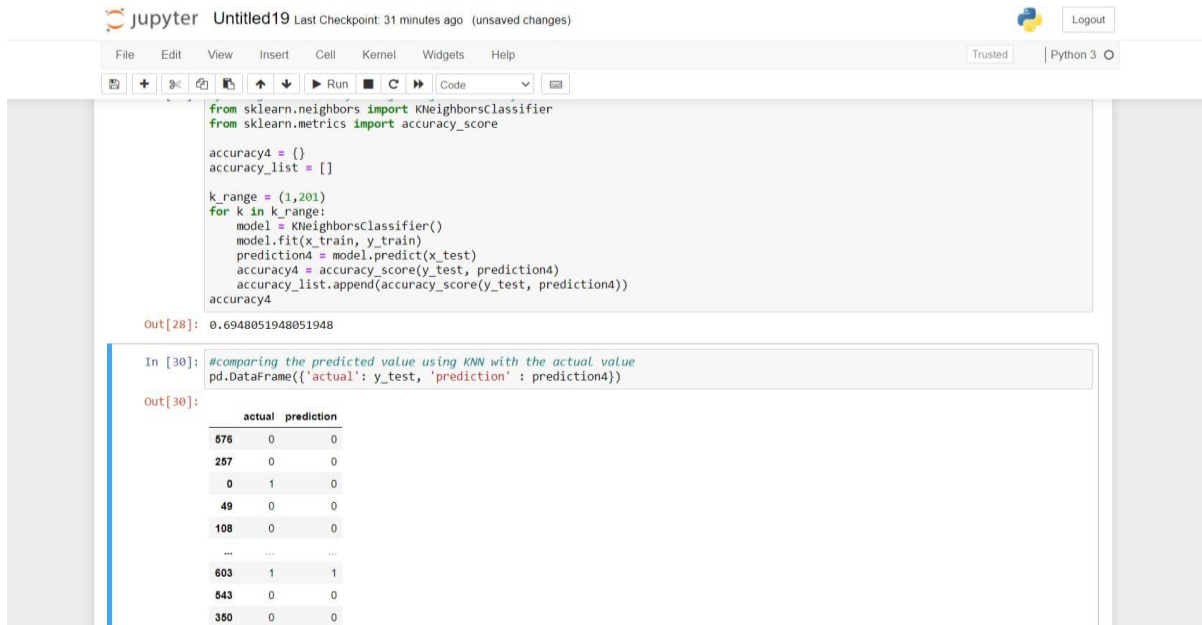
Out[27]:
```

	actual	prediction
576	0	0
257	0	0
0	1	1
49	0	0
108	0	0
...

Accuracy obtained using Logistic Regression is 82.47%.

4. Kneighbors Classifier:

Accuracy obtained using Kneighbors Classifier is 69.48%.



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

accuracy4 = {}
accuracy_list = []

k_range = (1,201)
for k in k_range:
    model = KNeighborsClassifier()
    model.fit(x_train, y_train)
    prediction4 = model.predict(x_test)
    accuracy4[k] = accuracy_score(y_test, prediction4)
    accuracy_list.append(accuracy_score(y_test, prediction4))

accuracy4
```

Out[28]: 0.6948051948051948

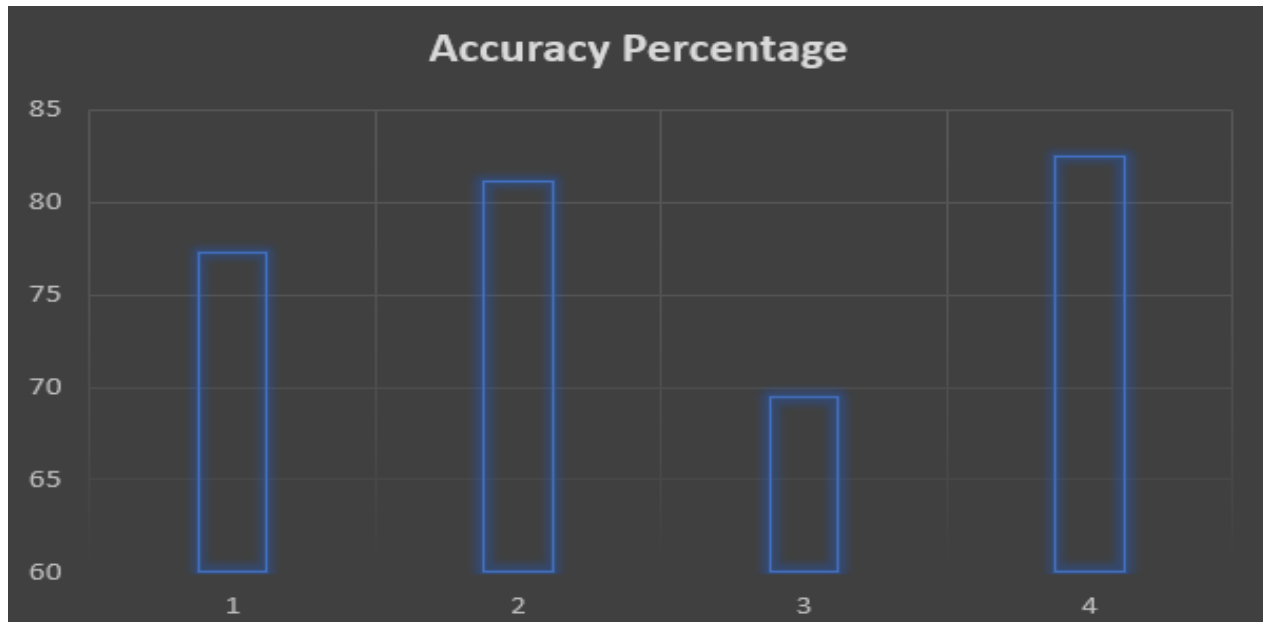
```
In [30]: #comparing the predicted value using KNN with the actual value
pd.DataFrame({'actual': y_test, 'prediction': prediction4})
```

Out[30]:

	actual	prediction
576	0	0
257	0	0
0	1	0
49	0	0
108	0	0
...
603	1	1
543	0	0
350	0	0

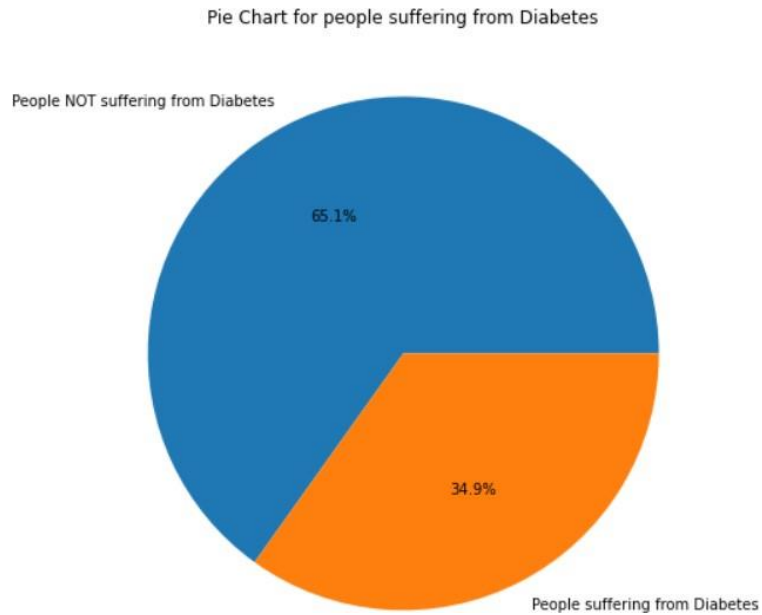
11. Accuracy Comparison Table –

Sl. No.	Algorithms	Accuracy Obtained (%)
1	Decision Tree Classifier	77.27
2	Random Forest	81.17
3	K-Neighbors Classifier	69.48
4	Logistic Regression	82.47

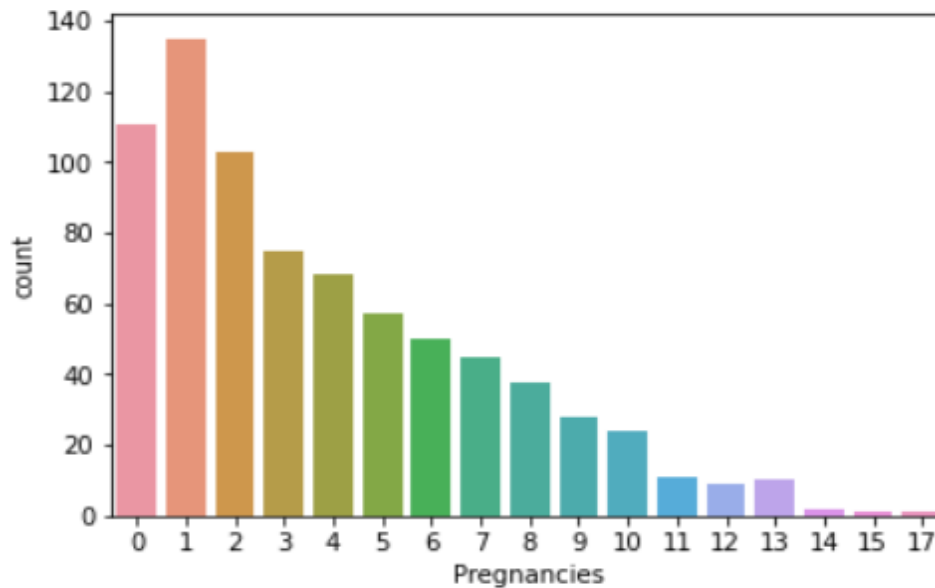


12. Visualization Graph:

12.1. Pie Chart of People Suffering and NOT Suffering from Diabetes –

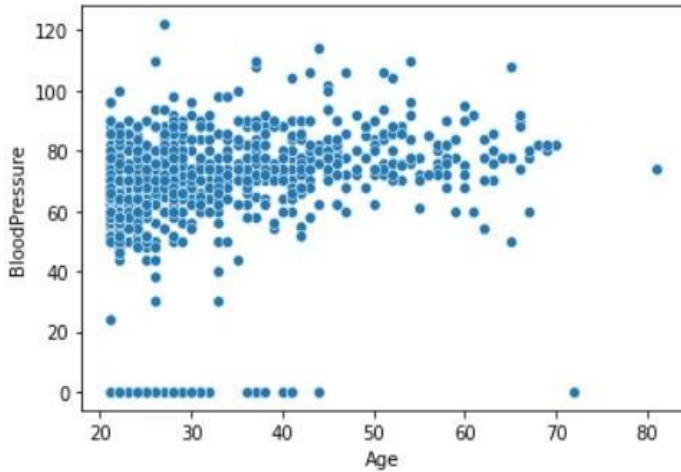


12.2. Count Plot of Pregnancies –

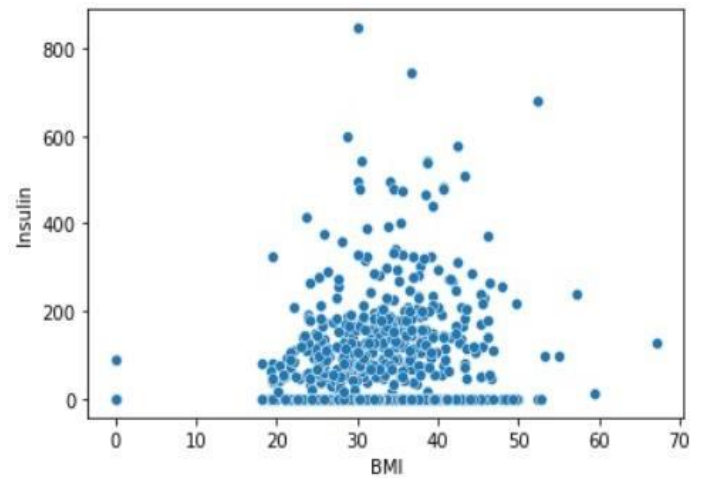


12.3. Scatter Plot of –

a) *Age vs Blood Pressure -*

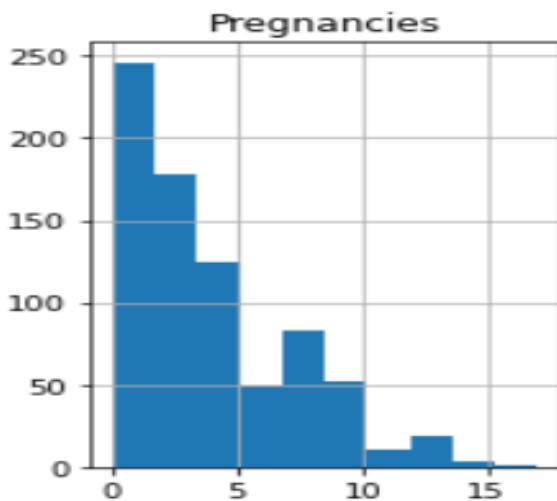


b) *BMI vs Insulin -*

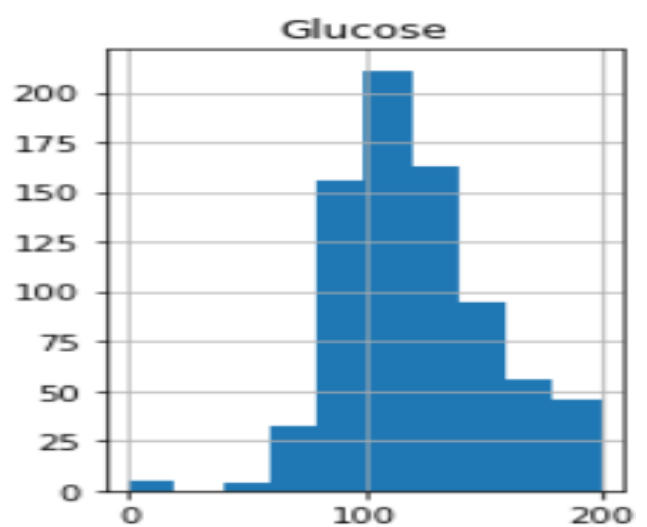


12.4. Histogram Graph of –

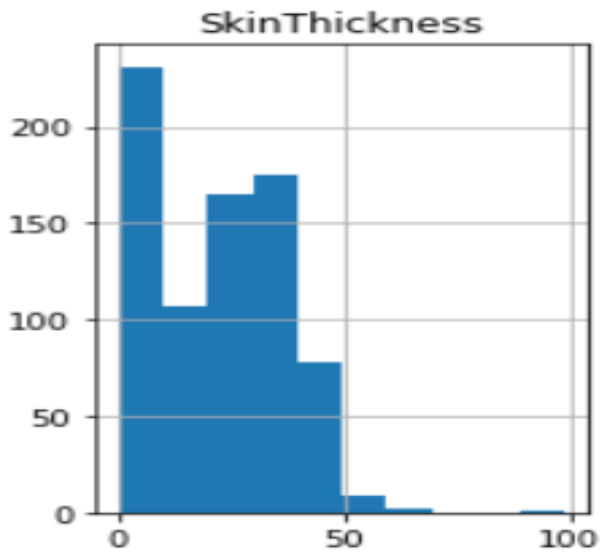
a) Pregnancies -



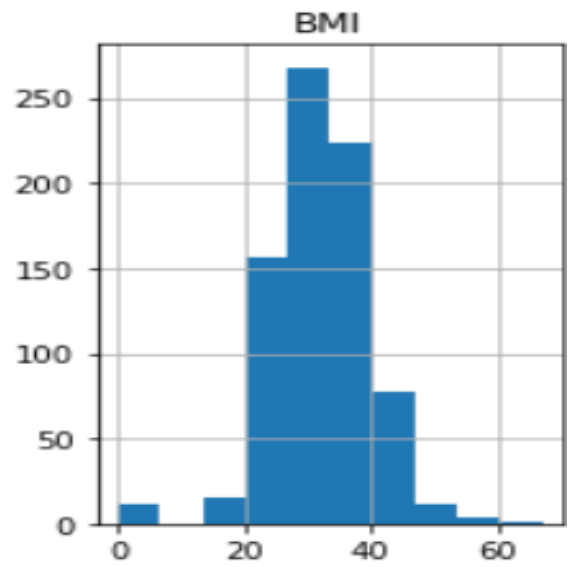
b) Glucose -



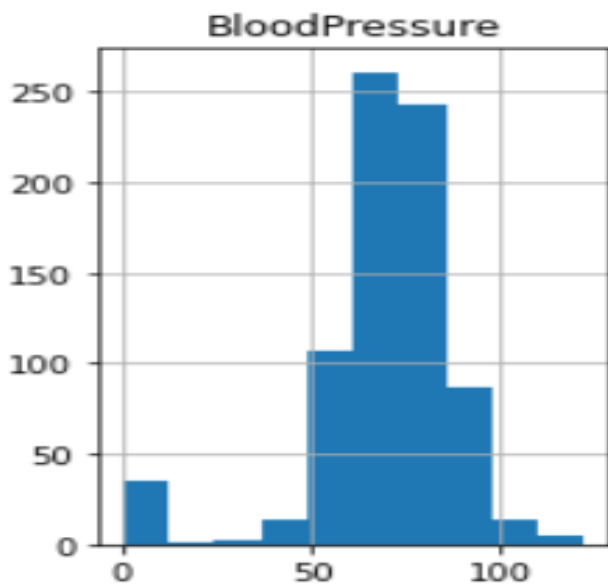
c) Skin Thickness -



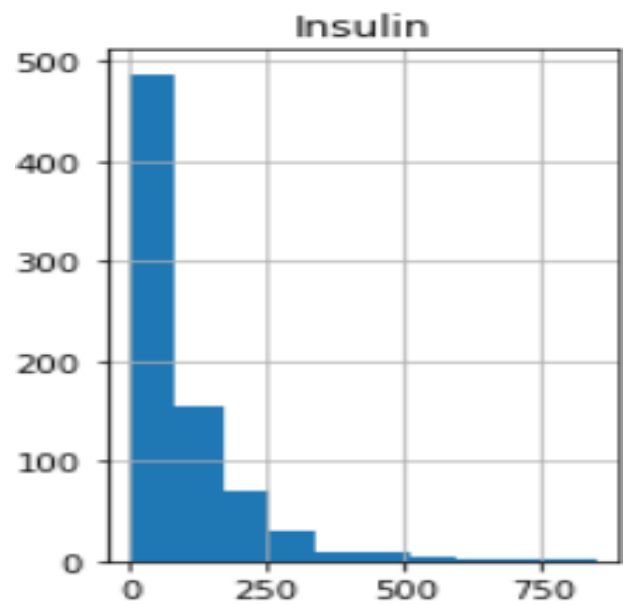
d) BMI -



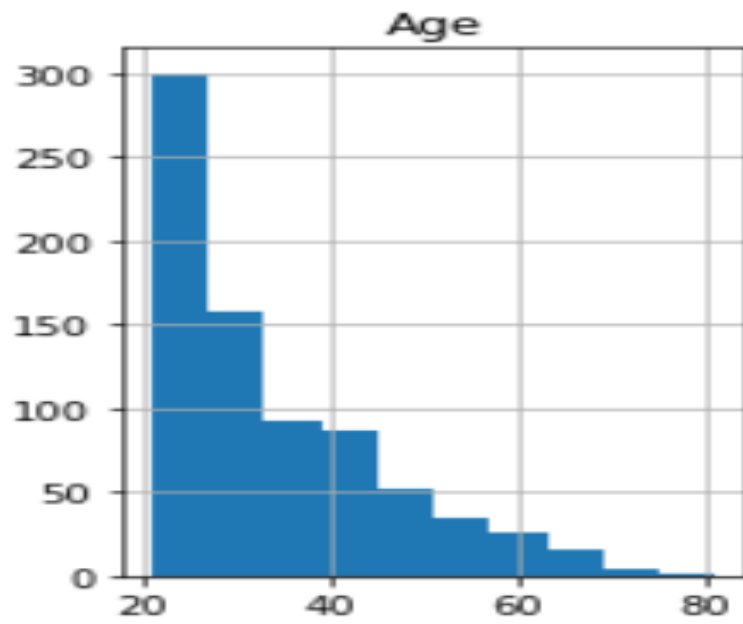
e) Blood Pressure -



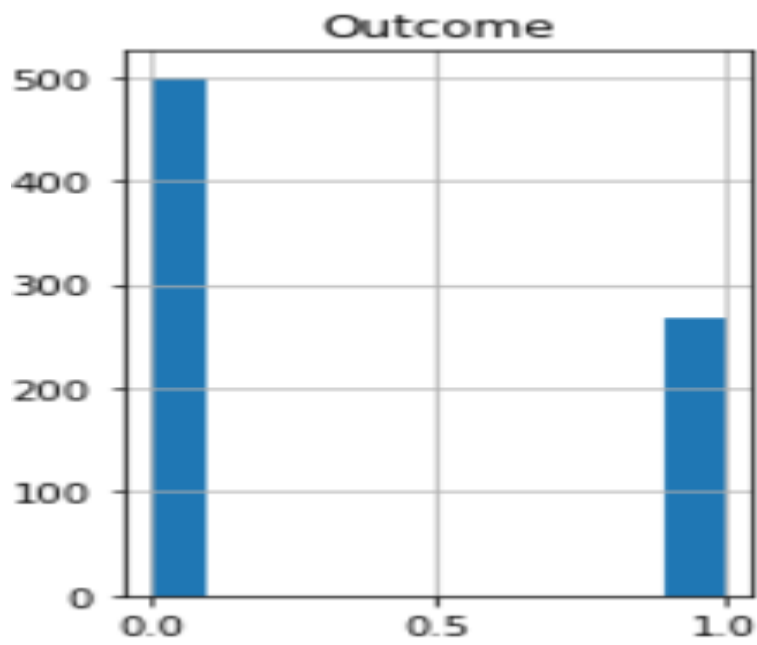
f) Insulin -



g) Age -



h) Outcome -



13. IMPLEMENTATION AND TESTING

A software system test plan is a document that describes the objectives, scope, approach and focus of software testing effort. The process of preparing a test plan is a usual way to think the efforts needed to validate the acceptability of a software product. The complete document will help people outside the test group understand the "WHY" and "HOW" product validation. It should be thorough enough to be useful but not so thorough that no one outside the test group will read it.

13.1. Introduction

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements.

The main purpose of testing is to detect errors and error prone areas in a system. Testing must be through well planned. A partially tested system is to detect errors and error prone areas in a system. Testing must be through well planned. A partially tested system is as bad as an untested system. And the price of an untested and under tested system is high.

13.2. Objectives Of Testing

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

13.3. Process Overview

The following represents the overall flow of the testing process:

- Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
- Identify which test(s) will be used to test each module.
- Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

13.4. Test Cases

A test case is a document that describes an input, action, or event and expected response, to determine if a feature of an application is working correctly. A test case should contain such as test case identifier, test condition, input data.

Requirement expected results. The process of developing test cases can help find problems in the requirements or design of an application since it requires completely thinking through the operations of the application.

13.5. Testing Steps

- **Unit Testing**

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

- **Integration Testing**

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface.

13.6. Validation

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

- a) The function or performance characteristics confirm to specification and are accepted.
- b) A deviation from specification is uncovered and a deficiency list is created.
- c) Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

Tested By:	Faizan Mahfooz , Noor Afsha		
Test Type	Unit Testing		
Test CaseNumber	1		
TestCaseName	Diabetes Prediction Model		
Test Case Description	The data is loaded in the form of csv file and the model will check different parameters (pregnancy, age, insulin, glucose, skin thickness, diabetes pedigree function, BMI, blood pressure) and whether the person is diabetic or not from the data that was imported. Then it will consider all the parameters and will give predictions whether the person is diabetic or not. We are checking for the accuracy of the model using accuracy_score which we have imported from sklearn.metrics. If we find that the accuracy is not good enough then we go back to shuffling and splitting the data again into training and testing sets for more accurate results.		
Item(s) to be tested			
1		Checking for the accuracy of the predictions made by the model	
Specifications			
Input		Expected Output/Result	
1) Loading and importing the dataset		1) Prediction whether the person is diabetic or not	
2) Checking for the correlation of the parameters		2) Establishing an accurate model	

13.7. White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

13.8. Black box testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

13.9. System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity.

13.10. Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also, the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

13.11. User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required.

This is done regarding the following point:

- [1] Input Screen Design.
- [2] Output Screen Design.
- [3] Format of reports and other outputs

13.12. Integration Testing

Software testing is always used in association with verification and validation. In the testing phase of this project our aim is to find the answer to following two questions.

- Whether the software matches with the specification (i.e., Process base) to verify the product.
- Whether this software in one client what wants (i.e., product base) to validate the product.
- Unit testing and integration testing has been carried out to find the answer to above questions.

In unit testing each individual module was tested to find any unexpected behavior if exists. Later all the module was integrated, and flat file was generated.

13.13. Functional Testing

These are the points concerned during the stress test:

- Nominal input: character is in putted in the place of digits and the system has to flash the message "Data error"
- Boundary value analysis: exhaustive test cases have designed to create an output report that produces themaximum (and minimum) allowable number of table entries.

14.SYSTEM SECURITY MEASURES

14.1. Database Security

System security measure is meant to be provided to make your system reliable and secured from unauthorized user may create threats to the system. So, you should follow some security measures. We have used security levels in database level at system level.

14.2. System Security

If we talk about the system security in our proposed system, we have implemented with the help of maintain the session throughout the system's use. Once a user has logged out than he/she will not be able to perform any task before signing back again.

A high level of authentic login is given to the system so this is a very tedious task to enter without authorizationand authentication.

15. COST ESTIMATION

The **Constructive Cost Model (COCOMO)** is a procedural **software cost estimation model** developed by **Barry W. Boehm**. Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

Types of COCOMO

1. Basic COCOMO

Basic COCOMO computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC).

COCOMO applies to three classes of software projects:

- Organic projects - small teams with good experience working with less than rigid requirements.
- Semi-detached projects - medium teams with mixed experience working with a mix of rigid and less than rigid requirements.
- Embedded projects - developed within a set of tight constraints. It is also combination of organic and semi-detached projects. (Hardware, software, operational, ...)

The basic COCOMO equations take the form: Effort Applied (E) =

$$a * (KLOC)^b [\text{man-months}] \text{ Development Time (D)} = c * (\text{Effort Applied})^d [\text{months}]$$

$$\text{People required (P)} = \text{Effort Applied} / \text{Development Time}$$

Time [count] Where,

KLOC is the estimated number of delivered lines (expressed in thousands) of code for project. The coefficients a, b, c and d are given in the following table:

Software project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick estimate of software costs. However, it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

2. Intermediate COCOMO

Intermediate COCOMO computes software development effort as function of program size and a set of cost drivers that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four cost drivers, each with a number of subsidiary attributes: -

Product attributes:

- 1) Required software reliability
- 2) Size of application database
- 3) Complexity of the

productHardware attributes:

- 1) Run-time performance constraints
- 2) Memory constraints
- 3) Volatility of the virtual machine environment
- 4) Required turnabout

timePersonnel attributes

- 1) Analyst capability
- 2) Software engineering capability

- 3) Applications experience
- 4) Virtual machine experience
- 5) Programming language

experienceProject attributes

- 1) Use of software tools
- 2) Application of software engineering methods
- 3) Required development schedule

Each of the 15 attributes receives a rating on a six-point scale that ranges from very low to extra high (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4. The Intermediate COCOMO formula now takes the form:

$$E=a*(KLOC)^b(EAF)$$

here,

E is the effort applied in person-months, KLOC is the estimated number of thousands of delivered lines of code for the project, and EAF is the factor calculated above. The coefficient a and the exponent b is given in the table:

Software project	a	B
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

The Development time D calculation uses P in the same way as in the Basic COCOMO.

3. Detailed COCOMO

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. These Phase Sensitive effort multipliers are each to determine the amount of effort required to complete each phase. In detailed COCOMO, the whole software is divided in different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

In detailed COCOMO, the effort is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle. A Detailed project schedule is never static. The Six phases of detailed COCOMO are: -

- Plan and requirement.
- System design.
- Detailed design.
- Module code and test.
- Integration and test.
- Cost Constructive model

Detailed cost estimation of Diabetes Prediction Model

So, by considering all the facts, if we calculate the cost, it will be like.

The OES is having 2.1 Kilo Lines of Code. According to the COCOMO model the comparison is:

Mode	Project Size	Innovation	Deadline	Development Environment
Organic	Typically, 2-50 KLOC	Little	Not tight	Familiar and in house
Semi-detached	Typically, 50-300 KLOC	Medium	Medium	Medium
Embedded	Typically, over 300 KLOC	Significant	Tight	Complex hardware/Customer interface is required

The project has 2.1 KLOC, so it's under **organic** category. Putting the facts in the formulas,

$$\text{Effort Applied (E)} = a * (\text{KLOC})^b \text{ [person-months]}$$

$$= 2.4 * (2.1)^{1.05} \text{ PM}$$

$$= 5 \text{ PM}$$

$$\text{Development Time (D)} = c * (\text{Effort Applied})^d \text{ [months]}$$

$$= 2.5 * (5)^{0.38} \text{ M}$$

$$= 4.6 \text{ M}$$

16. FUTURE SCOPE AND FURTHER ENHANCEMENTS

The Diabetes Prediction Model has a bright future in detecting and predicting early onset of diabetes in users through comparative analysis for a larger set of diabetic datasets. We might need a smarter and more accurate hybrid predictive analytics system, as the hybrid approaches yield better results than single classifiers, but nevertheless, this could act as a path to a better use of technology especially Machine Learning and Data Analytics in the field of medical and diagnostic sciences. Moreover, the prediction model can also be used to predict and detect other health issues and concerns, communicable/non-communicable disease such as various cancers and cardiovascular diseases, by obtaining user's medical reports.

17. CONCLUSION

The model was designed to implement accurate diabetic prediction using ML methods. We found that Logistic Regression was the most accurate as compared to the rest methods with an 82.47% classification accuracy. Machine learning can revolutionize the diabetes risk prediction with the help of advanced computational methods and availability of large amount of epidemiological and genetic diabetes risk dataset. This technique also has the potential to help researchers develop a far more accurate, effective, and fool-proof tool that will help health-care system to make better decision about the disease status. These experimental data can assist healthcare and may lead to cure diabetes and other diseases, to save precious human life.

18. BIBLIOGRAPHY

- Diabetes (who.int)
- GeeksforGeeks
- <https://www.javatpoint.com/>
- <https://numpy.org/>
- www.embitel.co.in
- www.marutitech.com