DIABETES MELLITUS PREDICTION USING MACHINE LEARNING

A project report submitted in partial fulfilment of the requirements for the award of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Swetha Annavarapu (16011A0537)
Afreen Sultana T (16011A0538)
Yashaswi Veerepalli(16011A0546)

Under the guidance of

Prof. Dr. D. Vasumathi

Department of Computer Science and Engineering
Jawaharlal Nehru Technological University Hyderabad
JNTUH College of Engineering, Hyderabad – 500085
2019

Department of Computer Science and Engineering
Jawaharlal Nehru Technological University Hyderabad
JNTUH College of Engineering, Hyderabad – 500085

DECLARATION

We, Swetha A (16011A0537), Afreen Sultana (16011A0538), Yashaswi V (16011A0546) , here by certify that the project reported entitled "DIABETES MELLITUS PREDICTION USING MACHINE LEARNING" carried out under the guidance of Dr.D.Vasumathi, is submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. This is a record of bonafide work carried out by me and the results embodied in this project have not been reproduced/ copied from any source and have not been submitted to any other University or Institute for the award of any other degree or diploma.

Swetha Annavarapu(16011A0537)
Afreen Sultana T (16011A0538)
Yashaswi Veerepalli (16011A0546)

Department of Computer Science & Engineering,
JNTUH College of Engineering,
Hyderabad.

Date:

Department of Computer Science and Engineering

Jawaharlal Nehru Technological University Hyderabad

JNTUH College of Engineering, Hyderabad – 500085

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report entitled "DIABETES MELLITUS PREDICTION USING MACHINE LEARNING", being submitted by , Swetha A (16011A0537), Afreen Sultana (16011A0538), Yashaswi V (16011A0546) in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by them. The results are verified and found satisfactory.

Dr.D.Vasumathi,
Professor,
Department of Computer Science and Engineering,
JNTUH College of Engineering,
Hyderabad.

Date:

Department of Computer Science and Engineering

Jawaharlal Nehru Technological University Hyderabad

JNTUH College of Engineering, Hyderabad – 500085

CERTIFICATE BY THE HEAD

This is to certify that the project report entitled "DIABETES MELLITUS PREDICTION USING MACHINE LEARNING", being submitted by , Swetha A (16011A0537), Afreen Sultana (16011A0538), Yashaswi V (16011A0546) , in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by them. The results are verified and found satisfactory.

Dr . R. Sridevi,
Professor & Head of the Department,
Department of Computer Science and Engineering,
JNTUH College of Engineering,
Hyderabad.

Date:

ACKNOWLEDGEMENT

We would like to express sincere thanks to our Supervisor Dr.D.Vasumathi ma’am, Professor for her admirable guidance and inspiration both theoretically and practically and most importantly for the drive to complete project successfully. Working under such an eminent guide was our privilege.

We owe a debt of gratitude to Dr.R.Sridevi ma’am, Professor & Head of the department of Computer Science & Engineering, for her kind considerations and encouragement in carrying out this project successfully.

We are grateful to the Project Review Committee members and Department of Computer Science & Engineering who have helped in successfully completing this project by giving their valuable suggestions and support.

We express thanks to our parents for their love, care and moral support without which we would have not been able to complete this project. It has been a constant source of inspiration for all our academic endeavours. Last but not the least, we thank the almighty for making us a part of the world.

Swetha Annavarapu (16011A0537)
Afreen Sultana T (16011A0538)
Yashaswi Veerepalli (16011A0546)

ABSTRACT

Diabetes mellitus (DM) is a chronic disease that is characterized by high blood glucose Nearly half of all diabetics have household heredity factors, which is one of the most important features of DM. Failure of the pancreas to produce enough insulin and the body's inefficient use insulin are both pathologic causes of DM. There are two types of DM. The pathogenesis of type 1 diabetes mellitus (T1DM) is that the pancreas secretes damaged β-cells, preventing it from lowering blood glucose level in time. Insulin resistance and insulin secretion deficiency are the pathogeneses of type 2 diabetes mellitus(T2DM), which is also called non-insulin dependent DM.
Keywords— Missing data, Data cleaning, Model Fitting.

OBJECTIVES

This website predicts the diabetes of the person using pima Indian diabetes dataset. Firstly we perform data cleaning by using some statistical manipulations like replacing the attributes of zero value with NA then replacing NA with mean of the attribute. Then we fit a model to the dataset. Initially we fit different models to the cleaned dataset and finally chooses one model which gives high prediction for that dataset. Now we start developing front end of the website with python so that web application will be up to date and easily adoptable.

System Requirements:

Software Requirements:-

➢ Operating system	:	Windows XP/7/10.
➢ Coding Language	:	Python
➢ Frame work	:	Flask 2.0, Anaconda 3.7

Hardware Requirements:-

➢ System	:	Pentium IV 2.4 GHz.
➢ Hard Disk	:	40 GB.
➢ Floppy Drive	:	1.44 Mb.
➢ Ram	:	4 Gb.

CONTENTS

1. INTRODUCTION

1.1 Introduction

Diabetes mellitus is a heterogeneous group of diseases characterized by chronic elevation of glucose in the blood. It arises because the body is unable to produce enough insulin for its own needs, either because of impaired insulin secretion, impaired insulin action, or both. Diabetes affects some 300 million people world-wide and is on the increase. Chronic exposure to high blood glucose is a leading cause of renal failure, visual loss and a range of other types of tissue damage. Diabetes also predisposes to arterial disease, not least because it is often accompanied by hypertension, lipid disorders and obesity. Many cases of diabetes and almost all of its unwanted long-term consequences are potentially avoidable, but this will require intervention at a societal as well as at a medical level.

1.2 Objective of the project

We want to design a web application such that it should predict where a person is suffering from diabetes or not. If he is not suffering from diabetes what are the chances that he may get affected by diabetes and we suggested diets separately for both the category people. So by this web application people can come to know that they may get affected by diabetes and they take preventive measures. We believe that **PREVENTION IS BETTER THAN CURE**.

1.3 Machine learning

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, in order 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 according

Machine learning algorithms are often categorized as supervised or unsupervised.

1.3.1 Models Used:

Since this website predicts the diabetes of the person using Pima Indian diabetes dataset, firstly we perform data cleaning by using some statistical manipulations like replacing the attributes of zero value with NA then replacing NA with mean of the attribute. Then we fit a model to the dataset. Initially we fit different models to the cleaned dataset and finally chooses one model which gives high prediction for that dataset. Different Models used are below:

1.3.1.1 K-Nearest Neighbours

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).
We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

Algorithm
Let m be the number of training data samples. Let p be an unknown point.

```
1. Store the training samples in an array of data points arr[]. This means each element of
   this array represents a tuple (x, y).
2. for i=0 to m:
3.   Calculate Euclidean distance d(arr[i], p).
4. Make set S of K smallest distances obtained. Each of these distances corresponds to an
   already classified data point.
5. Return the majority label among S.
```

1.3.1.2 SVC

The objective of a Linear **SVC** (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what is the "predicted" class.

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
>>>SVC()
```

After being fitted, the model can then be used to predict new values:

```
>>> clf.predict([[2., 2.]])array([1])
```

SVMs decision function depends on some subset of the training data, called the support vectors. Some properties of these support vectors can be found in members support_vectors_, support_ and n_support:

```
>>> # get support vectors
>>> clf.support_vectors_array([[0., 0.],[1., 1.]])
>>> # get indices of support vectors
>>> clf.support_array([0, 1]...)
>>> # get number of support vectors for each class
>>> clf.n_support_array([1, 1]...)
```

1.3.1.3 Logistic Regression

Logistic regression is another technique borrowed by machine learning from the field of statistics.

It is the go-to method for binary classification problems (problems with two class values).

Logistic Function

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

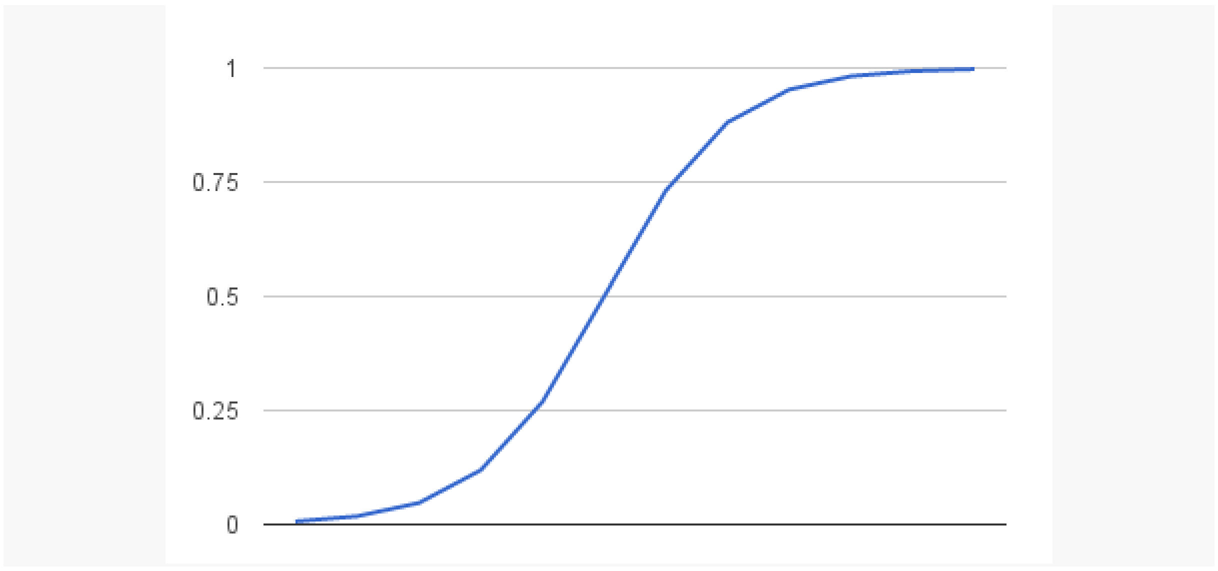


Fig 1.3.1.3 : Logistic Function

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modelled is a binary value (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

1.3.1.4 Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules.

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning.

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric **supervised learning** method used for both **classification** and **regression** tasks.

Tree models where the target variable can take a discrete set of values are called **classification trees**. Decision trees where the target variable can take continuous values (typically real numbers) are called **regression trees**. Classification And Regression Tree (CART) is a general term for this.

Data Format

Data comes in records of forms.
(x,Y)=(x1,x2,x3,....,xk,Y)
The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector x is composed of the features, x1, x2, x3 etc. That are used for that task.

Approach to make Decision Tree

While making decision tree, at each node of the tree we ask different types of questions. Based on the asked question we will calculate the information gain corresponding to it.

Information Gain

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information. For each node of the tree, the information value measures how much information a feature gives us about the class. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.

Steps for Making Decision Tree

- Get list of rows (dataset) which are taken into consideration for making decision tree (recursively at each node).
- Calculate uncertainty of our dataset or Gini impurity or how much our data is mixed up etc.
- Generate list of all questions which needs to be asked at that node.
- Partition rows into True rows and False rows based on each question asked.
- Calculate information gain based on Gini impurity and partition of data from previous step.

- Update highest information gain based on each question asked.
- Update best question based on information gain (higher information gain).
- Divide the node on best question. Repeat again from step 1 again until we get pure node (leaf nodes).

1.3.1.5 Gaussian Naive Bayes Classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

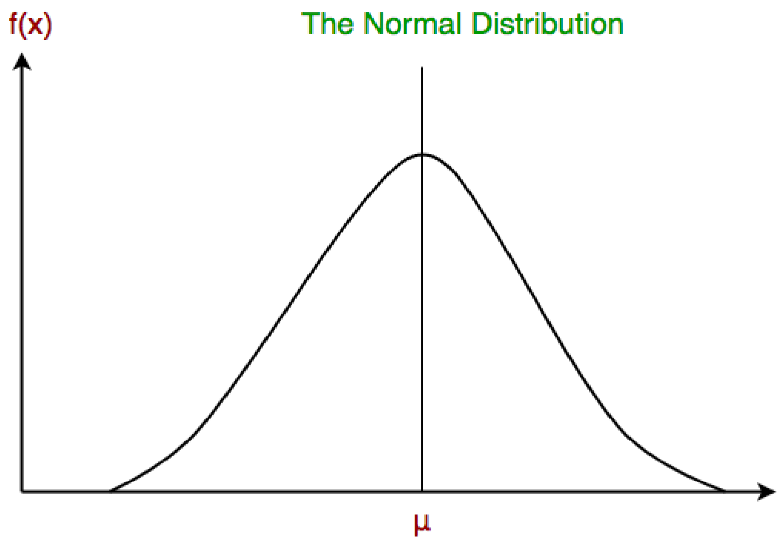


Fig 1.3.1.5 : Normal Distribution

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

1.3.1.6 Random Forest Classification Algorithm

Introduction

Random forest is a supervised learning algorithm which is used for both classification as well as regression. However, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from

each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Working of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps −

- Step 1 − First, start with the selection of random samples from a given dataset.
- Step 2 − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- Step 3 − In this step, voting will be performed for every predicted result.
- Step 4 − At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working −

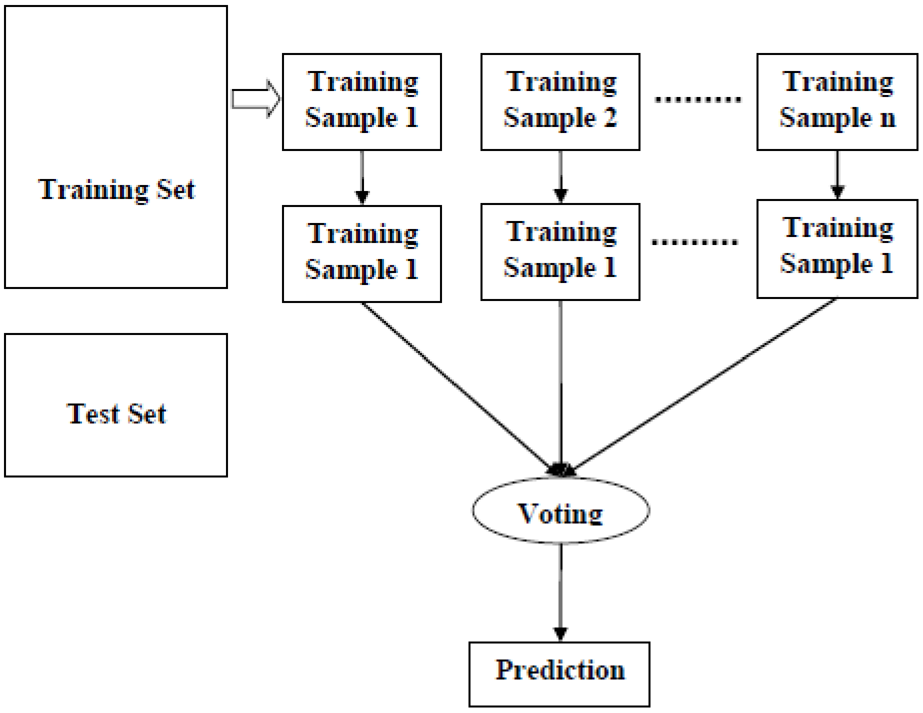


Fig 1.3.1.6 :Working of Algorithm

1.3.1.7 Gradient Boosting Classifiers

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets, and have recently been used to win many Kaggle data science competitions.

Gradient boosting classifiers are specific types of algorithms that are used for classification tasks, as the name suggests.

Features are the inputs that are given to the machine learning algorithm, the inputs that will be used to calculate an output value. In a mathematical sense, the features of the dataset are the variables used to solve the equation. The other part of the equation is the *label* or target, which are the classes the instances will be categorized into. Because the labels contain the target values for the machine learning classifier, when training a classifier you should split up the data into training and testing sets. The training set will have targets/labels, while the testing set won't contain these values.

Scikit-Learn, or "sklearn", is a machine learning library created for Python, intended to expedite machine learning tasks by making it easier to implement machine learning algorithms. It has easy-to-use functions to assist with splitting data into training and testing sets, as well as training a model, making predictions, and evaluating the model.

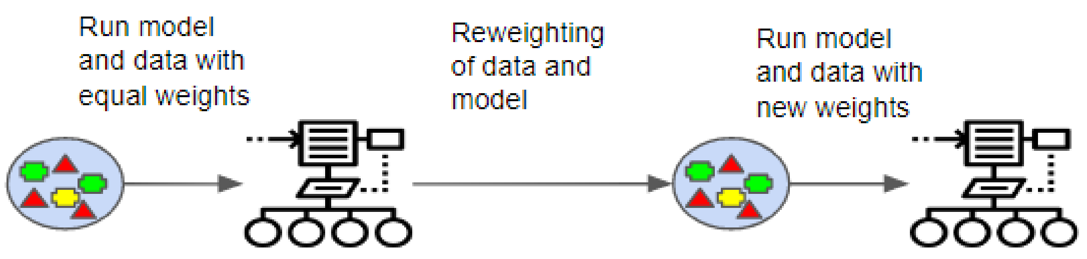
The idea behind "gradient boosting" is to take a weak hypothesis or weak learning algorithm and make a series of tweaks to it that will improve the strength of the hypothesis/learner. This type of Hypothesis Boosting is based on the idea of Probability Approximately Correct Learning (PAC).

This PAC learning method investigates machine learning problems to interpret how complex they are, and a similar method is applied to *Hypothesis Boosting*.

In hypothesis boosting, you look at all the observations that the machine learning algorithm is trained on, and you leave only the observations that the machine learning method successfully classified behind, stripping out the other observations. A new weak learner is created and tested on the set of data that was poorly classified, and then just the examples that were successfully classified are kept.

This idea was realized in the Adaptive Boosting (*AdaBoost*) algorithm. For AdaBoost, many weak learners are created by initializing many decision tree algorithms that only have a single split.The instances/observations in the training set are weighted by the algorithm, and more weight is assigned to instances which are difficult to classify. More weak learners are added into the system sequentially, and they are assigned to the most difficult training instances.

In AdaBoost, the predictions are made through majority vote, with the instances being classified according to which class receives the most votes from the weak learners.



Gradient boosting classifiers are the AdaBoosting method combined with weighted minimization, after which the classifiers and weighted inputs are recalculated. The objective of Gradient Boosting classifiers is to minimize the loss, or the difference between the actual class value of the training example and the predicted class value. It isn't required to understand the process for reducing the classifier's loss, but it operates similarly to gradient descent in a neural network.

Refinements to this process were made and Gradient Boosting Machines were created.

In the case of Gradient Boosting Machines, every time a new weak learner is added to the model, the weights of the previous learners are frozen or cemented in place, left unchanged as the new layers are introduced. This is distinct from the approaches used in AdaBoosting where the values are adjusted when new learners are added.

The power of gradient boosting machines comes from the fact that they can be used on more than binary classification problems, they can be used on multi-class classification problems and even regression problems.

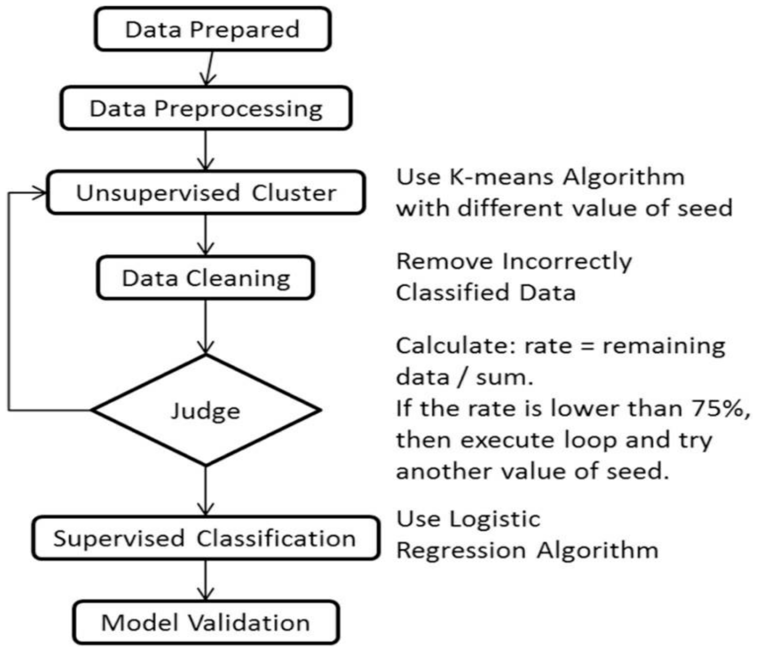
1.4 Tools and packages

You need to install the following software and certain packages in python for successful execution of report.

- Anaconda version 5.2
- Flask 2.0

2. SYSTEM DESIGN

2.1 Flow chart



Data Prepared
Data Preprocessing
Unsupervised Cluster
Use K-means Algorithm with different value of seed
Data Cleaning
Remove Incorrectly Classified Data
Judge
Calculate: rate = remaining data / sum.
If the rate is lower than 75%, then execute loop and try another value of seed.
Supervised Classification
Use Logistic Regression Algorithm
Model Validation

Figure 2.1: Flowchart representing the operations carried out to use Logistic Regression Algorithm

3. SOFTWARE AND TOOLS

3.1 Python

3.1.1 Introduction

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

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

Python interpreters are available for many operating systems. Python is managed by the non-profit Python Software Foundation.

Figure 3.1:Python Programming Language

3.1.2 Syntax and Semantics

The syntax of the Python programming language is the set of rules that defines how a Python program will be written and interpreted (by both the runtime system and by human readers).

- Since Python is a dynamically typed language, Python *values,* not variables, carry type. This has implications for many aspects of the way the language functions.
- Python has a broad range of basic data types. Alongside conventional integer and floating-point arithmetic, it transparently supports arbitrary-precision arithmetic, complex numbers, and decimal floating-point numbers.
- The language supports extensive introspection of types and classes. Types can be read and compared types are instances of type. The attributes of an object can be extracted as a dictionary.

3.1.3 Indentation

Python uses whitespace to delimit control flow blocks (following the off-side rule). Python borrows this feature from its predecessor ABC: instead of punctuation or keywords, it uses indentation to indicate the run of a block.

In so-called "free-format" languages — that use the block structure derived from ALGOL — blocks of code are set off with braces ({ }) or keywords. In most coding conventions for these languages, programmers conventionally indent the code within a block, to visually set it apart from the surrounding code (pretty printing).

Consider a function which is passed a single parameter and if the parameter is 0 will call and otherwise it will call passing and call itself recursively, passing as the parameter. Here are implementations of this function in both C and Python.

3.2 The Anaconda

3.2.1 Introduction

Anaconda is data science and machine learning platform for the Python and R programming languages. It is designed to make the process of creating and distributing projects simple, stable and reproducible across systems and is available on Linux, Windows, and OSX. Anaconda is a Python based platform that curates major data science packages including pandas, scikit-learn, SciPy, NumPy and Google's machine learning platform, TensorFlow. It comes packaged with conda (a pip like install tool), Anaconda navigator for a GUI experience, and spyder for an IDE.This tutorial will walk through some of the basics of Anaconda, conda, and spyder for the Python programming language and introduce you to the concepts needed to begin creating your own projects.

There are many great articles on this site for installing Anaconda on different distro's and native package management systems. For that reason, I will provide some links to this work below and skip to covering the tool itself.

- CentOS
- Ubuntu

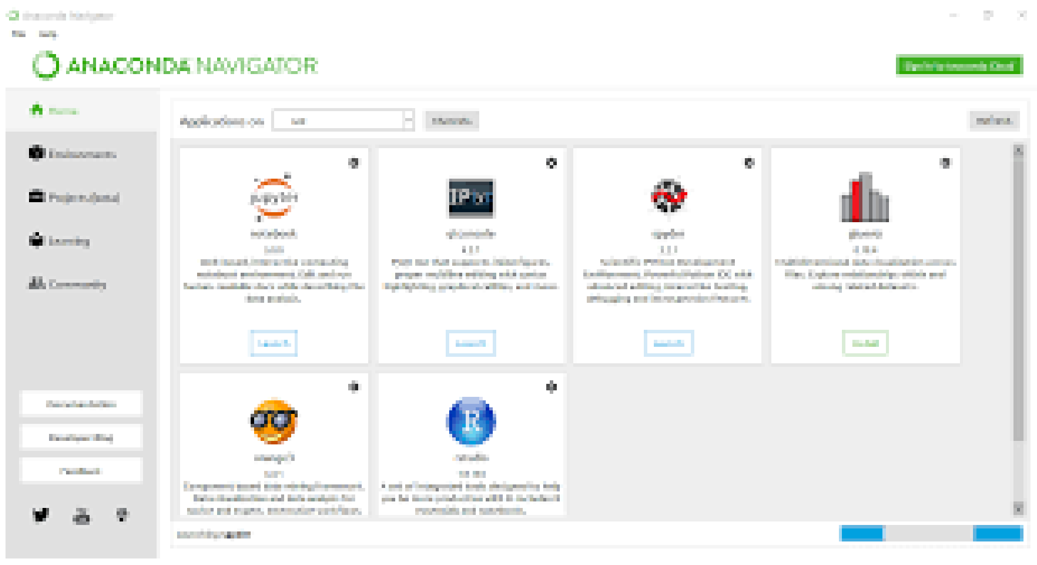


Figure 3.2.1 : Anaconda terminal

3.2.2 Code editors

Several excellent code editors are available that provide functionalities like python syntax highlighting, auto code indenting and utilities to send code/functions to the python console.

- Basic code editors provided by Rguis
- Spyder:Windows, Linux, macOS
- SPE:Windows, Linux, Mac OS X
- EditPad Pro:Linux, Windows
- Emacs (ESS add on package)
- Eclipse
- Notepad++(NppToR)

3.2.3 Benefits Of Using Anaconda

The diverse application of the Python language is a result of the combination of features which give this language an edge over others. Some of the benefits of programming in Python include:

1. Presence of Third Party Modules
2. Extensive Support Librarie
3. Open Source and Community Development
4. Learning Ease and Support Available
5. User-friendly Data Structures
6. Productivity and Speed

1. Presence of Third Party Modules:

The Python Package Index (PyPI) contains numerous third-party modules that make Python capable of interacting with most of the other languages and platforms.

2. Extensive Support Libraries:

Python provides a large standard library which includes areas like internet protocols, string operations, web services tools and operating system interfaces. Many high use programming tasks have already been scripted into the standard library which reduces length of code to be written significantly.

3. Open Source and Community Development:

Python language is developed under an OSI-approved open source license, which makes it free to use and distribute, including for commercial purposes.

Further, its development is driven by the community which collaborates for its code through hosting conferences and mailing lists, and provides for its numerous modules.

4. Learning Ease and Support Available:

Python offers excellent readability and uncluttered simple-to-learn syntax which helps beginners to utilize this programming language. The code style guidelines, PEP 8, provide a set of rules to facilitate the formatting of code. Additionally, the wide base of users and active developers has resulted in a rich internet resource bank to encourage development and the continued adoption of the language.

5. User-friendly Data Structures:

Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures. Further, Python also provides the option of dynamic high-level data typing which reduces the length of support code that is needed.

6. Productivity and Speed:

Python has clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing framework, all of which contribute to the increase in its speed and productivity. Python is considered a viable option for building complex multi-protocol network applications.

As can be seen from the above-mentioned points, Python offers a number of advantages for software development. As upgrading of the language continues, its loyalist base could grow as well.

3.3 Flask

3.3.1 Introduction

Flask is a micro web framework written in python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program. Flask is commonly used with MongoDB, which gives it more control over databases and history.

Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itself.

Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are:

- Werkzeug a WSGI utility library
- jinja2 which is its template engine

Figure 3.3.1 : Flask Framework

3.3.2 About

In 2004, Pocoo was formed as an international group of Python enthusiasts. Flask was created by Armin Ronacher of Pocoo:

"It came out of an April fool’s joke but proved popular enough to make into a serious application in its own right."

When Ronacher and Georg Brandl created a bulletin board system written in Python, the Pocoo projects Werkzeug and the Jinja2 were developed, too.

Despite the lack of a major release, Flask has become extremely popular among Python enthusiasts. As of mid 2016, it was the most popular Python web development framework on GitHub.

The microframework Flask is based on the *Pocoo* projects *Werkzeug* and *Jinja2*.

Werkzeug

Werkzeug is a utility library for the Python programming language, in other words a toolkit for Web Server Gateway Interface (WSGI) applications, and is licensed under a BSD License. Werkzeug can realize software objects for request, response, and utility functions. It can be used to build a custom software framework on top of it and supports Python 2.6, 2.7 and 3.3.

Jinja

Jinja is a template engine for the Python programming language and is licensed under a BSD License also by Ronacher. Similar to the Django web framework, furthermore it provides that templates are evaluated in a sandbox.

3.3.3 Features

- Contains development server and debugger
- Integrated support for unit testing
- RESTful request dispatching and Uses Jinja2 templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based and Extensive documentation
- Google App Engine compatibility and Extensions available to enhance features desired

4. IMPLEMENTATION & CODE

4.1 Procedure

4.1.1 Load the Dataset

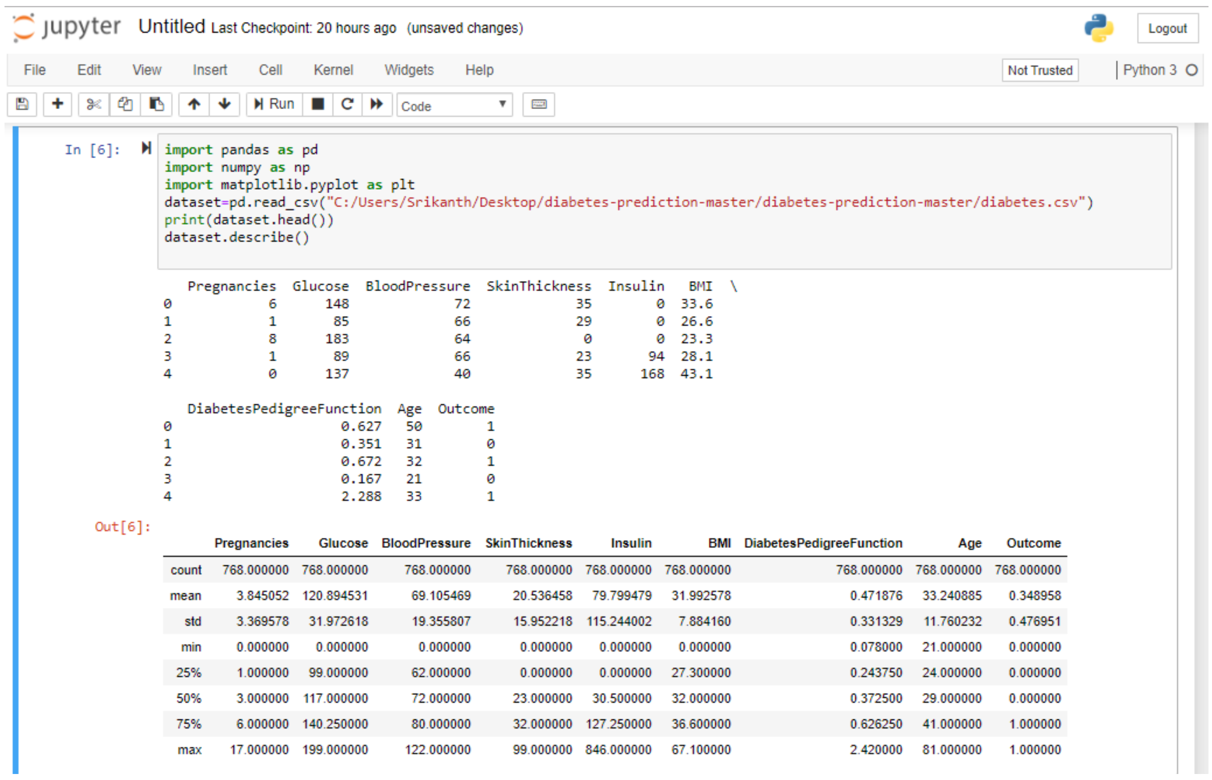
We are going to use the Pima Indians Diabetes Dataset.
This dataset describes the medical records for Pima Indians and whether or not each patient will have an onset of diabetes within five years.
Fields description follow:
preg = Number of times pregnant
plas = Plasma glucose concentration a 2 hours in an oral glucose tolerance test
pres = Diastolic blood pressure (mm Hg)
skin = Triceps skin fold thickness (mm)
test = 2-Hour serum insulin (mu U/ml)
mass = Body mass index (weight in kg/(height in m)^2)
pedi = Diabetes pedigree function
age = Age (years)
class = Class variable (1:tested positive for diabetes, 0: tested negative for diabetes)

4.1.1.1 Import Libraries

First, let's import all of the modules, functions and objects we are going to use in this tutorial.

4.1.1.2 Load Dataset

We can load the data directly from the UCI Machine Learning repository. We are using pandas to load the data. We will also use pandas next to explore the data both with descriptive statistics and data visualization. Note that we are specifying the names of each column when loading the data. This will help later when we explore the data.


jupyter Untitled Last Checkpoint: 20 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset=pd.read_csv("C:/Users/Srikanth/Desktop/diabetes-prediction-master/diabetes-prediction-master/diabetes.csv")
print(dataset.head())
dataset.describe()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

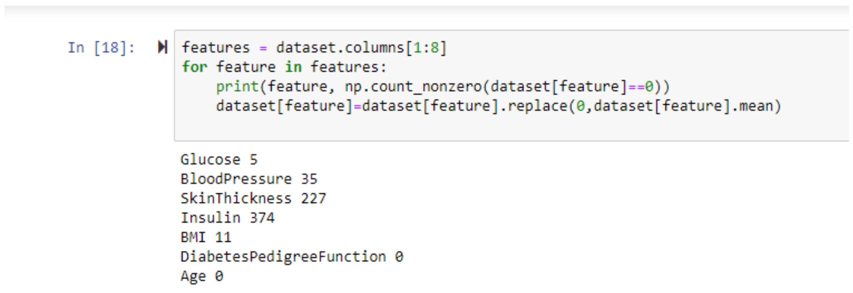
Out[6]:

	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	0.476951
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	36.600000	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

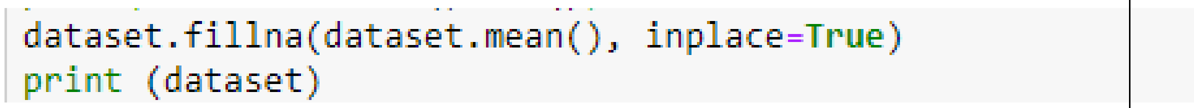
4.1.2 Taking care of Missing Data

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common idea to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.

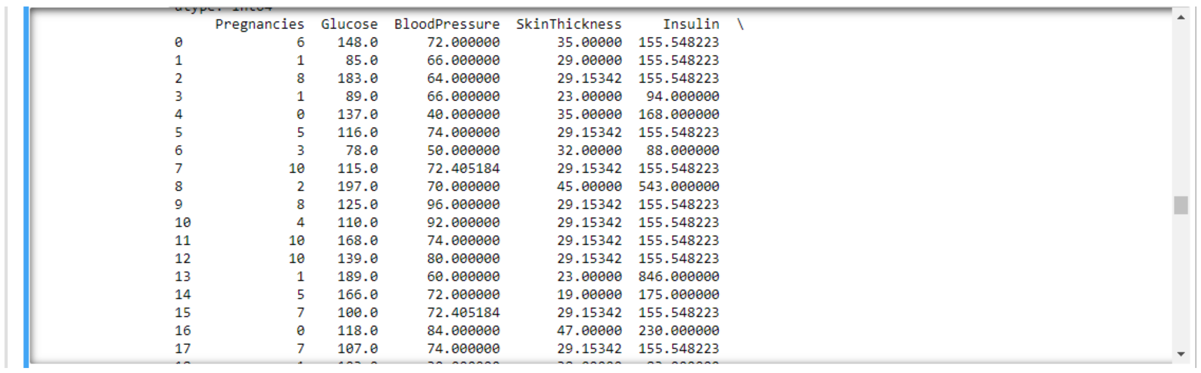
strategy — we will find the average so we will set it to mean. We can also set it to median or most_frequent (for mode) as necessary.



```
In [18]:  features = dataset.columns[1:8]
          for feature in features:
              print(feature, np.count_nonzero(dataset[feature]==0))
              dataset[feature]=dataset[feature].replace(0,dataset[feature].mean)

Glucose 5
BloodPressure 35
SkinThickness 227
Insulin 374
BMI 11
DiabetesPedigreeFunction 0
Age 0
```



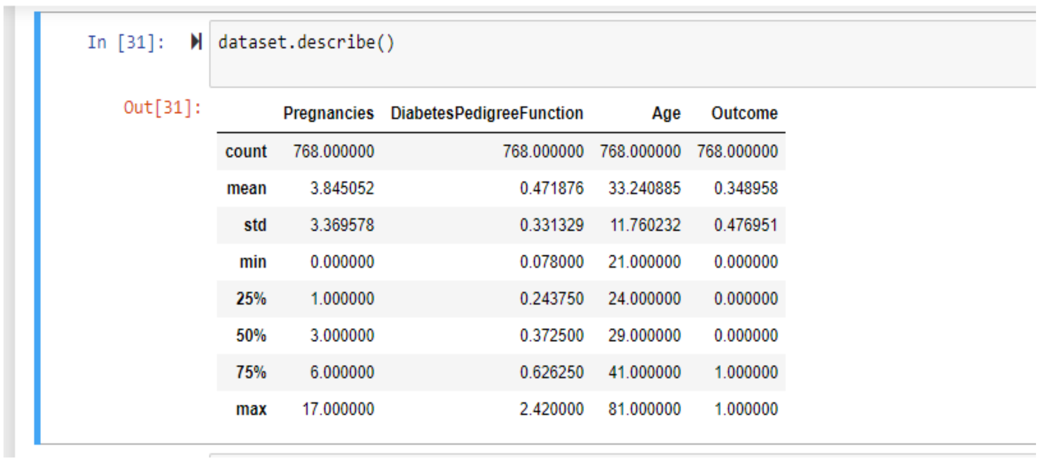
```
dataset.fillna(dataset.mean(), inplace=True)
print (dataset)
```



```
    Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin  \
0             6    148.0      72.000000       35.00000  155.548223
1             1     85.0      66.000000       29.00000  155.548223
2             8    183.0      64.000000       29.15342  155.548223
3             1     89.0      66.000000       23.00000   94.000000
4             0    137.0      40.000000       35.00000  168.000000
5             5    116.0      74.000000       29.15342  155.548223
6             3     78.0      50.000000       32.00000   88.000000
7            10    115.0      72.405184       29.15342  155.548223
8             2    197.0      70.000000       45.00000  543.000000
9             8    125.0      96.000000       29.15342  155.548223
10            4    110.0      92.000000       29.15342  155.548223
11           10    168.0      74.000000       29.15342  155.548223
12           10    139.0      80.000000       29.15342  155.548223
13            1    189.0      60.000000       23.00000  846.000000
14            5    166.0      72.000000       19.00000  175.000000
15            7    100.0      72.405184       29.15342  155.548223
16            0    118.0      84.000000       47.00000  230.000000
17            7    107.0      74.000000       29.15342  155.548223
```

4.1.3 Summarize the Dataset

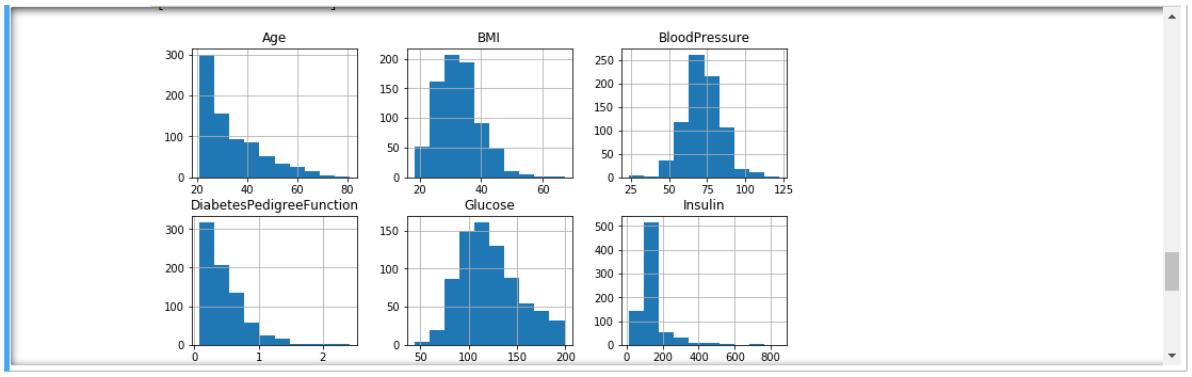
Now it is time to take a look at the data. In this step we are going to take a look at the data a few different ways:
Dimensions of the dataset. Peek at the data itself. Statistical summary of all attributes. Breakdown of the data by the class variable.

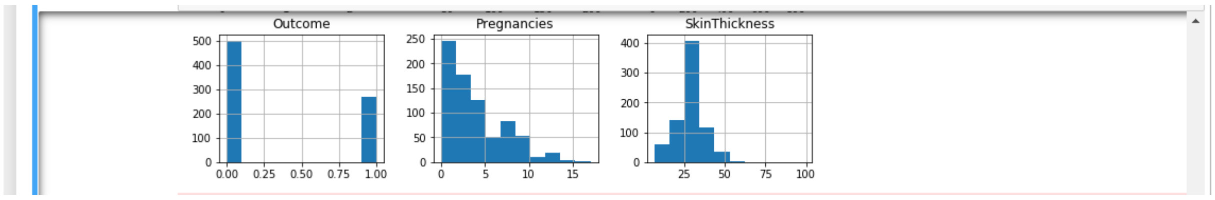


```
In [31]:  dataset.describe()

Out[31]:
       Pregnancies  DiabetesPedigreeFunction         Age     Outcome
count   768.000000                768.000000  768.000000  768.000000
mean      3.845052                  0.471876   33.240885    0.348958
std       3.369578                  0.331329   11.760232    0.476951
min       0.000000                  0.078000   21.000000    0.000000
25%       1.000000                  0.243750   24.000000    0.000000
50%       3.000000                  0.372500   29.000000    0.000000
75%       6.000000                  0.626250   41.000000    1.000000
max      17.000000                  2.420000   81.000000    1.000000
```

4.1.4 Data Visualisation

We use histogram plot to visualize the data





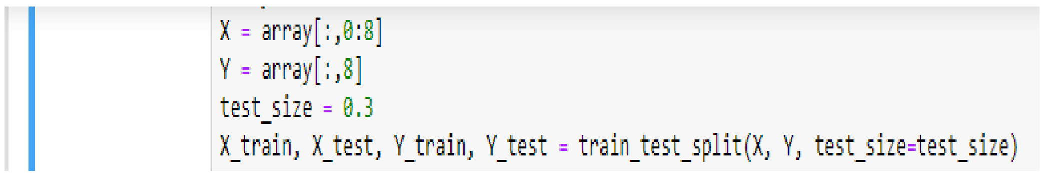
4.2 Evaluating Algorithms

4.2.1 Creating Validation Dataset

We need to know that the model we created is good.

Later, we will use statistical methods to estimate the accuracy of the models that we create on unseen data. We also want a more concrete estimate of the accuracy of the best model on unseen data by evaluating it on actual unseen data.

That is, we are going to hold back some data that the algorithms will not get to see and we will use this data to get a second and independent idea of how accurate the best model might actually be.



```
X = array[:,0:8]
Y = array[:,8]
test_size = 0.3
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size)
```

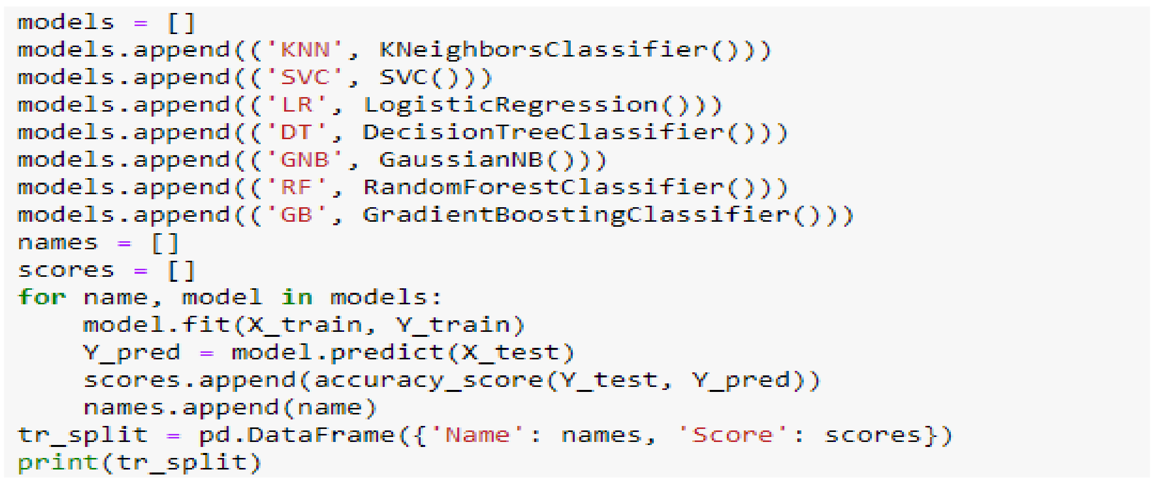
4.2.2 Build Models and Make Prediction

We don’t know which algorithms would be good on this problem or what configurations to use.

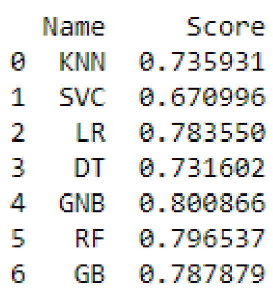
We get an idea from the plots that some of the classes are partially linearly separable in some dimensions, so we are expecting generally good results.

Let’s test 6 different algorithms:

KNeighboursClassifier (KNN)
Support Vector Classifier (SVC)
Logistic Regression (LR)
Decision Tree Classifier
Gaussian Naive Bayes (NB).
Gradient Boosting Classifier(GB)



```
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('SVC', SVC()))
models.append(('LR', LogisticRegression()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
models.append(('RF', RandomForestClassifier()))
models.append(('GB', GradientBoostingClassifier()))
names = []
scores = []
for name, model in models:
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    scores.append(accuracy_score(Y_test, Y_pred))
    names.append(name)
tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
```



```
  Name     Score
0  KNN  0.735931
1  SVC  0.670996
2   LR  0.783550
3   DT  0.731602
4  GNB  0.800866
5   RF  0.796537
6   GB  0.787879
```

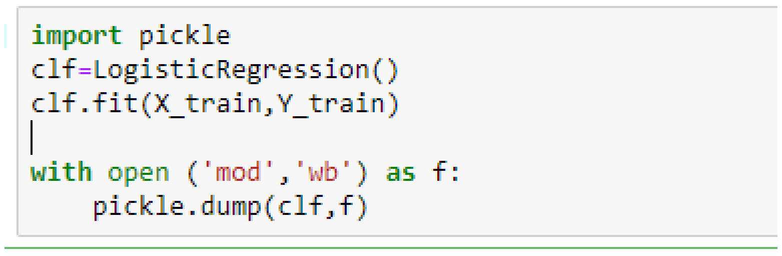
4.2.3 Select Best Model

We now have 6 models and accuracy estimations for each. We need to compare the models to each other and select the most accurate.
The results in the previous section suggest that the LR was perhaps the most accurate model. We will use this model as our final model.

4.2.4 Save and Load the Model

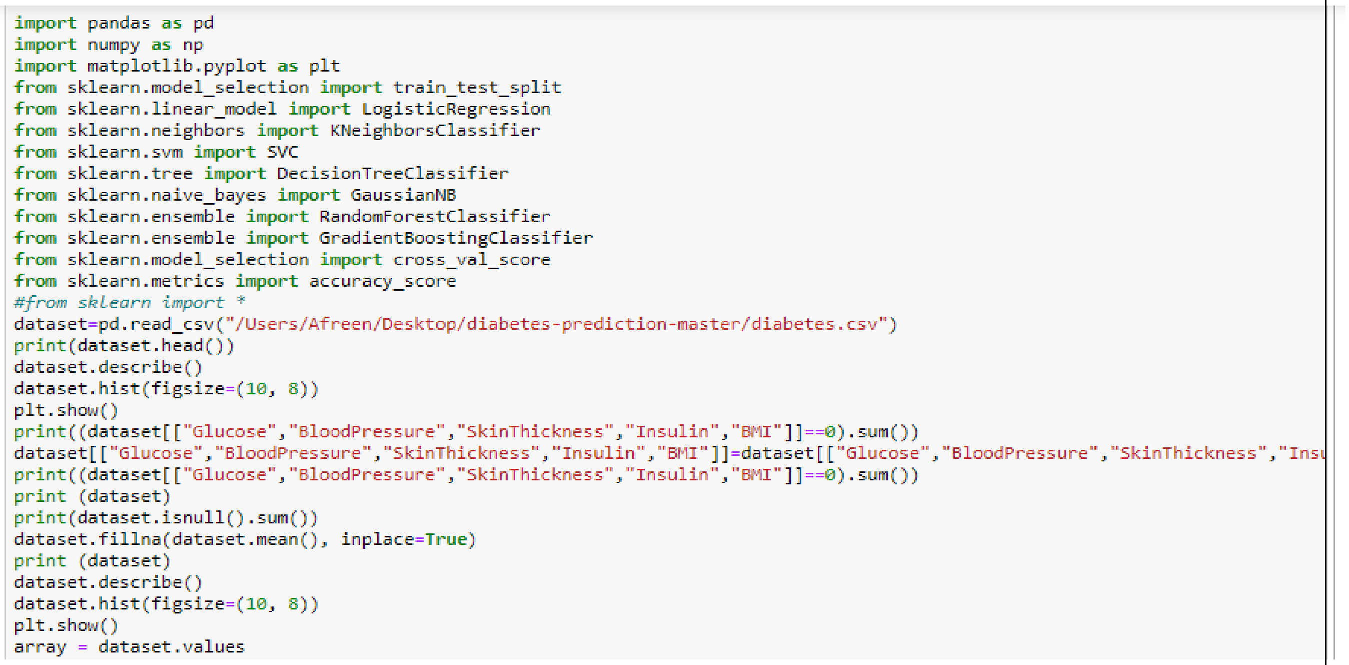
You can use the pickle operation to serialize your machine learning algorithms and save the serialized format to a file.
This allows you to save your model to file and load it later in order to make predictions.

Later you can load this file to deserialize your model and use it to make new predictions.

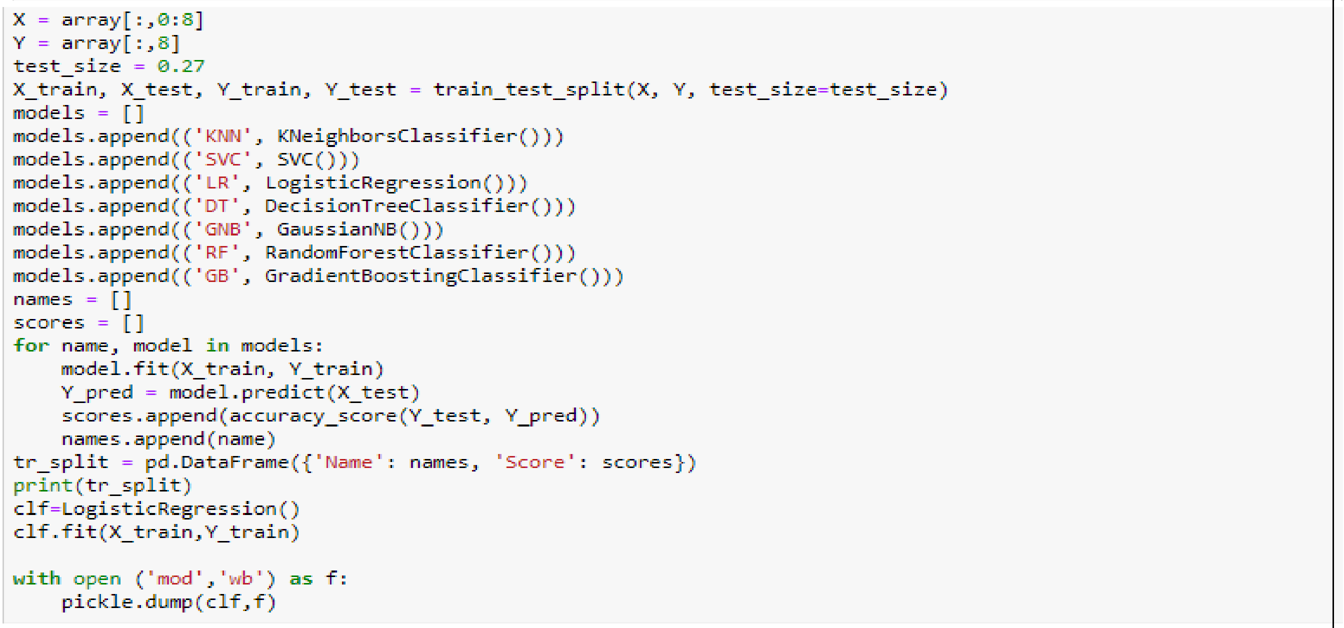


```python
import pickle
clf=LogisticRegression()
clf.fit(X_train,Y_train)

with open ('mod','wb') as f:
    pickle.dump(clf,f)
```

4.3 Code



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
#from sklearn import *
dataset=pd.read_csv("/Users/Afreen/Desktop/diabetes-prediction-master/diabetes.csv")
print(dataset.head())
dataset.describe()
dataset.hist(figsize=(10, 8))
plt.show()
print((dataset[["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]]==0).sum())
dataset[["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]]=dataset[["Glucose","BloodPressure","SkinThickness","Insu
print((dataset[["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]]==0).sum())
print (dataset)
print(dataset.isnull().sum())
dataset.fillna(dataset.mean(), inplace=True)
print (dataset)
dataset.describe()
dataset.hist(figsize=(10, 8))
plt.show()
array = dataset.values
```

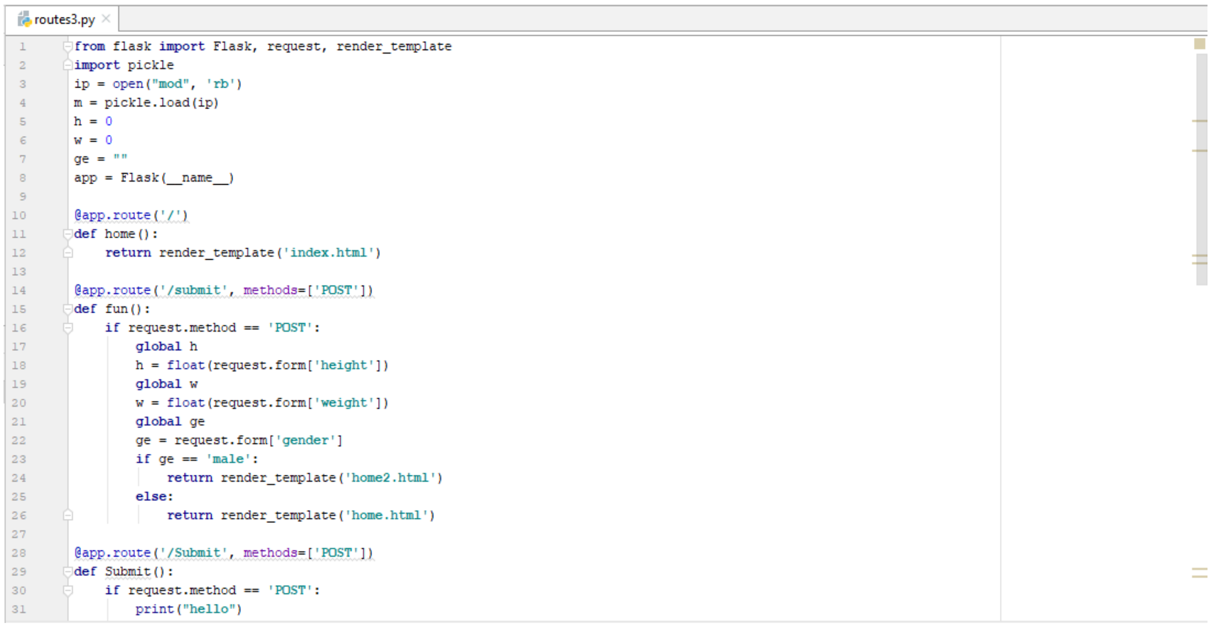


```
X = array[:,0:8]
Y = array[:,8]
test_size = 0.27
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size)
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('SVC', SVC()))
models.append(('LR', LogisticRegression()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
models.append(('RF', RandomForestClassifier()))
models.append(('GB', GradientBoostingClassifier()))
names = []
scores = []
for name, model in models:
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    scores.append(accuracy_score(Y_test, Y_pred))
    names.append(name)
tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
clf=LogisticRegression()
clf.fit(X_train,Y_train)

with open ('mod','wb') as f:
    pickle.dump(clf,f)
```

4.4 Make Predictions for the user Data

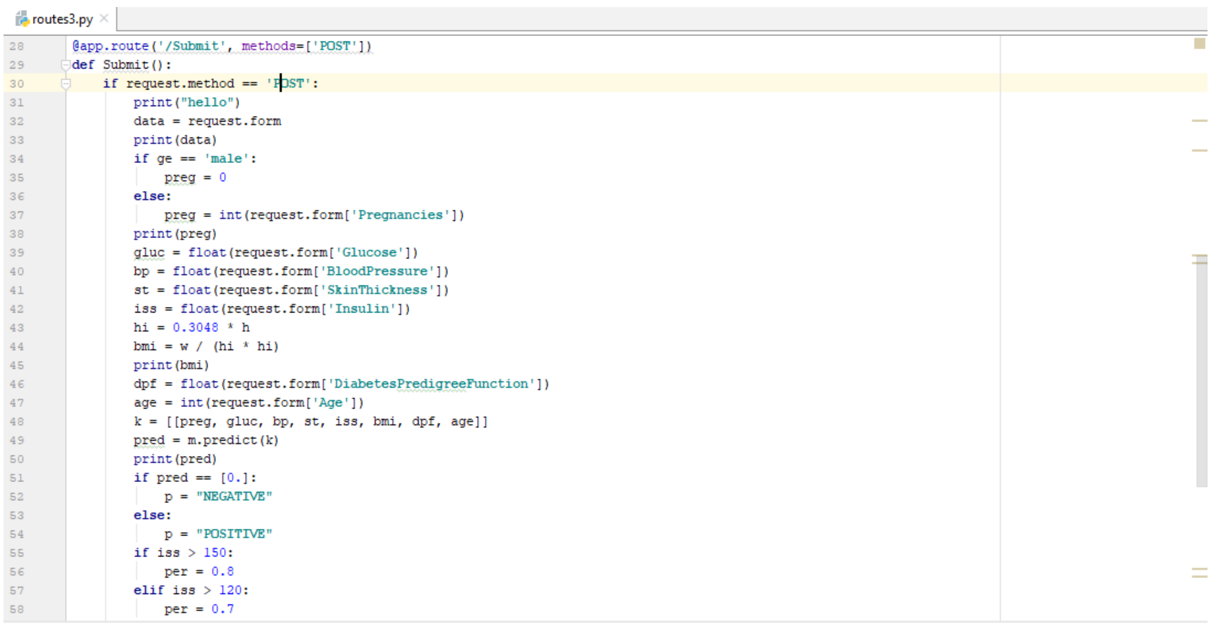
Now we start developing front end of the website with python so that web application will be upto date and easily adoptable. This website predicts the diabetes of the person using the model we developed after Machine Learning of the Pima Indian Dataset.

The Python Code for developing and predicting the result is below:



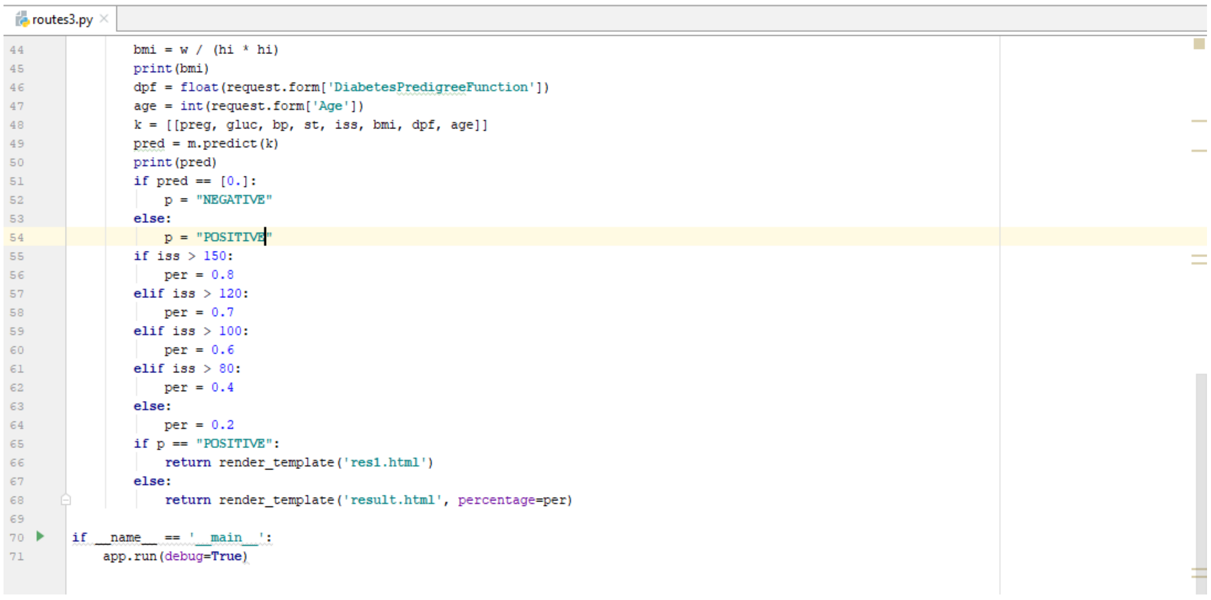
routes3.py

```python
from flask import Flask, request, render_template
import pickle
ip = open("mod", 'rb')
m = pickle.load(ip)
h = 0
w = 0
ge = ""
app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def fun():
    if request.method == 'POST':
        global h
        h = float(request.form['height'])
        global w
        w = float(request.form['weight'])
        global ge
        ge = request.form['gender']
        if ge == 'male':
            return render_template('home2.html')
        else:
            return render_template('home.html')

@app.route('/Submit', methods=['POST'])
def Submit():
    if request.method == 'POST':
        print("hello")
```



routes3.py

```python
@app.route('/Submit', methods=['POST'])
def Submit():
    if request.method == 'POST':
        print("hello")
        data = request.form
        print(data)
        if ge == 'male':
            preg = 0
        else:
            preg = int(request.form['Pregnancies'])
        print(preg)
        gluc = float(request.form['Glucose'])
        bp = float(request.form['BloodPressure'])
        st = float(request.form['SkinThickness'])
        iss = float(request.form['Insulin'])
        hi = 0.3048 * h
        bmi = w / (hi * hi)
        print(bmi)
        dpf = float(request.form['DiabetesPredigreeFunction'])
        age = int(request.form['Age'])
        k = [[preg, gluc, bp, st, iss, bmi, dpf, age]]
        pred = m.predict(k)
        print(pred)
        if pred == [0.]:
            p = "NEGATIVE"
        else:
            p = "POSITIVE"
        if iss > 150:
            per = 0.8
        elif iss > 120:
            per = 0.7
```

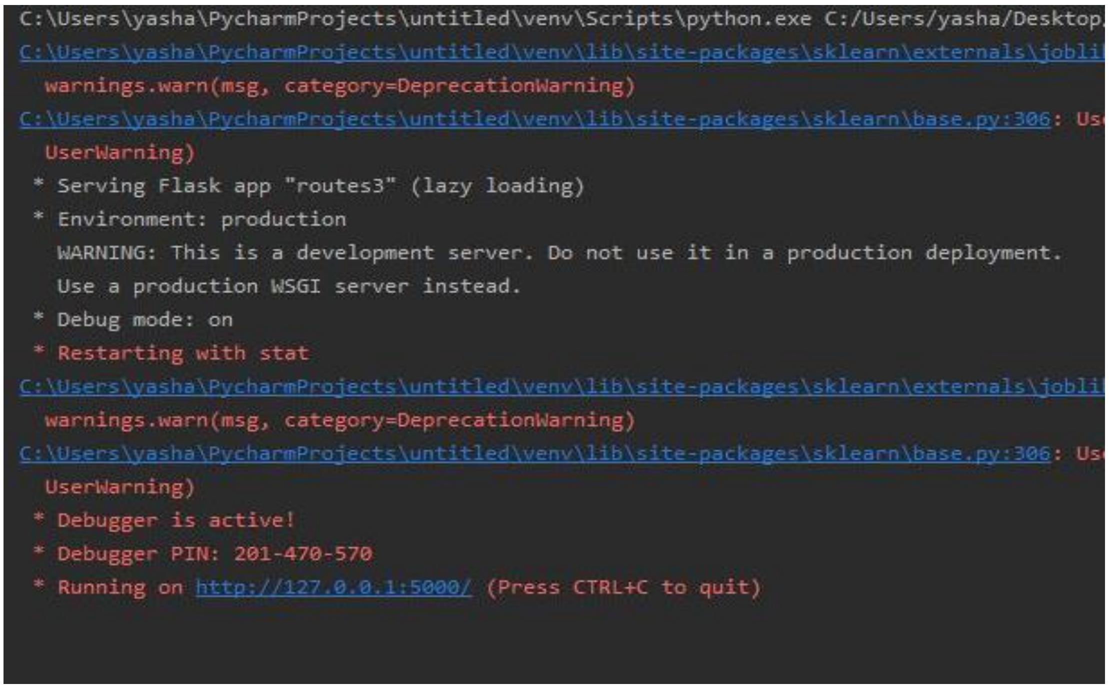


routes3.py

```python
        bmi = w / (hi * hi)
        print(bmi)
        dpf = float(request.form['DiabetesPredigreeFunction'])
        age = int(request.form['Age'])
        k = [[preg, gluc, bp, st, iss, bmi, dpf, age]]
        pred = m.predict(k)
        print(pred)
        if pred == [0.]:
            p = "NEGATIVE"
        else:
            p = "POSITIVE"
        if iss > 150:
            per = 0.8
        elif iss > 120:
            per = 0.7
        elif iss > 100:
            per = 0.6
        elif iss > 80:
            per = 0.4
        else:
            per = 0.2
        if p == "POSITIVE":
            return render_template('res1.html')
        else:
            return render_template('result.html', percentage=per)

if __name__ == '__main__':
    app.run(debug=True)
```

4.5 Execution

First of all we have to open our file in terminal. Then we have execute flask python file by using command PYTHON3 FILENAME.PY. We can see the results of that http address given in the terminal.



```
C:\Users\yasha\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/yasha/Desktop
C:\Users\yasha\PycharmProjects\untitled\venv\lib\site-packages\sklearn\externals\jobli
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\yasha\PycharmProjects\untitled\venv\lib\site-packages\sklearn\base.py:306: Us
  UserWarning)
 * Serving Flask app "routes3" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
C:\Users\yasha\PycharmProjects\untitled\venv\lib\site-packages\sklearn\externals\jobli
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\yasha\PycharmProjects\untitled\venv\lib\site-packages\sklearn\base.py:306: Us
  UserWarning)
 * Debugger is active!
 * Debugger PIN: 201-470-570
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

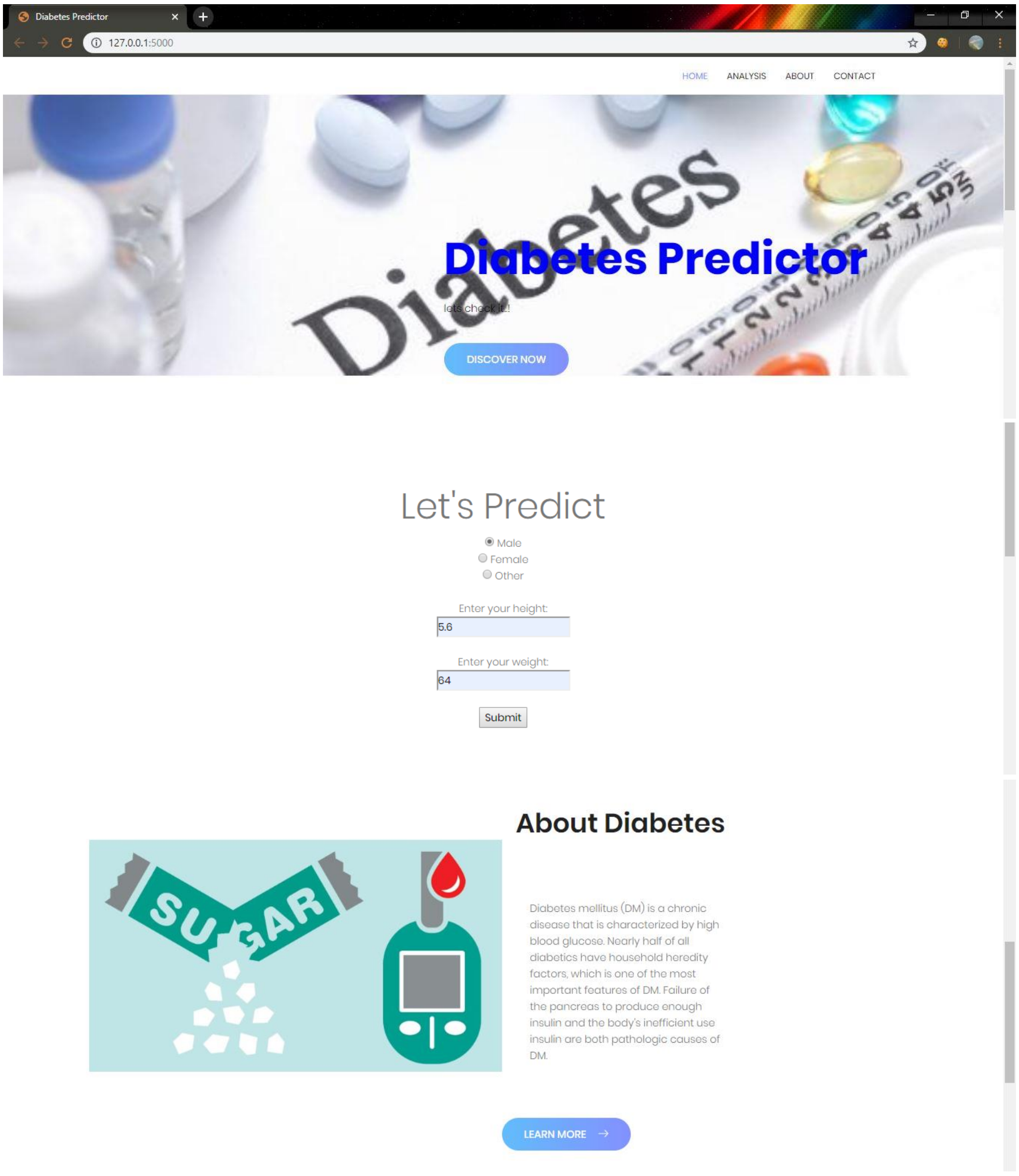
.

Figure 4.5 : Terminal showing http address of our web application

4.6 Result

4.6.1 Home

By copying the http link in browser, you can see our home page DIABETES PREDICTOR image as shown below



Diabetes Predictor

lets check it..!

Let's Predict

[✓ Male]
Female
Other

Enter your height: 5.6

Enter your weight: 64

About Diabetes

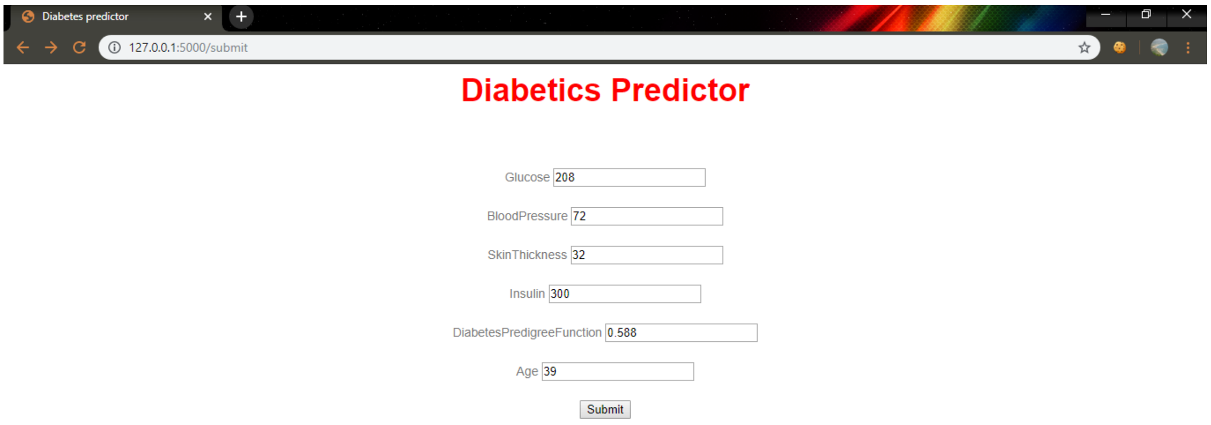
Diabetes mellitus (DM) is a chronic disease that is characterized by high blood glucose. Nearly half of all diabetics have household heredity factors, which is one of the most important features of DM. Failure of the pancreas to produce enough insulin and the body's inefficient use insulin are both pathologic causes of DM.

Figure 4.6.1 : Home page of our web application

By giving value input values of height, weight, gender in the lets predict section of the above figure the user will redirect to the another page which is given below

4.6.2 Core Attributes

In this page you need the core attribute values like glucose, blood pressure, skin thickness, insulin, diabetes predigree function, age



Diabetics Predictor

Glucose 208

BloodPressure 72

SkinThickness 32

Insulin 300

DiabetesPredigreeFunction 0.588

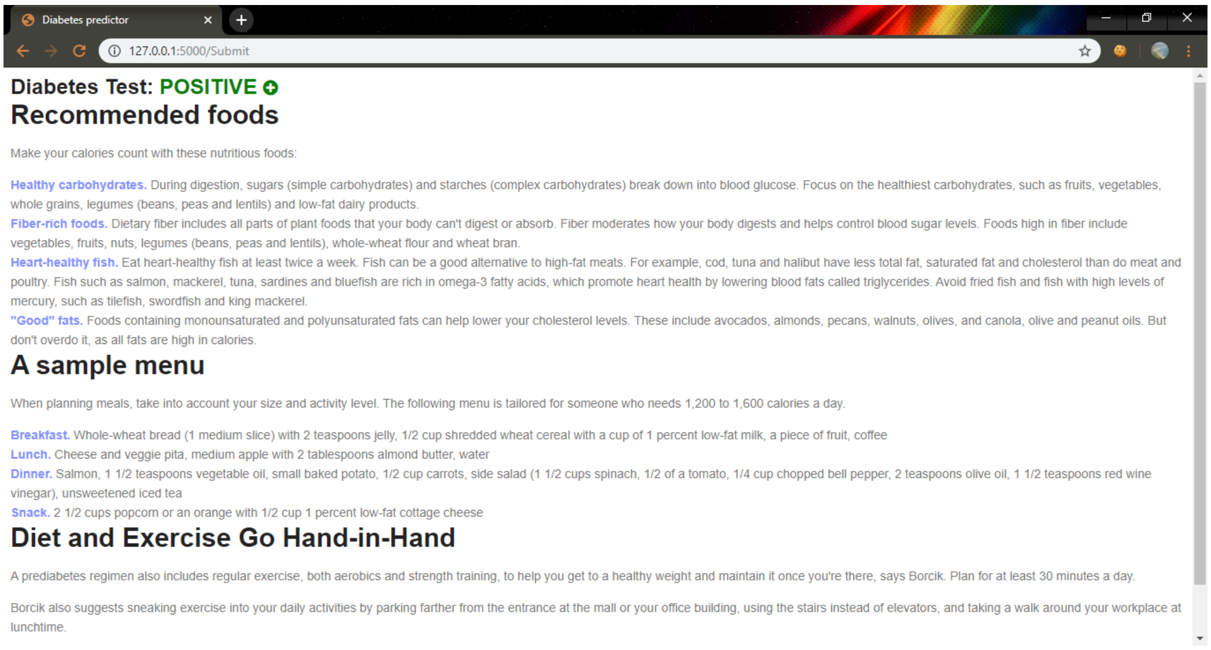
Age 39

Figure 4.6.2 : Form Page corresponding to the core diabetes attributes

After clicking Submit button your DIABETES TEST will predicted and the patient can Know whether they are suffering from DIABETES or not i.e. POSITIVE or NEGATIVE.

4.6.3 Positive Outcome

If the patient gets POSITIVE in result that means he is suffering from DIABETES. So he need to take care of his health by taking recommended nutritious food.
We also provided the sample menu i.e. what to take in Breakfast, Lunch, Dinner, Snacks. We suggested some physical exercise so by following those he can overcome the diabetes very soon.



Diabetes Test: POSITIVE

Recommended foods

Make your calories count with these nutritious foods:

Healthy carbohydrates. During digestion, sugars (simple carbohydrates) and starches (complex carbohydrates) break down into blood glucose. Focus on the healthiest carbohydrates, such as fruits, vegetables, whole grains, legumes (beans, peas and lentils) and low-fat dairy products.
Fiber-rich foods. Dietary fiber includes all parts of plant foods that your body can't digest or absorb. Fiber moderates how your body digests and helps control blood sugar levels. Foods high in fiber include vegetables, fruits, nuts, legumes (beans, peas and lentils), whole-wheat flour and wheat bran.
Heart-healthy fish. Eat heart-healthy fish at least twice a week. Fish can be a good alternative to high-fat meats. For example, cod, tuna and halibut have less total fat, saturated fat and cholesterol than do meat and poultry. Fish such as salmon, mackerel, tuna, sardines and bluefish are rich in omega-3 fatty acids, which promote heart health by lowering blood fats called triglycerides. Avoid fried fish and fish with high levels of mercury, such as tilefish, swordfish and king mackerel.
"Good" fats. Foods containing monounsaturated and polyunsaturated fats can help lower your cholesterol levels. These include avocados, almonds, pecans, walnuts, olives, and canola, olive and peanut oils. But don't overdo it, as all fats are high in calories.

A sample menu

When planning meals, take into account your size and activity level. The following menu is tailored for someone who needs 1,200 to 1,600 calories a day.

Breakfast. Whole-wheat bread (1 medium slice) with 2 teaspoons jelly, 1/2 cup shredded wheat cereal with a cup of 1 percent low-fat milk, a piece of fruit, coffee
Lunch. Cheese and veggie pita, medium apple with 2 tablespoons almond butter, water
Dinner. Salmon, 1 1/2 teaspoons vegetable oil, small baked potato, 1/2 cup carrots, side salad (1 1/2 cups spinach, 1/2 of a tomato, 1/4 cup chopped bell pepper, 2 teaspoons olive oil, 1 1/2 teaspoons red wine vinegar), unsweetened iced tea
Snack. 2 1/2 cups popcorn or an orange with 1/2 cup 1 percent low-fat cottage cheese

Diet and Exercise Go Hand-in-Hand

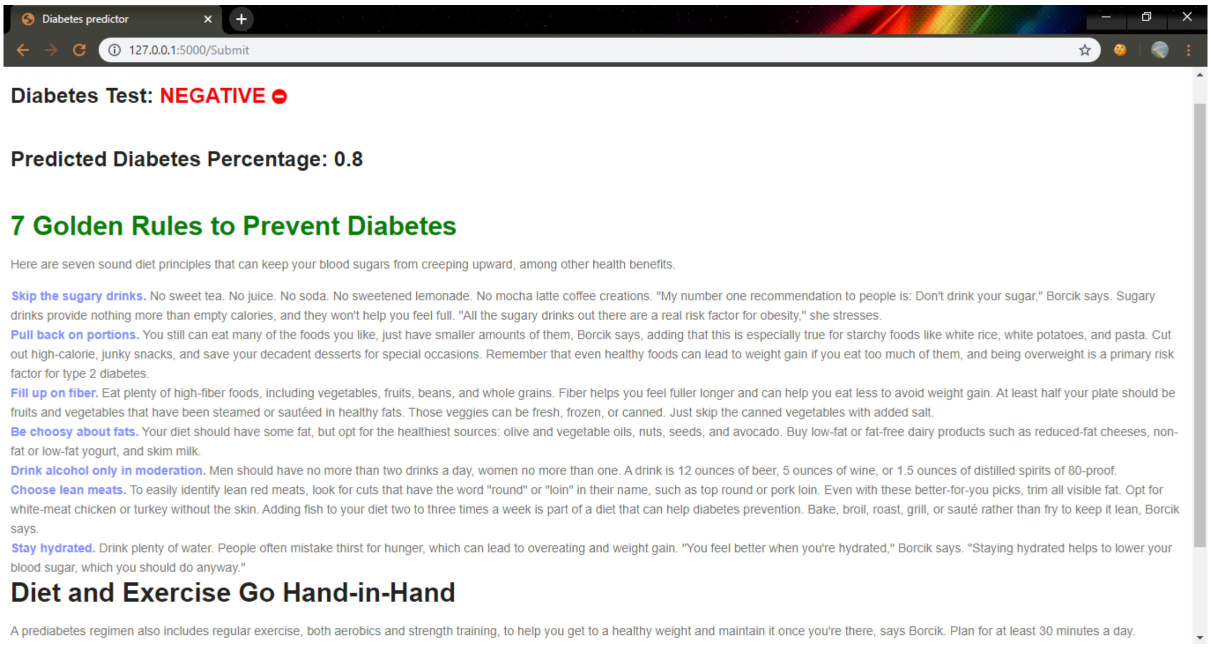
A prediabetes regimen also includes regular exercise, both aerobics and strength training, to help you get to a healthy weight and maintain it once you're there, says Borcik. Plan for at least 30 minutes a day.

Borcik also suggests sneaking exercise into your daily activities by parking farther from the entrance at the mall or your office building, using the stairs instead of elevators, and taking a walk around your workplace at lunchtime.

Figure 4.6.3 : Page corresponding to the positive result of Diabetes

4.6.4 Negative Outcome

If the Patient result is NEGATIVE then he is not suffering but the Patient can know how much chance is there that he can be attacked by DIABETES.
If the Patient knows that he can get attacked by DIABETES for him we are providing 7 Golden rules to prevent from DIABETES like food and exercise which is shown in the below figure.



Diabetes Test: NEGATIVE

Predicted Diabetes Percentage: 0.8

7 Golden Rules to Prevent Diabetes

Here are seven sound diet principles that can keep your blood sugars from creeping upward, among other health benefits.

Skip the sugary drinks. No sweet tea. No juice. No soda. No sweetened lemonade. No mocha latte coffee creations. "My number one recommendation to people is: Don't drink your sugar," Borcik says. Sugary drinks provide nothing more than empty calories, and they won't help you feel full. "All the sugary drinks out there are a real risk factor for obesity," she stresses.
Pull back on portions. You still can eat many of the foods you like, just have smaller amounts of them, Borcik says, adding that this is especially true for starchy foods like white rice, white potatoes, and pasta. Cut out high-calorie, junky snacks, and save your decadent desserts for special occasions. Remember that even healthy foods can lead to weight gain if you eat too much of them, and being overweight is a primary risk factor for type 2 diabetes.
Fill up on fiber. Eat plenty of high-fiber foods, including vegetables, fruits, beans, and whole grains. Fiber helps you feel fuller longer and can help you eat less to avoid weight gain. At least half your plate should be fruits and vegetables that have been steamed or sautéed in healthy fats. Those veggies can be fresh, frozen, or canned. Just skip the canned vegetables with added salt.
Be choosy about fats. Your diet should have some fat, but opt for the healthiest sources: olive and vegetable oils, nuts, seeds, and avocado. Buy low-fat or fat-free dairy products such as reduced-fat cheeses, non-fat or low-fat yogurt, and skim milk.
Drink alcohol only in moderation. Men should have no more than two drinks a day, women no more than one. A drink is 12 ounces of beer, 5 ounces of wine, or 1.5 ounces of distilled spirits of 80-proof.
Choose lean meats. To easily identify lean red meats, look for cuts that have the word "round" or "loin" in their name, such as top round or pork loin. Even with these better-for-you picks, trim all visible fat. Opt for white-meat chicken or turkey without the skin. Adding fish to your diet two to three times a week is part of a diet that can help diabetes prevention. Bake, broil, roast, grill, or sauté rather than fry to keep it lean, Borcik says.
Stay hydrated. Drink plenty of water. People often mistake thirst for hunger, which can lead to overeating and weight gain. "You feel better when you're hydrated," Borcik says. "Staying hydrated helps to lower your blood sugar, which you should do anyway."

Diet and Exercise Go Hand-in-Hand

A prediabetes regimen also includes regular exercise, both aerobics and strength training, to help you get to a healthy weight and maintain it once you're there, says Borcik. Plan for at least 30 minutes a day.

Figure 4.6.4 : Page corresponding to the negative result of Diabetes

5. CONCLUSION

Diabetes mellitus is reaching potentially epidemic proportions in India. The level of morbidity and mortality due to diabetes and its potential complications are enormous and pose significant healthcare burdens on both families and society. Worryingly, diabetes is now being shown to be associated with a spectrum of complications and to be occurring at a relatively younger age within the country. In India, the steady migration of people from rural to urban areas, the economic boom, and corresponding change in lifestyle are all affecting the level of diabetes. Yet despite the increase in diabetes there remains a paucity of studies investigating the precise status of the disease because of the geographical, socio-economic, and ethnic nature of such a large and diverse country. Given the disease is now highly visible across all sections of society within India, there is now the demand for urgent research and intervention - at regional and national levels - to try to mitigate the potentially catastrophic increase in diabetes that is predicted for the upcoming years.

6.BIBLIOGRAPHY

[1] PYTHON basics and syntax
https://www.tutorialspoint.com/python online training index.asp

[2] Python tutorials
https://www.youtube.com/watch?v=1EpYqtSlOr8.

[3] Anaconda: QuickStart and tutorials
https://www.youtube.com/watch?v=wbE5bgsk4RQ

[4] Flask tutorials
https://www.youtube.com/watch?v=5Qp5azTJ9XA