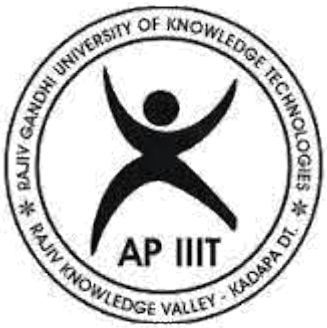
“DIABETES PREDICTION USING ML ”

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING



RGUKT
Rajiv Gandhi University of Knowledge Technologies
R.K. VALLEY

Submitted by

O NAVEEN – R170467

M NAGA VENKATA BHAVYA VANI – R170081

Under the Esteemed guidance
of Mr. SATYANANDARAM N
RGUKT RK Valley.

DECLARATION

We hereby declare that the report of the B. Tech Major Project Work entitled "DIABETES PREDICTION USING ML" which is being submitted to Rajiv Gandhi University of Knowledge Technologies, RK Valley, in partial fulfilment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide report of the work carried out by us. The material contained in this report has not been submitted to any university or institution for award of any degree.

O NAVEEN– R170467

M NAGA VENKATA BHAVYA VANI – R170081

Dept. Of Computer Science and Engineering.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

RGUKT

(A.P. Government Act 18 of 2008)
RGUKT, RK VALLEY
Department of Computer Science and Engineering

CERTIFICATE FOR PROJECT COMPLETION

This is to certify that the project entitled **"DIABETES PREDICTION USING ML" submitted by O NAVEEN(R170467),M NAGA VENKATA BHAVYA VANI(R170081),** under our guidance and supervision for the partial fulfilment for the degree Bachelor of Technology in Computer Science and Engineering during the academic year 2022-2023 at RGUKT, RK VALLEY. To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any University or Institute for the award of any degree or diploma.

Project Internal Guide

Mr. Satyanandaram N

Assistant Professor

RGUKT, RK Valley

Head of the Department

Mr. Satyanandaram N

HOD Of CSE

RGUKT, RK Valley

ABSTRACT

Diabetes Prediction :

Diabetes is a chronic disease with the potential to cause a world wide health care crisis. According to International Diabetes Federation 382 million people are living with diabetes across the whole world.By 2035,this will be doubled as 592 million. Diabetes mellitus or imply diabetes is a disease caused due to the increase level of blood glucose. Various traditional methods, based on physical and chemical tests, are available for diagnosing diabetes. However, early prediction of diabetes is quite challenging task for medical practitioners due to complex interdependence on various factors as diabetes affects human organs such as kidney, eye, heart, nerves, foot etc. Data science methods have the potential to benefit other scientific fields by shedding new light on common questions. One such task is to help make predictions on medical data. Machine learning is an emerging scientific field in data science dealing with the ways in which machines learn from experience. The aim of this project is to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by combining the results of different machine learning techniques.This project aims to predict diabetes via three different supervised machine learning methods including: SVM, Logistic regression, KNN. This project also aims to propose an effective technique for earlier detection of the diabetes disease using Machine learning algorithms and end to end deployment using Streamlit.

DIABETES PREDICTION USING ML

Introduction:

Diabetes is noxious diseases in the world. Diabetes caused because of obesity or high blood glucose level, and soforth. It affects the hormone insulin, resulting in abnormal metabolism of crabs and improves level of sugar in the blood. Diabetes occurs when body does not make enough insulin. According to (WHO) World Health Organizationabout 422 million people suffering from diabetes particu-larly from low or idle income countries. And this could be increased to 490 billion up to the year of 2030. However prevalence of diabetes is found among various Countrieslike Canada, China, and India etc. Population of India isnow more than 100 million so the actual number of diabet-ics in India is 40 million. Diabetes is major cause of death in the world. Early prediction of disease like diabetes can be controlled and save the human life. To accomplish this, this work explores prediction of diabetes by taking various attributes related to diabetes disease. For this purpose we use the Pima Indian Diabetes Dataset, we apply various Machine Learning classification and ensemble Techniques to predict diabetes. Machine Learning Is a method that is used to train computers or machines explicitly. Various
Machine Learning Techniques provide efficient result to collect Knowledge by building various classification and ensemble models from collected dataset. Such collected data can be useful to predict diabetes. Various techniques of Machine Learning can capable to do prediction, however it's tough to choose best technique. Thus for this purpose we apply popular classification and ensemble methods on
dataset for prediction.

Purpose

The purpose of this document is to gather the requirements that are needed for implementing the DIABETES PREDICTION. It also focuses on various key features, the product, product vision and scope, product overview. The aim of the project is to determine the appropriate classification model or algorithm that gives the best accuracy results ever possible. So, that algorithm proven to be the best can then be used in the prediction of diabetes to figure out if a person is diabetic or non-diabetic so far.

Intended Audience:

The diabetes prediction model is developed for health care purposes ,The intended audience would include policy makers, healthcare professionals ,public health officials and researchers.

Product Vision:

The product vision is to develop a Prediction System which is user friendly and accurate enough to predict the chance of a person is Diabetic or not . By using this system public researchers and healthcare professionals can early predict the chance of diabetes of a person.

Technologies:

- ➢ Machine Learning for Prediction System
- ➢ Python , Streamlit for UI

LITERATURE SURVEY

SUPPORT VECTOR MACHINE

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

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

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

SVM Algorithm used for Prediction:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

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

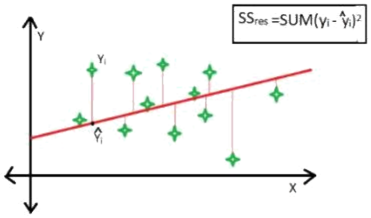
EVALUTION METRICS

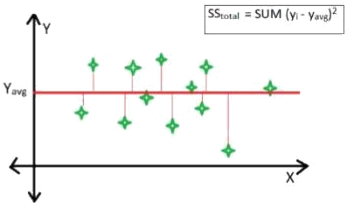
The main part of building a Machine Learning model is Evaluation of model. There are many methods of evaluation that can be used. While building our model we have used the below two metrics for the evaluation part.

R-Squared (R^2 or the coefficient of determination) :

- R-squared is a statistical method that determines the goodness of fit.
- It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
- The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
- It is also called a **coefficient of determination,** or **coefficient of multiple**
- **determination** for multiple regression.
- It can be calculated from the below formula:

$$R^2 = \frac{SS_{res}}{SS_{tot}}$$





Where,

SS_{res} = summation of squares of perpendicular distance between data points and the best-fitted line.

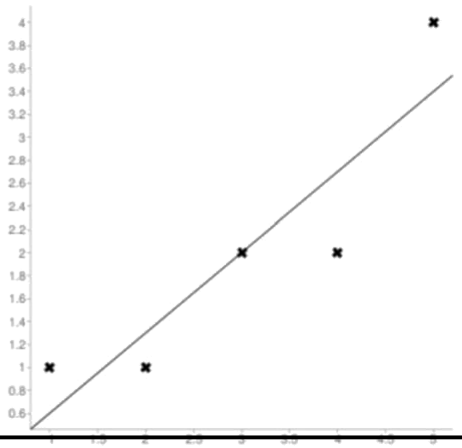
SS_{tot} = summation of squares of perpendicular distance between data points and the average line.

Mean Square Error (MSE):

Mean Square Error is the arithmetic mean of the squares of the predictions between the model and the observations. This is the value to be minimized in the context of a single or multiple regressions. It measures the average of error squares i.e. the average squared difference between the estimated values and true values. It is a risk function, corresponding to the expected value of the squared error loss.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

= Actual Output Value



Where,

n = Total no. of data points

Y_i

$\hat{Y}_I$ = Predicted Output Value

INITIAL RESULT OF ALGORITHMS

ALGORITHM	R2 (R SQUARED)	MEAN SQUARE ERROR
Support Vector Regression	0.8513	0.0792
Linear Regression	0.9417	0.0559
Random Forest Regression	0.9125	0.0492
Decision Tree Regression	0.8913	0.0743
Support Vector Machine	0.9126	0.0797

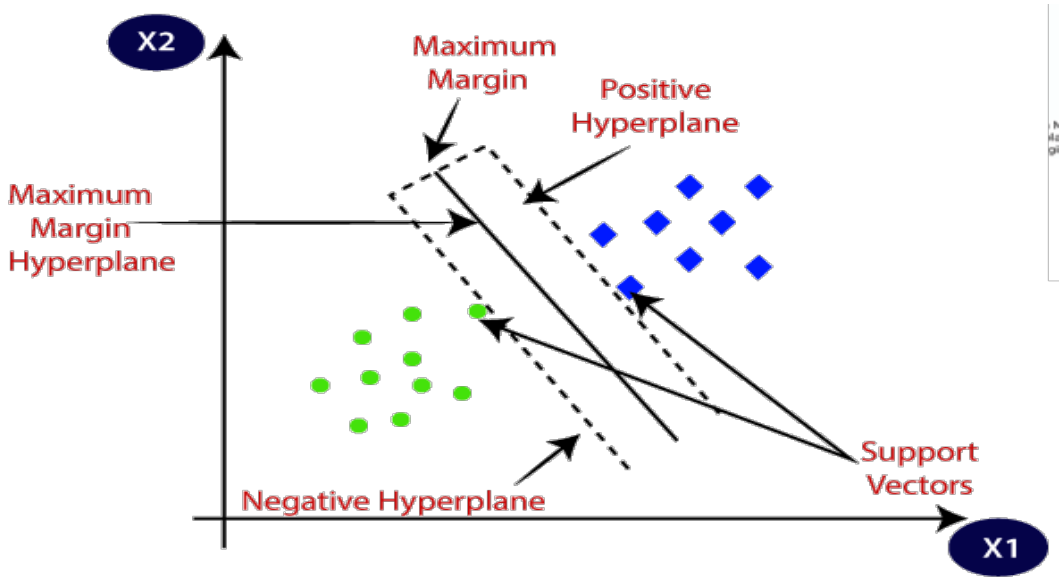
From table, we can see that the **Support vector Machine** suits best for our model.

SUPPORT VECTOR MACHINE

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

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

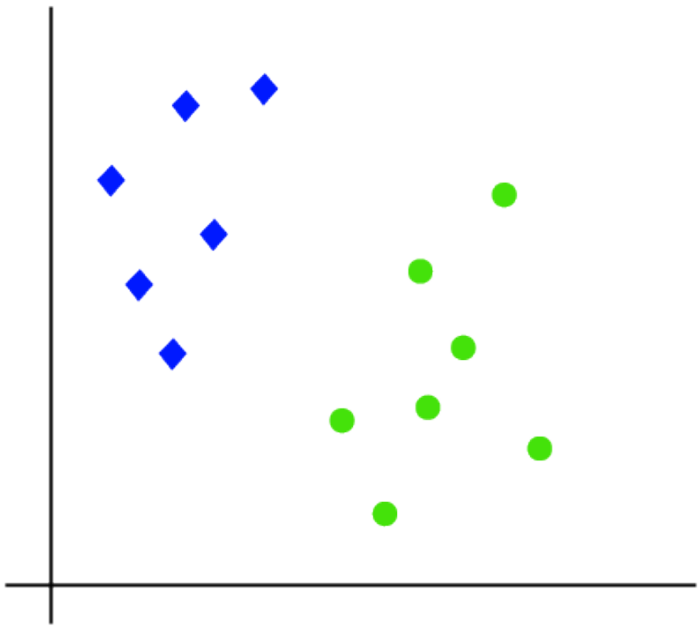
SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary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



How does SVM works?

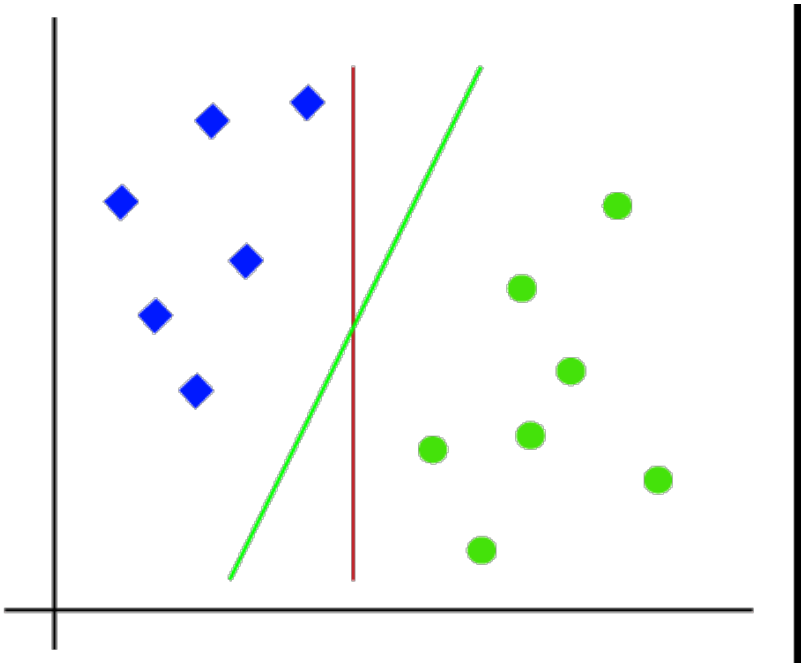
Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:

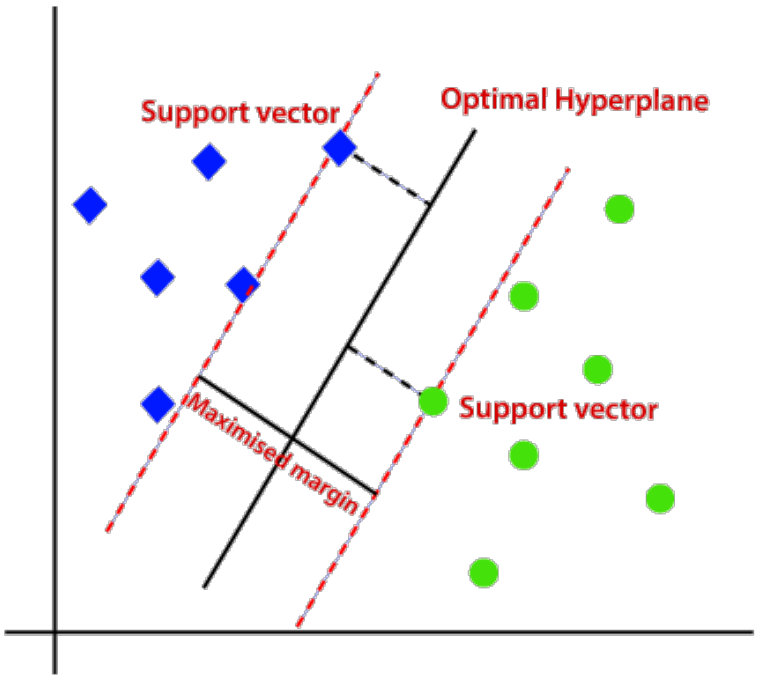


So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.

Consider the below image:



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

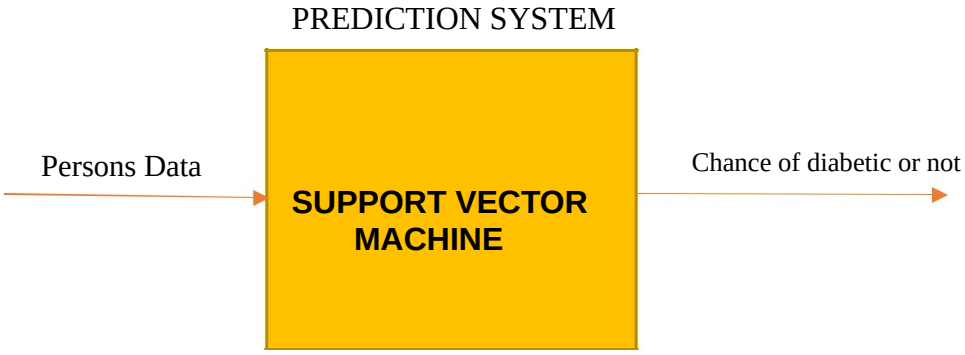


IMPLEMENTATION

PROPOSED SYSTEM

- So in this model we have proposed ‘Diabetes prediction using ML’ which predicts the percentage of chance of getting a person is Diabetic or not.

Working model



This Proposed System is built in following steps

1. Data Collection
2. Data Pre-processing
3. Model Building
4. Saving the Model using Pickle
5. Creating a WebApp using Streamlit

I used Jupyter notebook tool for this model building.

1.Data Collection

Initially I have collected the previous data related to admissions of IIIT and stored it in CSV file. After importing proper modules access the dataset using read_csv method

```
[1]: import numpy as np
     import pandas as pd
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn import svm
     from sklearn.metrics import accuracy_score
```

Data Collection and Analysis

PIMA Diabetes Dataset

```
[2]: # loading the diabetes dataset to a pandas DataFrame
```

```
In [5]: # printing the first 5 rows of the dataset
        diabetes_dataset.head()
```

Out[5]:

	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

Data features :

Given below table shows the data type of each attribute in our dataset.

SNO	Attribute Name	Type
1	No.of Pregnancies	Quantitative
2	Glucose	Quantitative
3	Blood Pressure	Quantitative
4	Skin Thickness	Quantitative
5	Insulin	Quantitative
6	BMI Value	Quantitative
7	DiabetespedgreeFunction Value	Quantitative
8	Age of the Person	Quantitative

Properties of Dataset :

```
In [6]: # number of rows and Columns in this dataset
        diabetes_dataset.shape
```

```
Out[6]: (768, 9)
```

```
In [7]: # getting the statistical measures of the data
        diabetes_dataset.describe()
```

Out[7]:

	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

2. Data Pre-Processing

It represents one of the most crucial steps in all Machine Learning projects because it involves formatting data, Improving data quality, feature engineering, and labelling .

Firstly we need to process the data such that there will be no null values and duplicate values. In the given data set there are so many parameters that are of type Integer.

We need to enter the values of data parameter such as No.of Pregnancies ,Glucose,Blood Pressure, Skin Thickness,Insulin , BMI Value , DiabetespedgreeFunction Value ,Age of the Person.

```
In [9]: diabetes_dataset['Outcome'].value_counts()
```

```
Out[9]: 0    500
        1    268
        Name: Outcome, dtype: int64

        0 --> Non-Diabetic

        1 --> Diabetic
```

```
In [10]: diabetes_dataset.groupby('Outcome').mean()
```

Out[10]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

```
In [11]: # separating the data and labels
         X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
         Y = diabetes_dataset['Outcome']
```

```
In [12]: print(X)
```

```
     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
```

Data Standardization : Data Standardization is used to converting a data into a standard format that computers can easily understand and use.We use this data standardization in support vector machine tries to maximize the instance between the separating plane and support vectors. Mainly Standardization is used for cleaning the data or transforms member data into a format that makes attribute comparision easier.

Data Standardization

```
In [17]: scaler = StandardScaler()

In [18]: scaler.fit(X)

Out[18]: StandardScaler
         StandardScaler()

In [19]: standardized_data = scaler.transform(X)

In [20]: print(standardized_data)

[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]

In [21]: X = standardized_data
         Y = diabetes_dataset['Outcome']

In [22]: print(X)
         print(Y)

[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

3 . Model Building

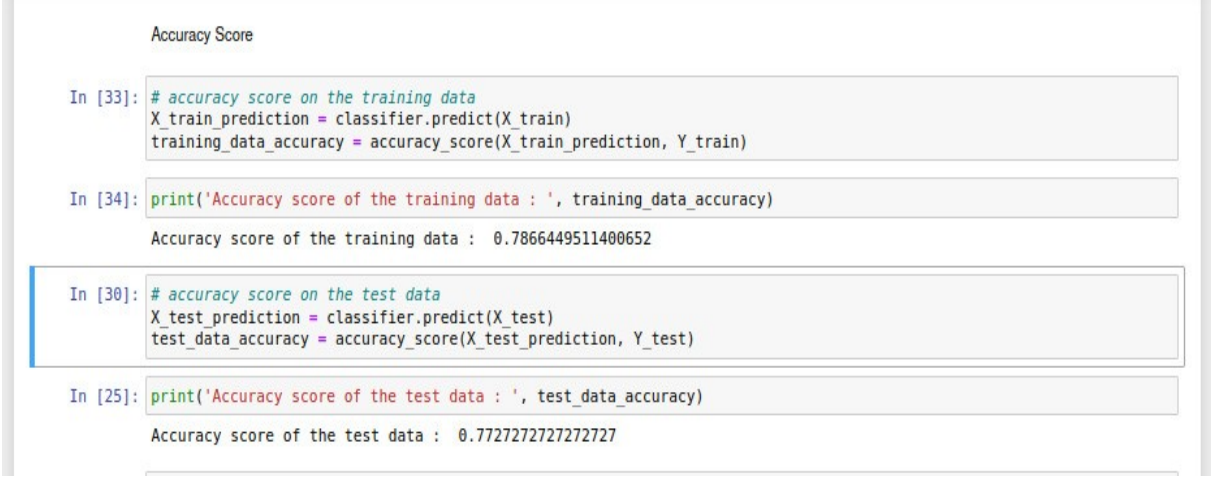
As the final dataset is ready, now we can build our model. Import the Support Vector Machine model from sklearn module.Divide the dataset into two parts such as Test data and Train Data.

Now train the test data using train data by fitting into Support Vector Machine Model.

```
Train Test Split

In [24]: X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)

In [25]: print(X.shape, X_train.shape, X_test.shape)

(768, 8) (614, 8) (154, 8)

Training the Model

In [26]: classifier = svm.SVC(kernel='linear')

In [27]: #training the support vector Machine Classifier
         classifier.fit(X_train, Y_train)

Out[27]: SVC
         SVC(kernel='linear')
```

Now the prediction model build successfully.

Testing Accuracy of Model :



Accuracy Score

```
In [33]: # accuracy score on the training data
         X_train_prediction = classifier.predict(X_train)
         training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

In [34]: print('Accuracy score of the training data : ', training_data_accuracy)

         Accuracy score of the training data :  0.7866449511400652

In [30]: # accuracy score on the test data
         X_test_prediction = classifier.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [25]: print('Accuracy score of the test data : ', test_data_accuracy)

         Accuracy score of the test data :  0.7727272727272727
```

4. Saving The Model :

```
In [28]: import pickle

In [29]: filename='trained_model.sav'
         pickle.dump(classifier,open(filename,'wb'))

In [30]:  #Loading the Saved Model
         loaded_model=pickle.load(open('trained_model.sav','rb'))

In [31]: input_data = (5,166,72,19,175,25.8,0.587,51)

         # changing the input_data to numpy array
         input_data_as_numpy_array = np.asarray(input_data)

         # reshape the array as we are predicting for one instance
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

         # standardize the input data
         std_data = scaler.transform(input_data_reshaped)
         print(std_data)

         prediction = loaded_model.predict(std_data)
         print(prediction)

         if (prediction[0] == 0):
           print('The person is not diabetic')
         else:
           print('The person is diabetic')

         [[ 0.3429808   1.41167241  0.14964075 -0.09637905  0.82661621 -0.78595734
            0.34768723  1.51108316]]
         [1]
         The person is diabetic
```

By using pickle python tool that allows you to save your ml models to minimize lengthy retraining and allow you to share,commit and reload pretrained machine learning models.

5.Creating WebApp Using Streamlit :

Streamlit is a open-source Python library that makes it easy to create and share beautiful custom web apps for machine learning and datascience.

Streamlit turns datascripts into shareable web apps in minutes.Streamlit is the easiest way especially for people with no front-end knowledge to put their code into a web application.

Source Code :

model.py

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('diabetes.csv')
# printing the first 5 rows of the dataset
diabetes_dataset.head()
# number of rows and Columns in this dataset
diabetes_dataset.shape
# getting the statistical measures of the data
diabetes_dataset.describe()
diabetes_dataset['Outcome'].value_counts()
diabetes_dataset.groupby('Outcome').mean()
# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
print(Y)
#Data Standardization
scaler = StandardScaler()
scaler.fit(X)
standardized_data = scaler.transform(X)
print(standardized_data)
X = standardized_data
Y = diabetes_dataset['Outcome']
print(X)
print(Y)
#Train Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y,
random_state=2)
print(X.shape, X_train.shape, X_test.shape)
#Training the Model
classifier = svm.SVC(kernel='linear')
#Accuracy Score
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy score of the training data : ', training_data_accuracy)
# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
#Making a Predictive System
nput_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')

#Saving the Trained Model
import pickle

filename='trained_model.sav'
pickle.dump(classifier,open(filename,'wb'))

#Loading the Saved Model
loaded_model=pickle.load(open('trained_model.sav','rb'))
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = loaded_model.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

Diabetes_prediction_webapp.py

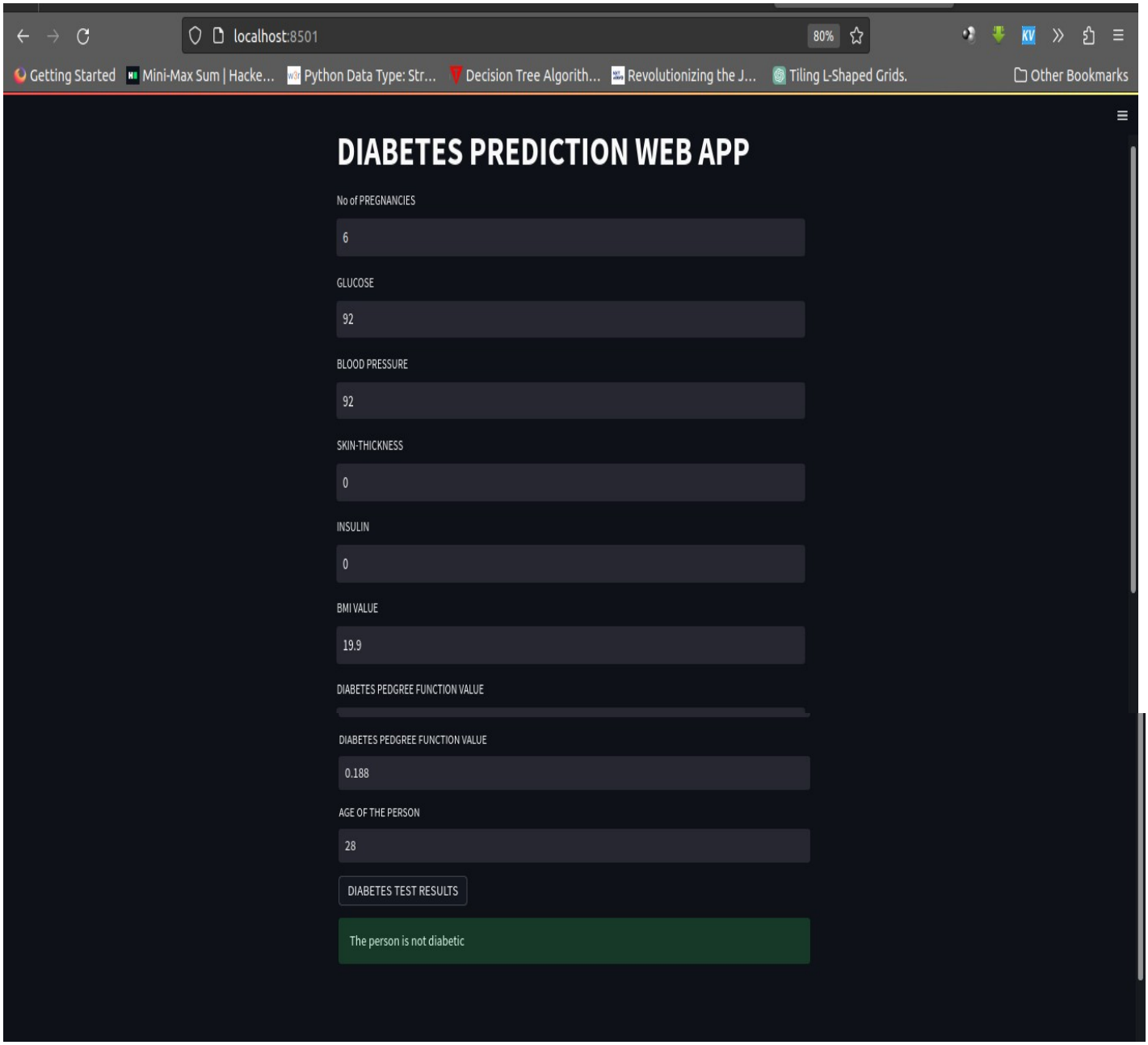
```
#!/usr/bin/env python3
# -- coding: utf-8 --
"""
Created on Wed Apr  5 15:29:48 2023

@author: student
"""
import numpy as np
import pickle
import streamlit as st
#Loading the Saved Model
loaded_model=pickle.load(open('/home/student/Desktop/diabeties
prediction/trained_model.sav','rb'))

# Creating a function for Prediction
def Diabetes_prediction(input_data):


    # changing the input_data to numpy array
    input_data_as_numpy_array = np.asarray(input_data)

    # reshape the array as we are predicting for one instance
    input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)


    prediction = loaded_model.predict(input_data_reshaped)
    print(prediction)

    if (prediction[0] == 0):
      return 'The person is not diabetic'
    else:
      return 'The person is diabetic'



def main():
    #Giving a title
    st.title('DIABETES PREDICTION WEB APP')

    Pregnancies= st.text_input('no of pregnancies')

    Glucose= st.text_input('Glucose')

    BloodPressure= st.text_input('BloodPressure')

    SkinThickness= st.text_input('SkinThickness')
```

```
    Insulin= st.text_input('Insulin')

    BMI= st.text_input('BMI VALUE')

    DiabetespedigreeFunction= st.text_input('DiabetespedgreeFunction Value')

    Age= st.text_input('Age of the person')

    #Code for prediction
    diagnosis=''

   #Creating a button for Prediction
    if st.button('Diabetes  Test Results'):
        diagnosis =
Diabetes_prediction([ Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,Diabetesp
edigreeFunction,Age])


    st.success(diagnosis)

if _name_ == '_main_':
    main()
```

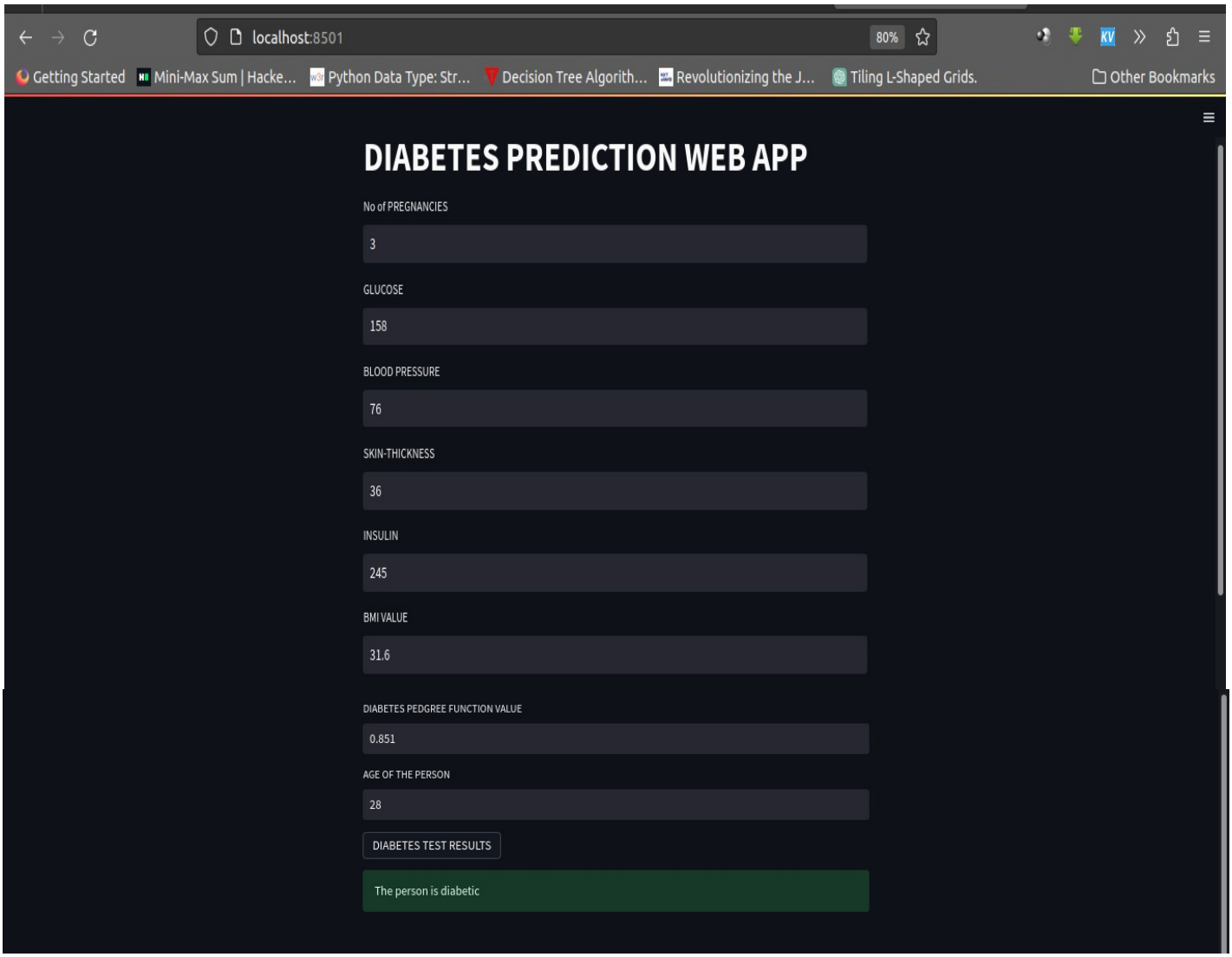
prediction.py

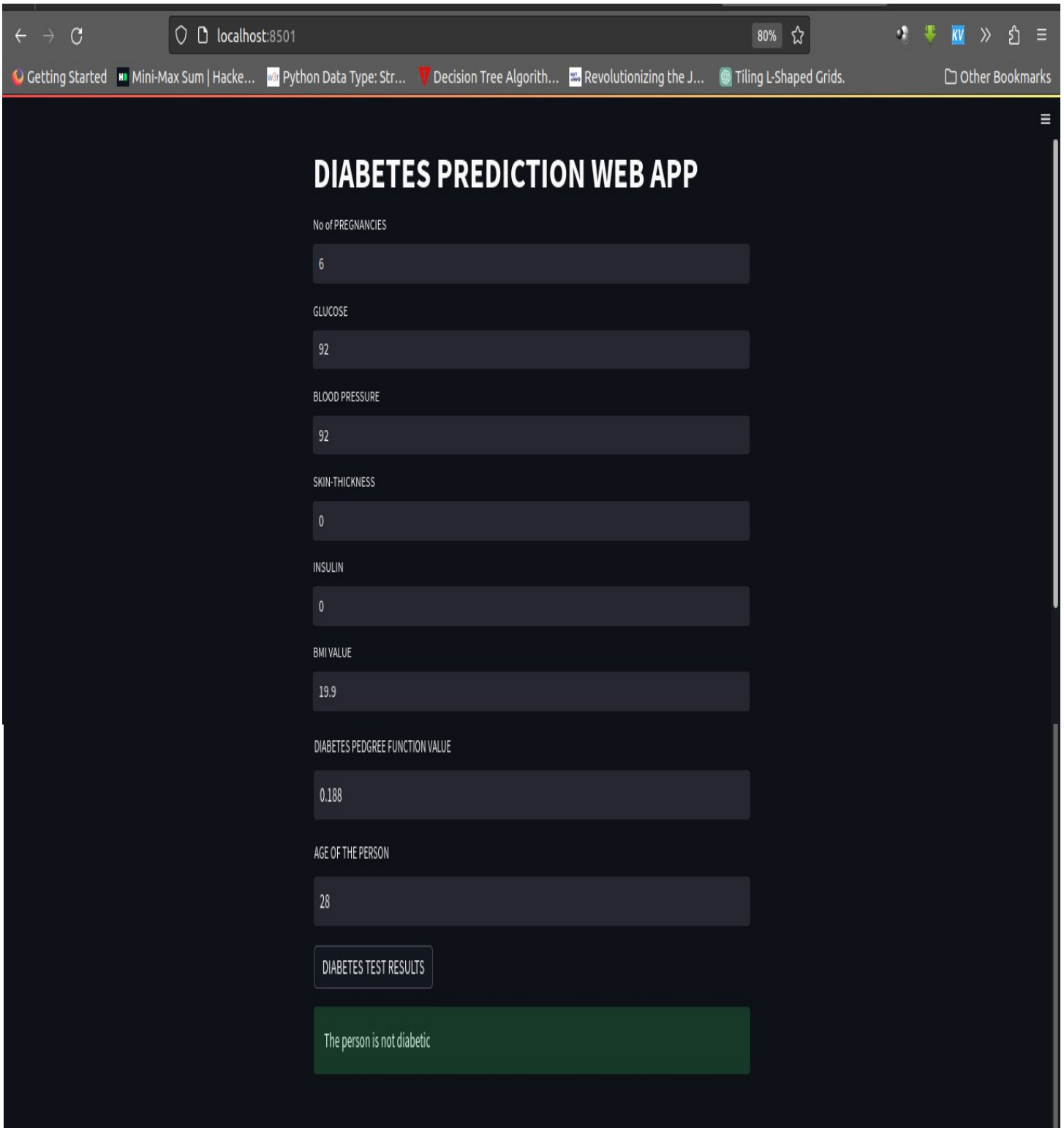
```
import numpy as np
import pickle
#Loading the Saved Model
loaded_model=pickle.load(open('/home/student/Desktop/diabeties
prediction/trained_model.sav','rb'))
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)


prediction = loaded_model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

RESULT :


localhost:8501
80%
Getting Started
Mini-Max Sum | Hacke...
Python Data Type: Str...
Decision Tree Algorith...
Revolutionizing the J...
Tiling L-Shaped Grids.
Other Bookmarks
DIABETES PREDICTION WEB APP
No of PREGNANCIES
6
GLUCOSE
92
BLOOD PRESSURE
92
SKIN-THICKNESS
0
INSULIN
0
BMI VALUE
19.9
DIABETES PEDGREE FUNCTION VALUE
DIABETES PEDGREE FUNCTION VALUE
0.188
AGE OF THE PERSON
28
DIABETES TEST RESULTS
The person is not diabetic

t


localhost:8501
80%
Getting Started
Mini-Max Sum | Hacke...
Python Data Type: Str...
Decision Tree Algorith...
Revolutionizing the J...
Tiling L-Shaped Grids.
Other Bookmarks
DIABETES PREDICTION WEB APP
No of PREGNANCIES
3
GLUCOSE
158
BLOOD PRESSURE
76
SKIN-THICKNESS
36
INSULIN
245
BMI VALUE
31.6
DIABETES PEDGREE FUNCTION VALUE
0.851
AGE OF THE PERSON
28
DIABETES TEST RESULTS
The person is diabetic


localhost:8501
80%
Getting Started
Mini-Max Sum | Hacke...
Python Data Type: Str...
Decision Tree Algorith...
Revolutionizing the J...
Tiling L-Shaped Grids.
Other Bookmarks
DIABETES PREDICTION WEB APP
No of PREGNANCIES
6
GLUCOSE
92
BLOOD PRESSURE
92
SKIN-THICKNESS
0
INSULIN
0
BMI VALUE
19.9
DIABETES PEDGREE FUNCTION VALUE
0.188
AGE OF THE PERSON
28
DIABETES TEST RESULTS
The person is not diabetic

CONCLUSION

In this project we used Support Vector Machine algorithm to provide chance of getting a person is Diabatic or Not. Here we used the persons health parameters data to train the model and Testing. . Whenever person or Health Researcher enter his details, the model gives the prediction based on similar previous data. So that sperson will know their chance of having a Diabetic Disease or not.

FUTURE ENHANCEMENTS

❖ In Future ,the designed system with the used machine learning classification algorithms can be used to predict or diagnose other diabetes analysis including some other machine Learning Algorithms.

❖ We will also try to improve the interface more interactive i.e. queries section, help section etc.

REFERENCE

[1] GeeksForGeeks : https://www.geeksforgeeks.org/ml-linear-regression/
[2] Scikit Learn :
https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[3] Javatpoint : https://www.javatpoint.com/machine-learning
[4] ResearchGate : https://www.researchgate.net/publication/348433004/
[5] Youtube :