Diabetes Mellitus Prediction

Submitted by
Hamza

in partial fulfillment for the award of the

Internship

C20/1 JSS-STEP
Sector 62, Noida, India
www.dexob.com

business@dexob.com

Under the supervision of

Dr. IHTIRAM RAZA KHAN

(2023)

DECLARATION

I, **Mr. HAMZA** a student of **Bachelors in Technology Computer Science and Engineering with Artificial Intelligence (B.Tech C.S.E-AI),)** hereby declare that the Project/Dissertation entitled **"Diabetes Prediction"** which is being submitted by me to the Dexob company, Noida in partial fulfillment of the requirement for the award of the internship certificate**,** is my original work and has not been submitted anywhere else for the award of any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

HAMZA

Date:

Place: Noida sector 62

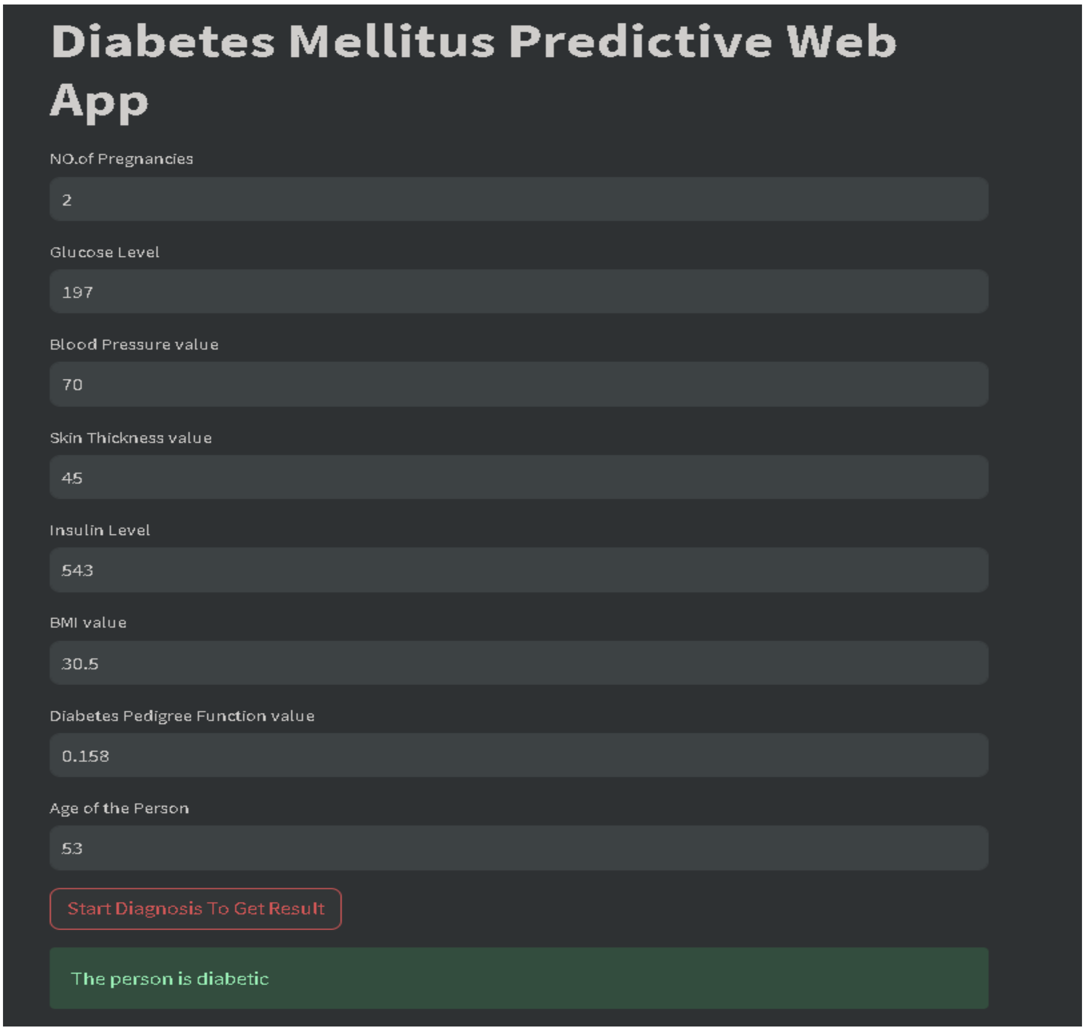
ACKNOWLEDGEMENT

I would want to express my gratitude to a number of persons who have encouraged and assisted me in the preparation of this project, both directly and indirectly. It allows me to look back and reflect on the support I've had during this process.

I would like to express my gratitude to **Dr. IHTIRAM RAZA KHAN**, my mentor, as well as the professors of Jamia Hamdard University, for their invaluable recommendations, innovative criticisms, and support throughout the writing process. I would like to offer my heartfelt gratitude to the entire faculty of Jamia Hamdard University (SEST).

My heartfelt gratitude goes out to my family and friends, as well as my esteemed university, Jamia Hamdard, for providing me with the chance and infrastructure to complete this project. Finally, I want to express my gratitude to everyone who assisted me in gathering data during the creation of the project, without whom it would not have been possible.

Diabetes Prediction



Diabetes Mellitus Predictive Web App

NO.of Pregnancies

2

Glucose Level

197

Blood Pressure value

70

Skin Thickness value

45

Insulin Level

543

BMI value

30.5

Diabetes Pedigree Function value

0.158

Age of the Person

53

The person is diabetic

Must watch page number 29 live working of this project

INTRODUCTION

Diabetes Prediction using Machine Learning:

1. I created machine learning system that can predict a person have diabetes or not.
2. I used one of the important machine learning algorithm called SVM support vector machine.
3. Coding done in Python programming language.

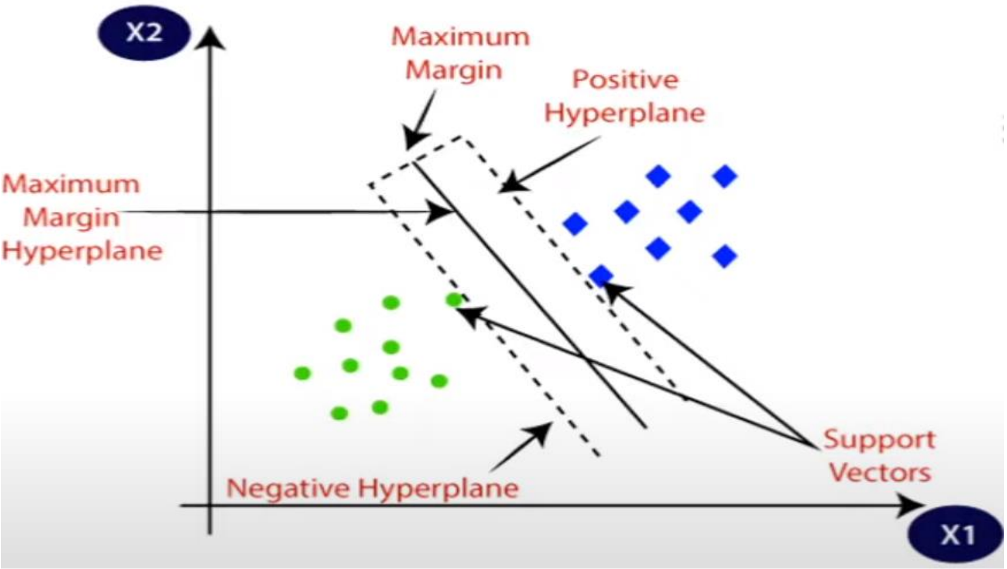
SVM→Is support vector machine which is supervised learning used for classification and regression problems.

- It used to separate two classes using hyper plane it decides data belongs to which class.

- To get a **margin** we draw 2 lines which is parallel to **hyperplane.**
- It decide which hyperplane exist.

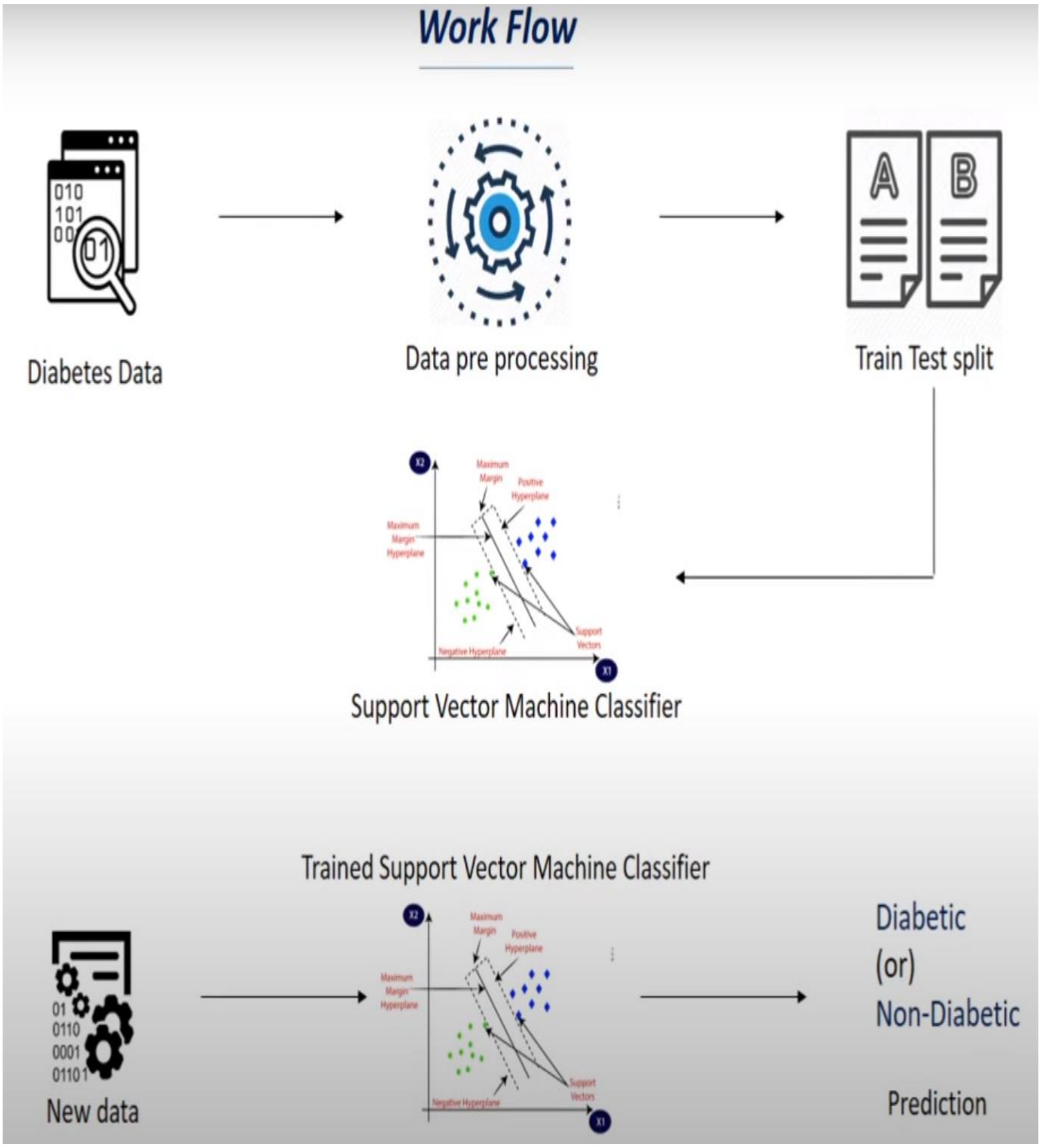
- **Support vector** are those points which are nearest to opponent class.

- In **supervised learning** we used label data we trained our model with blood pressure glucose BMI insulin and other attributes.



- After that hyperplane separates into two groups(diabetic or non diabetic).

Data flow Diagram



Data :Is taken from Kaggle PIMA diabetes data set which content information: for female patients having diabetes and non - diabetes and different attributes like: No. of pregnancy, Glucose level, Insulin, Blood pressure, Skinthickness, Body mass index, Age and Diabetes pedigree function.

Data preprocessing: I analyze data and making data suitable to feed in ML model by standardizing data to & make it in same range**.**

Splitting into test and training data sets: The 20% of data is used for testing purpose and remaining use for training and we also find the accuracy which is above 75%.
Now SVM classifier text the data then new data comes then system predict the user is diabetic or not bases of training data.

Need of Diabetes Prediction: From last few years we observe that lots of diabetes cases specially in females those who are homemakers so they can give input to check.

Some patient don’t want to visit to a doctor they can check with the system.

To learn and understand how SVM works with real life.

OBJECTIVE : **Use supervised machine learning algorithm and understand how it works with real life problem which is diabetes prediction.**

Make a system which can spread awareness for diabetes to make country healthier by preventing persons from diabetes it can be possible by time to time monitoring, So here user can give input for few things like body mass index, Glucose, Insulin level, Blood pressures and other to know weather they have diabetic or not. This system provides **75% percent above accuracy.**

Create a system which can predict person is diabetic or not.

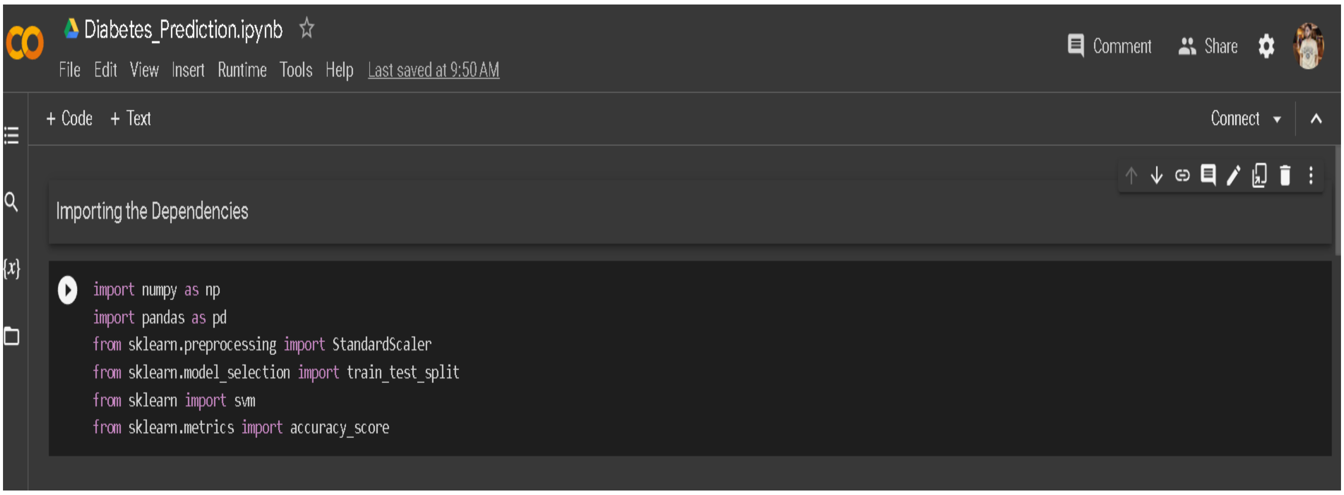
Aim was what approach will be taken to solve ML projects.

LIBRARIES USED :

- ➔ **Numpy→Use to create arrays which is helpful in processing.**
- ➔ **Pandas→Creating data frames useful to put data in structure table and to read CSV.**
- ➔ **Standard scalar→** For data standardization different attributes have different value range so it very difficult for machine learning model to make prediction.

  So we standardize to particular range fitting all inconsistent data with standard scalar function based on it transforming to common now all data is in range of zero to one it will help model to make better predictions.
- ➔ **Train tests split→**Used to split data into testing and training.

  Where the test data is unknown data for the model 20% of data we separated for best remaining will be used for training purpose.
- ➔ **SVM→I**s support vector machine which is the main factor to classify which is supervised machine learning algorithm, Used for classification and regression problem separates two class using hyper-planes.
- ➔ **Accuracy score→T**his library used to know the accuracy of the model means how much percentage prediction is correct so **out of 100 prediction 79 is correct which is above 75 which is good.**


CO
Diabetes_Prediction.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 9:50 AM
Comment
Share
+ Code + Text
Connect
{x}

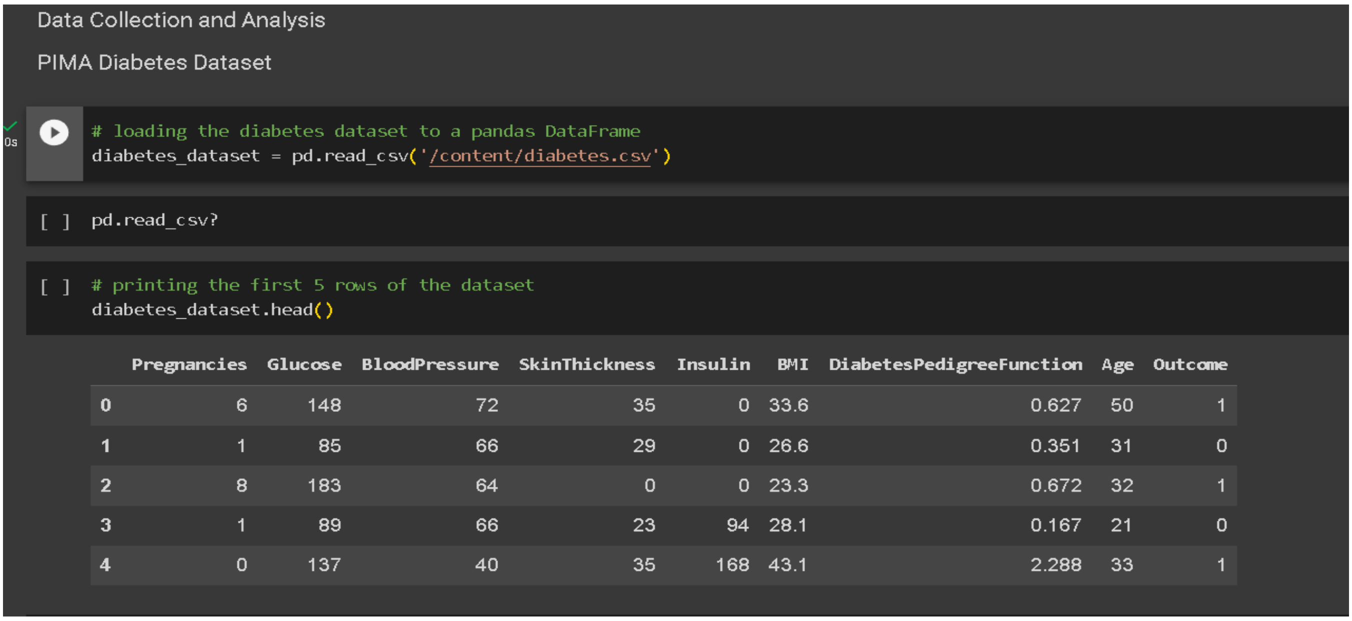
Importing the Dependencies

```
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
```

METHODOLOGY OF THE MODEL BUILDING FOR DIABETES PREDICTION:

Data collection and Analysis

importing the data to the colab and trying to understand the content and useful insight which are present data set.



Data Collection and Analysis

PIMA Diabetes Dataset

```
# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('/content/diabetes.csv')
```

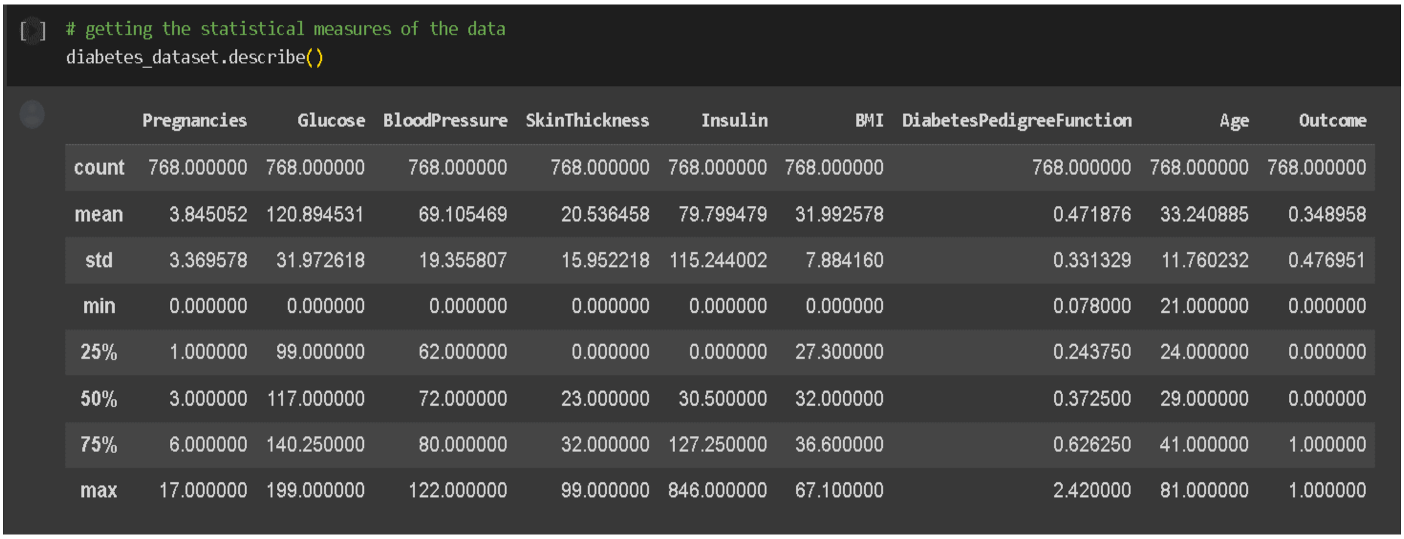
```
[ ] pd.read_csv?
```

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

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

The Pima Indian Diabetes Dataset, originally from the National Institute of Diabetes and Digestive and Kidney Diseases, contains information of 768 women from a population near Phoenix, Arizona, USA.

Content 9 attributes.

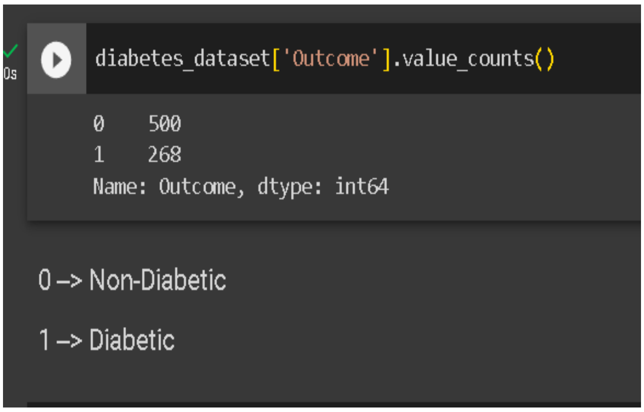


```
# getting the statistical measures of the data
diabetes_dataset.describe()
```

	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

This function gives the mean, std etc.

It provides statistical measures of data.
eg: Mean of glucose is 120



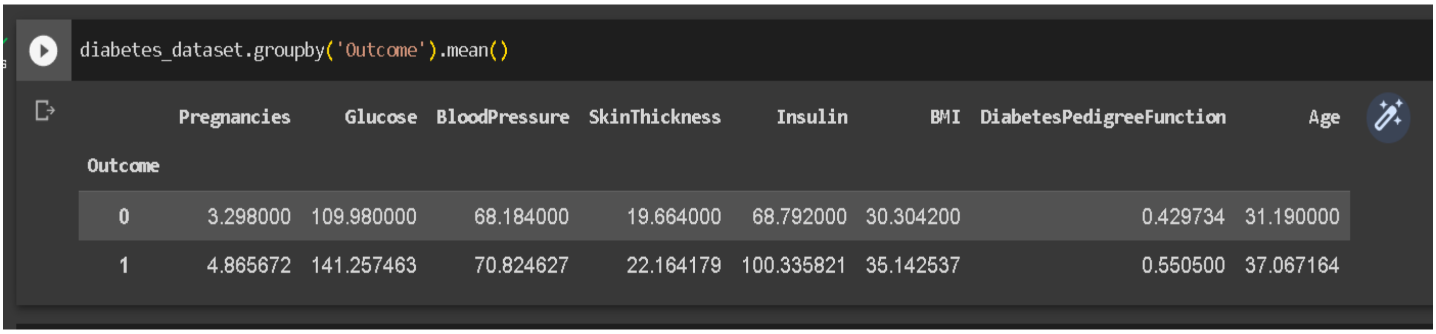
```
diabetes_dataset['Outcome'].value_counts()
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

0 –> Non-Diabetic

1 –> Diabetic

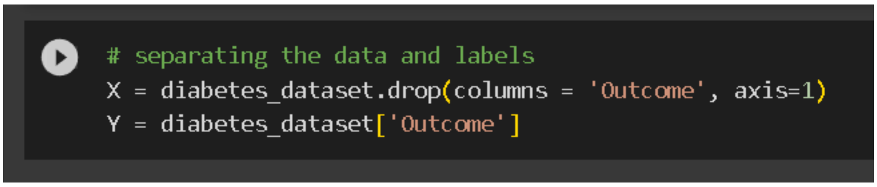
it is showing with label zero means non diabetic persons are 500 in this data set and one represent diabetic which are 268.



```
diabetes_dataset.groupby('Outcome').mean()
```

```
         Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin        BMI  DiabetesPedigreeFunction        Age
Outcome
      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
```

Trying to give mean to label 0 and 1 this insight will be used to tell a person is diabetic or not if a person glucose is 148 or 150 and other attributes will also be compared predict to predict.

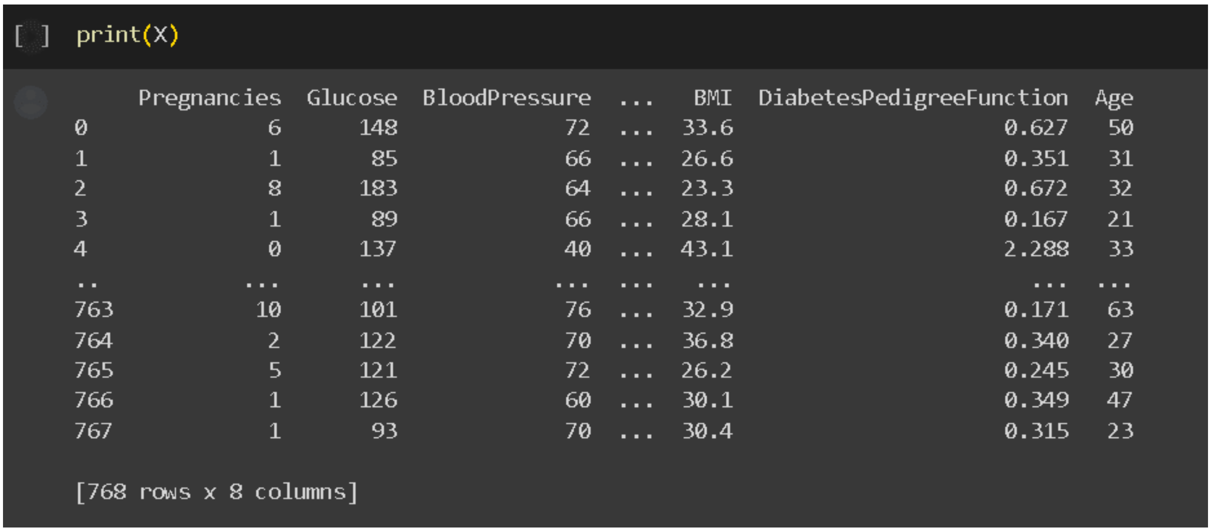


```
# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

Separating the data and labels using drop function.

axis=0 for row

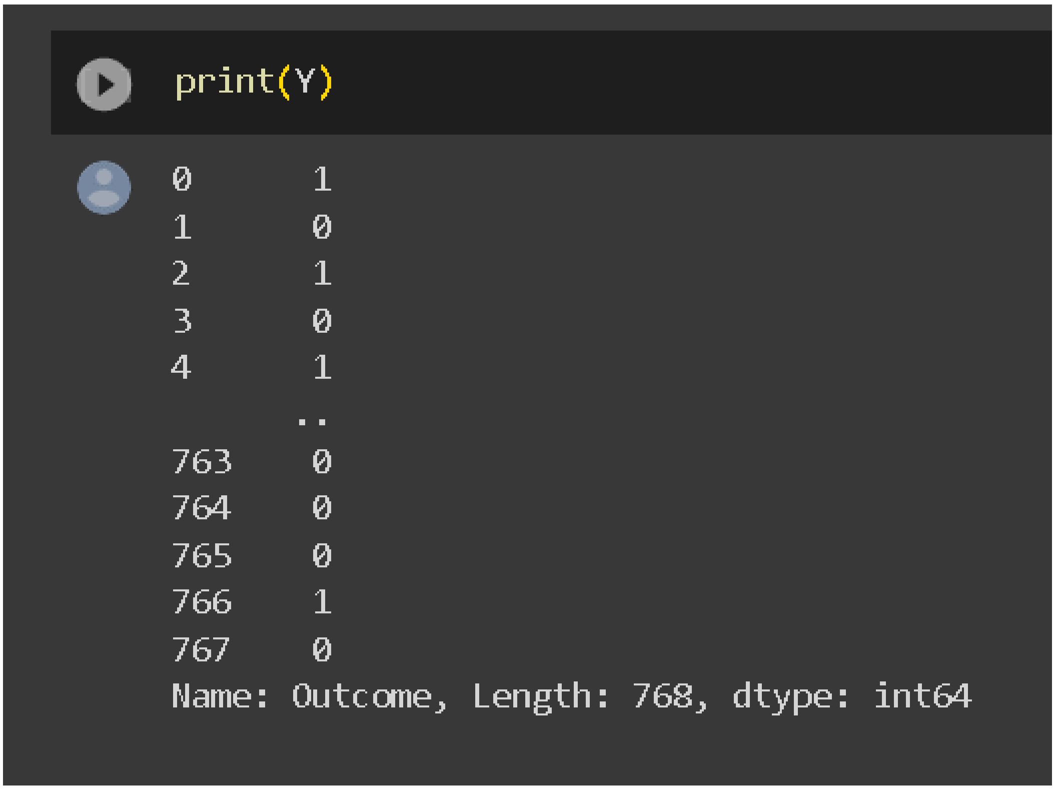
Now X having data.



```
[ ] print(X)
```

```
     Pregnancies  Glucose  BloodPressure  ...   BMI  DiabetesPedigreeFunction  Age
0              6      148             72  ...  33.6                     0.627   50
1              1       85             66  ...  26.6                     0.351   31
2              8      183             64  ...  23.3                     0.672   32
3              1       89             66  ...  28.1                     0.167   21
4              0      137             40  ...  43.1                     2.288   33
..           ...      ...            ...  ...   ...                       ...  ...
763           10      101             76  ...  32.9                     0.171   63
764            2      122             70  ...  36.8                     0.340   27
765            5      121             72  ...  26.2                     0.245   30
766            1      126             60  ...  30.1                     0.349   47
767            1       93             70  ...  30.4                     0.315   23

[768 rows x 8 columns]
```

Y is having labels.



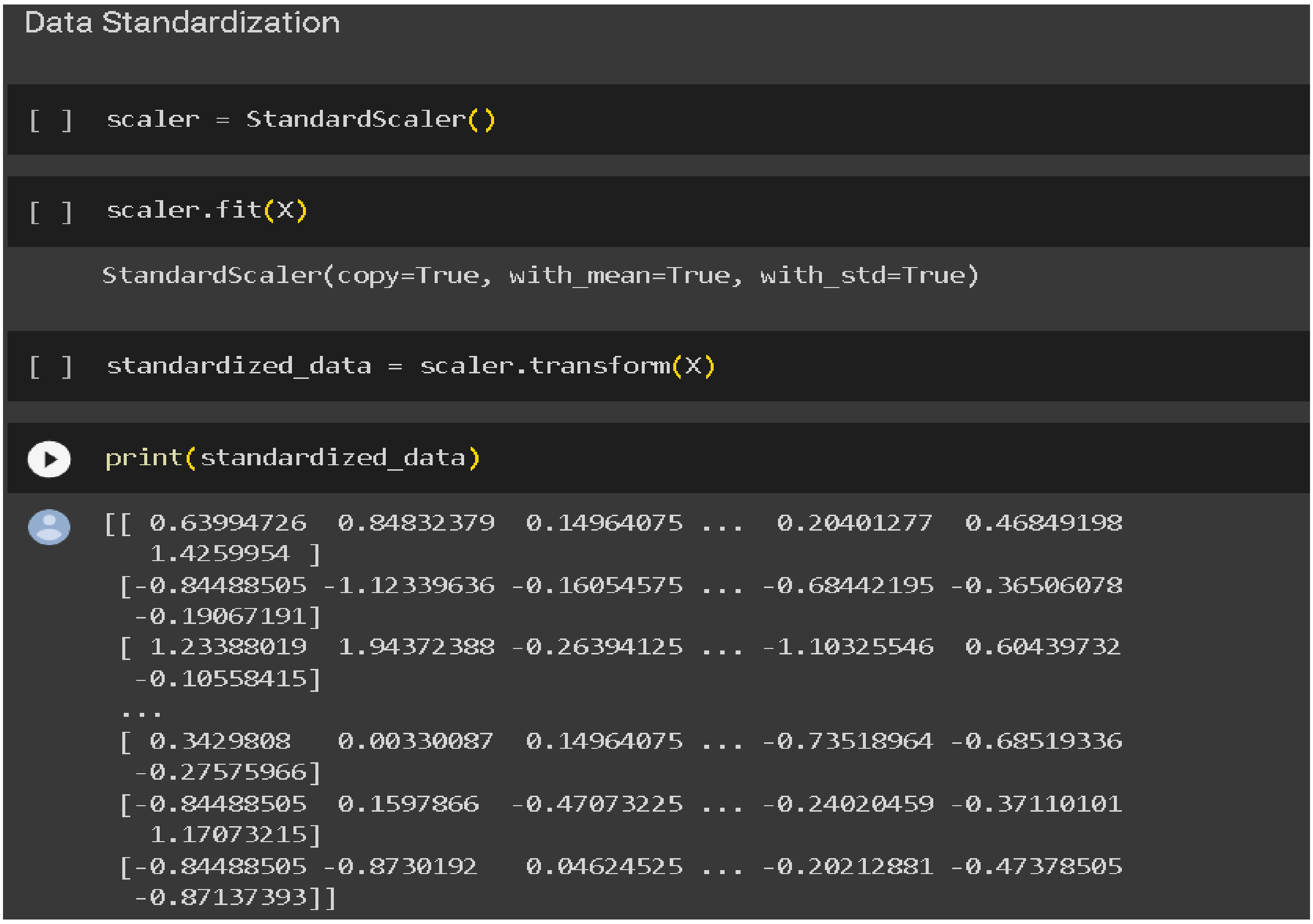
```
print(Y)
```

```
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
```

In data preprocessing: Data standardization is important step.

Why to do this? Because in data there are different attributes having different value range so it will difficult for machine learning model to make predictions .

In order to handle it we standardize data to particular range using a standard scalar function taking on instance fitting all inconsistent data with standard scalar function based on it transforming to common now, All data is in range of zero to one it will help ml model to predict do better predictions.



Data Standardization

```
[ ]  scaler = StandardScaler()
```

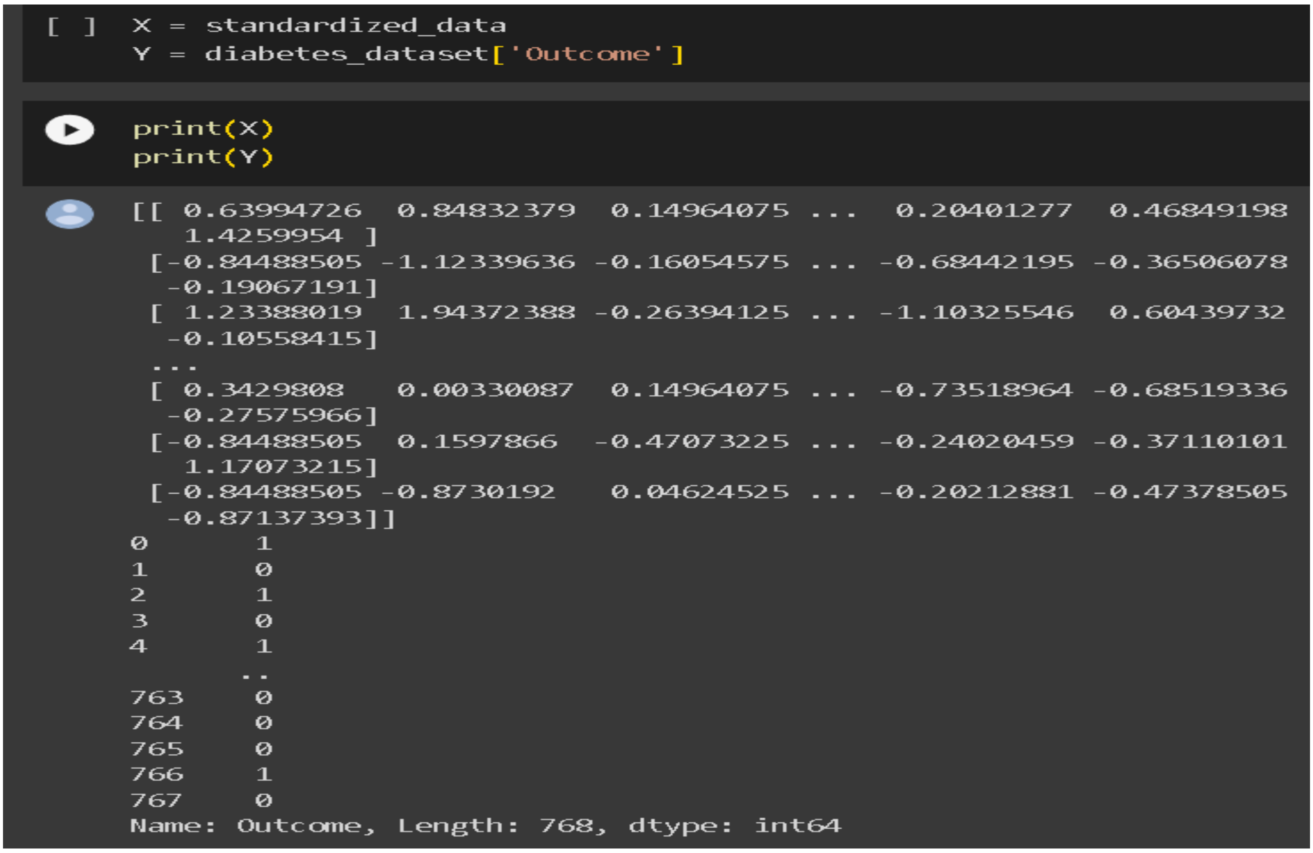
```
[ ]  scaler.fit(X)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
[ ]  standardized_data = scaler.transform(X)
```

```
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]]
```



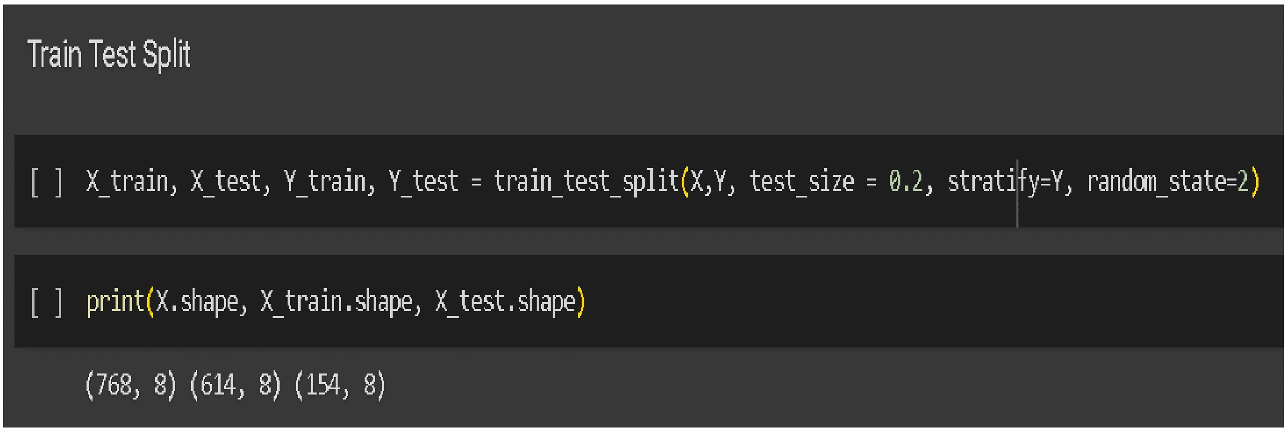
```
[ ]  X = standardized_data
     Y = diabetes_dataset['Outcome']
```

```
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
```

Now splitting the data→ Test data is unknown data for model.

- 0.2 means 20% of the data for test.
- satisfy is = y to avoid similar proposition.
- Random state splitting data into one form.

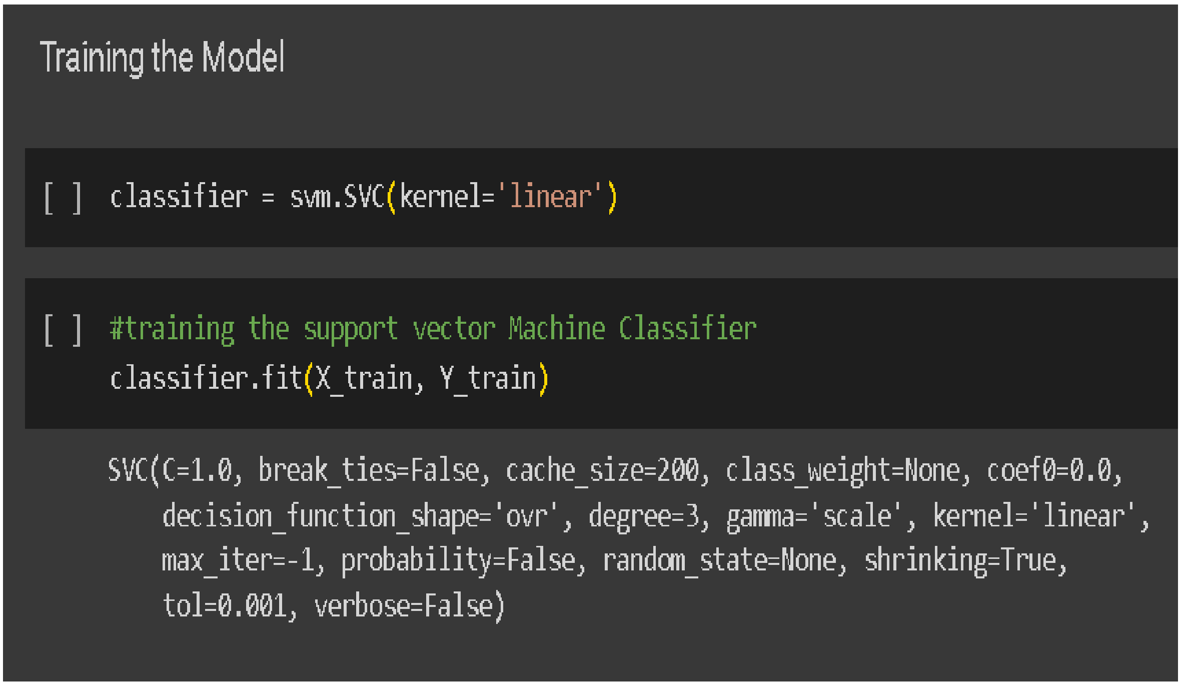


Train Test Split

```
[ ]  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)
```

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



Training the Model

```
classifier = svm.SVC(kernel='linear')
```

```
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

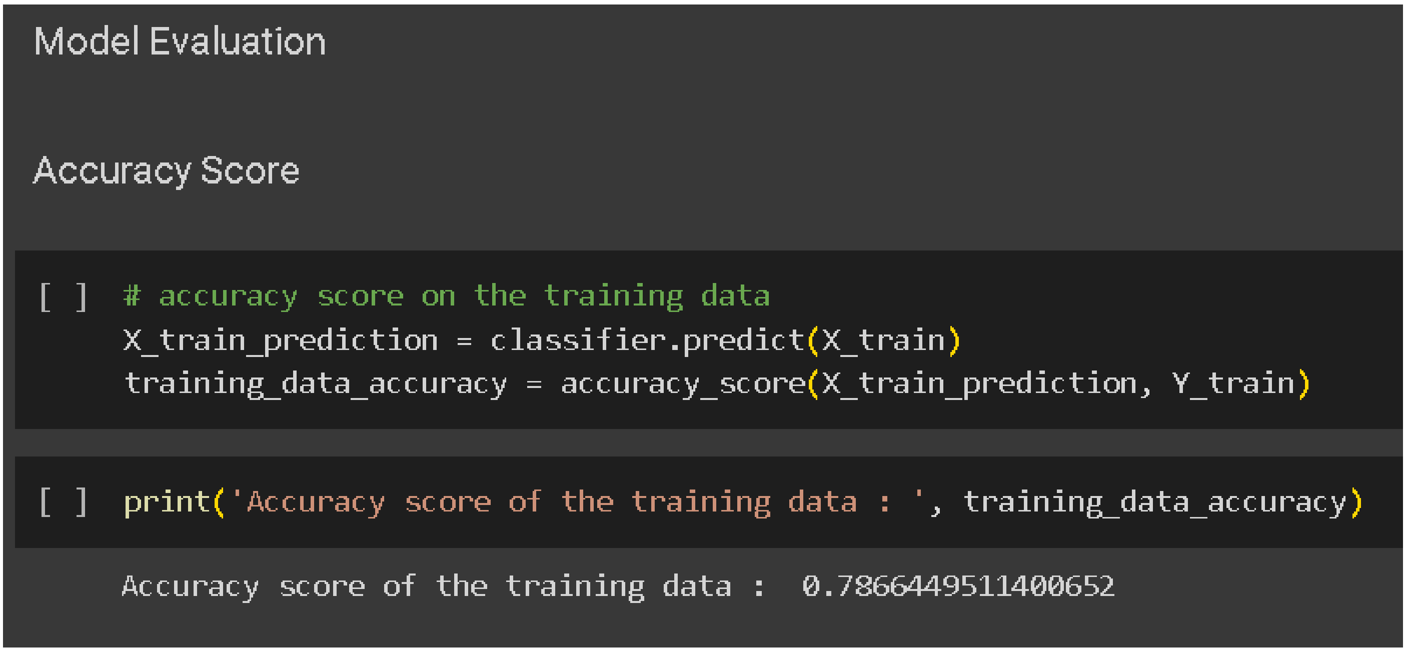
Support vector classifier with linear model

basically loading SVC

and providing training data with labels.

Benefit of small data training does not take much time.

Model evaluation



Model Evaluation

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 of the training data :  0.7866449511400652
```

We obtained accuracy of training data approx. 79%

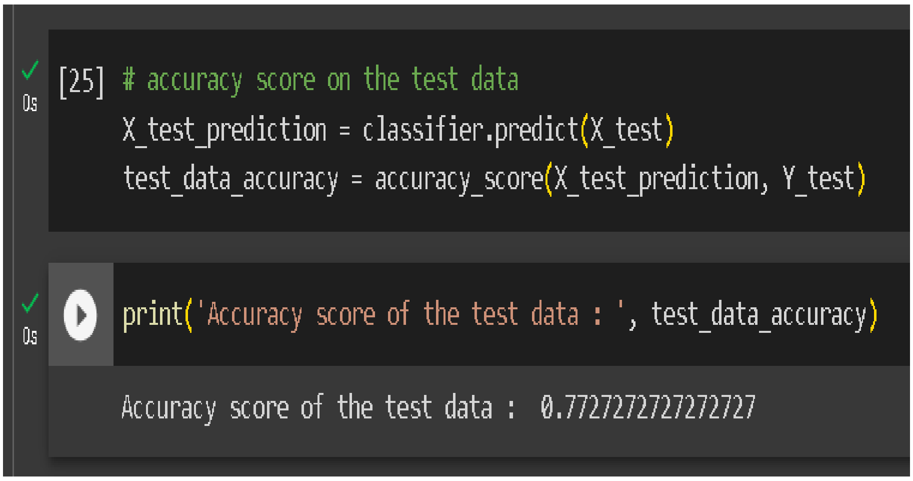
used predict function and accuracy score function.

Predicting all labels for training data and then comparing prediction with original y_train then predict accuracy score.

Optimization techniques can also be used to get more accuracy.

Now lets see **accuracy on test data**

we obtain the accuracy 77% which is pretty good for test data wish signifies that Overfitting is an undesirable machine learning behavior that occurs when the machine learning model gives accurate predictions for training data but not for new data.



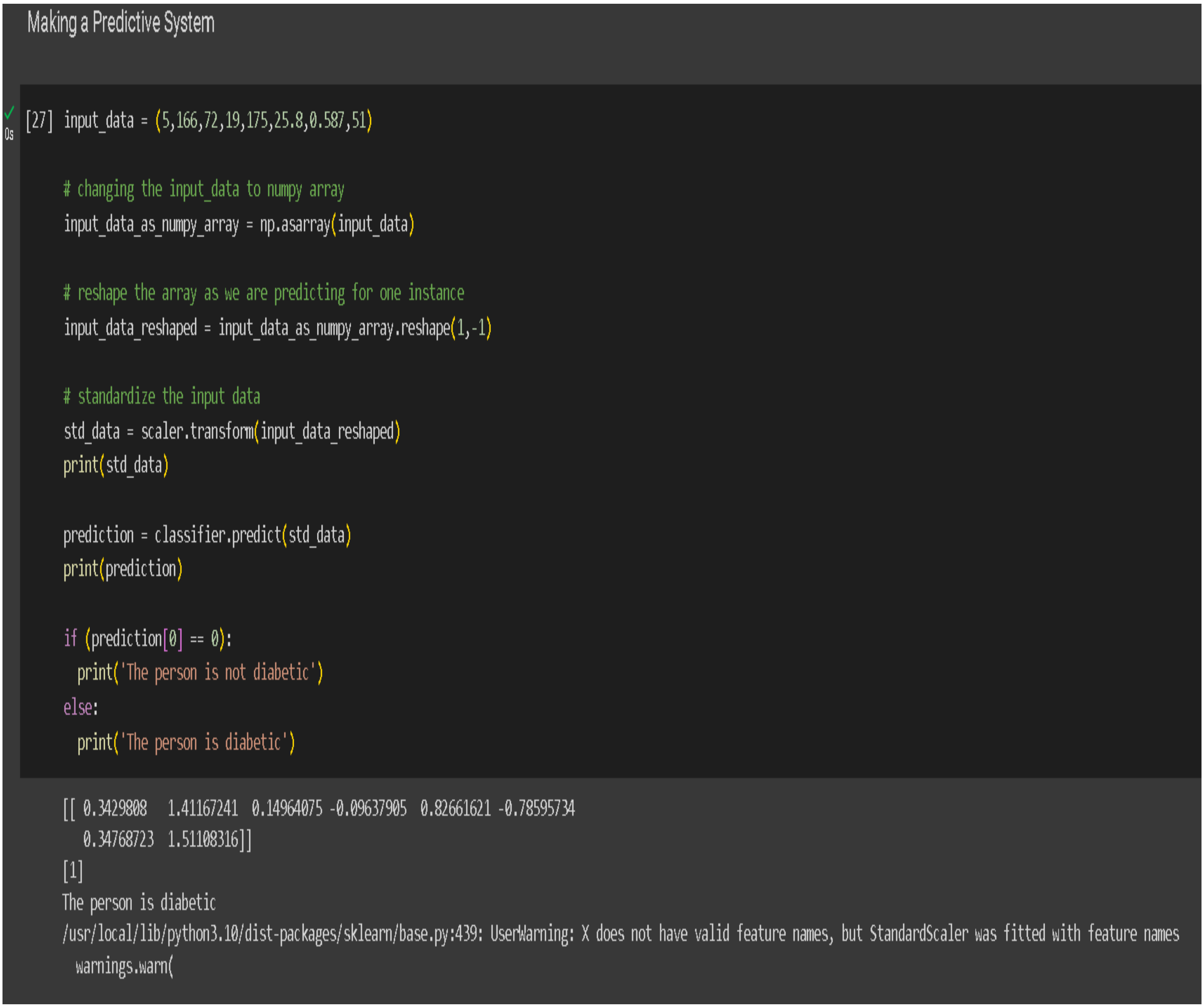
```
# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7727272727272727
```

Underfitting is a scenario in data science where a data model is unable to capture the relationship between the input and output variables accurately, generating a high error rate on both the training set and unseen data.

<u>RESULT : </u>System Which Can Predict

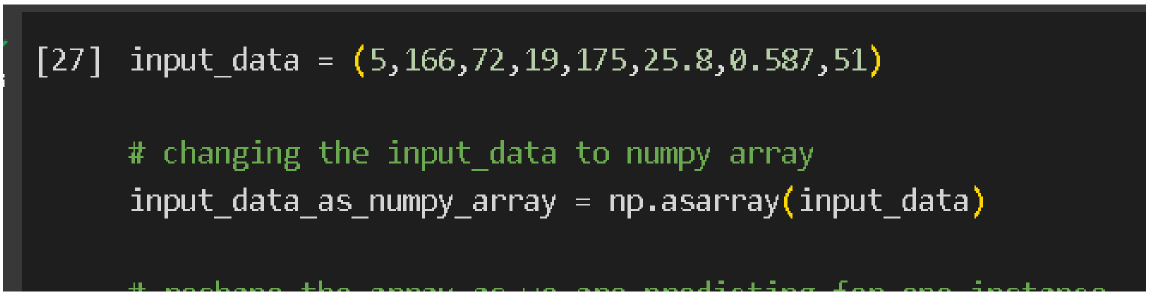


Making a Predictive System

```
[27] 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 = classifier.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
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

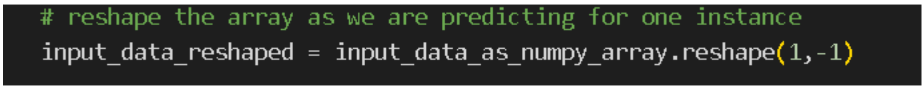
Numpy is also used here for **changing input data from list to numpy array**



```
[27] 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
```

reason so processing will be **easier** by the help of np,as array() which is present in numpy library which we imported.

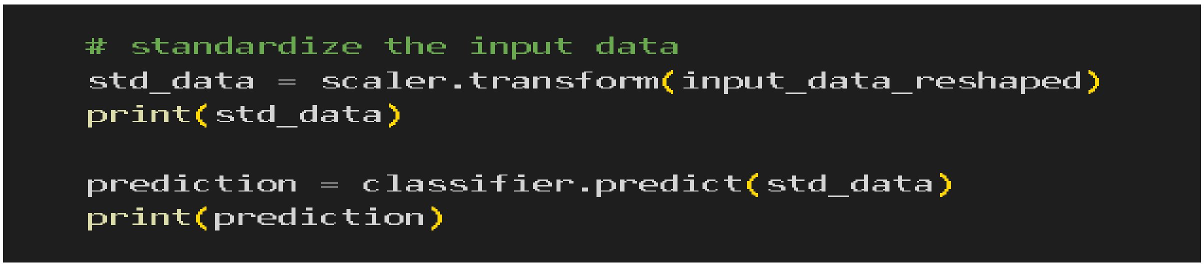
Re-shape our trained model on 768 example but in input we are giving only one,



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

So to prevent model from confusion.

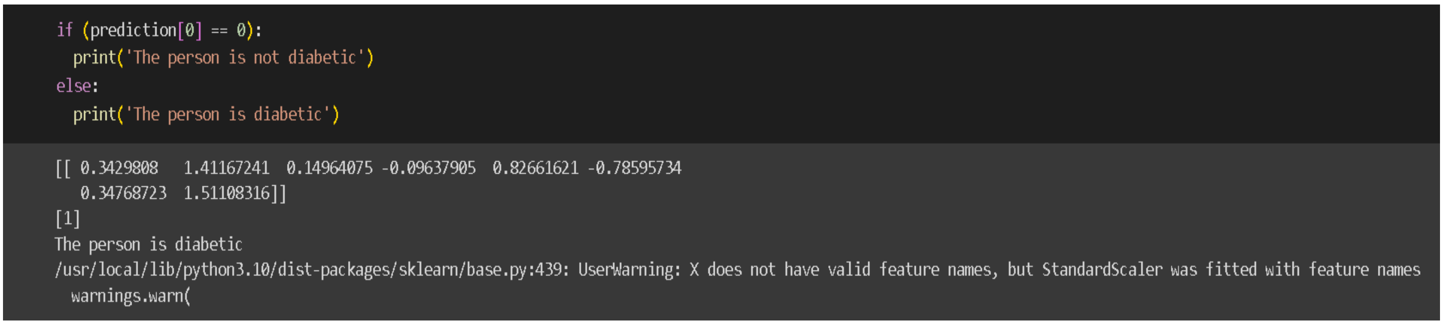
Standardized data to make a prediction variable



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

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

classifier in which we trained ML model

First two rows shows standardized data by the standard scalar function almost in same range.

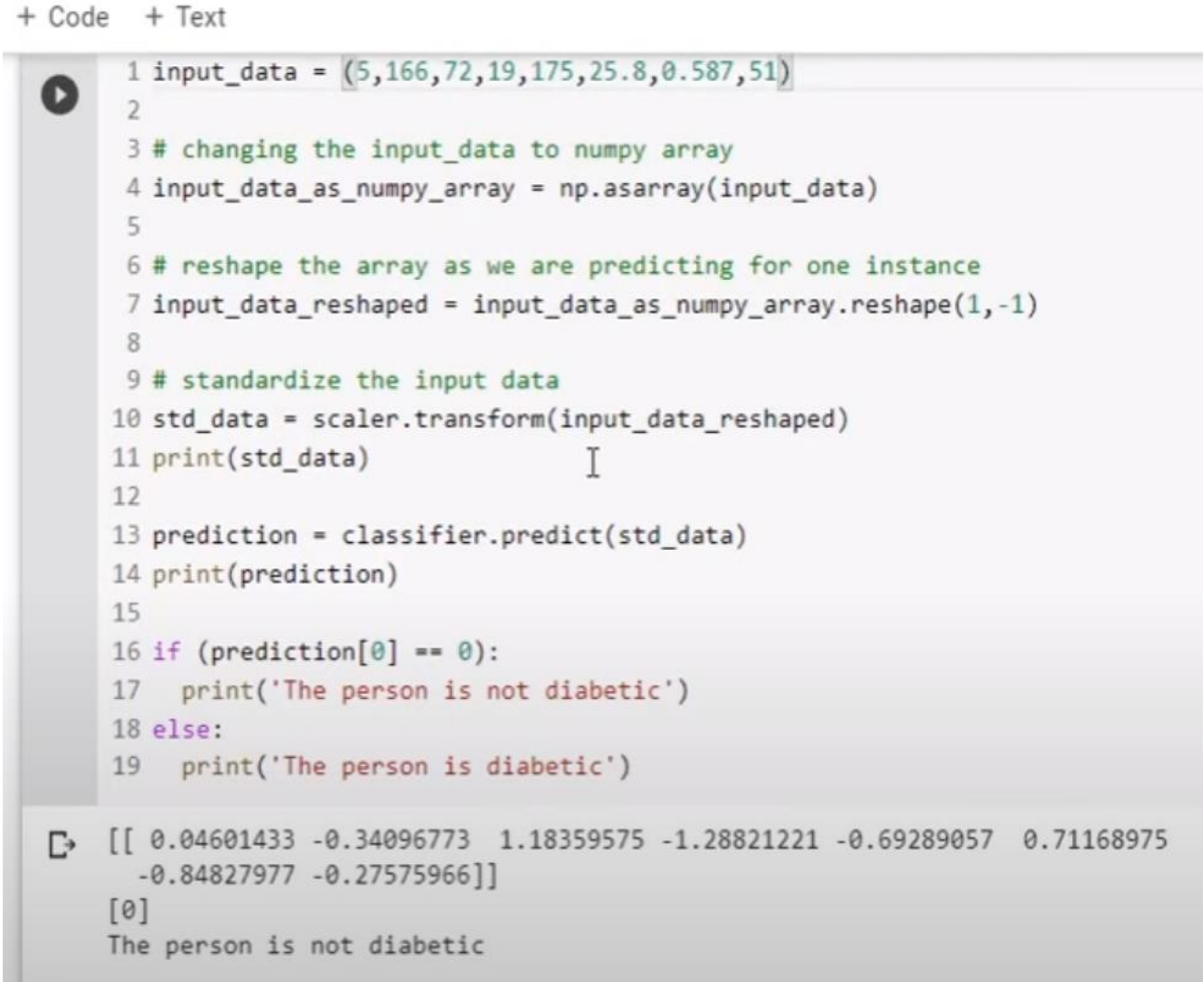


```
if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
[[ 0.3429808   1.41167241  0.14964075 -0.09637905  0.82661621 -0.78595734
   0.34768723  1.51108316]]
[1]
The person is diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

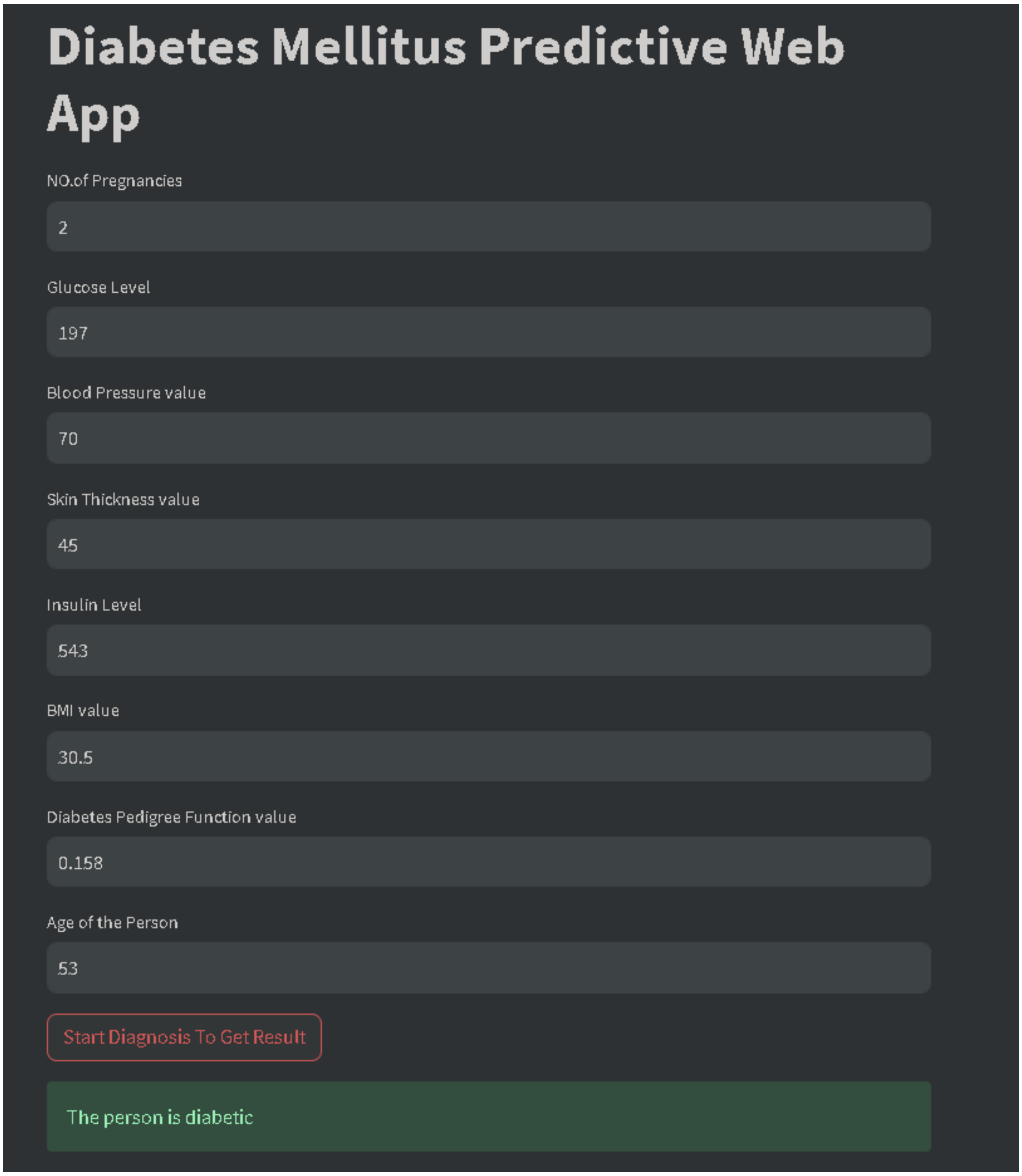
Used **if else** to see diabetic or not

Result for different input which is diabetic.



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

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

```
[[ 0.04601433 -0.34096773  1.18359575 -1.28821221 -0.69289057  0.71168975
  -0.84827977 -0.27575966]]
[0]
The person is not diabetic
```



Diabetes Mellitus Predictive Web App

NO.of Pregnancies

2

Glucose Level

197

Blood Pressure value

70

Skin Thickness value

45

Insulin Level

543

BMI value

30.5

Diabetes Pedigree Function value

0.158

Age of the Person

53

The person is diabetic

Deployment

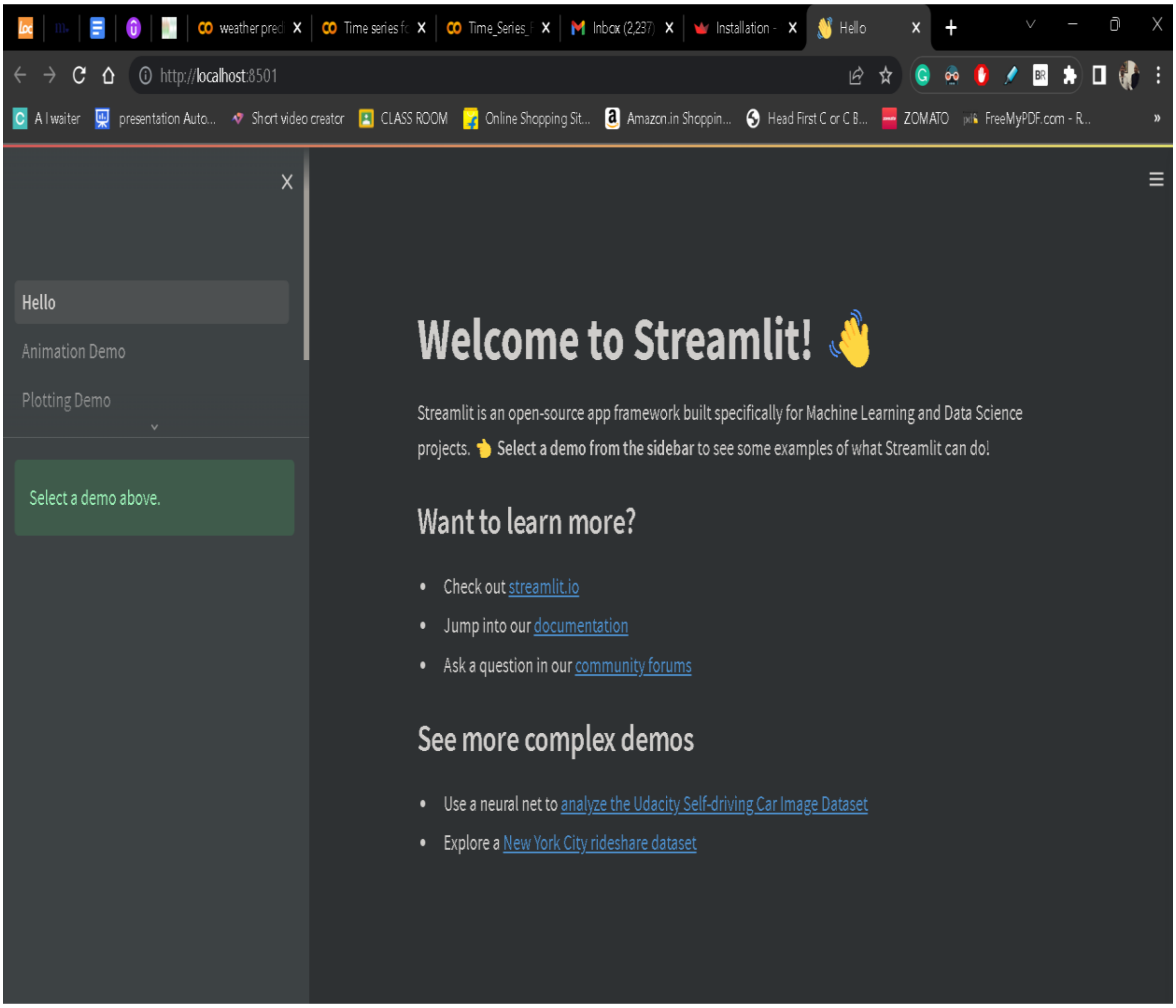
This will be first project of ML which will be who **hosted as web app** using Streamlit on HEROKU.

Streamlit →is the library to build machine learning model.

Open source framework which used to create web apps for machine learning and data science stuff.

Used to deploy machine learning models like Django, Flask used to deploy also but they required idea of HTML and CSS.

Streamlit we can create a simple to remarkable web app with few lines of code.



Select a demo above.

Welcome to Streamlit! 👋

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects. 👈 **Select a demo from the sidebar** to see some examples of what Streamlit can do!

Want to learn more?

- Check out streamlit.io
- Jump into our documentation
- Ask a question in our community forums

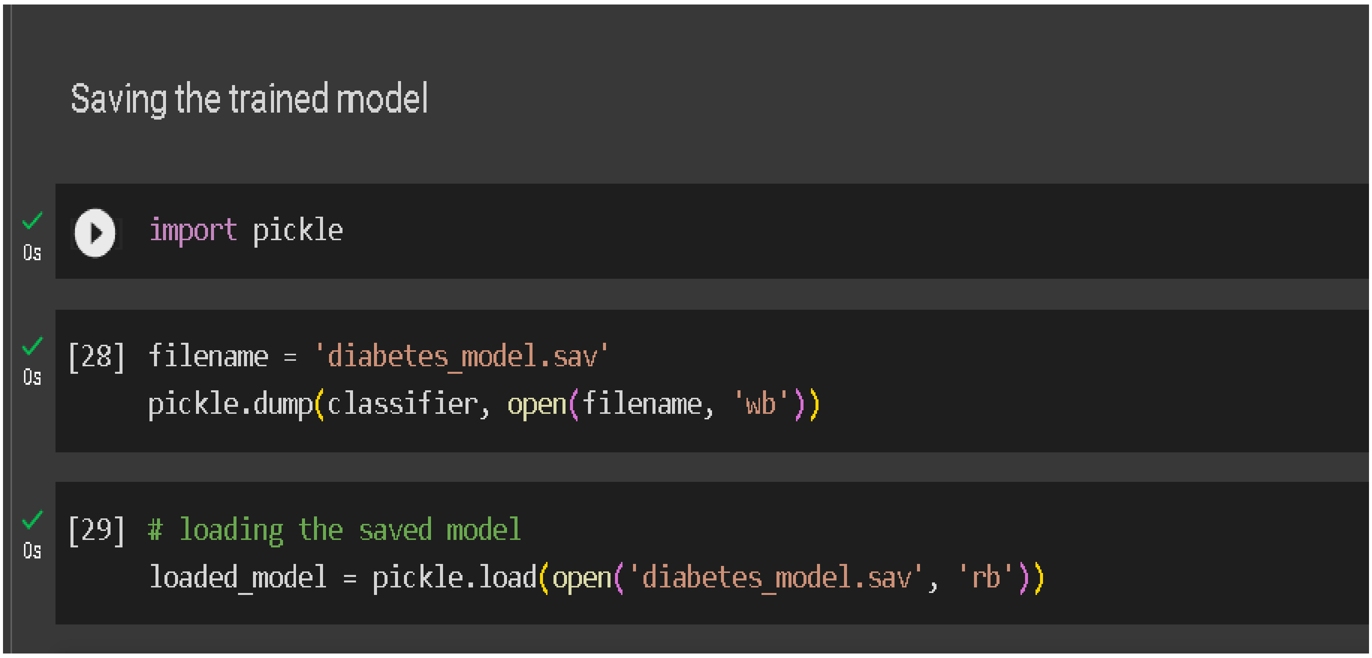
See more complex demos

- Use a neural net to analyze the Udacity Self-driving Car Image Dataset
- Explore a New York City rideshare dataset

Saving the trend model

by importing pickle library to save the model, Model name as classifier.

wr write binary



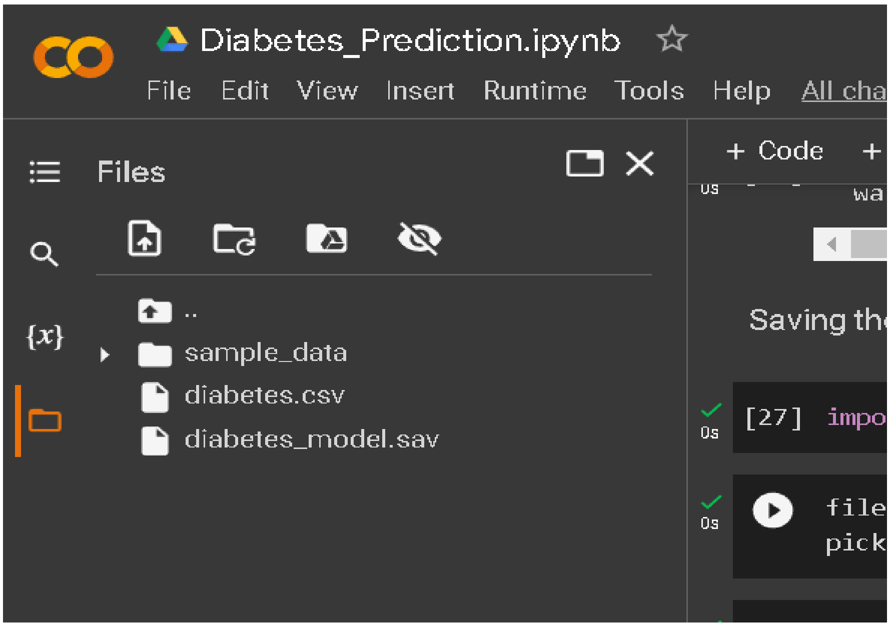
Saving the trained model

```
import pickle
```

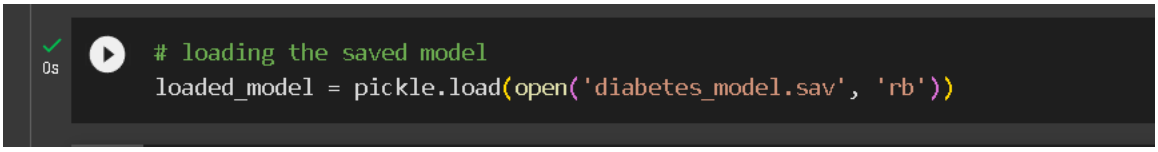
```
[28] filename = 'diabetes_model.sav'
     pickle.dump(classifier, open(filename, 'wb'))
```

```
[29] # loading the saved model
     loaded_model = pickle.load(open('diabetes_model.sav', 'rb'))
```

file created



load model and use it

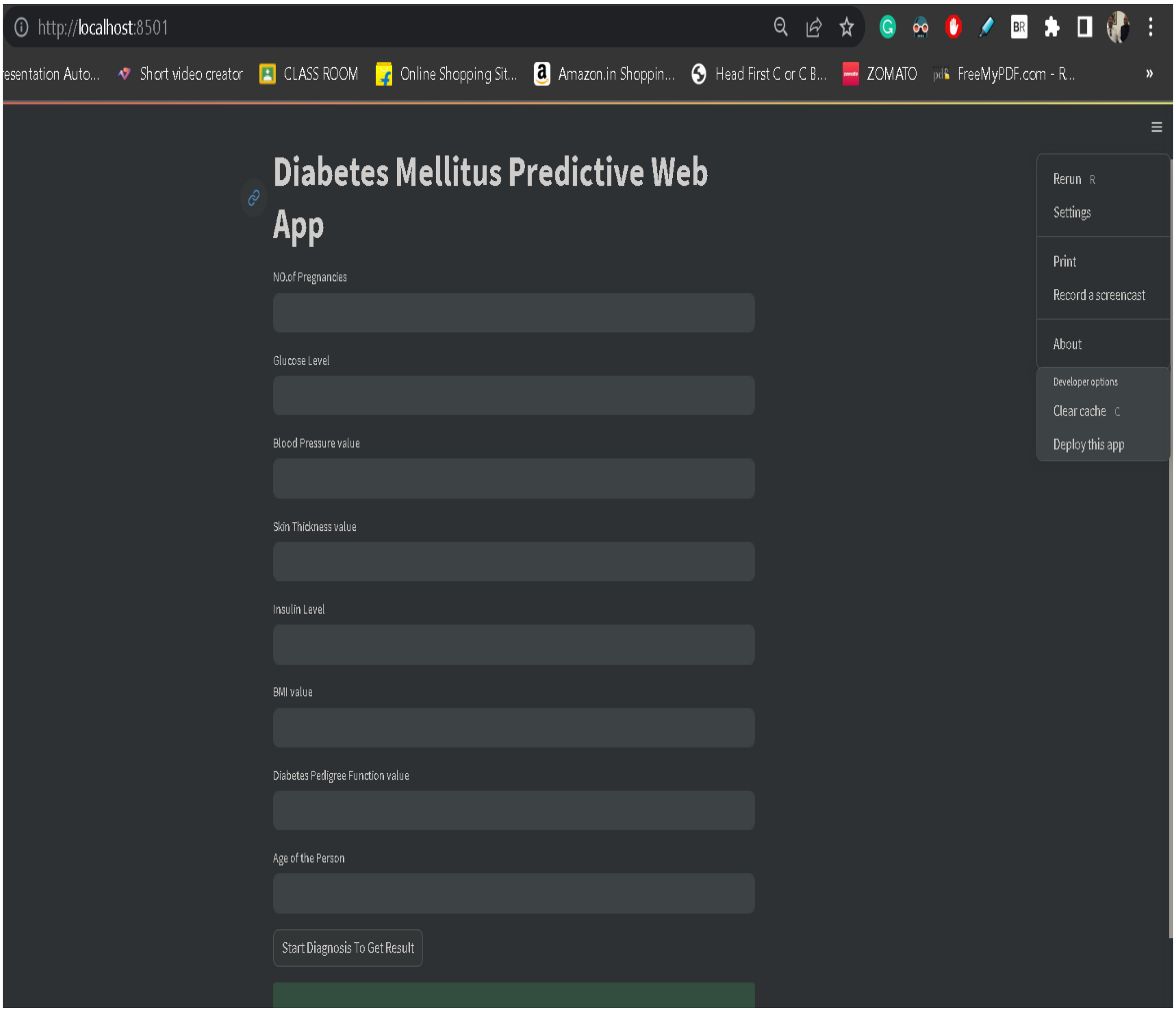


```
# loading the saved model
loaded_model = pickle.load(open('diabetes_model.sav', 'rb'))
```

dumb function used to save a trained model.

load function used to load the saved model.

RB is reading in binary format.



Diabetes Mellitus Predictive Web App

NO.of Pregnancies

Glucose Level

Blood Pressure value

Skin Thickness value

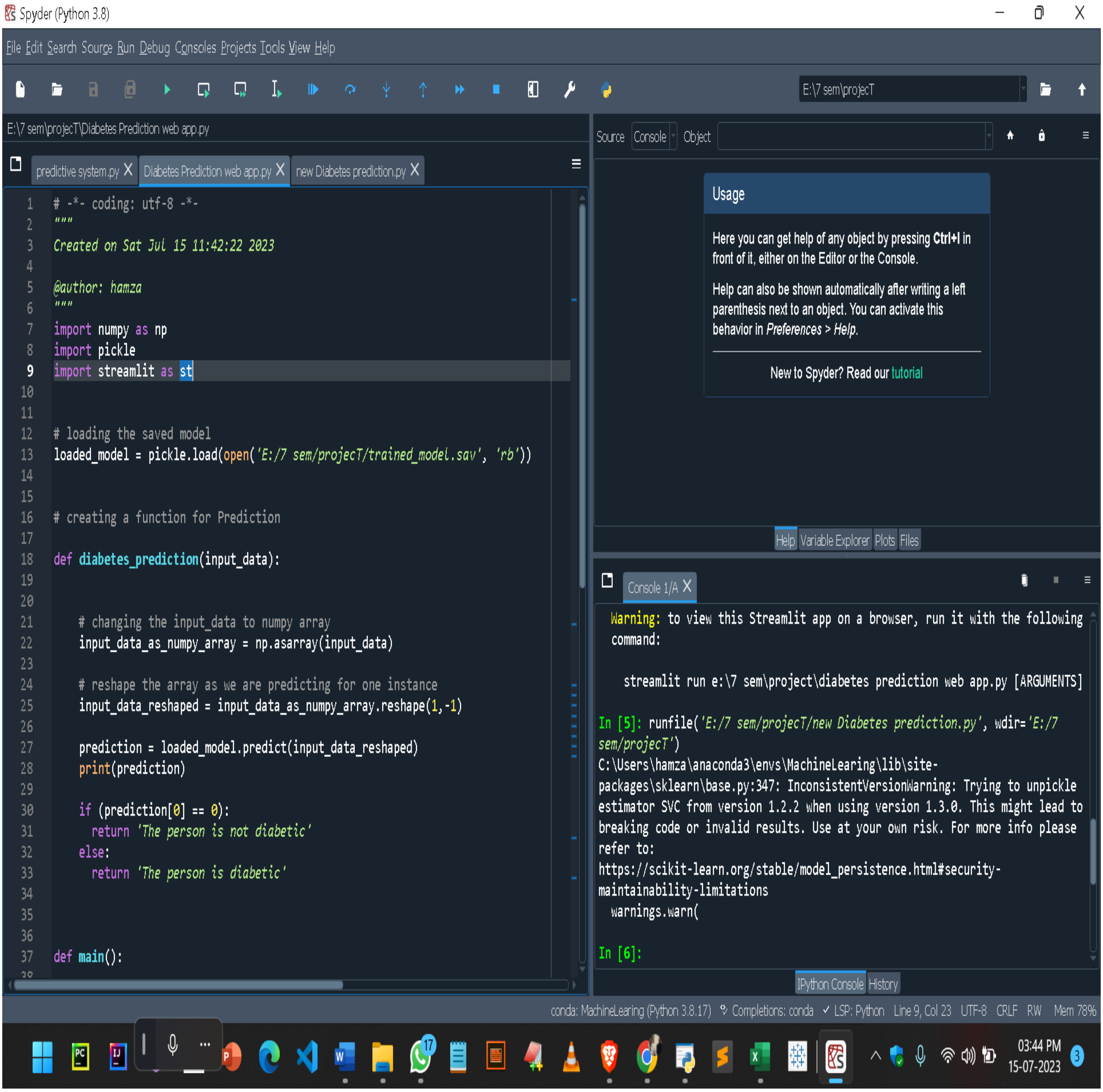
Insulin Level

BMI value

Diabetes Pedigree Function value

Age of the Person

Start Diagnosis To Get Result


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\7 sem\projecT
E:\7 sem\projecT\Diabetes Prediction web app.py
predictive system.py
Diabetes Prediction web app.py
new Diabetes prediction.py
Source Console Object
Help Variable Explorer Plots Files
Console 1/A
IPython Console History
conda: MachineLearing (Python 3.8.17) Completions: conda LSP: Python Line 9, Col 23 UTF-8 CRLF RW Mem 78%
03:44 PM
15-07-2023

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 15 11:42:22 2023

@author: hamza
"""
import numpy as np
import pickle
import streamlit as st


# loading the saved model
loaded_model = pickle.load(open('E:/7 sem/projecT/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():
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
 Warning: to view this Streamlit app on a browser, run it with the following
 command:

   streamlit run e:\7 sem\project\diabetes prediction web app.py [ARGUMENTS]

In [5]: runfile('E:/7 sem/projecT/new Diabetes prediction.py', wdir='E:/7
sem/projecT')
C:\Users\hamza\anaconda3\envs\MachineLearing\lib\site-
packages\sklearn\base.py:347: InconsistentVersionWarning: Trying to unpickle
estimator SVC from version 1.2.2 when using version 1.3.0. This might lead to
breaking code or invalid results. Use at your own risk. For more info please
refer to:
https://scikit-learn.org/stable/model_persistence.html#security-
maintainability-limitations
  warnings.warn(

In [6]:
```

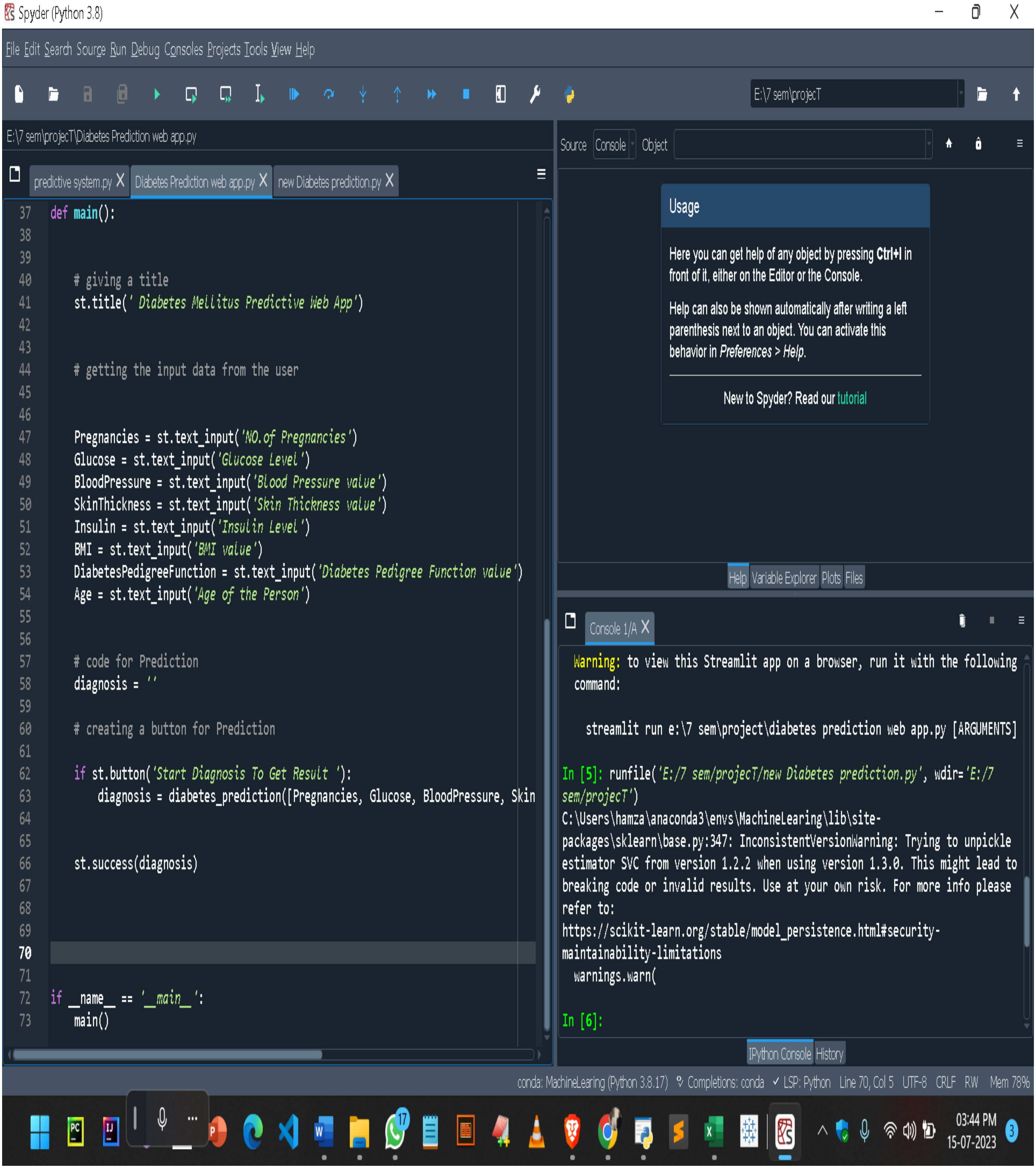
In Anaconda navigator created environment then used spider IDE to make simple user interface and host as a web app.Using pip command downloaded required packages. Usedtitle function and St.text input() get data from user interface.

Diagnosis variable is empty to store the result after all the processing

created button using function which lies under library streamlit.

gets input from the user through UI then user will pass values then data process start conversion list to array then reshape then compare with data set and labels then result.

Success() gives the final output password variable name.


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\7 sem\projecT
E:\7 sem\projecT\Diabetes Prediction web app.py
predictive system.py
Diabetes Prediction web app.py
new Diabetes prediction.py
Source Console Object
Help Variable Explorer Plots Files
Console 1/A
IPython Console History

```python
def main():


    # giving a title
    st.title(' Diabetes Mellitus Predictive Web App')


    # getting the input data from the user


    Pregnancies = st.text_input('NO.of Pregnancies')
    Glucose = st.text_input('Glucose Level')
    BloodPressure = st.text_input('Blood Pressure value')
    SkinThickness = st.text_input('Skin Thickness value')
    Insulin = st.text_input('Insulin Level')
    BMI = st.text_input('BMI value')
    DiabetesPedigreeFunction = st.text_input('Diabetes Pedigree Function value')
    Age = st.text_input('Age of the Person')


    # code for Prediction
    diagnosis = ''

    # creating a button for Prediction

    if st.button('Start Diagnosis To Get Result '):
        diagnosis = diabetes_prediction([Pregnancies, Glucose, BloodPressure, Skin


    st.success(diagnosis)





if __name__ == '__main__':
    main()
```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

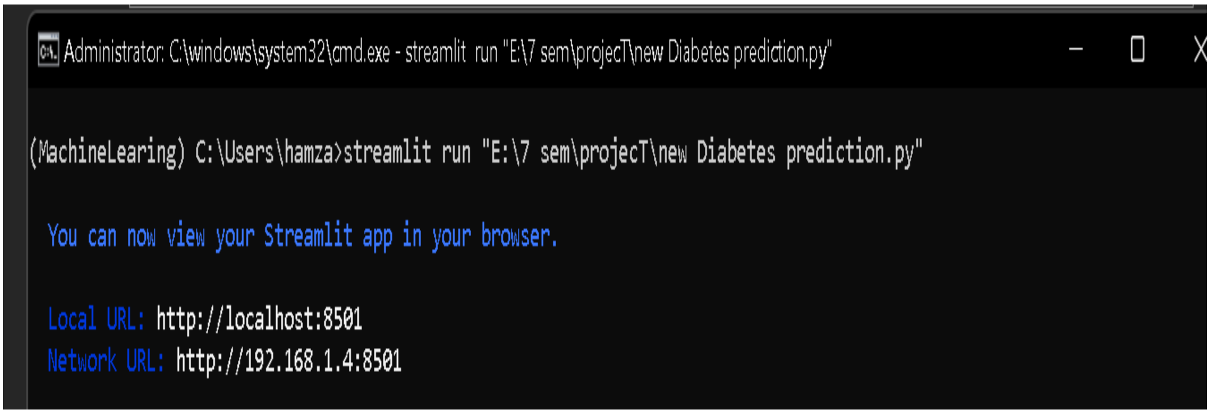
New to Spyder? Read our tutorial

```
Warning: to view this Streamlit app on a browser, run it with the following
command:

    streamlit run e:\7 sem\project\diabetes prediction web app.py [ARGUMENTS]

In [5]: runfile('E:/7 sem/projecT/new Diabetes prediction.py', wdir='E:/7
sem/projecT')
C:\Users\hamza\anaconda3\envs\MachineLearing\lib\site-
packages\sklearn\base.py:347: InconsistentVersionWarning: Trying to unpickle
estimator SVC from version 1.2.2 when using version 1.3.0. This might lead to
breaking code or invalid results. Use at your own risk. For more info please
refer to:
https://scikit-learn.org/stable/model_persistence.html#security-
maintainability-limitations
  warnings.warn(

In [6]:
```

conda: MachineLearing (Python 3.8.17) Completions: conda LSP: Python Line 70, Col 5 UTF-8 CRLF RW Mem 78%

03:44 PM
15-07-2023

To run the system open Anaconda navigator as administrator mode open your environment open terminal type strealit run “ path where file present\ file name.py”

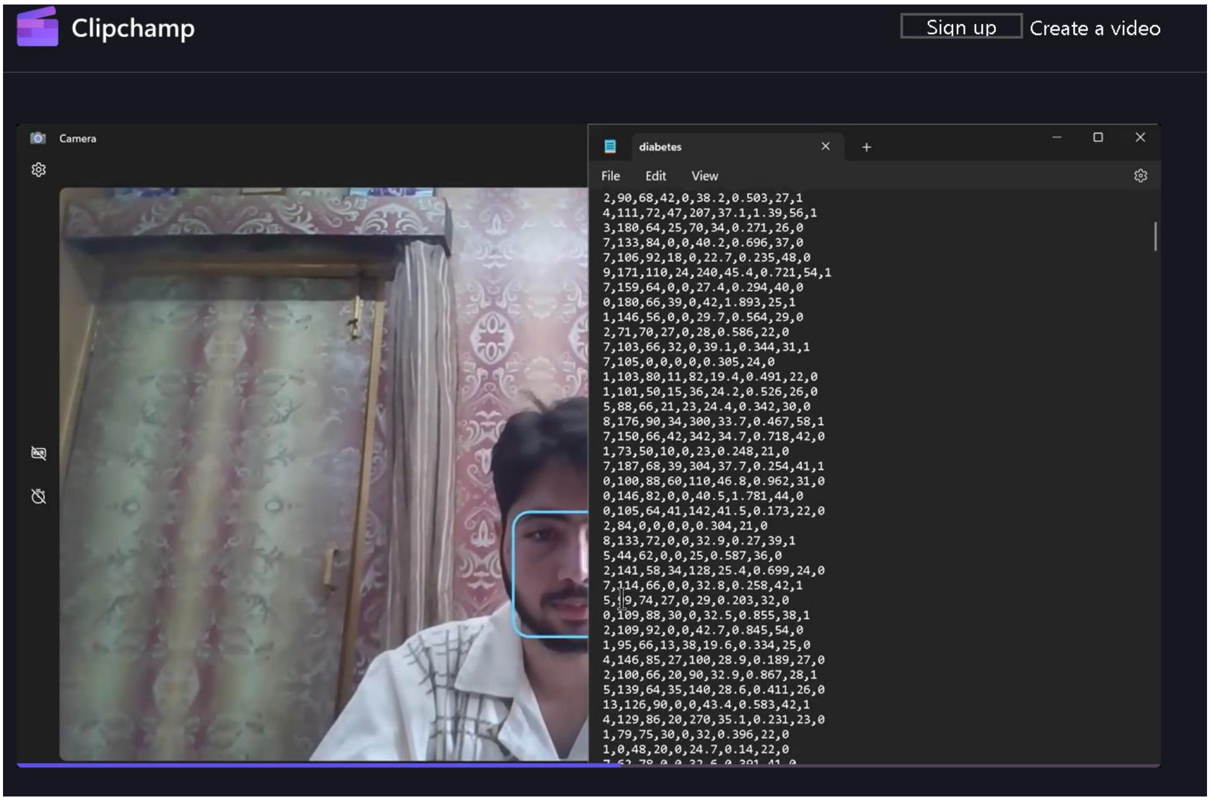


Administrator: C:\windows\system32\cmd.exe - streamlit run "E:\7 sem\projecT\new Diabetes prediction.py"

```
(MachineLearing) C:\Users\hamza>streamlit run "E:\7 sem\projecT\new Diabetes prediction.py"

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.4:8501
```

Live working clip to

watch please click on the image ctrl+ right click



```
2,90,68,42,0,38.2,0.503,27,1
4,111,72,47,207,37.1,1.39,56,1
3,180,64,25,70,34,0.271,26,0
7,133,84,0,0,40.2,0.696,37,0
7,106,92,18,0,22.7,0.235,48,0
9,171,110,24,240,45.4,0.721,54,1
7,159,64,0,0,27.4,0.294,40,0
0,180,66,39,0,42,1.893,25,1
1,146,56,0,0,29.7,0.564,29,0
2,71,70,27,0,28,0.586,22,0
7,103,66,32,0,39.1,0.344,31,1
7,105,0,0,0,0,0.305,24,0
1,103,80,11,82,19.4,0.491,22,0
1,101,50,15,36,24.2,0.526,26,0
5,88,66,21,23,24.4,0.342,30,0
8,176,90,34,300,33.7,0.467,58,1
7,150,66,42,342,34.7,0.718,42,0
1,73,50,10,0,23,0.248,21,0
7,187,68,39,304,37.7,0.254,41,1
0,100,88,60,110,46.8,0.962,31,0
0,146,82,0,0,40.5,1.781,44,0
0,105,64,41,142,41.5,0.173,22,0
2,84,0,0,0,0,0.304,21,0
8,133,72,0,0,32.9,0.27,39,1
5,44,62,0,0,25,0.587,36,0
2,141,58,34,128,25.4,0.699,24,0
7,114,66,0,0,32.8,0.258,42,1
5,99,74,27,0,29,0.203,32,0
0,109,88,30,0,32.5,0.855,38,1
2,109,92,0,0,42.7,0.845,54,0
1,95,66,13,38,19.6,0.334,25,0
4,146,85,27,100,28.9,0.189,27,0
2,100,66,20,90,32.9,0.867,28,1
5,139,64,35,140,28.6,0.411,26,0
13,126,90,0,0,43.4,0.583,42,1
4,129,86,20,270,35.1,0.231,23,0
1,79,75,30,0,32,0.396,22,0
1,0,48,20,0,24.7,0.14,22,0
7,62,78,0,0,32.6,0.391,41,0
```

CONCLUSION :**I created a system which can take input like insulin, blood pressure level and other to predict the person is diabetic or not** with accuracy of **79%** conversely (but) what will be trying to make it’s similar to the machines which are available in market but, I tried this programmatically that’s why it is different from market.

Further I can add other disease predicting model also in it. Like heart attack Parkinson disease and other.

Limitation: The person judge books by its cover so in future we will be working on remarkable UI as it was my first UI project so it is simple.

By applying optimization techniques we can get above 80 accuracy also.

BIBLIOGRAPHY

- **Software**

✦ Jupyter Notebook ,VS code, spider, Anaconda navigator, qt console

- **Sites**

  - https://stackoverflow.com/
  - www.google.com
  - www.youtube.com
  - www.kaggle.com

- video links for life illustration.
  https://clipchamp.com/watch/lrzH0mpb6lB
- link for download in your system

https://drive.google.com/uc?id=1SzE3jQDbPxbxK-iy7FcDUEj0rj1oIC27&export=download

thank you