Diabetes Prediciton

Project Report

MILIND DALAKOTI

milinddalakoti@gmail.com

9953653656

2021

Table of Contents

CONTENTS

ABSTRACT

Diabetes has become a common disease to mankind from young to old people nowadays. There are various reasons due to which the population of diabetic patients is increasing day by day are obesity, bad diet, autoimmune reaction, change in lifestyle, eating habits, environmental pollution, etc.

Diabetes is a metabolic disorder that is identified by the high blood sugar level. Increased blood glucose level damages the vital organs as well as other organs of the human body causing other potential health ailments.

Data analytics is one of the branches of computer science, which is a process of examining large datasets and find some useful hidden patterns and draw conclusions based upon those patterns. This analytical process is carried out using machine learning algorithms in the health care system. To carry out medical diagnoses, machine learning algorithms are used for analyzing large medical data to build machine learning models.

This project presents a diabetes prediction system for the diagnosis of diabetes. Early detection of diabetes is possible with the help of this model.

INTRODUCTION

Diabetes is brought about by the expansion level of the sugar (glucose) in the blood. Diabetes can be of two sorts, for example, type 1 diabetes and type 2 diabetes.

Type 1 diabetes can be found in kids and grown-ups at times. The grown-ups who are corpulent are predominantly influenced by this sort. For the most part, moderately aged individuals are influenced by type 2 diabetes.

Diabetes is a major reason for different ailments, for example, coronary illness, stroke, kidney sickness, eye issues, dental infection, nerve harm, foot issues. Side effects that can cause diabetes are over-the-top discharge of pee (polyuria), thirst, consistent appetite, weight reduction, vision changes, and weakness, which can happen suddenly.

The serious issue which is executing a great many individuals all through the world is diabetes.

The World Health Organization (WHO) survey has found that 1.2 million people died due to this chronic disease.

INTRODUCTION

Different AI and deep learning calculations are utilized for some sort of forecast offices. Often these calculations are utilized by business giants for benefit in deals. Given the subject how might we utilize innovations for human advancement?

To learn the different complexities of different highlights of the biomechanics of the human body and foresee the entangled issues of individuals. The machine must be prepared with the attitude of a specialist with different highlights and outer components gave from a valid dataset.

EXISTING METHOD

EXISTING METHOD TO DIAGNOSE DIABETES

If your doctor suspects that you have diabetes, they will order additional testing. There are different types of tests to diagnose types 1 and 2 diabetes as well as gestational diabetes.The following tests are used to diagnose diabetes:

01

A1c blood test

Also known as a glycated hemoglobin test. This test shows your average blood sugar level for the past 2 to 3 months by showing how much blood sugar is attached to your hemoglobin.

02

Random blood sugar test

Examines your blood sugar at an unspecified time. A level of 200 milligrams per deciliter indicates diabetes.

03

Fasting blood sugar test

conducted after an overnight fast. If your blood sugar level is 126 milligrams per deciliter, it’s considered diabetic.

04

Oral glucose tolerance test

Requires overnight fasting and then drinking a sugary liquid the next morning. After this, your blood sugar levels will be tested over the next 2 hours. A reading of more than 200 milligrams per deciliter is considered diabetic.

05

Initial glucose challenge test

This usually occurs at 24–28 weeks of gestation. If your blood sugar level readings are high for 2 of 3 readings, you’ll be diagnosed with gestational diabetes.

PROPOSED METHOD & ARCHITECTURE

PROPOSED METHOD

This project uses the data provided by the user and runs classification machine learning models to detect the various correlations between the data and the probability of diabetes.
Then using the correlations developed tries to predict if a person is diabetic or not.
Moreover, this model can improve as the dataset for the model grows, this opens the gate for further improvements and developments to our model.

Name	Advantage	Disadvantage
Support Vector Machine Classifier	• Effective in high dimensional spaces. • Still effective in cases where number of dimensions is greater than the number of samples. • Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. • Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.	• Avoid over-fitting in choosing Kernel functions if the number of features is much greater than the number of samples • SVMs do not directly provide probability estimates,
Random Forest Classifier	• It can be used to solve both classifications as well as regression problems. • It works well with both categorical and continuous variables. • It can automatically handle missing values. • It is usually robust to outliers and can handle them automatically.	• Complexity • Longer Training Period

PROPOSED METHOD

Name	Advantage	Disadvantage
Decision Tree Classifier	• Compared to other algorithms decision trees requires less effort for data preparation during pre-processing. • A decision tree does not require the normalization of data. • A decision tree does not require the scaling of data as well. • Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent. • A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.	• Decision tree often involves higher time to train the model. • Decision tree training is relatively expensive as the complexity and time has taken are more.

ARCHITECTURE

Fetching diagnosis data for diabetes patients and reports

Data Pre-processing

Splitting the data into Train and Test

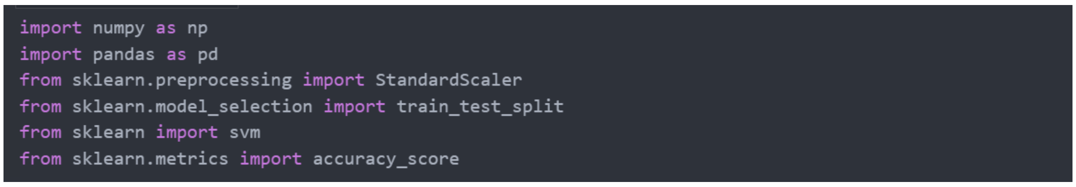
Training the Model

Calculating the Accuracy

Prediction

METHODOLOGY & IMPLEMENTATION

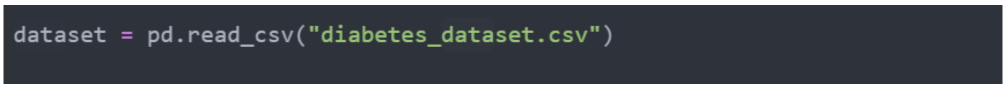
IMPORTING THE DEPENDENCIES



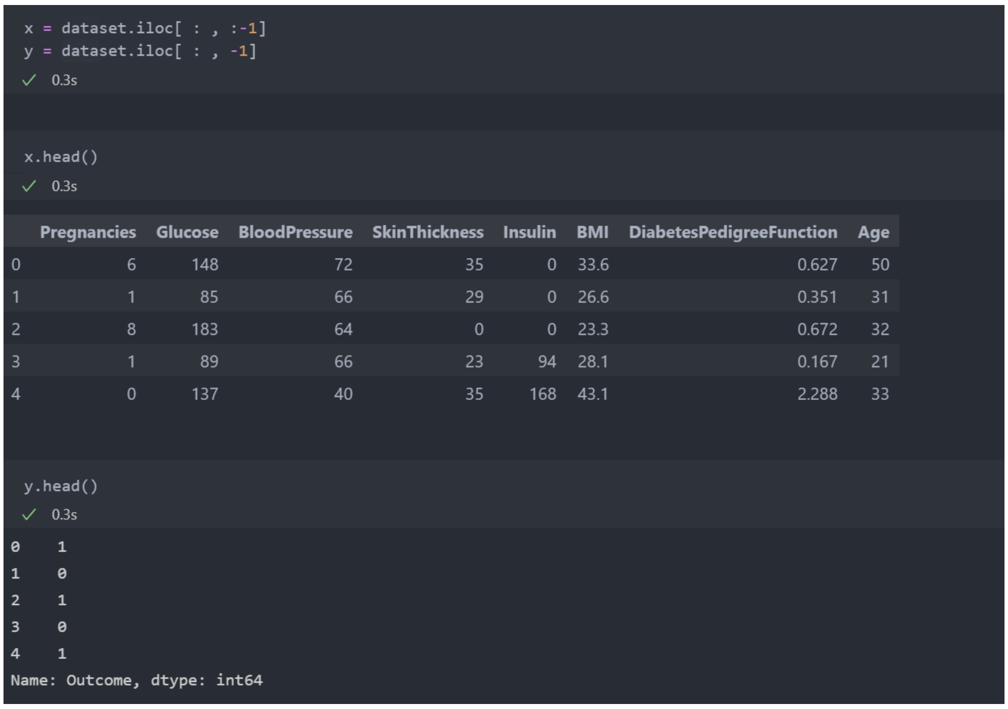
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

DATA COLLECTION

--> dataset link : https://www.kaggle.com/uciml/pima-indians-diabetes-database
:: The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.



```
dataset = pd.read_csv("diabetes_dataset.csv")
```



```
x = dataset.iloc[ : , :-1]
y = dataset.iloc[ : , -1]
✓ 0.3s
```

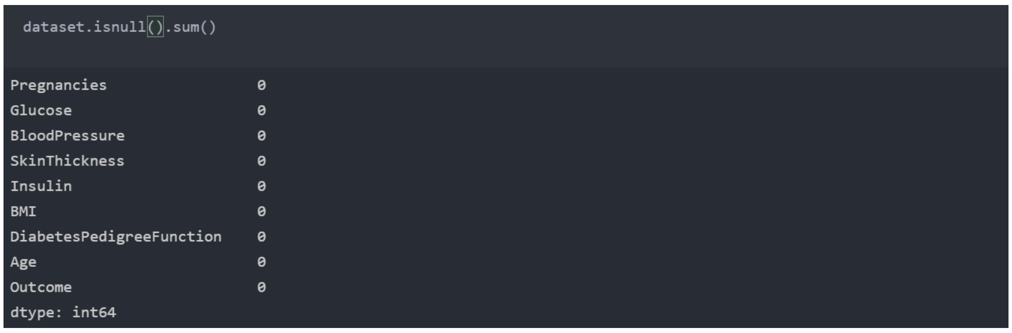
```
x.head()
✓ 0.3s
```

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

```
y.head()
✓ 0.3s
```

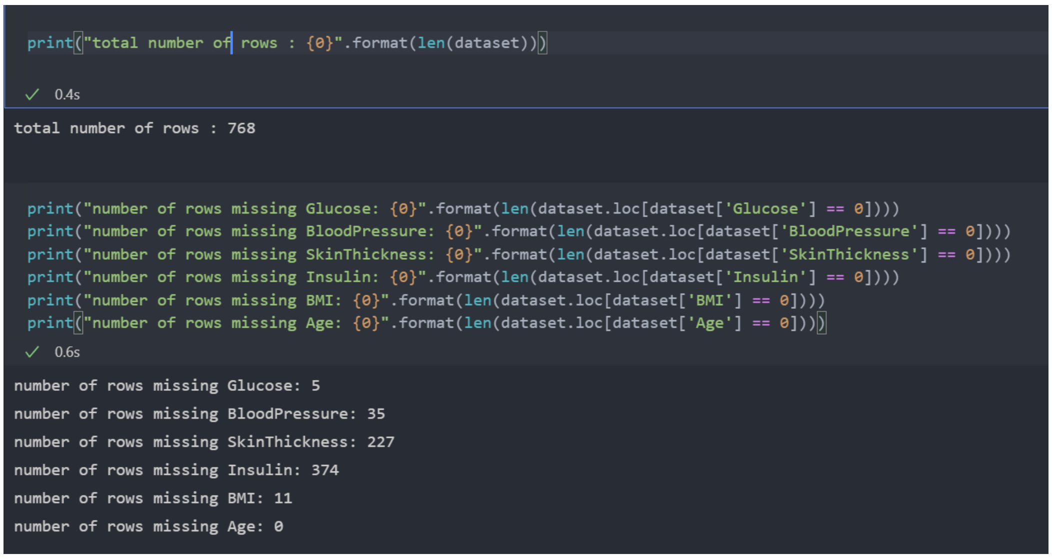
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

DATA PREPROCESSING



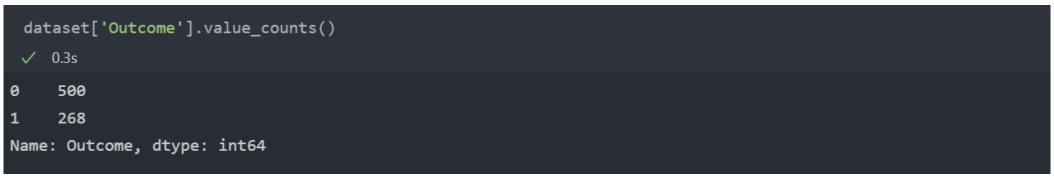
```
dataset.isnull().sum()
```

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

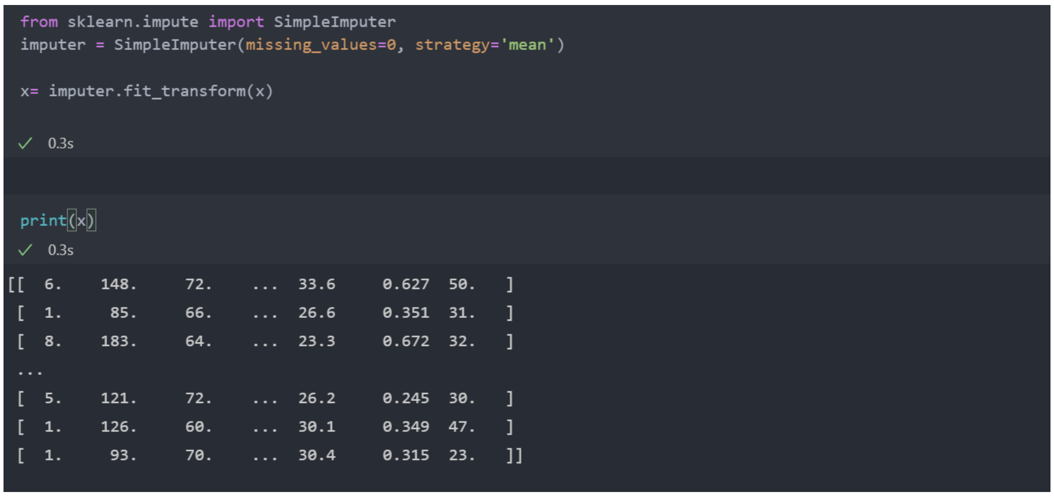


```
print("total number of rows : {0}".format(len(dataset)))
✓ 0.4s
total number of rows : 768
```

```
print("number of rows missing Glucose: {0}".format(len(dataset.loc[dataset['Glucose'] == 0])))
print("number of rows missing BloodPressure: {0}".format(len(dataset.loc[dataset['BloodPressure'] == 0])))
print("number of rows missing SkinThickness: {0}".format(len(dataset.loc[dataset['SkinThickness'] == 0])))
print("number of rows missing Insulin: {0}".format(len(dataset.loc[dataset['Insulin'] == 0])))
print("number of rows missing BMI: {0}".format(len(dataset.loc[dataset['BMI'] == 0])))
print("number of rows missing Age: {0}".format(len(dataset.loc[dataset['Age'] == 0])))
✓ 0.6s
number of rows missing Glucose: 5
number of rows missing BloodPressure: 35
number of rows missing SkinThickness: 227
number of rows missing Insulin: 374
number of rows missing BMI: 11
number of rows missing Age: 0
```

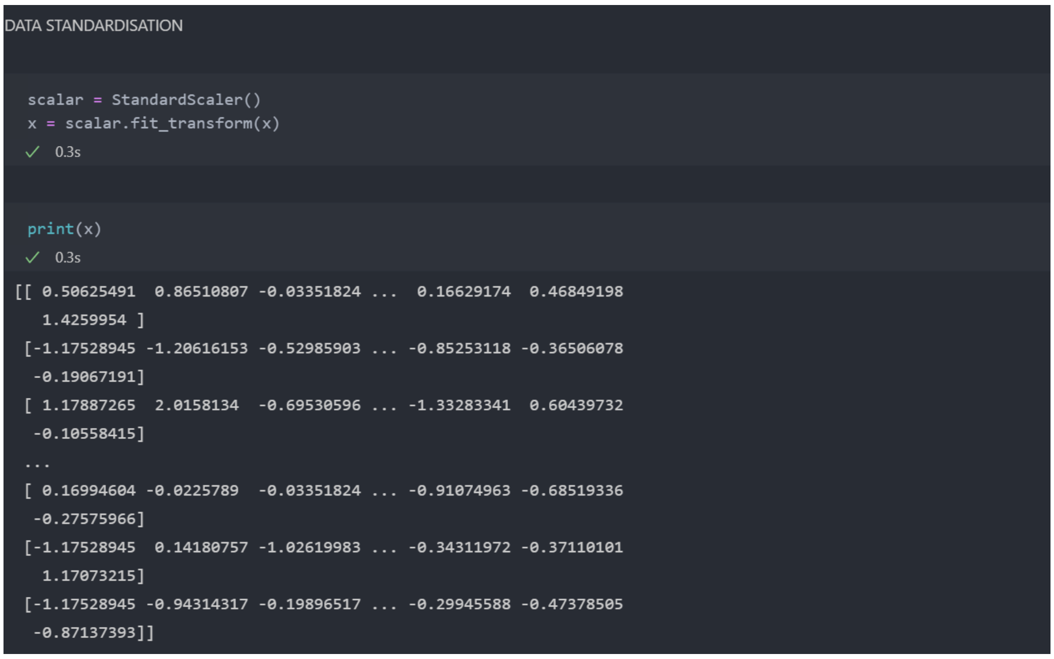


```
dataset['Outcome'].value_counts()
✓ 0.3s
0    500
1    268
Name: Outcome, dtype: int64
```



```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=0, strategy='mean')

x= imputer.fit_transform(x)

✓ 0.3s
```

```
print(x)
✓ 0.3s
[[  6.    148.     72.    ...  33.6     0.627  50.   ]
 [  1.     85.     66.    ...  26.6     0.351  31.   ]
 [  8.    183.     64.    ...  23.3     0.672  32.   ]
 ...
 [  5.    121.     72.    ...  26.2     0.245  30.   ]
 [  1.    126.     60.    ...  30.1     0.349  47.   ]
 [  1.     93.     70.    ...  30.4     0.315  23.   ]]
```

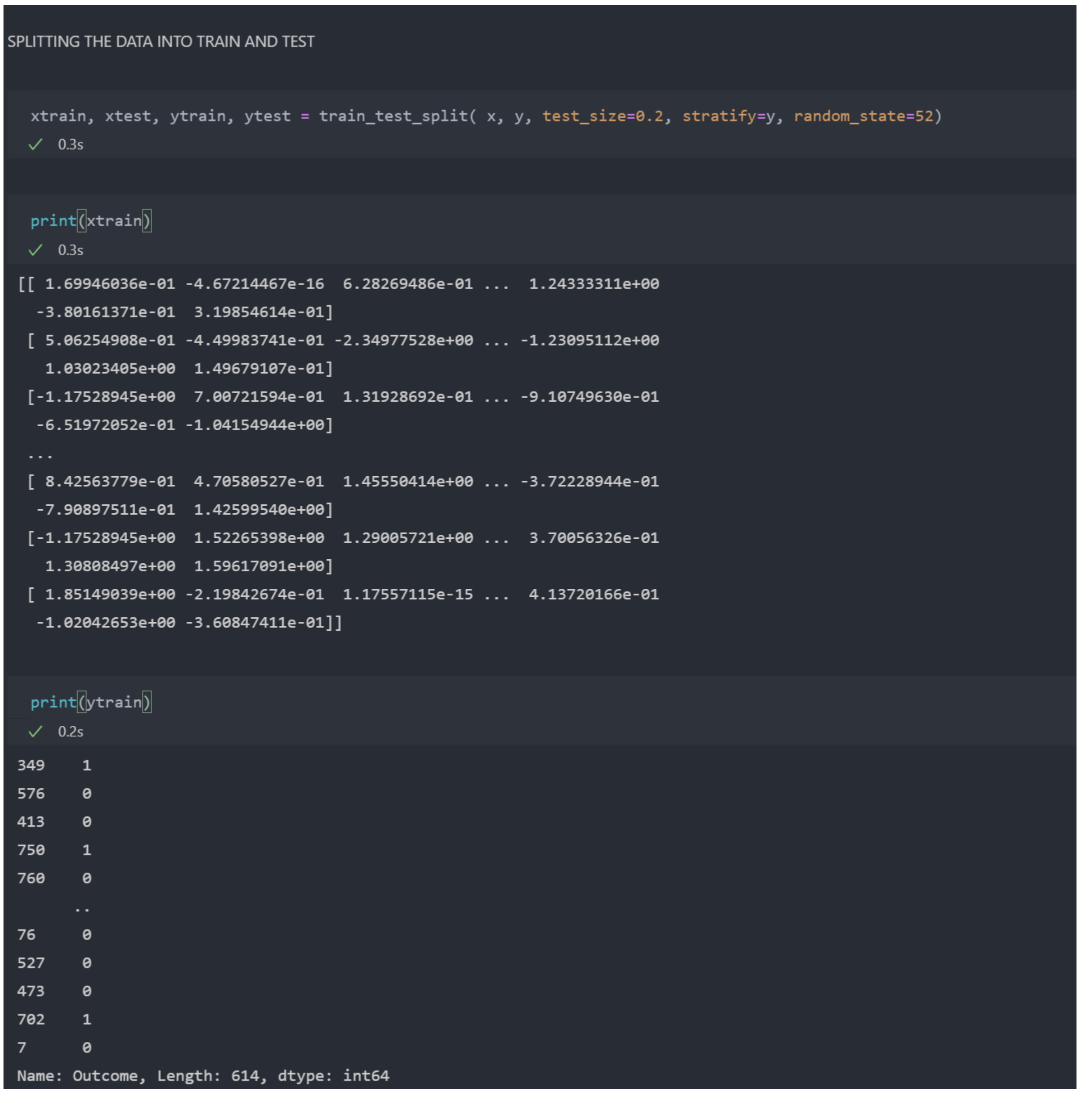


DATA STANDARDISATION

```
scalar = StandardScaler()
x = scalar.fit_transform(x)
✓ 0.3s
```

```
print(x)
✓ 0.3s
[[ 0.50625491  0.86510807 -0.03351824 ...  0.16629174  0.46849198
   1.4259954 ]
 [-1.17528945 -1.20616153 -0.52985903 ... -0.85253118 -0.36506078
  -0.19067191]
 [ 1.17887265  2.0158134  -0.69530596 ... -1.33283341  0.60439732
  -0.10558415]
 ...
 [ 0.16994604 -0.0225789  -0.03351824 ... -0.91074963 -0.68519336
  -0.27575966]
 [-1.17528945  0.14180757 -1.02619983 ... -0.34311972 -0.37110101
   1.17073215]
 [-1.17528945 -0.94314317 -0.19896517 ... -0.29945588 -0.47378505
  -0.87137393]]
```

SPLITTING THE DATASET INTO TRAIN AND TEST

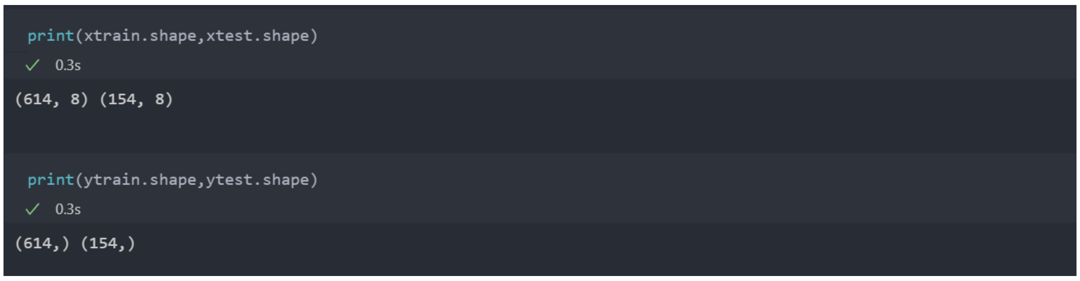


SPLITTING THE DATA INTO TRAIN AND TEST

```
xtrain, xtest, ytrain, ytest = train_test_split( x, y, test_size=0.2, stratify=y, random_state=52)
✓ 0.3s
```

```
print(xtrain)
✓ 0.3s
[[ 1.69946036e-01 -4.67214467e-16  6.28269486e-01 ...  1.24333311e+00
  -3.80161371e-01  3.19854614e-01]
 [ 5.06254908e-01 -4.49983741e-01 -2.34977528e+00 ... -1.23095112e+00
   1.03023405e+00  1.49679107e-01]
 [-1.17528945e+00  7.00721594e-01  1.31928692e-01 ... -9.10749630e-01
  -6.51972052e-01 -1.04154944e+00]
 ...
 [ 8.42563779e-01  4.70580527e-01  1.45550414e+00 ... -3.72228944e-01
  -7.90897511e-01  1.42599540e+00]
 [-1.17528945e+00  1.52265398e+00  1.29005721e+00 ...  3.70056326e-01
   1.30808497e+00  1.59617091e+00]
 [ 1.85149039e+00 -2.19842674e-01  1.17557115e-15 ...  4.13720166e-01
  -1.02042653e+00 -3.60847411e-01]]
```

```
print(ytrain)
✓ 0.2s
349    1
576    0
413    0
750    1
760    0
      ..
76     0
527    0
473    0
702    1
7      0
Name: Outcome, Length: 614, dtype: int64
```

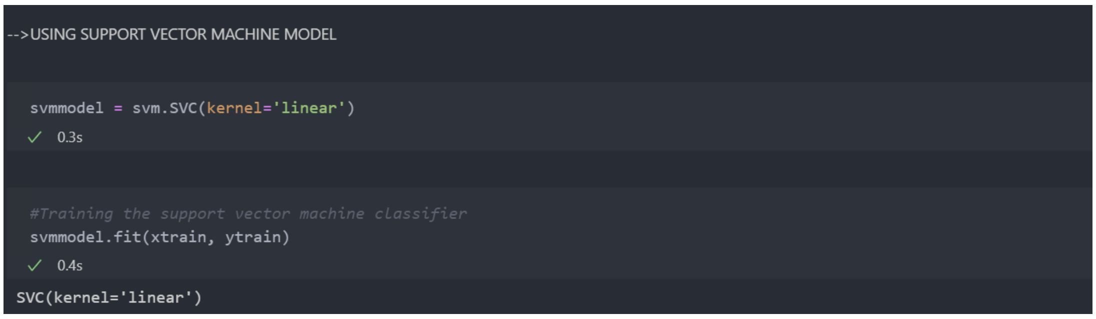


```
print(xtrain.shape,xtest.shape)
✓ 0.3s
(614, 8) (154, 8)
```

```
print(ytrain.shape,ytest.shape)
✓ 0.3s
(614,) (154,)
```

TRAINING THE MODEL

--: SUPPORT VECTOR MACHINE MODEL

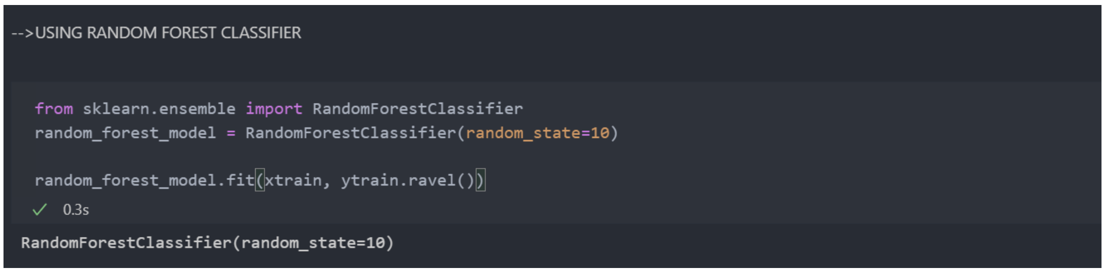


-->USING SUPPORT VECTOR MACHINE MODEL

```
svmmodel = svm.SVC(kernel='linear')
✓ 0.3s
```

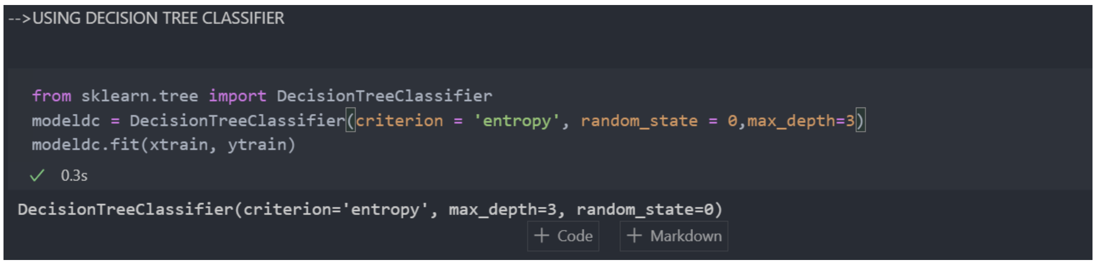
```
#Training the support vector machine classifier
svmmodel.fit(xtrain, ytrain)
✓ 0.4s
SVC(kernel='linear')
```

--: RANDOM FOREST CLASSIFIER


-->USING RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
random_forest_model = RandomForestClassifier(random_state=10)

random_forest_model.fit(xtrain, ytrain.ravel())
```
✓ 0.3s
```
RandomForestClassifier(random_state=10)
```

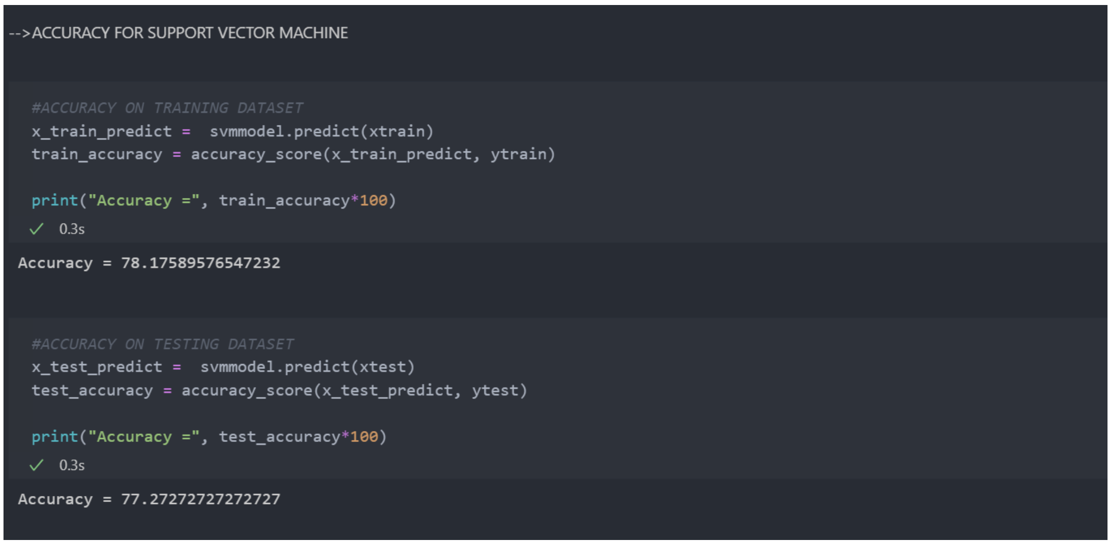
--: DECISION TREE CLASSIFIER


-->USING DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier
modeldc = DecisionTreeClassifier(criterion = 'entropy', random_state = 0,max_depth=3)
modeldc.fit(xtrain, ytrain)
```
✓ 0.3s
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

FINDING THE ACCURACY SCORE

--: SUPPORT VECTOR MACHINE MODEL

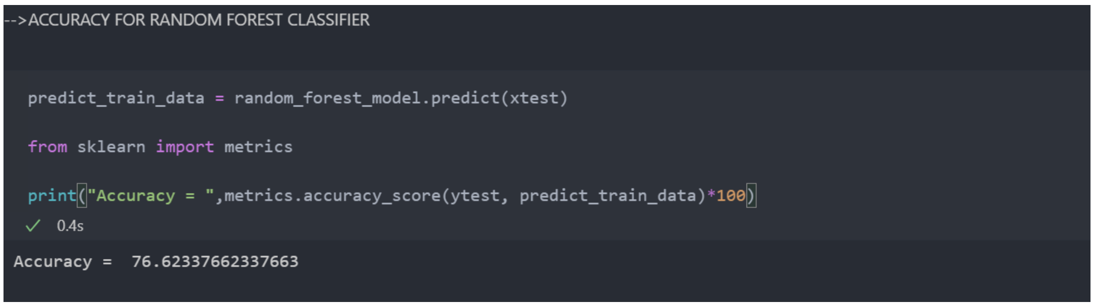

-->ACCURACY FOR SUPPORT VECTOR MACHINE

```
#ACCURACY ON TRAINING DATASET
x_train_predict =  svmmodel.predict(xtrain)
train_accuracy = accuracy_score(x_train_predict, ytrain)

print("Accuracy =", train_accuracy*100)
```
✓ 0.3s
```
Accuracy = 78.17589576547232
```

```
#ACCURACY ON TESTING DATASET
x_test_predict =  svmmodel.predict(xtest)
test_accuracy = accuracy_score(x_test_predict, ytest)

print("Accuracy =", test_accuracy*100)
```
✓ 0.3s
```
Accuracy = 77.27272727272727
```

METHODOLOGY &IMPLEMENTATION

--: RANDOM FOREST CLASSIFIER



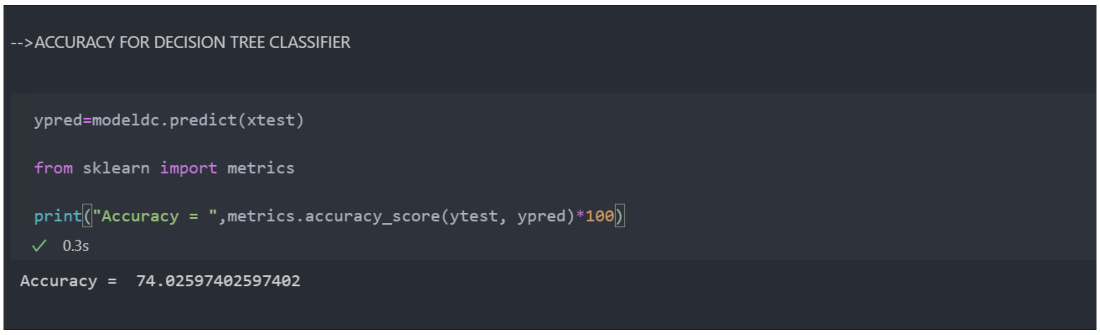
-->ACCURACY FOR RANDOM FOREST CLASSIFIER

```
predict_train_data = random_forest_model.predict(xtest)

from sklearn import metrics

print("Accuracy = ",metrics.accuracy_score(ytest, predict_train_data)*100)
```

✓ 0.4s

```
Accuracy =  76.62337662337663
```

--: DECISION TREE CLASSIFIER



-->ACCURACY FOR DECISION TREE CLASSIFIER

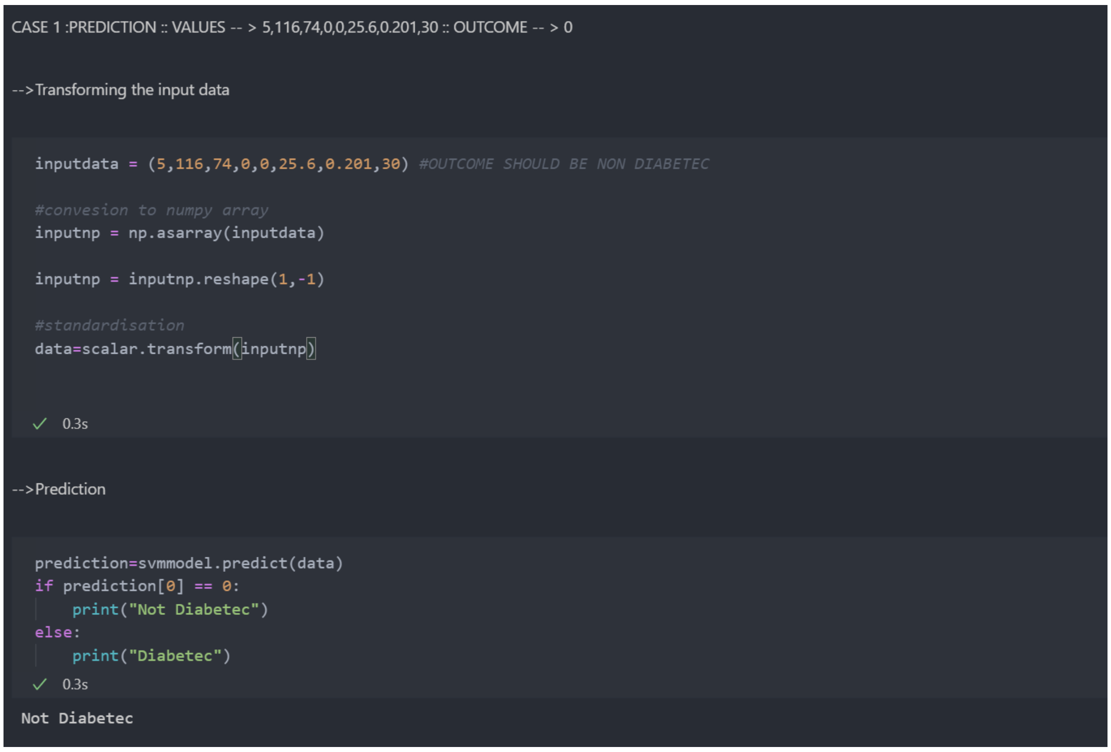
```
ypred=modeldc.predict(xtest)

from sklearn import metrics

print("Accuracy = ",metrics.accuracy_score(ytest, ypred)*100)
```

✓ 0.3s

```
Accuracy =  74.02597402597402
```

SINCE WE GET THE BEST ACCURACY USING SUPPORT VECTOR MACHINE MODEL WE USE IT TO PREDICT THE TEST CASES

PREDICTION



CASE 1 :PREDICTION :: VALUES -- > 5,116,74,0,0,25.6,0.201,30 :: OUTCOME -- > 0

-->Transforming the input data

```
inputdata = (5,116,74,0,0,25.6,0.201,30) #OUTCOME SHOULD BE NON DIABETEC

#convesion to numpy array
inputnp = np.asarray(inputdata)

inputnp = inputnp.reshape(1,-1)

#standardisation
data=scalar.transform(inputnp)
```

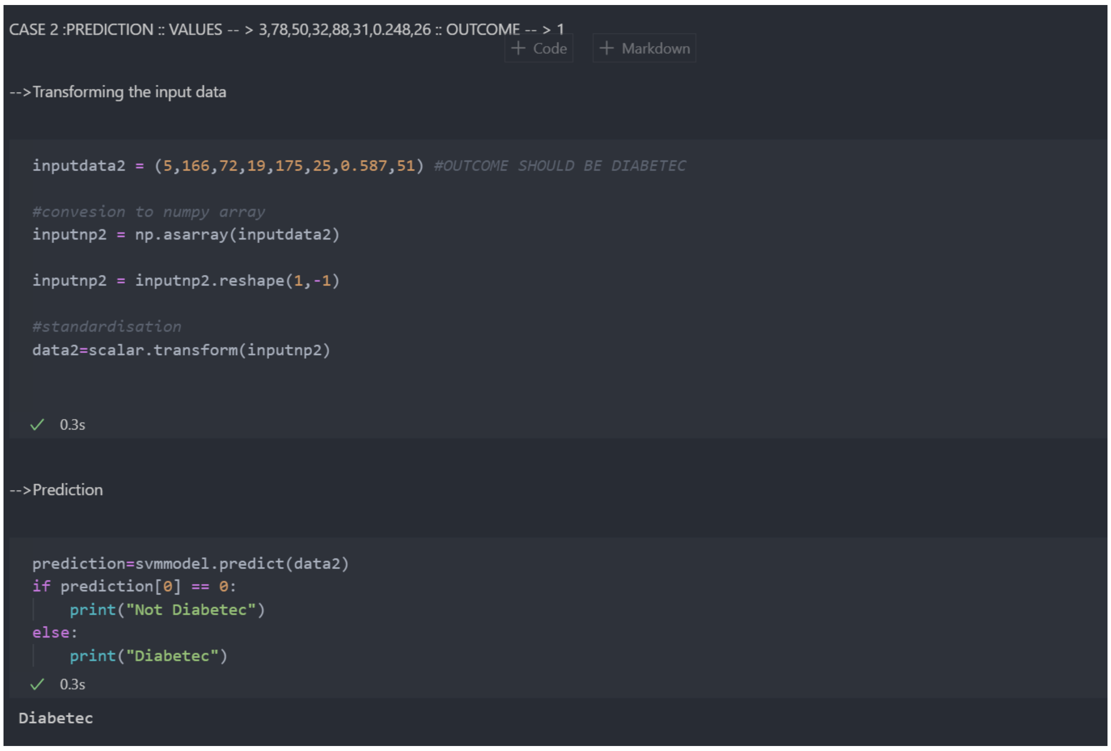
✓ 0.3s

-->Prediction

```
prediction=svmmodel.predict(data)
if prediction[0] == 0:
    print("Not Diabetec")
else:
    print("Diabetec")
```

✓ 0.3s

```
Not Diabetec
```



CASE 2 :PREDICTION :: VALUES -- > 3,78,50,32,88,31,0.248,26 :: OUTCOME -- > 1

-->Transforming the input data

```python
inputdata2 = (5,166,72,19,175,25,0.587,51) #OUTCOME SHOULD BE DIABETEC

#convesion to numpy array
inputnp2 = np.asarray(inputdata2)

inputnp2 = inputnp2.reshape(1,-1)

#standardisation
data2=scalar.transform(inputnp2)
```

✓ 0.3s

-->Prediction

```python
prediction=svmmodel.predict(data2)
if prediction[0] == 0:
    print("Not Diabetec")
else:
    print("Diabetec")
```

✓ 0.3s

```
Diabetec
```

CONCLUSION

Here we applied different Machine Learning techniques to construct a diabetes classifier. Accomplished an accuracy score of 78 % through the use of Support Vector Machine Classifier.

I thank you for providing me with an opportunity to develop such an interesting project using machine learning.

CONTACT

MILIND DALAKOTI

9953653656

milinddalakoti@gmail.com
github.com/milinddalakoti
linked.in/milind-dalakoti/