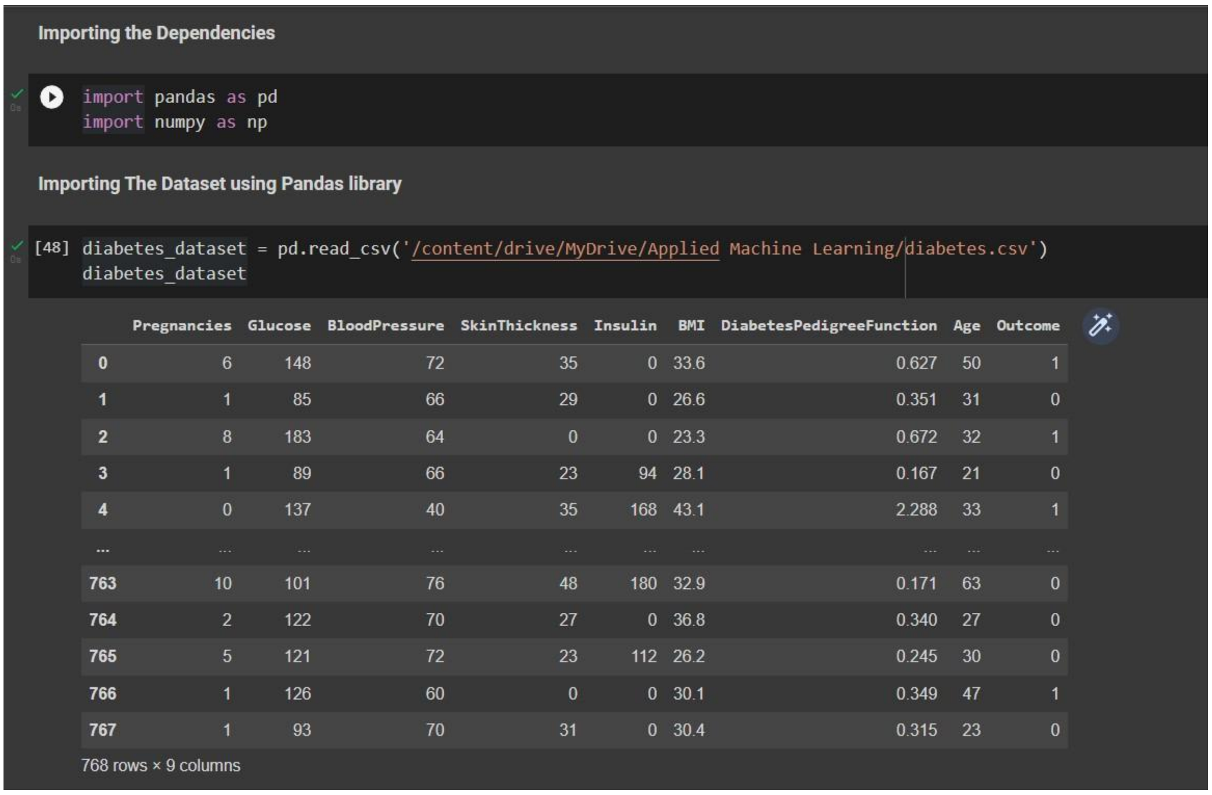
Mini Project

Aim: Build a diabetes predictive system using Machine Learning.

Code:



Importing the Dependencies

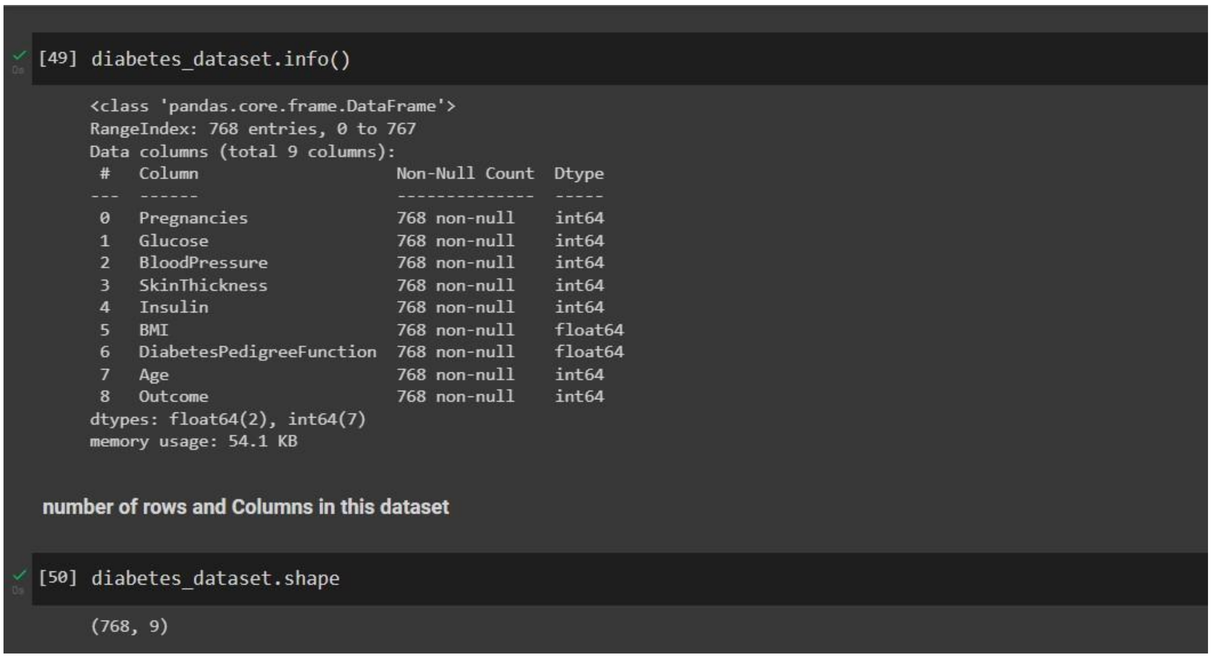
```
import pandas as pd
import numpy as np
```

Importing The Dataset using Pandas library

```
[48] diabetes_dataset = pd.read_csv('/content/drive/MyDrive/Applied Machine Learning/diabetes.csv')
     diabetes_dataset
```

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns



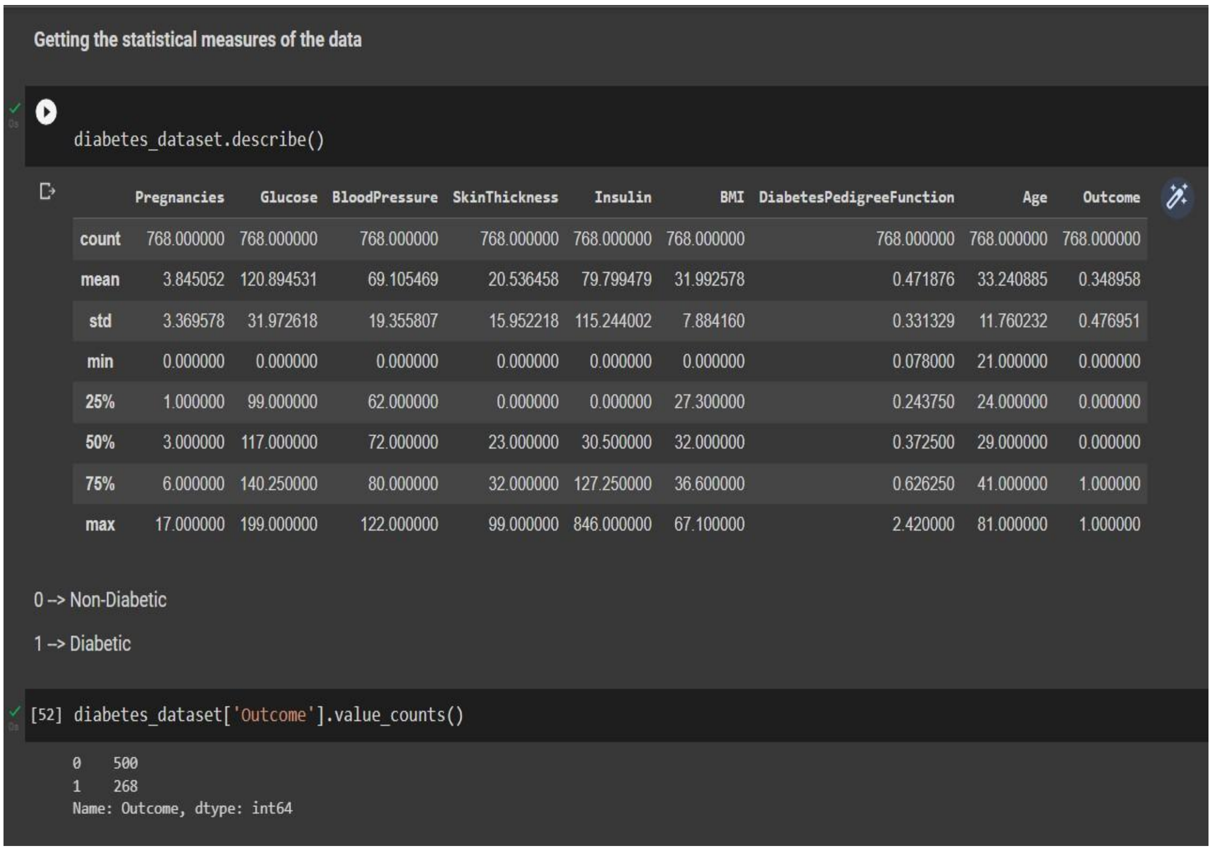
```
[49] diabetes_dataset.info()
```

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

number of rows and Columns in this dataset

```
[50] diabetes_dataset.shape
```

```
(768, 9)
```



Getting the statistical measures of the data

```
diabetes_dataset.describe()
```

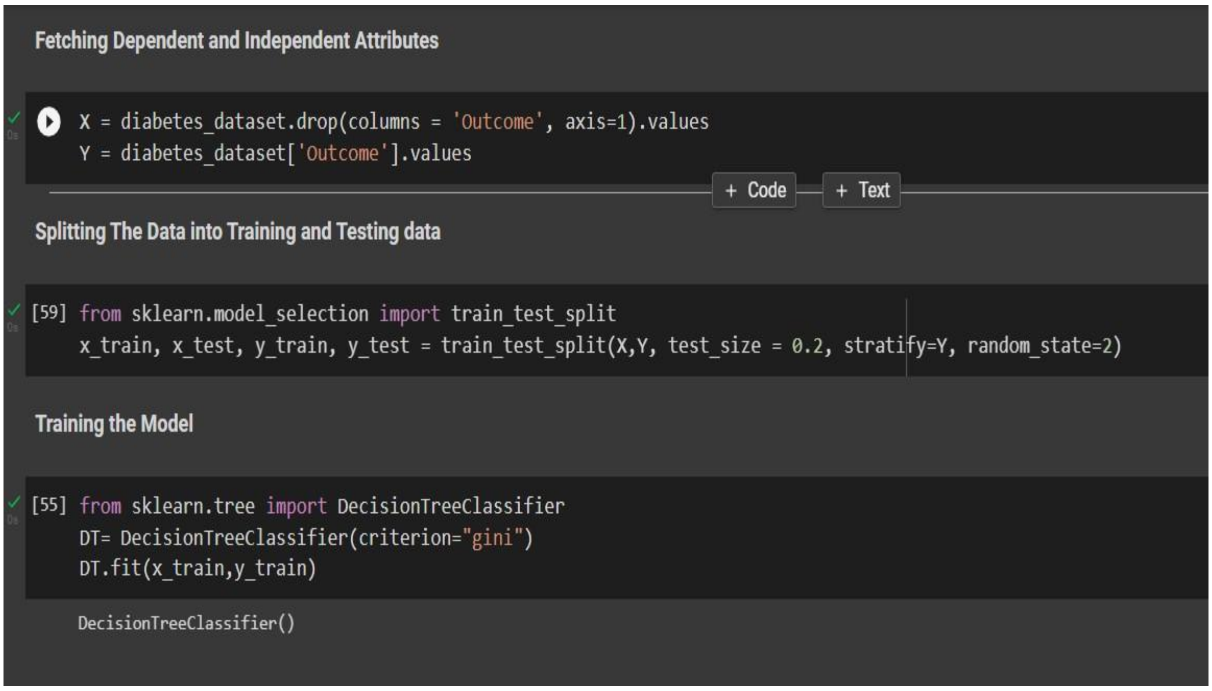
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	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

0 --> Non-Diabetic

1 --> Diabetic

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

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



Fetching Dependent and Independent Attributes

```
X = diabetes_dataset.drop(columns = 'Outcome', axis=1).values
Y = diabetes_dataset['Outcome'].values
```

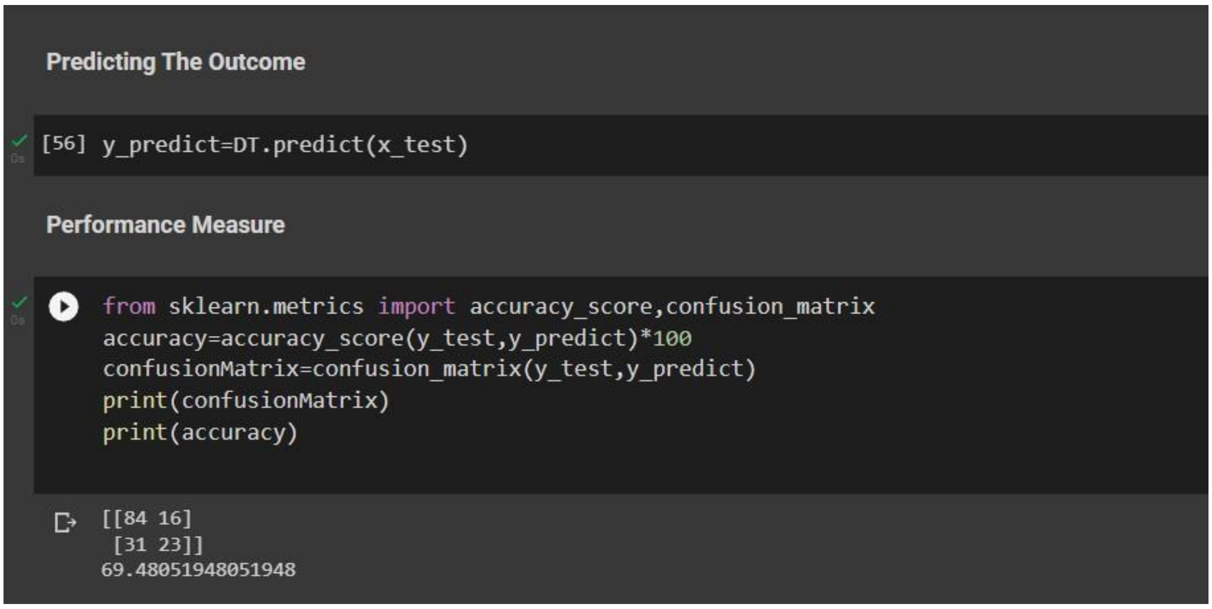
Splitting The Data into Training and Testing data

```
[59] from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)
```

Training the Model

```
[55] from sklearn.tree import DecisionTreeClassifier
     DT= DecisionTreeClassifier(criterion="gini")
     DT.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```



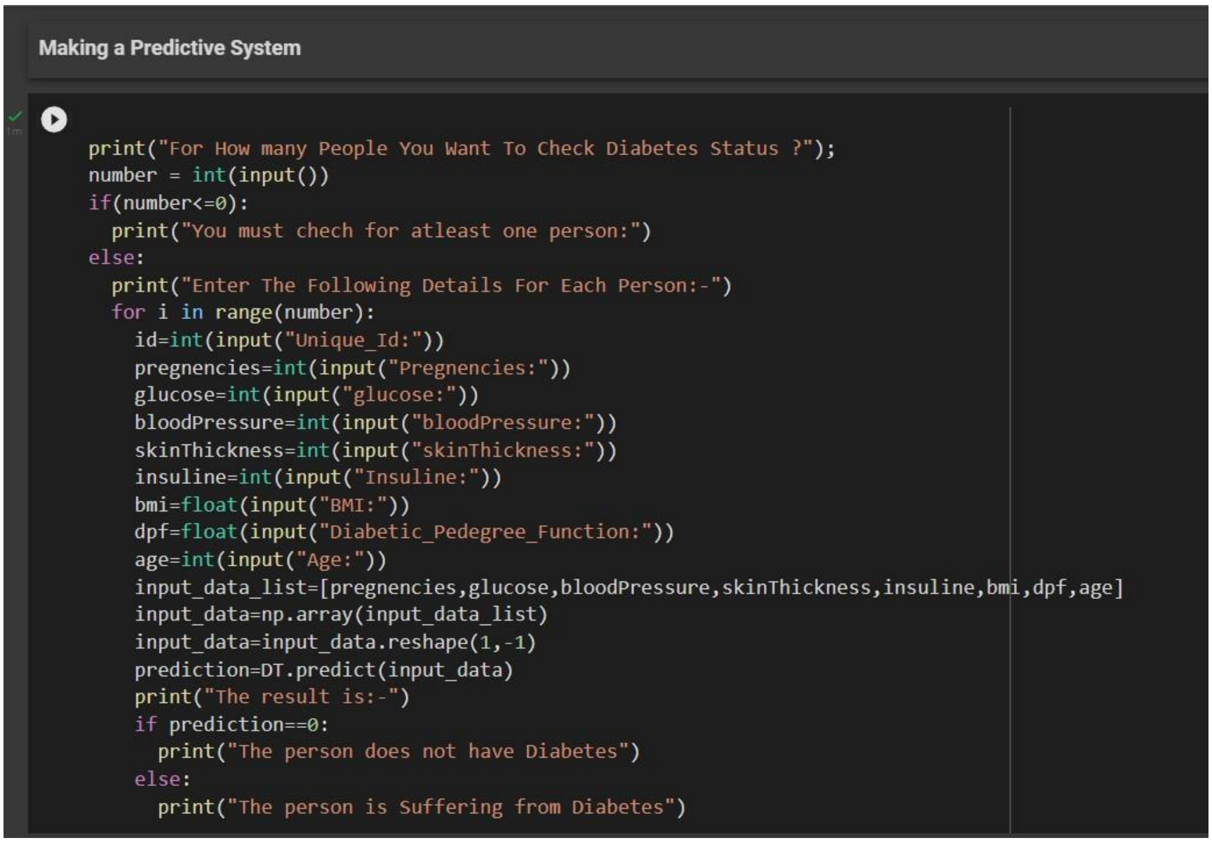
Predicting The Outcome

```
[56] y_predict=DT.predict(x_test)
```

Performance Measure

```
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy=accuracy_score(y_test,y_predict)*100
confusionMatrix=confusion_matrix(y_test,y_predict)
print(confusionMatrix)
print(accuracy)
```

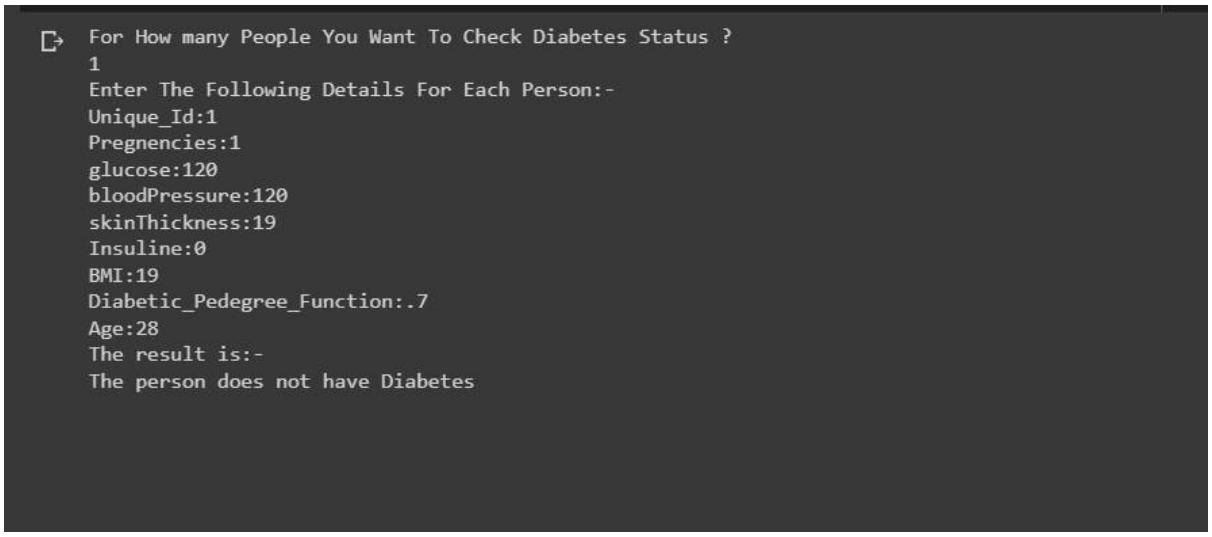
```
[[84 16]
 [31 23]]
69.48051948051948
```



Making a Predictive System

```
print("For How many People You Want To Check Diabetes Status ?");
number = int(input())
if(number<=0):
  print("You must chech for atleast one person:")
else:
  print("Enter The Following Details For Each Person:-")
  for i in range(number):
    id=int(input("Unique_Id:"))
    pregnencies=int(input("Pregnencies:"))
    glucose=int(input("glucose:"))
    bloodPressure=int(input("bloodPressure:"))
    skinThickness=int(input("skinThickness:"))
    insuline=int(input("Insuline:"))
    bmi=float(input("BMI:"))
    dpf=float(input("Diabetic_Pedegree_Function:"))
    age=int(input("Age:"))
    input_data_list=[pregnencies,glucose,bloodPressure,skinThickness,insuline,bmi,dpf,age]
    input_data=np.array(input_data_list)
    input_data=input_data.reshape(1,-1)
    prediction=DT.predict(input_data)
    print("The result is:-")
    if prediction==0:
      print("The person does not have Diabetes")
    else:
      print("The person is Suffering from Diabetes")
```

Output:-



```
For How many People You Want To Check Diabetes Status ?
1
Enter The Following Details For Each Person:-
Unique_Id:1
Pregnencies:1
glucose:120
bloodPressure:120
skinThickness:19
Insuline:0
BMI:19
Diabetic_Pedegree_Function:.7
Age:28
The result is:-
The person does not have Diabetes
```