

# **DIABETIC CELL DETECTION USING LOGISTIC REGRESSION**

## **A PROJECT REPORT**

*In partial fulfilment of the requirements for the award for this  
training in **VALUABLE ADDED TRAINING (VAT)** Under the  
guidance of*

**Sofikul Mullick**

BY

<b>SWARNADEEP JANA</b>	<b>10905519038</b>
<b>KIRAN DEY</b>	<b>10905519037</b>
<b>KRISHANU ADAK</b>	<b>10905519039</b>



**Netaji Subhash Engineering College, Garia**

**In association with  
Engineering Study Centre (ESC)**

*(Note: All entries of the proforma of approval should be filled up with  
appropriate and complete information. Incomplete proforma of approval in any  
respect will be summarily rejected.)*

1.	Title of the Project:	<b>DIABETIC CELL DETECTION USING LOGISTIC REGRESSION</b>
2.	Project Members:	<b>SWARNADEEP JANA KIRAN DEY KRISHANU ADAK</b>
3.	Name of the guide:	<b>SOFIKUL MULLICK</b>

### Project Version Control History

Version	Primary Author	Description of Version	Date Completed
Final	Swarnadeep Jana  Kiran Dey  Krishanu Adak	Project Report	1 <sup>st</sup> June, 2021

Swarnadeep Jana\_\_\_\_\_

Signature of Head of the Team

Date: 1.06.2021

For Office Use Only

**Approved**

**Not Approved**

\_\_\_\_\_  
Signature of Approver

Date:

**SOFIKUL MULLICK**

Project Proposal Evaluator

## **DECLARATION**

We hereby declare that the project work being presented in the project proposal entitled “**DIABETIC CELL DETECTION USING LOGISTIC REGRESSION**” in partial fulfilment of the requirements for the award for this training in **VALUABLE ADDED TRAINING (VAT)** Under the at **ENGINEERING STUDY CENTER, SALT LAKE, KOLKATA, WEST BENGAL**, is an authentic work carried out under the guidance of **SOFIKUL MULLICK**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

**Date:**

**Name of the Student:** Swarnadeep Jana  
Kiran Dey  
Krishanu Adak

**Signature of the students:**

1. *Swarnadeep Jana*
2. *Kiran Dey*
3. *Krishanu Adak*

## **CERTIFICATE**

This is to certify that this proposal of minor project entitled “**DIABETIC CELL DETECTION USING LOGISTIC REGRESSION**” is a record of bona fide work, carried out by **Swarnadeep Jana, Kiran Dey, Krishanu Adak** under my guidance at **Engineers Study Center**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award for this training in **VALUABLE ADDED TRAINING (VAT)** and as per regulations of the **Engineers Study Center®**. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

**Guide / Supervisor**

-----  
**SOFIKUL MULLICK**

Project Engineer  
Engineers Study Center

## **ACKNOWLEDGEMENT**

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to **SOFIKUL MULLICK, Project Engineer at Engineering Study Center, Kolkata**. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Engineers Study Center. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

# **INDEX**

<b>Sl No</b>	<b>Topic</b>	<b>Page No</b>
1.	Abstract	7
2.	Introduction	8-10
3.	Problem Definition	11
4.	Project Goal	12
5.	Methodology	13-14
6.	Project Objective	15
7.	Project Implementation	16-17
8.	Step-by-Step Working	18-29
9.	Future Scope	30
10.	Summary	31
11.	Code	32-41

## **ABSTRACT**

Due to its continuously increasing occurrence, more and more families are influenced by diabetes mellitus. Most diabetics know little about their health quality or the risk factors they face prior to diagnosis. In this study, we have proposed a novel model based on data mining techniques for predicting type 2 diabetes mellitus (T2DM). The main problems that we are trying to solve are to improve the accuracy of the prediction model, and to make the model adaptive to more than one dataset. Based on a series of preprocessing procedures, the model is comprised of two parts, the improved K-means algorithm and the logistic regression algorithm. The Pima Indians Diabetes Dataset and the Waikato Environment for Knowledge Analysis toolkit were utilized to compare our results with the results from other researchers. The conclusion shows that the model attained a 3.04% higher accuracy of prediction than those of other researchers. Moreover, our model ensures that the dataset quality is sufficient. To further evaluate the performance of our model, we applied it to two other diabetes datasets. Both experiments' results show good performance. As a result, the model is shown to be useful for the realistic health management of diabetes.

## **INTRODUCTION**

**Python** is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

**Machine learning (ML)** is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

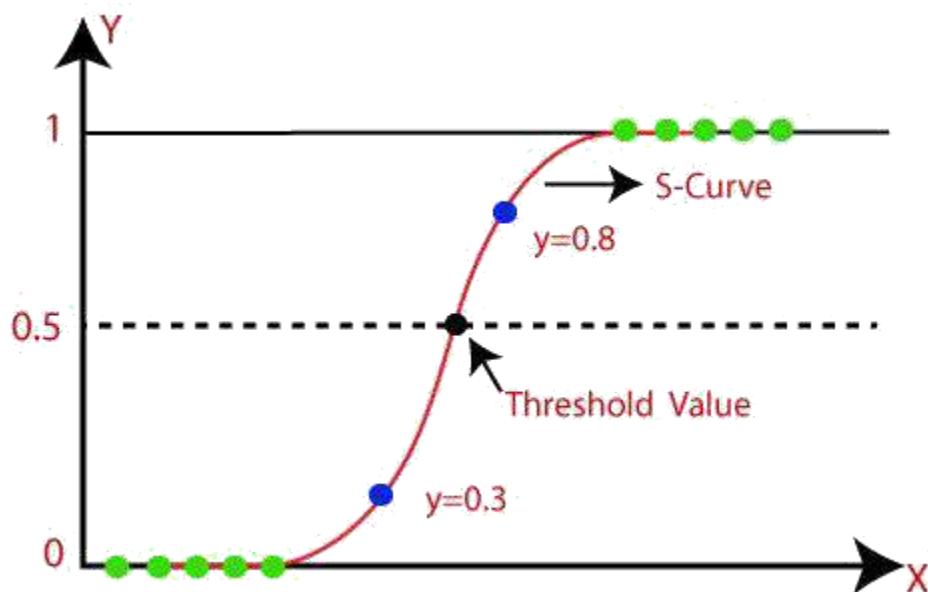
**Logistic regression** is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is



dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts  $P(Y=1)$  as a function of  $X$ . It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.



### **Type of Logistic Regression:**

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

## **PROBLEM DEFINITION**

Diabetes mellitus, commonly known as diabetes, is a metabolic disease that causes high blood sugar. The hormone insulin moves sugar from the blood into your cells to be stored or used for energy. With diabetes, your body either doesn't make enough insulin or can't effectively use the insulin it does make.

So here we calculate how much the possibilities to have a diabetic cell. We can know that from where the diabetic Cell started increasing.

## **PROJECT GOAL**

The goal is to predict whether a person has diabetic cell or not. The goal is achieved by using logistic regression in machine learning. Diabetes is a common for all people around the world, mainly in india we know that every 100 people there will be 5 people who have diabetes. Early detection of diabetic cell can greatly improve prognosis by promoting clinical treatment to patients.

## **METHODOLOGY**

- **Data Selection:** Data is the foundation for any machine learning project. The job is to find ways and sources of collecting relevant and comprehensive data, interpreting it, and analyzing results with the help of statistical techniques.
- **Data cleaning:** This set of procedures allows for removing noise and fixing inconsistencies in data. A data scientist can fill in missing data using imputation techniques. A specialist also detects outliers — observations that deviate significantly from the rest of distribution.
- **Dependent and independent of data:** If the values in one sample affect the values in the sample, then the samples are dependent. If the values in one sample reveal no information about those of the other sample, then the samples are independent.
- **Data Visualization:** A large amount of information represented in graphic form is easier to understand and analyze. Some companies specify that a data analyst must know how to create slides, diagrams, charts, and templates.

- **Data Splitting:** A dataset used for machine learning should be partitioned into three subsets — training, test, and validation sets.
- **Model Selection:** After a data scientist has preprocessed the collected data and split it into three subsets, he or she can proceed with a model training. This process entails “feeding” the algorithm with training data. An algorithm will process data and output a model that is able to find a target value in new data. The purpose of model training is to develop a model.
- **Model Evaluation:** The goal of this step is to develop the simplest model able to formulate a target value fast and well enough and check the accuracy

## **PROJECT OBJECTIVE**

The main objective of this is diabetic Cell Detection using Logistic Regression.

## **PROJECT IMPLEMENTATION**

- **SELECTION OF DATA:** The process of selecting data depends on the type of project we desire to. The data set can be collected from various sources such as a file, database, sensor and many other such sources.
- **VISUALIZATION OF DATA:** Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns.
- **SELECTION OF DEPENDENT AND INDEPENDENT DATA:**  
We need to select the dependent and independent data and store them in y and x.
- **SPLITTING OF THE DATA:** We train the classifier using ‘training data set’, then test the performance of your classifier on unseen ‘test data set’. We split the data for training and testing by using the ‘train\_test\_split’.



- **FITTING THE MODEL:** In a data set, a training set is implemented to build up a model. Once the model is trained, we can use the same trained model to predict using the testing data i.e., the unseen data. Once this is done, we can develop a confusion matrix, this tells us how well our model is trained.

- **MODEL EVALUATION:** It is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future.

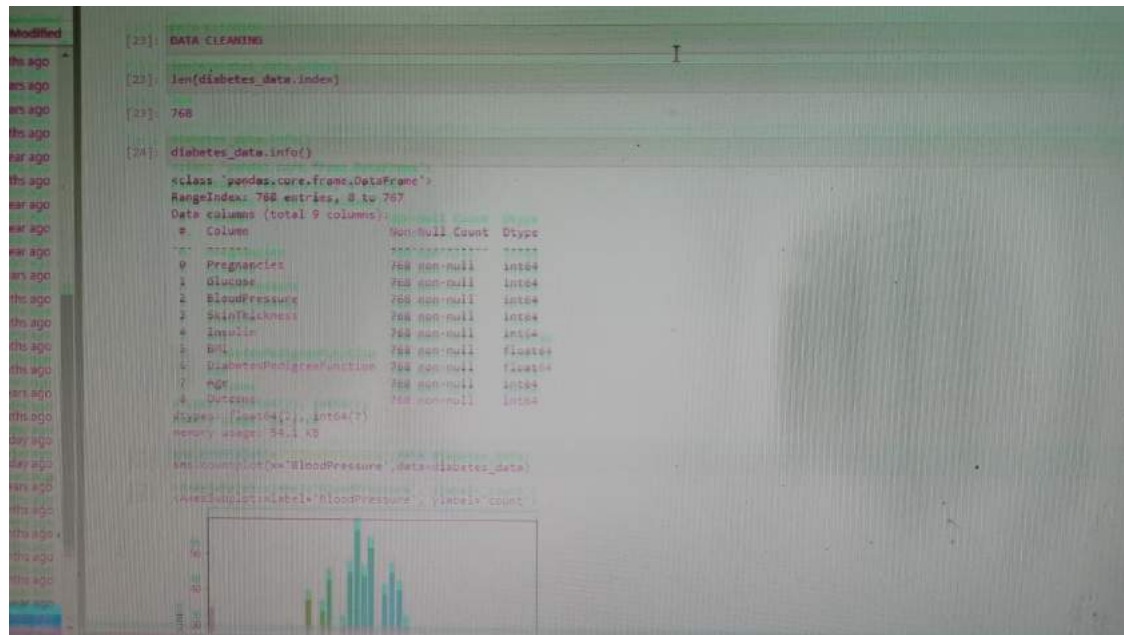
### DATA SELECTION:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

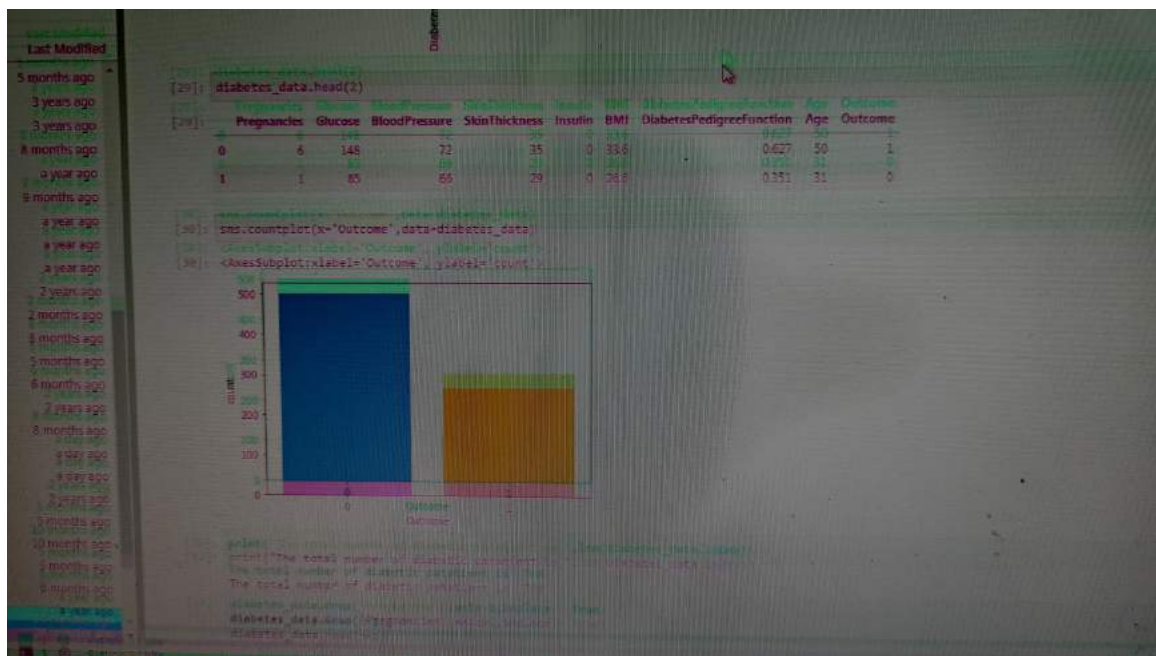
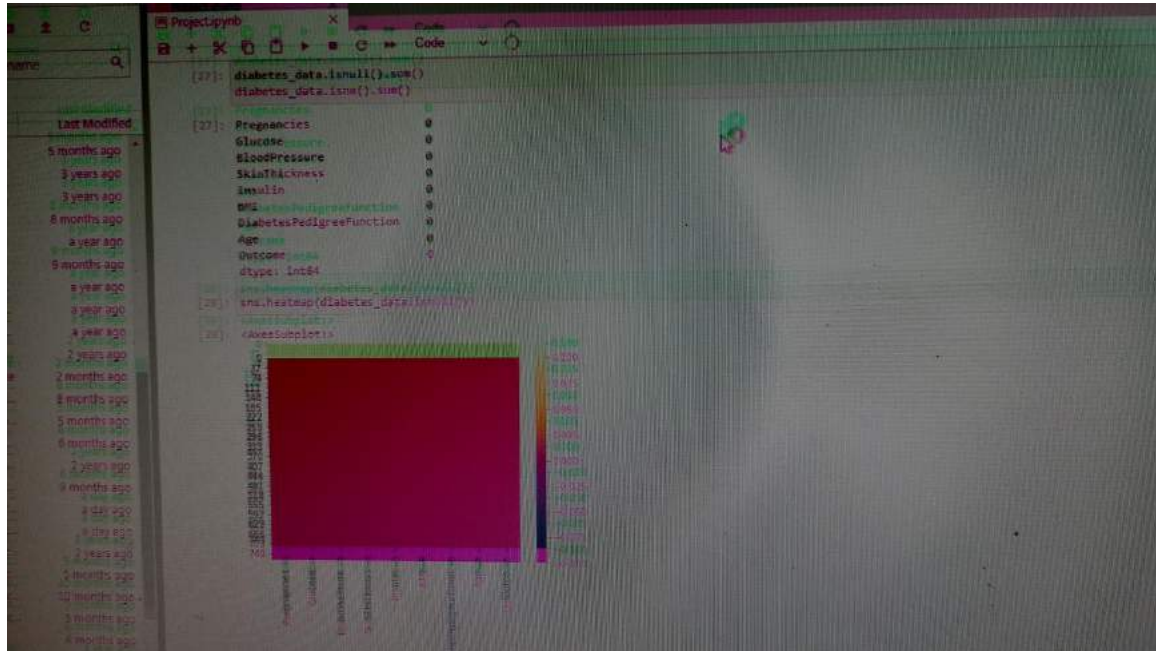
# DATA SELECTION
diabetes_data = pd.read_csv('C:\Users\usmannadeep\Desktop\Var training on python on ML and DA\diabetes1.csv')
diabetes_data.head()
diabetes_data.tail()
diabetes_data.info()
diabetes_data.describe()
diabetes_data.groupby('Outcome').describe()
```

[illegible]

## CLEANING DATA:



# CLEANING DATA:





# CLEANING DATA:

```

[1]: print("The total number of diabetic patients is: ", len(diabetes_data))
# the total number of diabetic patients is: 768

[2]: diabetes_data.drop("Pregnancies", axis=1, inplace=True)
diabetes_data.head(n=5)

[3]:
Disease BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
0 0 140 72 35 1 0.336 33 1
1 1 101 65 40 0 0.258 31 0
2 1 103 64 40 0 0.234 33 1
3 1 89 80 28 40 0.281 26 0
4 0 127 40 35 100 0.191 33 1

[4]: diabetes_data.tail(n=5)

[5]:
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
763 10 201 76 40 160 22.5 0.171 45 0
764 2 222 100 27 0 0.433 30 0
765 5 221 72 33 100 26.1 0.191 30 0
766 1 124 80 35 0 0.198 27 1
767 1 85 70 35 0 0.154 25 0

[6]: diabetes_data.info()
[7]: diabetes_data.describe()

```

diabetes\_data

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	140	72	35	1	0.336	33	1	
1	1	101	65	40	0	0.258	31	0	
2	1	103	64	40	0	0.234	33	1	
3	1	89	80	28	40	0.281	26	0	
4	0	127	40	35	100	0.191	33	1	
5	0	173	70	45	160	0.171	30	0	
6	1	158	76	40	160	0.171	45	0	
7	1	159	68	27	0	0.433	30	0	
8	1	153	78	35	0	0.354	30	0	
9	5	221	72	33	100	0.191	30	0	
10	1	124	80	35	0	0.198	27	1	
11	1	99	96	30	0	0.246	30	1	
12	1	85	70	35	0	0.154	25	0	

768 rows x 9 columns

768 rows x 9 columns

diabetes\_data.head()

diabetes\_data.tail()

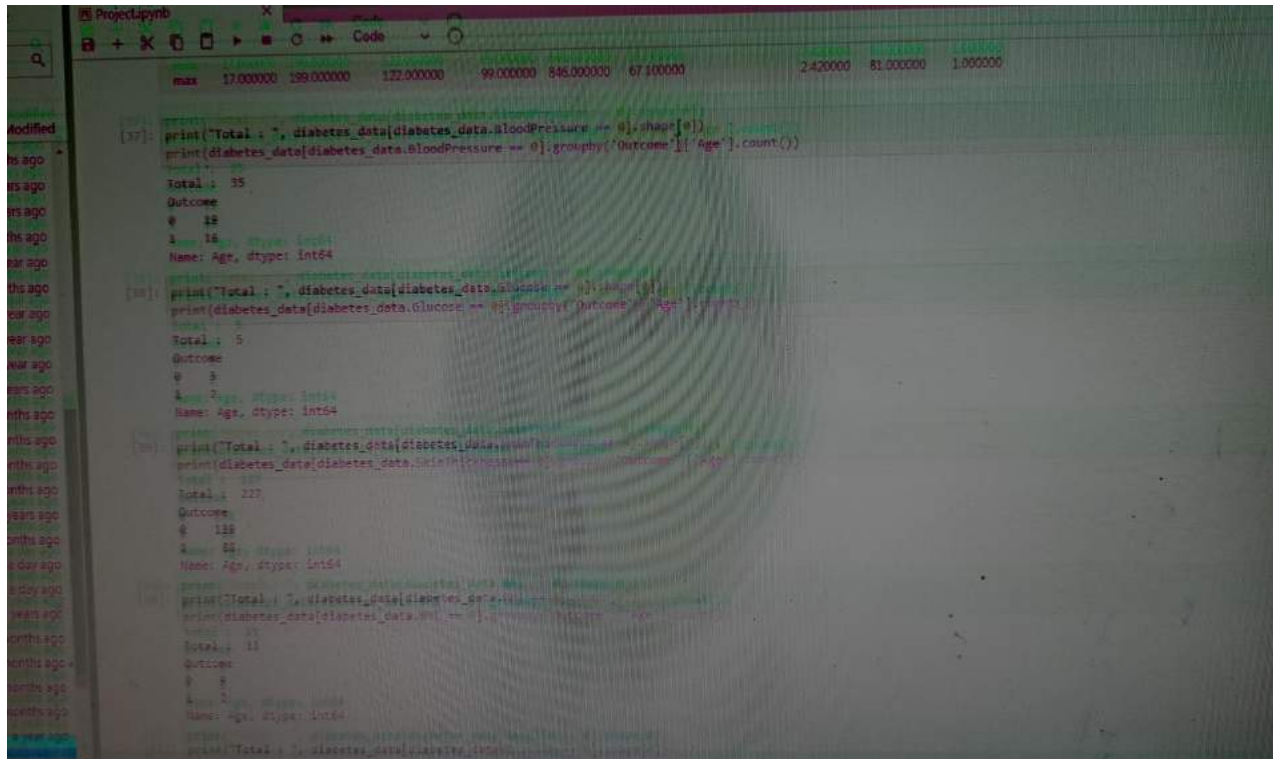
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	140	72	35	1	0.336	33	1	
1	1	101	65	40	0	0.258	31	0	
2	1	103	64	40	0	0.234	33	1	
3	1	89	80	28	40	0.281	26	0	
4	0	127	40	35	100	0.191	33	1	
5	0	173	70	45	160	0.171	30	0	
6	1	158	76	40	160	0.171	45	0	
7	1	159	68	27	0	0.433	30	0	
8	1	153	78	35	0	0.354	30	0	
9	5	221	72	33	100	0.191	30	0	
10	1	124	80	35	0	0.198	27	1	
11	1	99	96	30	0	0.246	30	1	
12	1	85	70	35	0	0.154	25	0	
13	0	183	64	40	0	0.234	33	1	
14	0	196	64	40	0	0.234	33	1	
15	0	175	64	40	0	0.234	33	1	
16	0	146	64	40	0	0.234	33	1	
17	0	140	64	40	0	0.234	33	1	
18	0	133	64	40	0	0.234	33	1	
19	0	133	64	40	0	0.234	33	1	
20	0	133	64	40	0	0.234	33	1	
21	0	133	64	40	0	0.234	33	1	
22	0	133	64	40	0	0.234	33	1	
23	0	133	64	40	0	0.234	33	1	
24	0	133	64	40	0	0.234	33	1	
25	0	133	64	40	0	0.234	33	1	
26	0	133	64	40	0	0.234	33	1	
27	0	133	64	40	0	0.234	33	1	
28	0	133	64	40	0	0.234	33	1	
29	0	133	64	40	0	0.234	33	1	
30	0	133	64	40	0	0.234	33	1	
31	0	133	64	40	0	0.234	33	1	
32	0	133	64	40	0	0.234	33	1	
33	0	133	64	40	0	0.234	33	1	
34	0	133	64	40	0	0.234	33	1	
35	0	133	64	40	0	0.234	33	1	
36	0	133	64	40	0	0.234	33	1	
37	0	133	64	40	0	0.234	33	1	
38	0	133	64	40	0	0.234	33	1	
39	0	133	64	40	0	0.234	33	1	
40	0	133	64	40	0	0.234	33	1	
41	0	133	64	40	0	0.234	33	1	
42	0	133	64	40	0	0.234	33	1	
43	0	133	64	40	0	0.234	33	1	
44	0	133	64	40	0	0.234	33	1	
45	0	133	64	40	0	0.234	33	1	
46	0	133	64	40	0	0.234	33	1	
47	0	133	64	40	0	0.234	33	1	
48	0	133	64	40	0	0.234	33	1	
49	0	133	64	40	0	0.234	33	1	
50	0	133	64	40	0	0.234	33	1	
51	0	133	64	40	0	0.234	33	1	
52	0	133	64	40	0	0.234	33	1	
53	0	133	64	40	0	0.234	33	1	
54	0	133	64	40	0	0.234	33	1	
55	0	133	64	40	0	0.234	33	1	
56	0	133	64	40	0	0.234	33	1	
57	0	133	64	40	0	0.234	33	1	
58	0	133	64	40	0	0.234	33	1	
59	0	133	64	40	0	0.234	33	1	
60	0	133	64	40	0	0.234	33	1	
61	0	133	64	40	0	0.234	33	1	
62	0	133	64	40	0	0.234	33	1	
63	0	133	64	40	0	0.234	33	1	
64	0	133	64	40	0	0.234	33	1	
65	0	133	64	40	0	0.234	33	1	
66	0	133	64	40	0	0.234	33	1	
67	0	133	64	40	0	0.234	33	1	
68	0	133	64	40	0	0.234	33	1	
69	0	133	64	40	0	0.234	33	1	
70	0	133	64	40	0	0.234	33	1	
71	0	133	64	40	0	0.234	33	1	
72	0	133	64	40	0	0.234	33	1	
73	0	133	64	40	0	0.234	33	1	
74	0	133	64	40	0	0.234	33	1	
75	0	133	64	40	0	0.234	33	1	
76	0	133	64	40	0	0.234	33	1	
77	0	133	64	40	0	0.234	33	1	
78	0	133	64	40	0	0.234	33	1	
79	0	133	64	40	0	0.234	33	1	
80	0	133	64	40	0	0.234	33	1	
81	0	133	64	40	0	0.234	33	1	
82	0	133	64	40	0	0.234	33	1	
83	0	133	64	40	0	0.234	33	1	
84	0	133	64	40	0	0.234	33	1	
85	0	133	64	40	0	0.234	33	1	
86	0	133	64	40	0	0.234	33	1	
87	0	133	64	40	0	0.234	33	1	
88	0	133	64	40	0	0.234	33	1	
89	0	133	64	40	0	0.234	33	1	
90	0	133	64	40	0	0.234	33	1	
91	0	133	64	40	0	0.234	33	1	
92	0	133	64	40	0	0.234	33	1	
93	0	133	64	40	0	0.234	33	1	
94	0	133	64	40	0	0.234	33	1	
95	0	133	64	40	0	0.234	33	1	
96	0	133	64	40	0	0.234	33	1	
97	0	133	64	40	0	0.234	33	1	
98	0	133	64	40	0	0.234	33	1	
99	0	133	64	40	0	0.234	33	1	

diabetes\_data.describe()

768 rows x 9 columns

	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	3.845000	120.884501	69.125000	31.612500	39.779167	25.962278	0.471875	33.343750	0.348594
std	3.369578	31.972418	19.359375	17.957219	115.344578	7.614670	0.137129	11.760222	0.479504
min	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	21.000000	0.343750	24.000000	0.000000
50%	1.000000	117.000000	72.000000	0.000000	0.000000	25.000000	0.375000	29.000000	0.000000
75%	1.000000	142.250000	80.000000	0.000000	0.000000	31.000000	0.468750	31.000000	1.000000
max	17.000000	199.000000	126.000000	99.000000	1900.000000	43.000000	0.671000	50.000000	1.000000

## CLEANING DATA:



```
[37]: print("Total : ", diabetes_data[diabetes_data.bloodPressure == 0].shape[0])
      print(diabetes_data[diabetes_data.bloodPressure == 0].groupby('Outcome')['Age'].count())

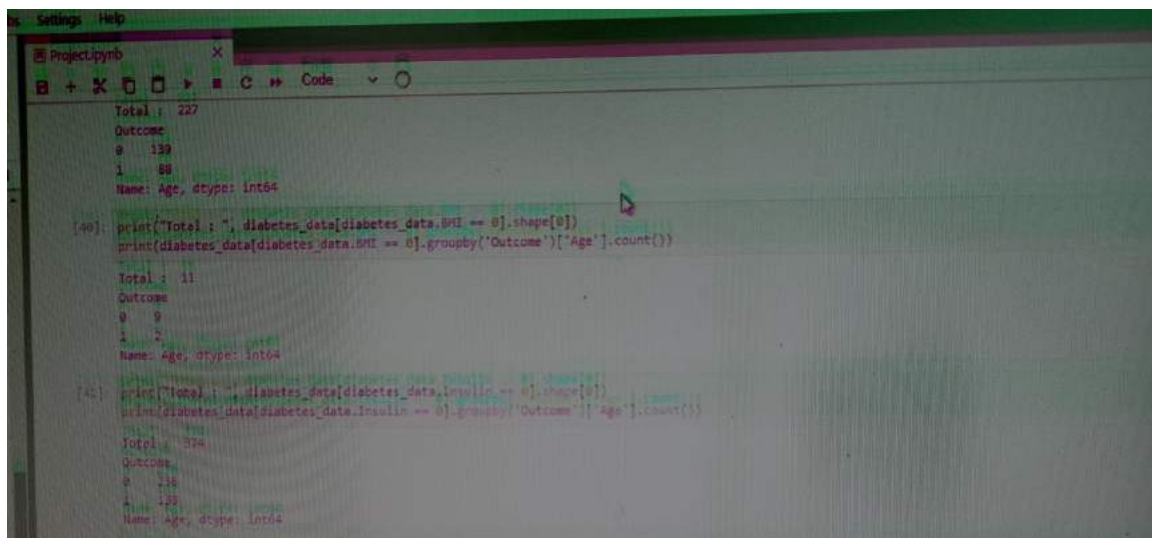
      Total : 35
      Outcome
      0    18
      1    16
      Name: Age, dtype: int64

[38]: print("Total : ", diabetes_data[diabetes_data.Glucose == 0].shape[0])
      print(diabetes_data[diabetes_data.Glucose == 0].groupby('Outcome')['Age'].count())

      Total : 5
      Outcome
      0    3
      1    2
      Name: Age, dtype: int64

[39]: print("Total : ", diabetes_data[diabetes_data.SkinThickness == 0].shape[0])
      print(diabetes_data[diabetes_data.SkinThickness == 0].groupby('Outcome')['Age'].count())

      Total : 227
      Outcome
      0   139
      1    88
      Name: Age, dtype: int64
```



```
[40]: print("Total : ", diabetes_data[diabetes_data.BMI == 0].shape[0])
      print(diabetes_data[diabetes_data.BMI == 0].groupby('Outcome')['Age'].count())

      Total : 11
      Outcome
      0    9
      1    2
      Name: Age, dtype: int64

[41]: print("Total : ", diabetes_data[diabetes_data.Insulin == 0].shape[0])
      print(diabetes_data[diabetes_data.Insulin == 0].groupby('Outcome')['Age'].count())

      Total : 179
      Outcome
      0   136
      1   129
      Name: Age, dtype: int64
```

## DEPENDENT AND INDEPENDENT OF DATA :

```
[43]: Dependent and independent of data
File: C:\python\input\33-e1e2d1f0652d1", line 1
dependent and independent of data
SyntaxError: Invalid syntax

[44]: y = diabetes_data['Outcome']

[45]: x = diabetes_data.drop('Pregnancies', axis=1)
from sklearn.feature_selection import RFE
model = LogisticRegression(solver='lbfgs', max_iter=1000)
rfe = RFE(model, n_features_to_select=10)
fit = rfe.fit(x,y)
dropColumns=[]
for i in range(0,30):
    if (fit.ranking_[i]>10):
        dropColumns.append(X.columns[i])

IndexError: Input 49 is out of bounds for axis 0 with size 30
File: C:\python\input\33-e1e2d1f0652d1", line 1
IndexError: Input 49 is out of bounds for axis 0 with size 30
SyntaxError: Invalid syntax

[46]: diabetes_data.head()
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
0           6         120             70              35         0   23.6              0.357                30         1
1           1          85             66              29         0   26.6              0.687                31         0
2           8         183             60              43         0   23.3              0.361                33         1
3           1          89             69              27         94  26.6              0.167                21         0
4           0         137             40              35         160  43.1              0.268                33         1
```

```
[ ]: x.drop(dropColumns,axis=1,inplace=True)
x.head()

[46]: diabetes_data.head()
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
0           6         120             70              35         0   23.6              0.357                30         1
1           1          85             66              29         0   26.6              0.687                31         0
2           8         183             60              43         0   23.3              0.361                33         1
3           1          89             69              27         94  26.6              0.167                21         0
4           0         137             40              35         160  43.1              0.268                33         1

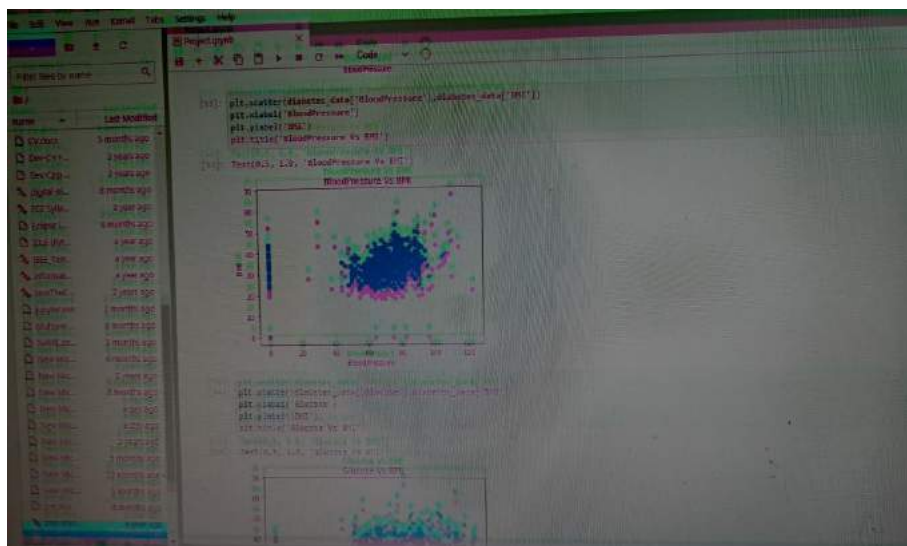
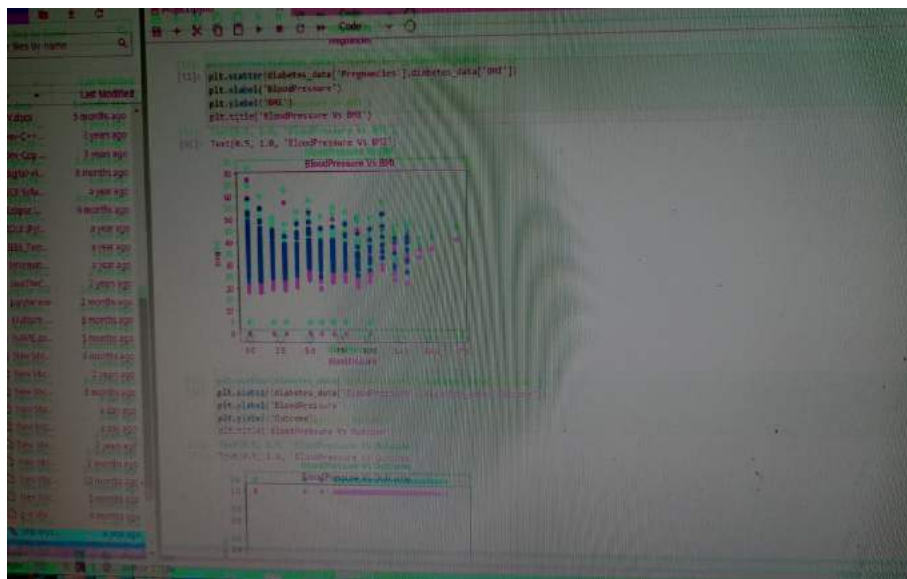
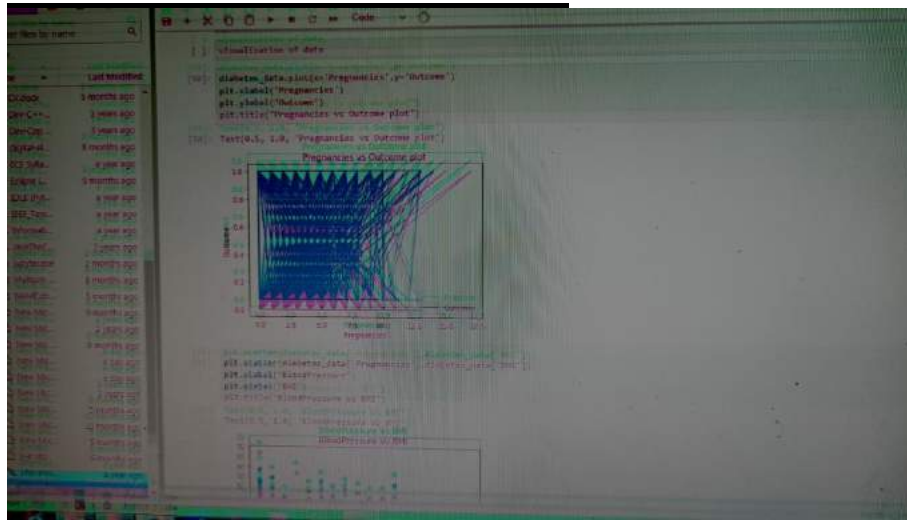
[47]: diabetes_data.head()
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
0           6         120             70              35         0   23.6              0.357                30         1
1           1          85             66              29         0   26.6              0.687                31         0
2           8         183             60              43         0   23.3              0.361                33         1
3           1          89             69              27         94  26.6              0.167                21         0
4           0         137             40              35         160  43.1              0.268                33         1

[48]: y = diabetes_data['Outcome']
y.head()
0      1
1      0
2      1
3      0
4      1
Name: Outcome, dtype: int64

[49]: visualization of data
[50]: visualization of data
File: C:\python\input\33-e1e2d1f0652d1", line 1
```

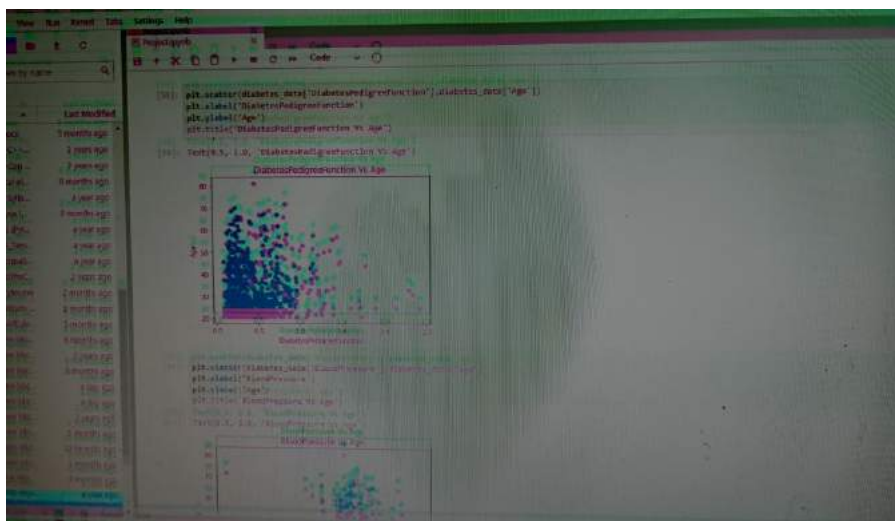
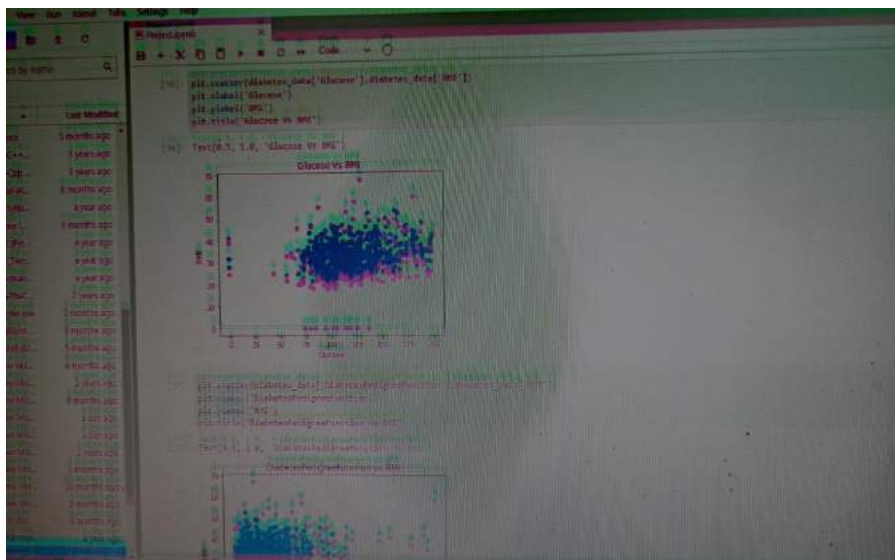
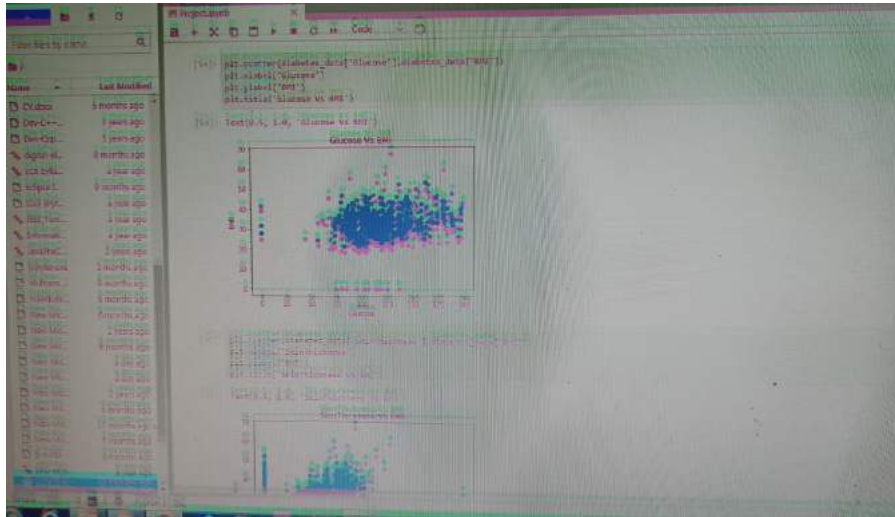


# VISUALIZATION OF DATA:

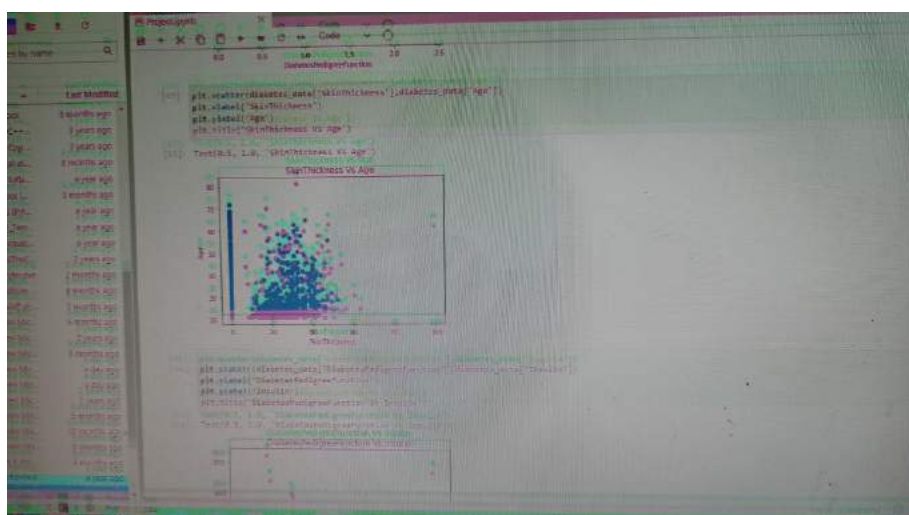
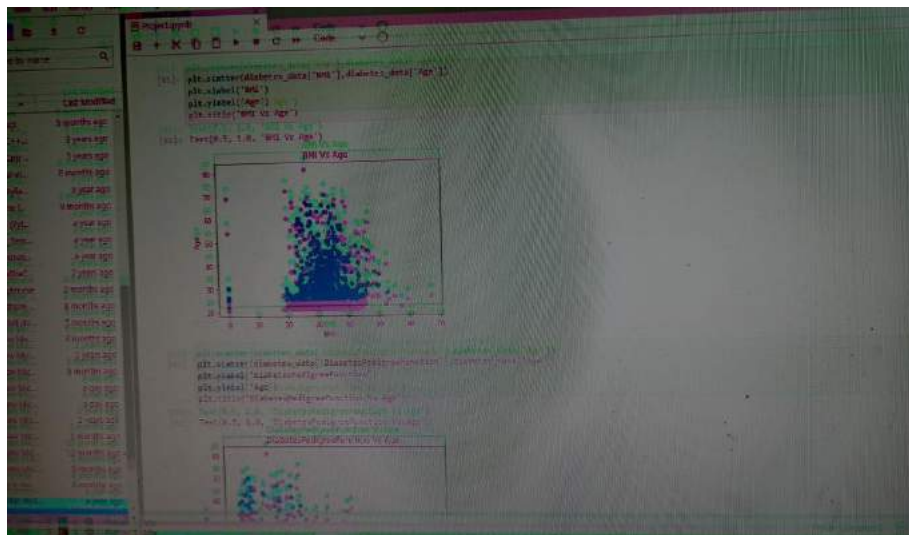
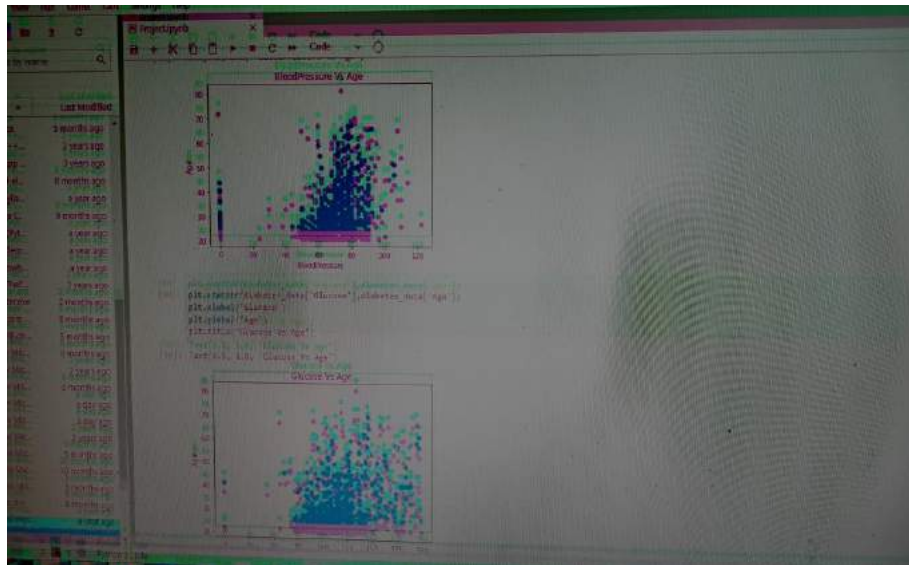




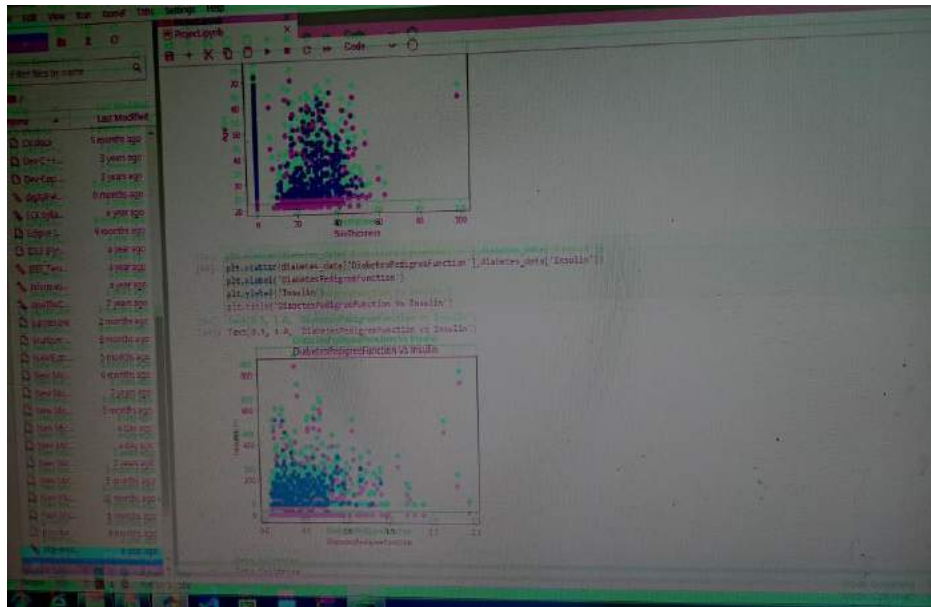
# VISUALIZATION OF DATA:



# VISUALIZATION OF DATA:



## VISUALIZATION OF DATA:

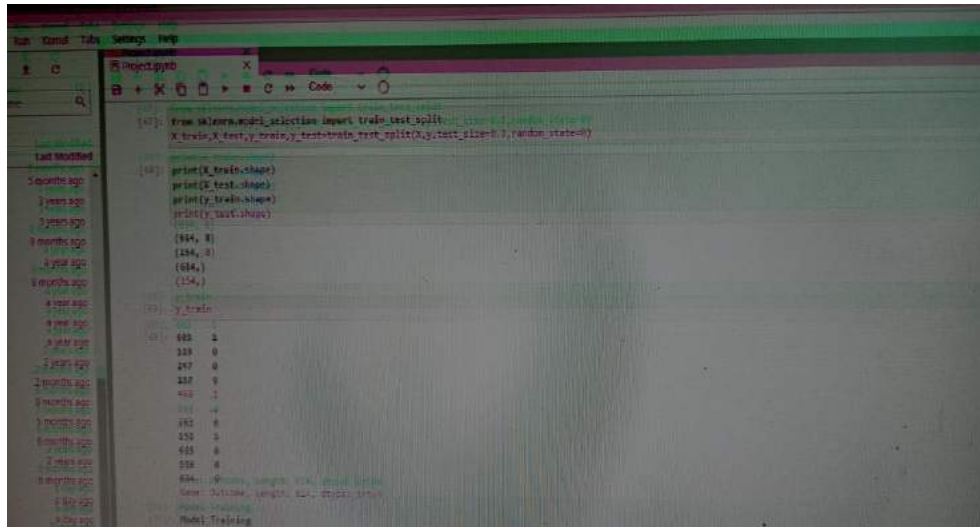


## SPLITTING THE DATA:

diabetes\_data.head

	diabetes_data.head	diabetes_data.head	diabetes_data.head	diabetes_data.head	diabetes_data.head	diabetes_data.head	diabetes_data.head	diabetes_data.head	diabetes_data.head
0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10
2	3	4	5	6	7	8	9	10	11
3	4	5	6	7	8	9	10	11	12
4	5	6	7	8	9	10	11	12	13
5	6	7	8	9	10	11	12	13	14
6	7	8	9	10	11	12	13	14	15
7	8	9	10	11	12	13	14	15	16
8	9	10	11	12	13	14	15	16	17
9	10	11	12	13	14	15	16	17	18
10	11	12	13	14	15	16	17	18	19
11	12	13	14	15	16	17	18	19	20
12	13	14	15	16	17	18	19	20	21
13	14	15	16	17	18	19	20	21	22
14	15	16	17	18	19	20	21	22	23
15	16	17	18	19	20	21	22	23	24
16	17	18	19	20	21	22	23	24	25
17	18	19	20	21	22	23	24	25	26
18	19	20	21	22	23	24	25	26	27
19	20	21	22	23	24	25	26	27	28
20	21	22	23	24	25	26	27	28	29
21	22	23	24	25	26	27	28	29	30
22	23	24	25	26	27	28	29	30	31
23	24	25	26	27	28	29	30	31	32
24	25	26	27	28	29	30	31	32	33
25	26	27	28	29	30	31	32	33	34
26	27	28	29	30	31	32	33	34	35
27	28	29	30	31	32	33	34	35	36
28	29	30	31	32	33	34	35	36	37
29	30	31	32	33	34	35	36	37	38
30	31	32	33	34	35	36	37	38	39
31	32	33	34	35	36	37	38	39	40
32	33	34	35	36	37	38	39	40	41
33	34	35	36	37	38	39	40	41	42
34	35	36	37	38	39	40	41	42	43
35	36	37	38	39	40	41	42	43	44
36	37	38	39	40	41	42	43	44	45
37	38	39	40	41	42	43	44	45	46
38	39	40	41	42	43	44	45	46	47
39	40	41	42	43	44	45	46	47	48
40	41	42	43	44	45	46	47	48	49
41	42	43	44	45	46	47	48	49	50
42	43	44	45	46	47	48	49	50	51
43	44	45	46	47	48	49	50	51	52
44	45	46	47	48	49	50	51	52	53
45	46	47	48	49	50	51	52	53	54
46	47	48	49	50	51	52	53	54	55
47	48	49	50	51	52	53	54	55	56
48	49	50	51	52	53	54	55	56	57
49	50	51	52	53	54	55	56	57	58
50	51	52	53	54	55	56	57	58	59
51	52	53	54	55	56	57	58	59	60
52	53	54	55	56	57	58	59	60	61
53	54	55	56	57	58	59	60	61	62
54	55	56	57	58	59	60	61	62	63
55	56	57	58	59	60	61	62	63	64
56	57	58	59	60	61	62	63	64	65
57	58	59	60	61	62	63	64	65	66
58	59	60	61	62	63	64	65	66	67
59	60	61	62	63	64	65	66	67	68
60	61	62	63	64	65	66	67	68	69
61	62	63	64	65	66	67	68	69	70
62	63	64	65	66	67	68	69	70	71
63	64	65	66	67	68	69	70	71	72
64	65	66	67	68	69	70	71	72	73
65	66	67	68	69	70	71	72	73	74
66	67	68	69	70	71	72	73	74	75
67	68	69	70	71	72	73	74	75	76
68	69	70	71	72	73	74	75	76	77
69	70	71	72	73	74	75	76	77	78
70	71	72	73	74	75	76	77	78	79
71	72	73	74	75	76	77	78	79	80
72	73	74	75	76	77	78	79	80	81
73	74	75	76	77	78	79	80	81	82
74	75	76	77	78	79	80	81	82	83
75	76	77	78	79	80	81	82	83	84
76	77	78	79	80	81	82	83	84	85
77	78	79	80	81	82	83	84	85	86
78	79	80	81	82	83	84	85	86	87
79	80	81	82	83	84	85	86	87	88
80	81	82	83	84	85	86	87	88	89
81	82	83	84	85	86	87	88	89	90
82	83	84	85	86	87	88	89	90	91
83	84	85	86	87	88	89	90	91	92
84	85	86	87	88	89	90	91	92	93
85	86	87	88	89	90	91	92	93	94
86	87	88	89	90	91	92	93	94	95
87	88	89	90	91	92	93	94	95	96
88	89	90	91	92	93	94	95	96	97
89	90	91	92	93	94	95	96	97	98
90	91	92	93	94	95	96	97	98	99

## SPLITTING THE DATA:



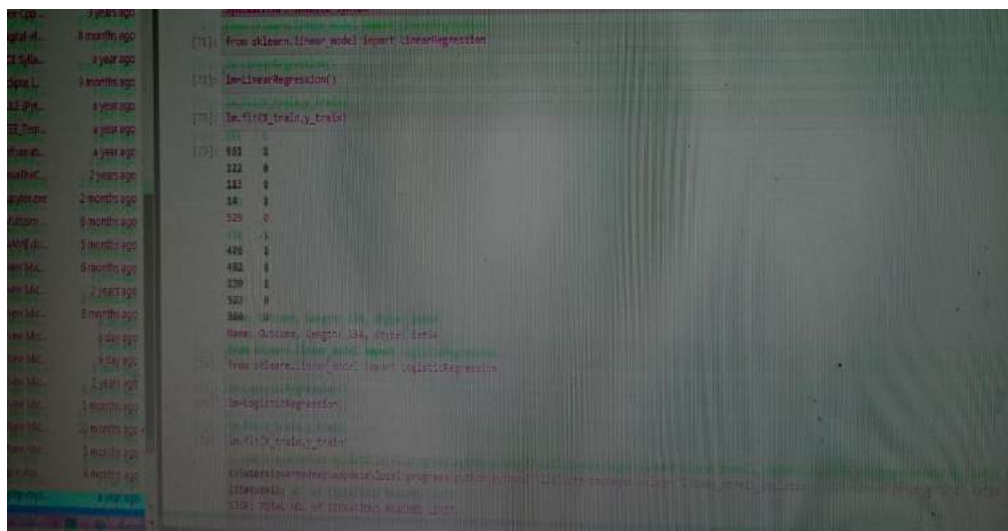
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

(934, 8)
(334, 8)
(934,)
(334,)
```

Output: (934, 8), (334, 8), (934,), (334,)

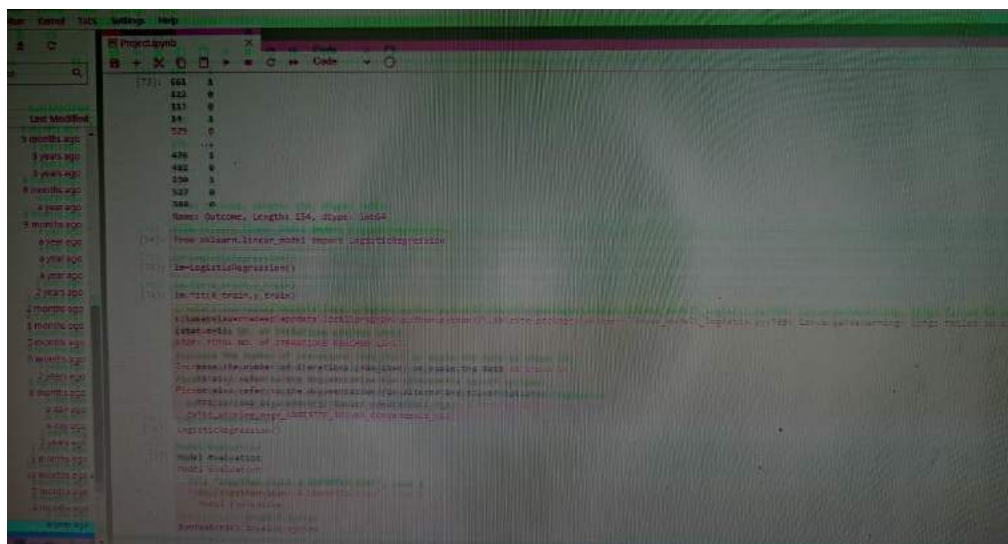
## MODEL TRAINING:



```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, y_train)

(934, 8)
(334, 8)
(934,)
(334,)
```

Output: (934, 8), (334, 8), (934,), (334,)



```
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression()
logit.fit(X_train, y_train)

(934, 8)
(334, 8)
(934,)
(334,)
```

Output: (934, 8), (334, 8), (934,), (334,)



[illegible][illegible][illegible]

## **FUTURE SCOPE**

Proposed system uses “Logistic regression” to find the diabetes disease, in data science we have many algorithms for classification such as Naive Bayes, SVM, Decision Tree, ID3 etc... in future we can add more algorithms to find outputs and algorithms can be compared to find the efficient algorithm. We can add visitor query module, where visitors can post queries to administrator and admin can send reply to those queries. We can add treatment module, where doctors upload treatment details for patients and patient can view those treatment details.

## **SUMMARY**

Logistic Regression is a powerful Machine Learning tool, and we can use it successfully for predicting categorical outputs of biomedical data. Data wrangling and data mining can benefit from excellent performances offered by Python and its libraries so well supported by the community. Linear Algebra programming has intrinsic advantages in avoiding, where possible, ‘while’ and ‘for’ loops. It is implementable by NumPy, a package that vectorizes the matrixes. NumPy makes working on them more comfortable, and guarantees better control over the operations, especially for large arrays.

Moreover, the Machine Learning scenario with Python is enriched by the presence of many powerful packages (i.e., Scikit-learn), which provide excellently optimized classifications and predictions on data.

The Data Set, with its patients and features, offers an exhaustive assortment of parameters for classification and for this reason represents a perfect example for Machine Learning applications. Anyway, many of these features seem to be redundant, and a definite impact on classification and prediction by some of them remains still unknown.

## CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt %matplotlib inline
import seaborn as sns

from sklearn.linear_model
import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report

diabetes_data=pd.read_csv('C:\\Users\\swarnadeep\\Desktop\\V
at traning on python on ML and DA\\diabetes2.csv')

diabetes_data.head

diabetes_data

diabetes_data.head(n=11)

diabetes_data.info()

sns.countplot(x='BMI',data=diabetes_data)
```



```

diabetes_data.isnull().sum()
diabetes_data.isna().sum()
sns.heatmap(diabetes_data.isnull())
diabetes_data.head(2)
sns.countplot(x='Outcome',data=diabetes_data)
print("The total number of diabetic patient is
",len(diabetes_data.index))
diabetes_data.drop('Pregnancies',axis=1,inplace = True)
diabetes_data.head(n=1)
diabetes_data.SkinThickness.unique()

diabetes_data=pd.read_csv('C:\\Users\\swarnadeep\\Desktop\\V
at training on python on ML and DA\\diabetes2.csv')
diabetes_data

diabetes_data.describe()

print("Total : ", diabetes_data[diabetes_data.BloodPressure ==
0].shape[0])

print(diabetes_data[diabetes_data.BloodPressure ==
0].groupby('Outcome')['Age'].count())

```

```
print("Total : ", diabetes_data[diabetes_data.Glucose ==
0].shape[0])

print(diabetes_data[diabetes_data.Glucose ==
0].groupby('Outcome')['Age'].count())

print("Total : ", diabetes_data[diabetes_data.SkinThickness
== 0].shape[0])

print(diabetes_data[diabetes_data.SkinThickness ==
0].groupby('Outcome')['Age'].count())

print("Total : ", diabetes_data[diabetes_data.BMI == 0].shape[0])

print(diabetes_data[diabetes_data.BMI ==
0].groupby('Outcome')['Age'].count())

print("Total : ", diabetes_data[diabetes_data.Insulin ==
0].shape[0])

print(diabetes_data[diabetes_data.Insulin ==
0].groupby('Outcome')['Age'].count())
```

### Dependent and independent of data

```
y = diabetes_data['Outcome']
X.drop(dropColumns,axis=1,inplace=True)
X.head()
diabetes_data.head()
diabetes_data.head(0)
y = diabetes_data['Outcome']
```

```
y.head()
```

## Visualization of data

```
diabetes_data.plot(x='Pregnancies',y='Outcome')
```

```
plt.xlabel('Pregnancies')
```

```
plt.ylabel('Outcome')
```

```
plt.title("Pregnancies vs Outcome plot")
```

```
plt.scatter(diabetes_data['Pregnancies'],diabetes_data['BMI'])
```

```
plt.xlabel('BloodPressure')
```

```
plt.ylabel('BMI')
```

```
plt.title('BloodPressure Vs BMI')
```

```
plt.scatter(diabetes_data['BloodPressure'],diabetes_data['Outcome'])
```

```
plt.xlabel('BloodPressure')
```

```
plt.ylabel('Outcome')
```

```
plt.title('BloodPressure Vs Outcome')
```

```
plt.scatter(diabetes_data['BloodPressure'],diabetes_data['BMI'])
```

```
plt.xlabel('BloodPressure')
```

```
plt.ylabel('BMI')
```

```
plt.title('BloodPressure Vs BMI')
```

```
plt.scatter(diabetes_data['Glucose'],diabetes_data['BMI'])
```

```
plt.xlabel('Glucose')
```

```
plt.ylabel('BMI')
```

```
plt.title('Glucose Vs BMI')
```

```
plt.scatter(diabetes_data['SkinThickness'],diabetes_data['BMI'])
```

```
plt.xlabel('SkinThickness')
```

```
plt.ylabel('BMI')
```

```
plt.title('SkinThickness Vs BMI')
```

```
plt.scatter(diabetes_data['Glucose'],diabetes_data['BMI'])
```

```
plt.xlabel('Glucose')
```

```
plt.ylabel('BMI')
```

```
plt.title('Glucose Vs BMI')
```

```
plt.scatter(diabetes_data['DiabetesPedigreeFunction'],diabetes_data['BMI'])
```

```
plt.xlabel('DiabetesPedigreeFunction')
```

```
plt.ylabel('BMI')
```

```
plt.title('DiabetesPedigreeFunction Vs BMI')
```

```
plt.scatter(diabetes_data['DiabetesPedigreeFunction'],diabetes_data['Age'])
```

```
plt.xlabel('DiabetesPedigreeFunction')
```

```
plt.ylabel('Age')
```

```
plt.title('DiabetesPedigreeFunction Vs Age')
```

```
plt.scatter(diabetes_data['BloodPressure'],diabetes_data['Age'])
```

```
plt.xlabel('BloodPressure')
```

```
plt.ylabel('Age')
```

```
plt.title('BloodPressure Vs Age')
```

```
plt.scatter(diabetes_data['Glucose'],diabetes_data['Age'])
```

```
plt.xlabel('Glucose')
```

```
plt.ylabel('Age')
```

```
plt.title('Glucose Vs Age')
```

```
plt.scatter(diabetes_data['BMI'],diabetes_data['Age'])
```

```
plt.xlabel('BMI')
```

```
plt.ylabel('Age')
```

```
plt.title('BMI Vs Age')
```

```
plt.scatter(diabetes_data['DiabetesPedigreeFunction'],diabetes_data['Age'])
```

```
plt.xlabel('DiabetesPedigreeFunction')
```

```
plt.ylabel('Age')
```

```
plt.title('DiabetesPedigreeFunction Vs Age')
```

```
plt.scatter(diabetes_data['SkinThickness'],diabetes_data['Age'])
```

```
plt.xlabel('SkinThickness')
```

```
plt.ylabel('Age')
```

```
plt.title('SkinThickness Vs Age')
```

```
plt.scatter(diabetes_data['DiabetesPedigreeFunction'],diabetes_data['Insulin'])
```

```
plt.xlabel('DiabetesPedigreeFunction')
```

```
plt.ylabel('Insulin')
```

```
plt.title('DiabetesPedigreeFunction Vs Insulin')
```

### Data splitting

```
diabetes_data.head  
  
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,random_state=0)  
  
print(X_train.shape)  
print(X_test.shape)  
print(y_train.shape)  
print(y_test.shape)  
  
y_train  
  
from sklearn.metrics import confusion_matrix, accuracy_score  
print(confusion_matrix(y_test,y_pred))  
print(accuracy_score(y_test,y_pred))
```

### Model training

```
[ from sklearn.linear_model import LinearRegression  
lm=LinearRegression()  
lm.fit(X_train,y_train) ] For linear regression  
  
from sklearn.linear_model import LogisticRegression  
lm=LogisticRegression()  
lm.fit(X_train,y_train)
```

## Data prediction/model evaluation

```
y_pred=lm.predict(X_test)
```

```
y_pred
```

```
X_test
```

```
df=pd.DataFrame({'Actual':y_test,'MC predicted':y_pred})
```

```
df
```

```
test1=X_test.head(1)
```

```
test1
```

```
pred1=lm.predict(test1)
```

```
if pred1==1:
```

```
    print("diabetic-- +ve")
```

```
else:
```

```
    print("diabetic -----ve")
```

```
pred1
```

```
from sklearn.metrics import accuracy_score
```

```
print("So my model accuracy is =  
",accuracy_score(y_test,y_pred)*100)
```



## Data prediction/model evaluation

```
y_pred=lm.predict(X_test)
y_pred
X_test
df=pd.DataFrame({'Actual':y_test,'MC predicted':y_pred})
df
test1=X_test.head(1)
test1
pred1=lm.predict(test1)
if pred1==1:
    print("diabetic-- +ve")
else:
    print("diabetic -----ve")
pred1
from sklearn.metrics import accuracy_score
print("So my model accuracy is =
",accuracy_score(y_test,y_pred)*100)
```