# AI Based Diabetes Prediction System

For AI based Diabetes Prediction system we use the Logistic Regression  machine learning algorithm for implement the code.

## Algorithm (Logistic Regression):

```
# === libraries ===
import numpy as np
# === sigmoid === #
description: the S shape function which gives us one or zero.
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
# === Logistic Regression ===
class LogisticRegression():
    def __init__(self, lr=0.001, n_iters=1000):
        self.lr = lr
        self.n_iters = n_iters
        self.weights = None
        self.bias = None
    # X ===> Training inputs samples
    # y ===> Target values
    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0
        # this is gradient descent
    for _ in range(self.n_iters):
        linear_pred = np.dot(X, self.weights) + self.bias
        predictions = sigmoid(linear_pred)
        # formula: dw = (1 / n_samples) * X.T * (predictions - y)

        # formula: db = (1 / n_samples) * sum(predictions - y)
        dw = (1 / n_samples) * np.dot(X.T, (predictions - y))
        db = (1 / n_samples) * np.sum(predictions - y)
        self.weights = self.weights - self.lr * dw
        self.bias = self.bias - self.lr * db

    # labeling the data
    def predict(self, X):
        linear_pred = np.dot(X, self.weights) + self.bias
        y_pred = sigmoid(linear_pred)
        class_pred = [0 if y <= 0.5 else 1 for y in y_pred]
        return class_pred
```

## Training the Model:

### Code:

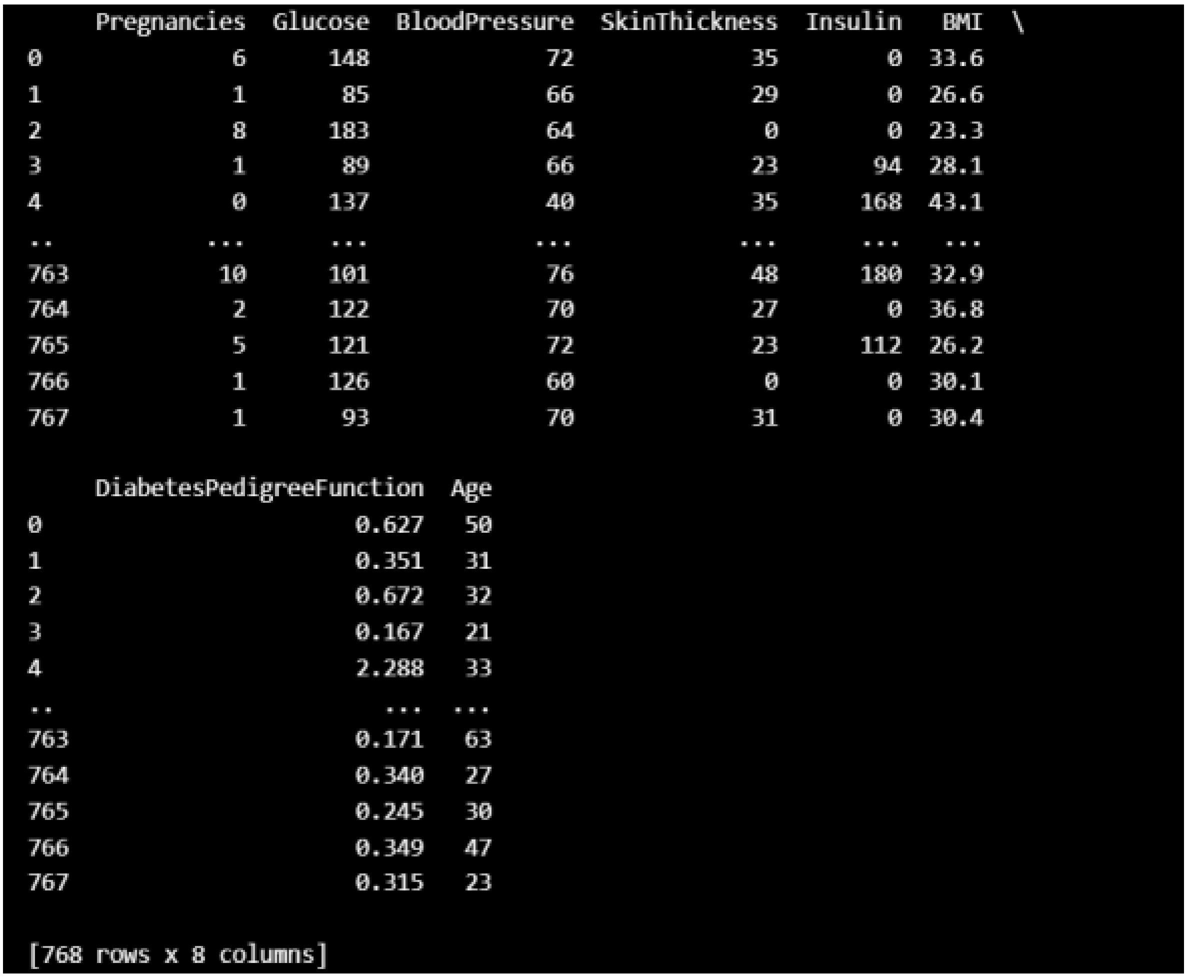
```
 import numpy as np
 from sklearn.model_selection import train_test_split
```

```
from logisticRegression import LogisticRegression
import pandas as pd
```

**Code:**

```
learn_d = pd.read_csv("diabetes.csv")
X = learn_d.drop('Outcome', axis=1)
# Assuming 'Outcome' is the diabetes label
y = learn_d['Outcome']
print(X)
```

**Output :**

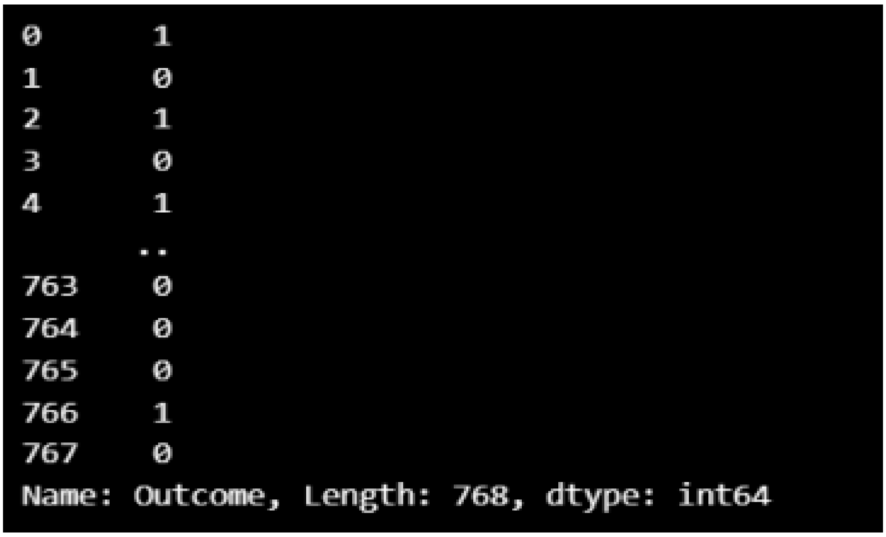


```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

## Code:

```
print(y)
```

**Output:**

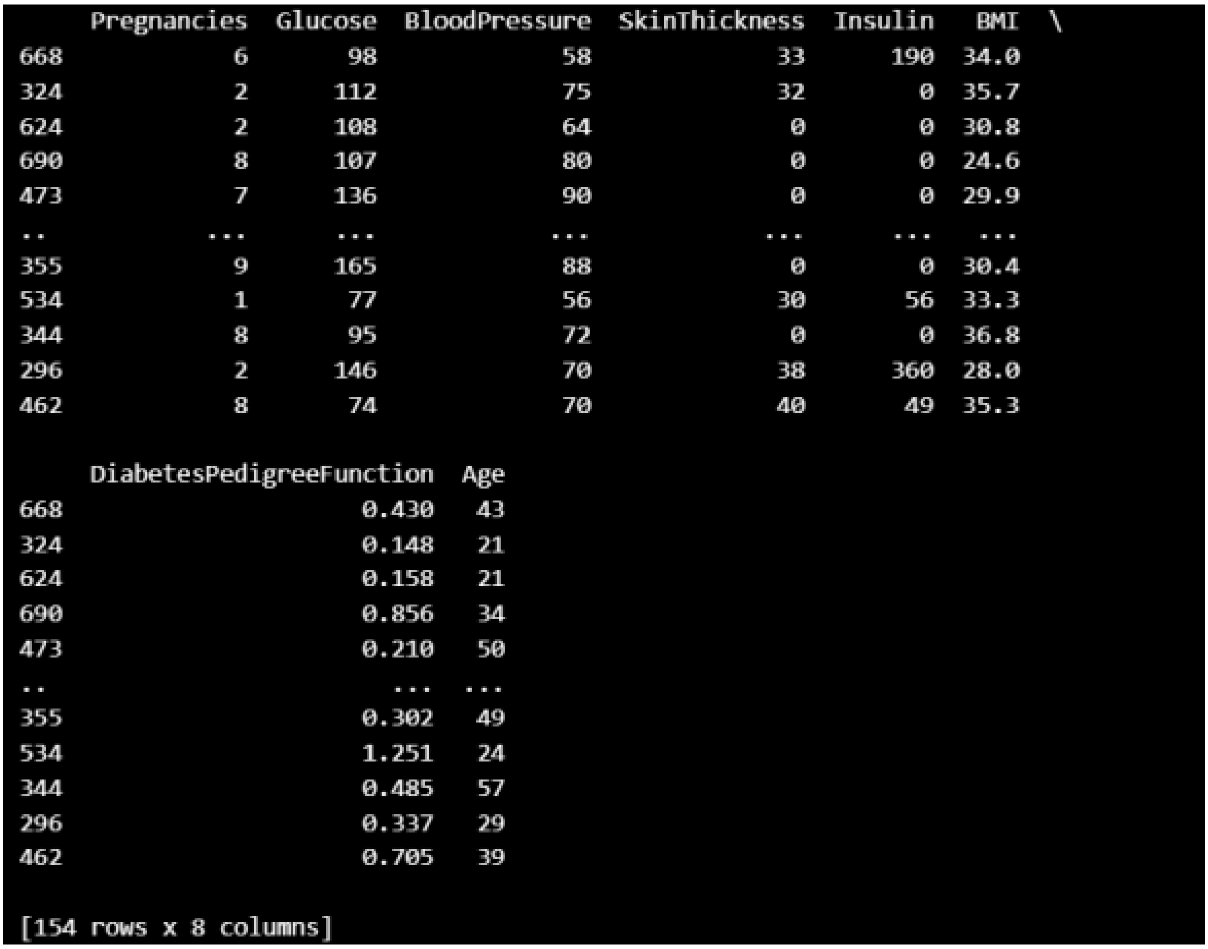


```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

## Code:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
print(X_test)
```

## Output :

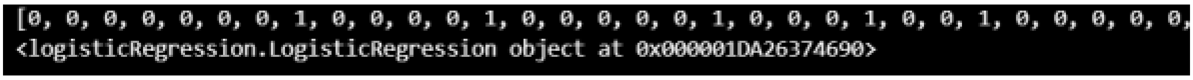


```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
668            6       98             58             33      190  34.0
324            2      112             75             32        0  35.7
624            2      108             64              0        0  30.8
690            8      107             80              0        0  24.6
473            7      136             90              0        0  29.9
..           ...      ...            ...            ...      ...   ...
355            9      165             88              0        0  30.4
534            1       77             56             30       56  33.3
344            8       95             72              0        0  36.8
296            2      146             70             38      360  28.0
462            8       74             70             40       49  35.3

     DiabetesPedigreeFunction  Age
668                     0.430   43
324                     0.148   21
624                     0.158   21
690                     0.856   34
473                     0.210   50
..                        ...  ...
355                     0.302   49
534                     1.251   24
344                     0.485   57
296                     0.337   29
462                     0.705   39

[154 rows x 8 columns]
```

## Code:

```
LR = LogisticRegression(lr = 0.0001, n_iters= 100000)
LR.fit(X_train, y_train)

Y_pred = LR.predict(X_test)
print(Y_prediction)
```

**Output :**

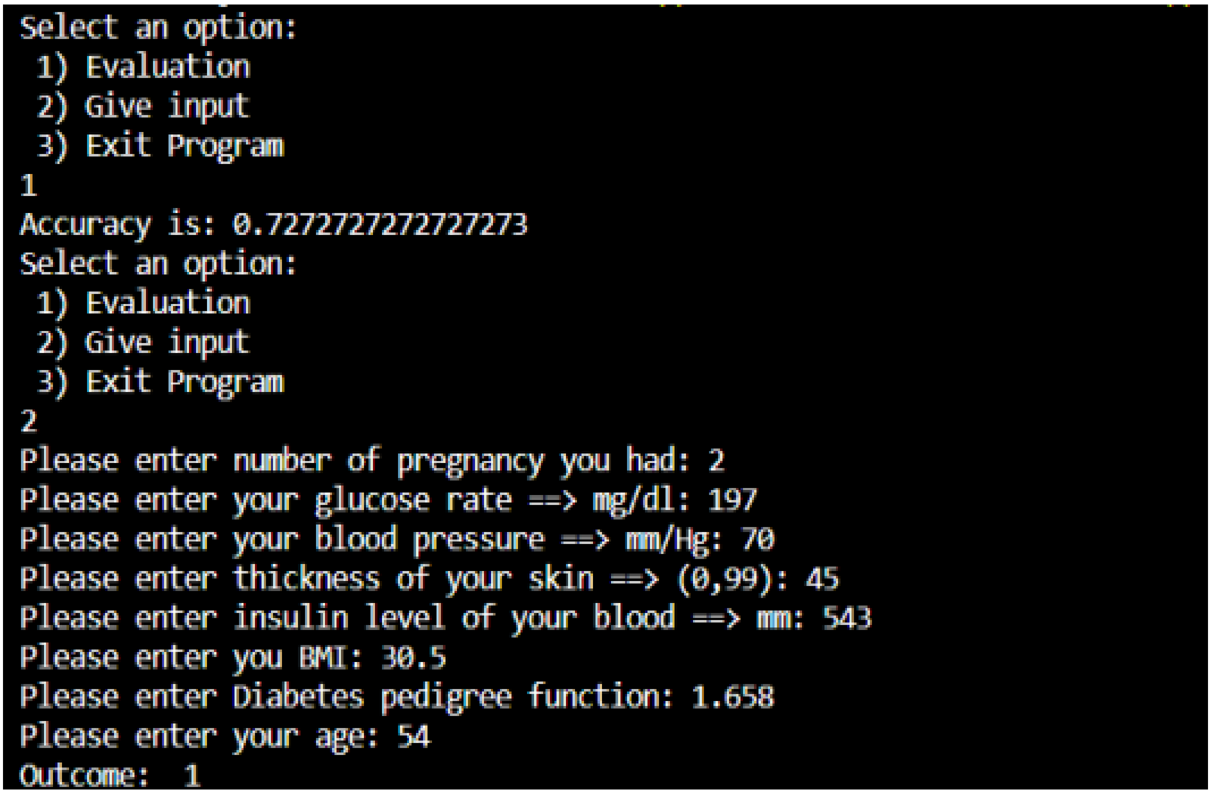


```
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
<logisticRegression.LogisticRegression object at 0x000001DA26374690>
```

## Code:

```
def accuracy(y_pred, y_test):
    return np.sum(y_pred == y_test)/len(y_test)


# result = accuracy(Y_prediction, Y_test)
# print(result)


# === main ===
def new_func():
    option = int(input("select the option:"))
    return option
while(True):
    print("Select an option: \n 1) Evaluation\n 2) Give input\n 3) Exit Program")
    option = new_func()


    if(option == 1):
        acc = accuracy(y_test, Y_pred)
        print("Accuracy is:",acc)
     elif(option == 2):
        pregnancies = float(input("Please enter number of
        pregnancy you had: "))

        glucose = float(input("Please enter your glucose rate ==>
        mg/dl: "))

        bloodPressure = float(input("Please enter your blood
        pressure ==> mm/Hg: "))

        skinThickness = float(input("Please enter thickness of your
        skin ==> (0,99): "))

        insulin = float(input("Please enter insulin level of your
        blood ==> mm: "))

        bmi = float(input("Please enter you BMI: "))

        diabetesPedigreeFunction = float(input("Please enter
        Diabetes pedigree function: ")) age = float(input("Please
        enter your age: "))

        x_input = [[pregnancies, glucose, bloodPressure,
        skinThickness, insulin, bmi, diabetesPedigreeFunction,
        age]]

        prob = LR.predict(x_input)

        print("Outcome: ", prob[0])


    elif(option == 3):

        print("exit")

        break
```

## Output :



```
Select an option:
 1) Evaluation
 2) Give input
 3) Exit Program
1
Accuracy is: 0.7272727272727273
Select an option:
 1) Evaluation
 2) Give input
 3) Exit Program
2
Please enter number of pregnancy you had: 2
Please enter your glucose rate ==> mg/dl: 197
Please enter your blood pressure ==> mm/Hg: 70
Please enter thickness of your skin ==> (0,99): 45
Please enter insulin level of your blood ==> mm: 543
Please enter you BMI: 30.5
Please enter Diabetes pedigree function: 1.658
Please enter your age: 54
Outcome:  1
```