ASSIGNMENT-11

CODE:

```
# Import required libraries
import pandas as pd
import numpy as np

# Load the diabetes dataset
diabetes_df = pd.read_csv('diabetes.csv')

# Split the dataset into input features and target variable
X = diabetes_df.drop('Outcome', axis=1).values
y = diabetes_df['Outcome'].values

# Define a function to train a logistic regression model on the
dataset
def train_model(X, y):
    # Initialize weights and bias to zeros
    w = np.zeros(X.shape[1])
    b = 0

    # Define the sigmoid function
    def sigmoid(z):
       return 1 / (1 + np.exp(-z))

    # Define the loss function (cross-entropy)
    def loss(X, y, w, b):
       z = np.dot(X, w) + b
       y_pred = sigmoid(z)
       return -(np.sum(y*np.log(y_pred) + (1-y)*np.log(1-y_pred))) /
X.shape[0]

    # Define the gradient of the loss function with respect to
weights and bias
    def gradient(X, y, w, b):
       z = np.dot(X, w) + b
       y_pred = sigmoid(z)
       dw = np.dot(X.T, (y_pred - y)) / X.shape[0]
       db = np.sum(y_pred - y) / X.shape[0]
       return dw, db

    # Train the model using batch gradient descent
    alpha = 0.01
    num_iterations = 10000
    for i in range(num_iterations):
       dw, db = gradient(X, y, w, b)
       w -= alpha * dw
       b -= alpha * db
       if i % 1000 == 0:
           print("Iteration: {}, Loss: {}".format(i, loss(X, y, w,
b)))

   # Return the trained weights and bias
   return w, b

# Train a logistic regression model on the diabetes dataset
w, b = train_model(X, y)

# Define a function to take user inputs and predict diabetes
def predict_diabetes():
    # Take user inputs
    preg = float(input("Enter the number of times the person has been
pregnant: "))
    glucose = float(input("Enter the person's glucose level: "))
    bp = float(input("Enter the person's blood pressure: "))
    skin = float(input("Enter the person's skin thickness: "))
    insulin = float(input("Enter the person's insulin level: "))
    bmi = float(input("Enter the person's BMI: "))
    dpf = float(input("Enter the person's Diabetes Pedigree Function:
"))
    age = float(input("Enter the person's age: "))

    # Create a numpy array of the inputs
    input_data = np.array([preg, glucose, bp, skin, insulin,bmi,dpf,
age])

    # Use the trained model to make a prediction
    z = np.dot(input_data, w) + b
    prediction = 1 / (1 + np.exp(-z))

    # Print the prediction
    if prediction < 0.5:
       print("The person is not likely to have diabetes.")
    else:
       print("The person is likely to have diabetes.")

# Call the predict_diabetes function to take inputs and predict
diabetes
predict_diabetes()
```

OUTPUT:

```
Iteration: 0, Loss: 10.775606658350837
Iteration: 1000, Loss: nan
Iteration: 2000, Loss: nan
Iteration: 3000, Loss: 31.876076967118546
Iteration: 4000, Loss: nan
<ipython-input-4-596598af5407>:22: RuntimeWarning: divide by zero
encountered in log
  return -(np.sum(y*np.log(y_pred) + (1-y)*np.log(1-y_pred))) /
X.shape[0]
<ipython-input-4-596598af5407>:22: RuntimeWarning: invalid value
encountered in multiply
  return -(np.sum(y*np.log(y_pred) + (1-y)*np.log(1-y_pred))) /
X.shape[0]
Iteration: 5000, Loss: nan
Iteration: 6000, Loss: nan
Iteration: 7000, Loss: nan
Iteration: 8000, Loss: nan
Iteration: 9000, Loss: nan
Enter the number of times the person has been pregnant: 2
Enter the person's glucose level: 10
Enter the person's blood pressure: 97
Enter the person's skin thickness: 22
Enter the person's insulin level: 107
Enter the person's BMI: 45
Enter the person's Diabetes Pedigree Function: 67
Enter the person's age: 34

The person is not likely to have diabetes.
```