COIMBATORE INSTITUTE OF TECHNOLOGY

PROBLEM STATEMENT - 1 : PREDICTION OF DIABETES

AIM:

Predict which patient has diabetes from Diabetes Database.csv and try to understand the dataset attributes and try to figure out type ML model suits and build from scratch.

DATASET:

The dataset diabetes.csv contains pregnancies, glucose , Age etc., and using this i'm going to predict the which person has most probability of the getting the diabetes

SOLUTION :

I have imported the functions like numpy, pandas, matplotlib.pyplot, seaborn, logisticregression and train_test_split. Then I have read the dataset and displayed the dataset. Then I have used the read() function for viewing the top five rows in the dataset and displayed the values of outcome attribute by using the above command. Then finally I have used logistic regression and I have calculated the cost and the test accuracy.

Here i got the test accuracy as 77%

OUTPUT:

diabetes_ques2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

+ Code + Text

RAM Disk Editing

..
sample_data
diabetes.csv

```
[8] import pandas as pd
    import numpy as np

    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split
```

```
df = pd.read_csv('diabetes.csv')
print(df)
```

```
     Pregnancies  Glucose  ...  Age  Outcome
0              6      148  ...   50        1
1              1       85  ...   31        0
2              8      183  ...   32        1
3              1       89  ...   21        0
4              0      137  ...   33        1
..           ...      ...  ...  ...      ...
763           10      101  ...   63        0
764            2      122  ...   27        0
765            5      121  ...   30        0
766            1      126  ...   47        1
767            1       93  ...   23        0

[768 rows x 9 columns]
```

diabetes_ques2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

+ Code + Text

RAM Disk Editing

..
sample_data
diabetes.csv

```
765            5      121  ...   30        0
[10] 766            1      126  ...   47        1
     767            1       93  ...   23        0

     [768 rows x 9 columns]
```

```
df.head()
```

	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

```
[12] y = df.Outcome.values
     y
```

```
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
```

Disk 69.63 GB available

diabetes_ques2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

..
sample_data
diabetes.csv

Disk 69.63 GB available

+ Code + Text

RAM Disk Editing

```
[12]        0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
            1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
            0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
            0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0,
            0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
            0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
            1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

```
[21] x_data = df.drop(['Outcome'], axis = 1)
```

```
[22] x = (x_data - np.min(x_data)) / (np.max(x_data) - np.min(x_data)).values
```

```
[23] x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state=0)
```

```
[24] x_train = x_train.T
     y_train = y_train.T
     x_test = x_test.T
     y_test = y_test.T
     def initialize(dimension):

         weight = np.full((dimension,1),0.01)
         bias = 0.0
         return weight,bias
     def sigmoid(z):

         y_head = 1/(1+ np.exp(-z))
         return y_head
     def forwardBackward(weight,bias,x_train,y_train):
         y_head = sigmoid(np.dot(weight.T,x_train) + bias)
         loss = -(y_train*np.log(y_head) + (1-y_train)*np.log(1-y_head))
         cost = np.sum(loss) / x_train.shape[1]

         derivative_weight = np.dot(x_train,((y_head-y_train).T))/x_train.shape[1]
         derivative_bias = np.sum(y_head-y_train)/x_train.shape[1]
         gradients = {"Derivative Weight" : derivative_weight, "Derivative Bias" : derivative_bias}
```

diabetes_ques2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

.. sample_data diabetes.csv

+ Code + Text

RAM Disk Editing

```
[24]  def update(weight,bias,x_train,y_train,learningRate,iteration) :
          costList = []
          index = []

          #for each iteration, update weight and bias values
          for i in range(iteration):
              cost,gradients = forwardBackward(weight,bias,x_train,y_train)
              weight = weight - learningRate * gradients["Derivative Weight"]
              bias = bias - learningRate * gradients["Derivative Bias"]

              costList.append(cost)
              index.append(i)

          parameters = {"weight": weight,"bias": bias}

          print("iteration:",iteration)
          print("cost:",cost)

          plt.plot(index,costList)
          plt.xlabel("Number of Iteration")
          plt.ylabel("Cost")
          plt.show()

          return parameters, gradients
      def predict(weight,bias,x_test):
          z = np.dot(weight.T,x_test) + bias
          y_head = sigmoid(z)

          y_prediction = np.zeros((1,x_test.shape[1]))

          for i in range(y_head.shape[1]):
              if y_head[0,i] <= 0.5:
                  y_prediction[0,i] = 0
              else:
                  y_prediction[0,i] = 1
          return y_prediction
```

diabetes_ques2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

.. sample_data diabetes.csv

+ Code + Text

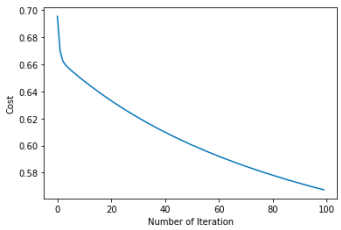
RAM Disk Editing

```
[24]          else:
                  y_prediction[0,i] = 1
          return y_prediction
```

```
[25] def logistic_regression(x_train,y_train,x_test,y_test,learningRate,iteration):
          dimension = x_train.shape[0]
          weight,bias = initialize(dimension)

          parameters, gradients = update(weight,bias,x_train,y_train,learningRate,iteration)

          y_prediction = predict(parameters["weight"],parameters["bias"],x_test)

          print("Manuel Test Accuracy: {:.2f}%".format((100 - np.mean(np.abs(y_prediction - y_test))*100)))
```

```
logistic_regression(x_train,y_train,x_test,y_test,1,100)
```

```
iteration: 100
cost: 0.567352562819793
```




```
Manuel Test Accuracy: 76.62%
```