TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

Central Department of Computer Science and Information Technology
Kirtipur, Kathmandu
2022

Lab Report: II
Machine Learning

Submitted By:
Sudhan Kandel
Semester: 1st
Roll no: 2

Submitted To:
MR. Arjun Singh Saud

1. **Write python programs to predict diabetes using logistic regression. Implement the algorithm using library and without using library.**

Using Library:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import math
from math import exp
df=pd.read_csv('Diabetes.csv')
independent_variable=df[['Pragnency', 'Glucose', 'Blod Pressure', 'Skin Thikness', 'Insulin',
    'BMI', 'DFP', 'Age']]
target=df['Diabetes']
x_train,x_test,y_train,y_test=train_test_split(independent_variable,target,train_size=0.7)
model=LogisticRegression()
model.fit(x_train,y_train)
predicted_value=model.predict(x_test)
```

OUTPUT:

Updated Coefficient:

```
[ 9.38517161e-02,  2.97177075e-02, -1.25419359e-02, -7.61877955e-03, -4.56
503428e-04,  9.95041767e-02,1.46393596e+00,  2.30949033e-02]
```

Classification Report:

```
                precision    recall  f1-score   support

           0       0.93      0.80      0.86       172
           1       0.58      0.81      0.68        59

    accuracy                           0.80       231
   macro avg       0.75      0.81      0.77       231
weighted avg       0.84      0.80      0.81       231
```

Without Using Library:

```
def coffiecient_bgd(train,l_rate,n_epoch):
    train=train.to_numpy()

    coef=[0.0 for i in range(len(train[0]))]
    for j in range(n_epoch):
```

```
        print("..............epoch"+str(j)+"..............")
        for i in train:


            z=coef[0]+coef[1]*i[0]+coef[2]*i[1]+coef[3]*i[2]+coef[4]*i[3]+coef[5]*i[4]+coe[6]*i[5]+coef[7]*i[6]+coef[8]*i[7]
            yhat=1/(1+exp(-z))
            coef[0]-=l_rate*(yhat-i[8])
            coef[1]-=l_rate*(yhat-i[8])*i[0]
            coef[2]-=l_rate*(yhat-i[8])*i[1]
            coef[3]-=l_rate*(yhat-i[8])*i[2]
            coef[4]-=l_rate*(yhat-i[8])*i[3]
            coef[5]-=l_rate*(yhat-i[8])*i[4]
            coef[6]-=l_rate*(yhat-i[8])*i[5]
            coef[7]-=l_rate*(yhat-i[8])*i[6]
            coef[8]-=l_rate*(yhat-i[8])*i[7]
            print("Updated Weights values are ",coef)
    return coef


def prediction(coef,test):
    test=test.to_numpy()
    predict=[]
    for i in test:
        y=coef[0]+coef[1]*i[0]+coef[2]*i[1]+coef[3]*i[2]+coef[4]*i[3]+coef[5]*i[4]+coef[6]*i[5]+coef[7]*i[6]+coef[8]*i[7]
        yhat=1/(1+exp(-y))
        if yhat>0.5:
            predict.append(1)
        else:
            predict.append(0)
    return predict
```

Output:

Weight update for every training data set

```
Updated Weights values are  [-0.46362362673893237, 0.9686254208866975, 0.2
79286552466961, -0.17143204292078668, 0.07841400892370243, 0.14310311657127017, 0.061760644282604554, 0.12268004677551507, -0.21578916356523187]
Updated Weights values are  [-0.4646236147810785, 0.9686254208866975, 0.17828776021020432, -0.2474311341238907, 0.07841400892370243, 0.14310311657127017, 0.02606107117798858, 0.12248204914317014, -0.24178885266103062]
Updated Weights values are  [-0.4654571487388707, 0.9677918869289053, 0.11077715096290391, -0.3091126470005108, 0.04423911665422374, 0.09559168097711687, -0.012531551067788568, 0.12156849592542993, -0.2684619393103798]
```

```
Updated Weights values are  [-0.46545715305035323, 0.9677918783059402, 0.1
1077104398892214, -0.3091129229353949, 0.04423911665422374, 0.095591680977
11687, -0.012531683861451554, 0.12156849524421567, -0.2684620298515137]
Updated Weights values are  [-0.4644571530573525, 0.9717918782779431, 0.22
577104318400554, -0.23711292343934265, 0.04423911665422374, 0.095591680977
11687, 0.016368315936269407, 0.12194449524158395, -0.22246203017348032]
Updated Weights values are  [-0.46445715305734847, 0.9717918782779796, 0.2
2577104318459487, -0.237112923438985, 0.04423911665436193, 0.0955916809777
8749, 0.016368315936392555, 0.12194449524158708, -0.2224620301732649]
Updated Weights values are  [-0.4644571530523866, 0.9717918783126983, 0.2
2577104397320547, -0.2371129231116372, 0.04423911665436193, 0.095591680977
78749, 0.01636831608717093, 0.12194449524348669, -0.22246202999471157]
Updated Weights values are  [-0.46445715302343155, 0.9717918785733123, 0.2
2577104872217088, -0.2371129208529829, 0.04423911665436193, 0.095591680977
78749, 0.016368317036964015, 0.12194449524777234, -0.22246202869164178]
```

Final Updated Coefficient:

```
[-0.46445715302343155, 0.9717918785733123, 0.22577104872217088, -0.2371129
208529829, 0.04423911665436193, 0.09559168097778749, 0.016368317036964015,
0.12194449524777234, -0.22246202869164178]
```

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.67	0.80	223
1	0.05	0.50	0.09	8
accuracy			0.67	231
macro avg	0.51	0.59	0.44	231
weighted avg	0.94	0.67	0.77	231

Conclusion:

Here I implement logistic regression from sketch and using library. And classification report by using library is better than manual calculation, this is because hyperparameters in library are tune d properly.

program link: https://github.com/sudhankandel/Machine-Learning-Lab/blob/main/lab2.ipynb