Lab 14 04-11-2022

November 3, 2022

Program on multi layered feedforward network using any standard dataset available in the public domain and find the accuracy of the algorithm

Ajay Jeevan Jose

1 Multi Layered Perceptron

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
```

1.1 Dataset loading

[2]:
```
df = pd.read_csv('diabetes.csv')
df.describe().transpose()
```

[2]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000			

```
Outcome                     0.0000     1.00000     1.00
```

[3]:
```
df.Outcome.value_counts()
```

[3]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

1.2 Data Preprocessing

1.2.1 Extracting independent and dependent variables

[4]:
```
y = df['Outcome']
print(y)
df = df.drop(['Outcome'],axis=1)
x = df
print(x)
```

```
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
     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]
```

1.2.2 Creating test&train dataset

[5]:
```
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)
```

```
(537, 8)
(231, 8)
```

1.2.3 Feature Scaling

[6]:
```
from sklearn.preprocessing import MinMaxScaler
st_x = MinMaxScaler()
x_train = st_x.fit_transform(x_train)
x_test = st_x.fit_transform(x_test)
print(x_train)
```

```
[[0.         0.76262626 0.73770492 … 0.62742176 0.12285959 0.        ]
 [0.05882353 0.41919192 0.55737705 … 0.27123696 0.23116438 0.1       ]
 [0.11764706 0.61616162 0.57377049 … 0.54843517 0.10958904 0.1       ]
 …
 [0.23529412 0.47474747 0.53278689 … 0.3681073  0.02739726 0.        ]
 [0.64705882 0.42929293 0.60655738 … 0.4485842  0.09246575 0.23333333]
 [0.29411765 0.68686869 0.67213115 … 0.         0.2380137  0.8       ]]
```

1.3 Fitting Dataset

[32]:
```
from sklearn.neural_network import MLPClassifier
classifier = MLPClassifier(hidden_layer_sizes=(8,8,8), activation='relu',␣
 ↪solver='adam', max_iter=600)
classifier.fit(x_train,y_train)
```

[32]:
```
MLPClassifier(hidden_layer_sizes=(8, 8, 8), max_iter=600)
```

[33]:
```
x_pred = classifier.predict(x_train)
y_pred = classifier.predict(x_test)
```

[34]:
```
print(y_test.values)
print(y_pred)
```

```
[1 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0
 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0
 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1
 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0
 1 1 0 0 1 1 0 0 0]
[1 0 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 1 1
 0 1 1 1 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0
 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1 0
 0 1 1 0 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0
 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 1
 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0
 1 1 1 1 1 0 0 0 1]
```

1.4 Visualisation

1.4.1 Creating CM

[35]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[124  33]
 [ 21  53]]
```

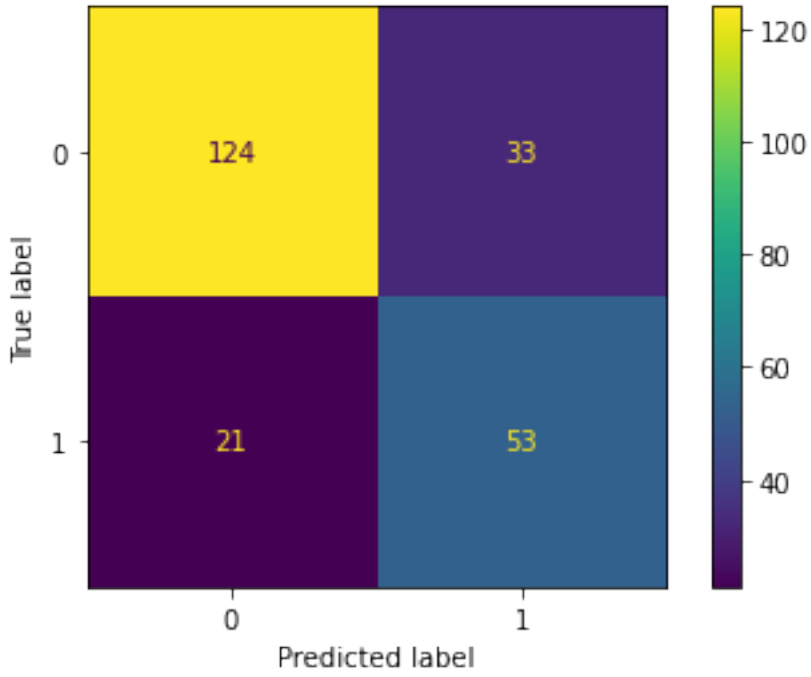
[36]:
```
from sklearn.metrics import confusion_matrix
cm1= confusion_matrix(y_train, x_pred)
print(cm1)
```

```
[[298  45]
 [ 68 126]]
```

1.4.2 CM Display

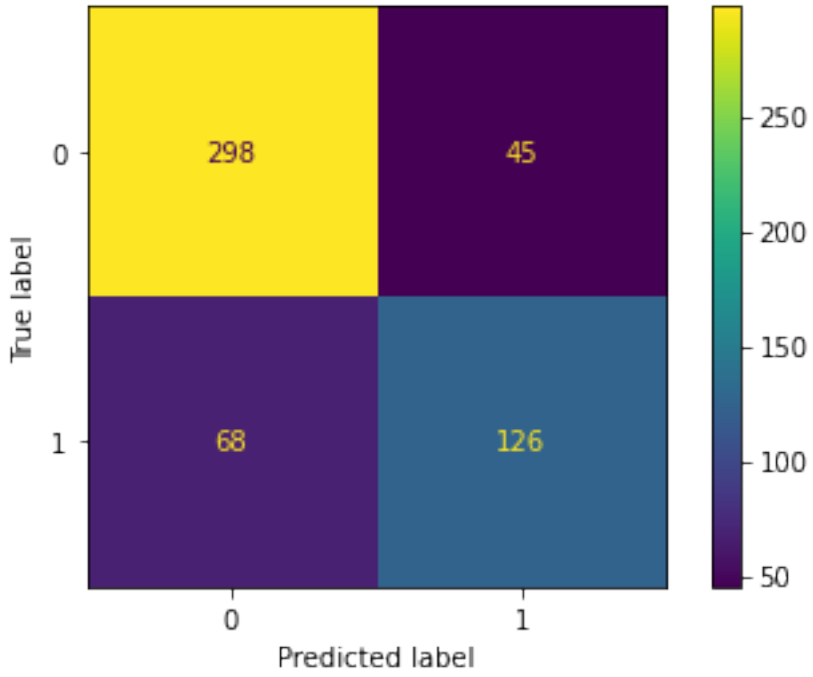
[37]:
```
from sklearn.metrics import ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=classifier.
 ↪classes_)
disp.plot()
```

[37]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f337e997dc0>
```



[39]:
```
from sklearn.metrics import ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=cm1,display_labels=classifier.
 ↪classes_)
disp.plot()
```

[39]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f3393889bb0>
```



1.5 Accuracy

[40]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
print('Accuracy score: {}'.format(accuracy_score(y_test, y_pred)))
print('Precision score: {}'.format(precision_score(y_test, y_pred,pos_label=0)))
print('Recall score: {}'.format(recall_score(y_test, y_pred,pos_label=0)))
print('F1 score: {}'.format(f1_score(y_test, y_pred,pos_label=0)))
```

```
Accuracy score: 0.7662337662337663
Precision score: 0.8551724137931035
Recall score: 0.7898089171974523
F1 score: 0.8211920529801324
```

[41]:
```
classifier.coefs_
```

[41]:
```
[array([[-1.21213208e-01, -4.01082274e-18,  2.61444638e-01,
          4.21342863e-01,  3.64567395e-02, -1.29308885e-02,
          4.54771789e-01, -4.37710733e-02],
        [ 7.72151607e-01, -2.13245404e-02, -3.28944740e-01,
         -1.40335526e-02, -5.97857284e-01, -6.63901486e-03,
          1.38611715e-01, -1.08389295e-01],
        [ 1.44503827e-01, -2.36426144e-02, -5.49842166e-01,
          4.27535120e-01,  3.56963120e-01, -2.86577891e-05,
```

```
         5.12870531e-01, -1.14161739e-01],
       [ 2.77244135e-01, -3.42383672e-04, -1.07354758e+00,
         5.13205073e-01,  9.24973436e-02,  1.76709400e-04,
        -4.13597924e-01, -2.63116509e-01],
       [-6.29573710e-02, -3.30104694e-08, -4.95954014e-01,
         4.34076676e-01,  1.84323043e-01, -4.33127418e-02,
         7.72026733e-01, -1.87279990e-01],
       [ 2.84249209e-01, -3.23524693e-02,  4.94886540e-01,
         2.68178817e-02, -5.24342595e-01, -6.75936458e-03,
         3.18724440e-02, -1.05803075e-02],
       [ 3.16646322e-01, -1.18246535e-04,  3.10433930e-01,
         5.44074077e-01, -1.72019953e-01,  1.02911919e-03,
        -1.55394322e-01,  2.42406490e-01],
       [ 2.31963776e-01,  1.66080345e-03,  4.55217029e-01,
         7.74063066e-01, -5.84626933e-01, -3.85304458e-07,
         2.89095870e-01,  8.37770411e-01]]),
array([[-2.45572083e-01,  4.33544940e-01, -7.90192192e-02,
         1.92293530e-01, -4.48944124e-01, -3.13151969e-04,
        -6.04041152e-01,  4.75295035e-01],
       [-1.11366856e-03,  1.76419836e-17, -2.80838864e-07,
        -1.71013817e-03, -1.09472810e-05, -2.24633514e-21,
        -1.24594203e-02, -1.08792840e-03],
       [ 6.78253234e-01, -1.00052208e+00,  7.06625981e-01,
        -5.05876908e-01, -2.64580750e-01, -7.82977643e-04,
         2.03238420e-01, -3.15098691e-01],
       [-1.68464132e-01, -4.15579174e-01, -3.79848940e-03,
        -2.44542852e-02, -8.36193979e-01,  3.19010734e-05,
        -4.40878410e-01, -6.19463163e-01],
       [ 9.53809670e-01, -2.35868773e-01,  1.00553747e+00,
        -1.01436435e+00,  3.20465034e-01,  1.37653090e-16,
         1.18572474e+00, -3.21066906e-01],
       [ 2.04303263e-03,  3.78856061e-10,  2.17574996e-03,
        -9.98724783e-04,  6.14783198e-22,  4.69648802e-02,
        -2.81791778e-07,  6.05913264e-03],
       [-3.29123870e-01,  4.44351764e-01,  8.84288108e-02,
        -5.30883677e-01, -5.37371832e-02, -5.34194880e-03,
        -1.31076712e-01, -4.54887990e-01],
       [-7.06255970e-01, -1.20553665e+00,  1.13347284e+00,
        -8.21274446e-01,  1.10213644e+00, -2.18172100e-09,
         7.22396805e-01,  4.95317227e-01]]),
array([[ 1.45537425e-02, -7.29392666e-01,  4.23039764e-01,
        -8.86776120e-04,  8.08813169e-01,  2.27874271e-01,
        -3.42497274e-02,  1.69954940e-22],
       [ 7.34182304e-01,  7.06554049e-01,  4.39333658e-02,
         6.74130630e-09, -5.21537757e-01,  3.35510787e-01,
         4.54447947e-01, -2.95224196e-09],
       [ 1.43305321e-02, -5.68300272e-01,  1.04569865e+00,
```

```
         -7.71073262e-09,  1.03390258e+00,  9.28514721e-01,
         -9.12764823e-01, -4.98247811e-03],
        [ 1.87946918e-01,  6.58283091e-01, -5.53004390e-01,
         -2.17010884e-04, -4.63881572e-01, -1.01223217e+00,
          2.53475322e-01, -6.21359769e-05],
        [-3.57407943e-01,  2.24009724e-01,  5.10720399e-01,
          9.05996606e-04, -2.81165812e-01,  7.47052635e-01,
          2.40698341e-01, -8.99785508e-23],
        [ 4.18524212e-02,  9.85007621e-10, -2.43252165e-03,
          1.37771883e-05, -1.91361019e-05, -3.16769025e-02,
         -6.09657564e-11,  2.85356799e-02],
        [ 1.37716601e-01, -6.93326058e-01,  6.71746003e-01,
         -2.20800987e-02,  1.72611011e-01,  3.33055865e-01,
         -4.26444172e-01,  3.38296331e-02],
        [-4.38214625e-02, -4.37238871e-01,  3.19166477e-01,
          3.00867676e-02,  1.07174210e-01, -1.04865130e-01,
         -4.38892010e-01,  1.68086514e-02]]),
 array([[ 5.19961774e-01],
        [ 1.06490310e+00],
        [-1.03215366e+00],
        [-3.56616773e-08],
        [-4.82777297e-01],
        [-8.92429118e-01],
        [ 5.55932150e-01],
        [-1.28268147e-03]])]
```

[42]:
```
https://drive.google.com/file/d/1BlPFuxHik3nRkF2plMc982HOojpXv3OJ/classifier.
↪intercepts_
```

[42]:
```
[array([-0.10589611, -0.02269364, -0.15621345, -0.44414801,  0.74041077,
        -0.47258228,  0.34872073, -0.14670669]),
 array([ 0.53276176,  0.24347454,  0.04586402,  0.75058861,  0.53345441,
        -0.51631305,  0.78200071, -0.17653702]),
 array([ 0.35796254,  0.60463265,  0.21077278, -0.42458932,  0.57897916,
         0.12702859,  0.13067793, -0.42122155]),
 array([-0.14614943])]
```