Experiment: 07

Neural Network Using Scikit Learn

In [1]:
```
print('-------------------------------EXPERIMENT-07-------------------
--------------')
print('NAME:  Pratyush Srivastava')
print('ROLL NO: 18SCSE1010128')
```

```
-------------------------------EXPERIMENT-07-------------------------
-------
NAME:  Pratyush Srivastava
ROLL NO: 18SCSE1010128
```

In [2]:
```
print('----------------------------Neural Network Using Scikit Learn--
---------------')
```

```
----------------------------Neural Network Using Scikit Learn---------
--------
```

In [3]:
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import sklearn

from sklearn.neural_network import MLPClassifier

from sklearn.neural_network import MLPRegressor
```

```
# Import necessary modules

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

from math import sqrt

from sklearn.metrics import r2_score
```

In [4]:
```
from sklearn import datasets
df = pd.read_csv('diabetes.csv')

print(df.shape)

df.describe().transpose()
```

```
(768, 9)
```

Out[4]:

	count	mean	std	min	25%	50%	75%
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000

In [5]:
```
target_column = ['Age']
```

```
predictors = list(set(list(df.columns))-set(target_column))

df[predictors] = df[predictors]/df[predictors].max()

df.describe().transpose()
```

Out[5]:

	count	mean	std	min	25%	50%	75%
Pregnancies	768.0	0.226180	0.198210	0.000000	0.058824	0.176471	0.35294
Glucose	768.0	0.607510	0.160666	0.000000	0.497487	0.587940	0.70477
BloodPressure	768.0	0.566438	0.158654	0.000000	0.508197	0.590164	0.65573
SkinThickness	768.0	0.207439	0.161134	0.000000	0.000000	0.232323	0.32323
Insulin	768.0	0.094326	0.136222	0.000000	0.000000	0.036052	0.15041
BMI	768.0	0.476790	0.117499	0.000000	0.406855	0.476900	0.54545
DiabetesPedigreeFunction	768.0	0.194990	0.136913	0.032231	0.100723	0.153926	0.25878
Age	768.0	33.240885	11.760232	21.000000	24.000000	29.000000	41.00000
Outcome	768.0	0.348958	0.476951	0.000000	0.000000	0.000000	1.00000

In [6]:
```
X = df[predictors].values

y = df[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
0, random_state=40)

print(X_train.shape); print(X_test.shape)
```

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

In [7]:
```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(8,8,8), activation='relu', solv
```

```
er='adam', max_iter=500)

mlp.fit(X_train,y_train)

predict_train = mlp.predict(X_train)

predict_test = mlp.predict(X_test)
```

```
C:\Users\praty\New folder\lib\site-packages\sklearn\utils\validation.p
y:73: DataConversionWarning: A column-vector y was passed when a 1d arr
ay was expected. Please change the shape of y to (n_samples, ), for exa
mple using ravel().
  return f(**kwargs)
C:\Users\praty\New folder\lib\site-packages\sklearn\neural_network\_mul
tilayer_perceptron.py:582: ConvergenceWarning: Stochastic Optimizer: Ma
ximum iterations (500) reached and the optimization hasn't converged ye
t.
  warnings.warn(
```

In [8]:
```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_train,predict_train))
print(classification_report(y_train,predict_train))
```

```
[[36  7  0 ...  0  0  0]
 [35  2  0 ...  0  0  0]
 [16  2  0 ...  0  0  0]
 ...
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 1  0  0 ...  0  0  0]]
              precision    recall  f1-score   support

          21       0.22      0.75      0.34        48
          22       0.05      0.04      0.05        48
          23       0.00      0.00      0.00        28
          24       0.14      0.15      0.14        39
          25       0.18      0.13      0.15        31
          26       0.00      0.00      0.00        27
          27       0.00      0.00      0.00        20
```

28	0.00	0.00	0.00	23
29	0.07	0.29	0.12	24
30	0.00	0.00	0.00	13
31	0.00	0.00	0.00	17
32	0.00	0.00	0.00	9
33	0.04	0.07	0.05	15
34	0.00	0.00	0.00	11
35	0.00	0.00	0.00	8
36	0.00	0.00	0.00	9
37	0.00	0.00	0.00	14
38	0.05	0.15	0.07	13
39	0.00	0.00	0.00	5
40	0.00	0.00	0.00	8
41	0.07	0.06	0.07	16
42	0.00	0.00	0.00	8
43	0.00	0.00	0.00	9
44	0.00	0.00	0.00	5
45	0.10	0.55	0.17	11
46	0.00	0.00	0.00	10
47	0.00	0.00	0.00	3
48	0.00	0.00	0.00	2
49	0.00	0.00	0.00	4
50	0.00	0.00	0.00	6
51	0.00	0.00	0.00	5
52	0.00	0.00	0.00	4
53	0.00	0.00	0.00	4
54	0.00	0.00	0.00	3
55	0.00	0.00	0.00	2
56	0.00	0.00	0.00	3
57	0.00	0.00	0.00	4
58	0.00	0.00	0.00	5
59	0.00	0.00	0.00	1
60	0.00	0.00	0.00	2
61	0.00	0.00	0.00	1
62	0.00	0.00	0.00	3
63	0.00	0.00	0.00	4
64	0.00	0.00	0.00	1
65	0.00	0.00	0.00	2
66	0.00	0.00	0.00	3

```
           67       0.00      0.00      0.00         2
           68       0.00      0.00      0.00         1
           69       0.00      0.00      0.00         2
           72       0.00      0.00      0.00         1

     accuracy                           0.12       537
    macro avg       0.02      0.04      0.02       537
 weighted avg       0.05      0.12      0.07       537
```

```
C:\Users\praty\New folder\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [9]:
```
print(confusion_matrix(y_test,predict_test))

print(classification_report(y_test,predict_test))
```

```
[[11  1  0 ...  0  0  0]
 [15  3  0 ...  0  0  0]
 [ 7  0  0 ...  0  0  0]
 ...
 [ 1  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]]
              precision    recall  f1-score   support

          21       0.19      0.73      0.30        15
          22       0.15      0.12      0.14        24
          23       0.00      0.00      0.00        10
          24       0.15      0.43      0.22         7
          25       0.29      0.12      0.17        17
          26       0.00      0.00      0.00         6
          27       0.00      0.00      0.00        12
          28       0.00      0.00      0.00        12
          29       0.04      0.40      0.08         5
          30       0.00      0.00      0.00         8
```

31	0.00	0.00	0.00	7
32	0.00	0.00	0.00	7
33	0.00	0.00	0.00	2
34	0.00	0.00	0.00	3
35	0.00	0.00	0.00	2
36	0.00	0.00	0.00	7
37	0.00	0.00	0.00	5
38	0.00	0.00	0.00	3
39	0.00	0.00	0.00	7
40	0.00	0.00	0.00	5
41	0.00	0.00	0.00	6
42	0.00	0.00	0.00	10
43	0.00	0.00	0.00	4
44	0.00	0.00	0.00	3
45	0.04	0.25	0.06	4
46	0.00	0.00	0.00	3
47	0.00	0.00	0.00	3
48	0.00	0.00	0.00	3
49	0.00	0.00	0.00	1
50	0.00	0.00	0.00	2
51	0.00	0.00	0.00	3
52	0.00	0.00	0.00	4
53	0.00	0.00	0.00	1
54	0.00	0.00	0.00	3
55	0.00	0.00	0.00	2
57	0.00	0.00	0.00	1
58	0.00	0.00	0.00	2
59	0.00	0.00	0.00	2
60	0.00	0.00	0.00	3
61	0.00	0.00	0.00	1
62	0.00	0.00	0.00	1
65	0.00	0.00	0.00	1
66	0.00	0.00	0.00	1
67	0.00	0.00	0.00	1
70	0.00	0.00	0.00	1
81	0.00	0.00	0.00	1
accuracy			0.10	231
macro avg	0.02	0.04	0.02	231

```
weighted avg       0.06      0.10      0.06       231
```

In [10]:
```
print('-------------------------EXPERIMENT-07 Ended--------------------
-------------')
```

```
-------------------------EXPERIMENT-07 Ended--------------------------
------
```

In [ ]:
```

```