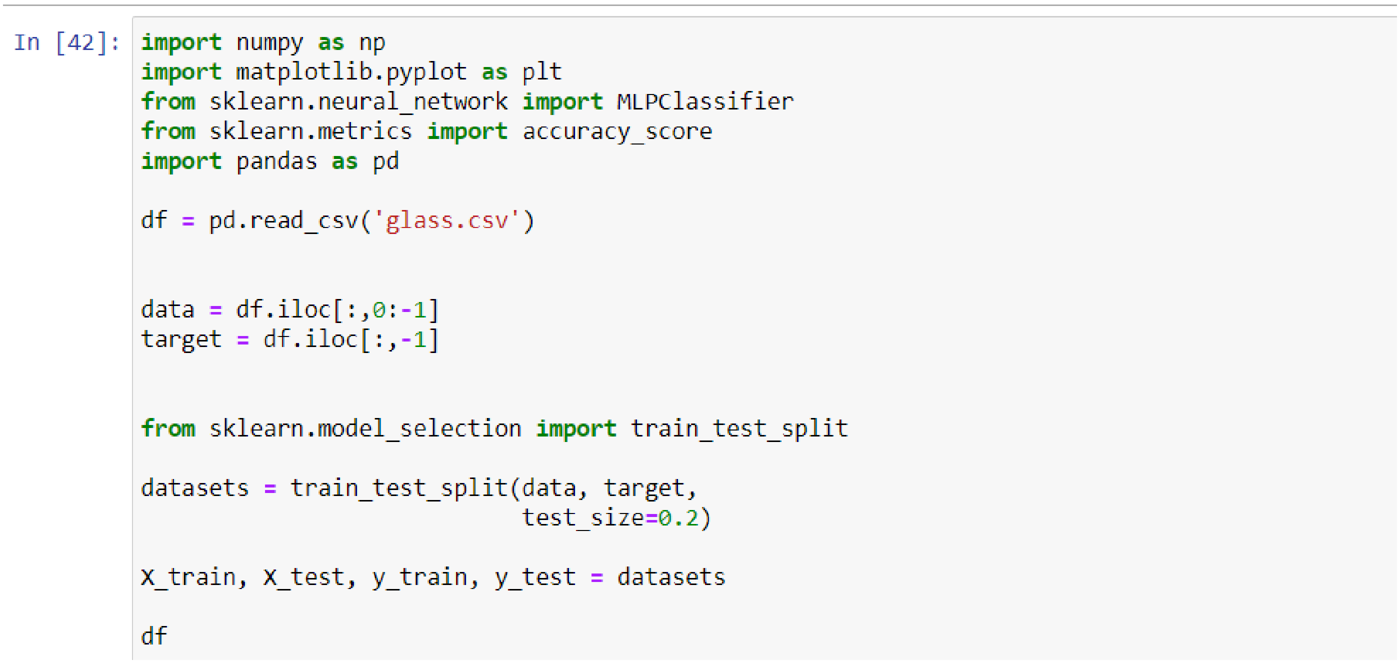
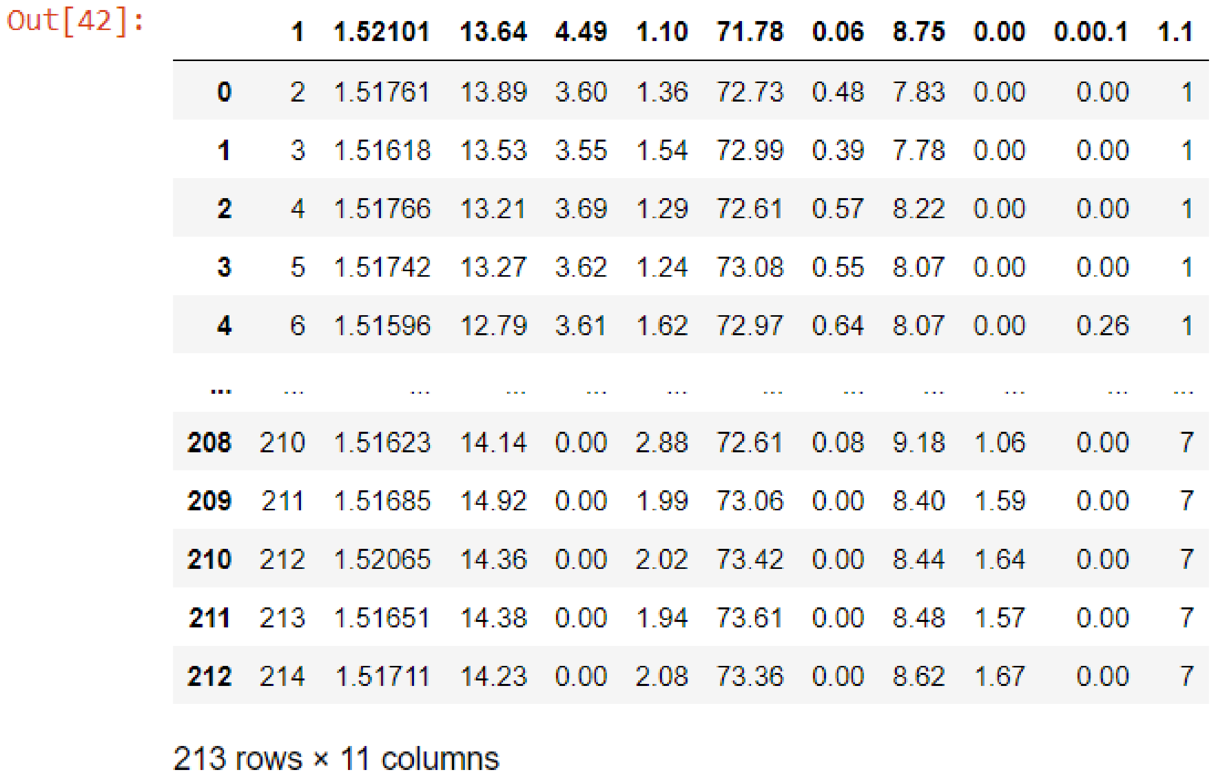


```
In [42]: import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.neural_network import MLPClassifier
         from sklearn.metrics import accuracy_score
         import pandas as pd

         df = pd.read_csv('glass.csv')


         data = df.iloc[:,0:-1]
         target = df.iloc[:,-1]


         from sklearn.model_selection import train_test_split

         datasets = train_test_split(data, target,
                                     test_size=0.2)

         X_train, X_test, y_train, y_test = datasets

         df
```

DATASET:



Out[42]:

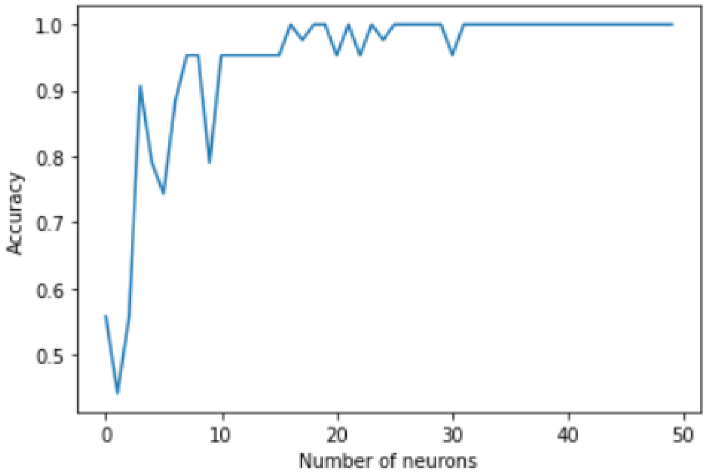
	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.00.1	1.1
0	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.00	1
1	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.00	1
2	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.00	1
3	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.00	1
4	6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.00	0.26	1
...	...	...	...	...	...	...	...	...	...	...	...
208	210	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.00	7
209	211	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.00	7
210	212	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.00	7
211	213	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.00	7
212	214	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.00	7

213 rows × 11 columns

Number of neurons X accuracy:

```
In [33]: acc = np.zeros(50)

         for i in range(50):
             mlp = MLPClassifier(hidden_layer_sizes=i+1, max_iter=5000, activation = 'logistic')
             mlp.fit(X_train, y_train)
             predictions_test = mlp.predict(X_test)
             acc[i] = accuracy_score(predictions_test, y_test)
             if i == np.argmax(acc):
                 max_prediction = predictions_test

         plt.xlabel("Number of neurons")
         plt.ylabel("Accuracy")
         plt.plot(acc)
```

```
Out[33]: [<matplotlib.lines.Line2D at 0x7fd37bf60130>]
```

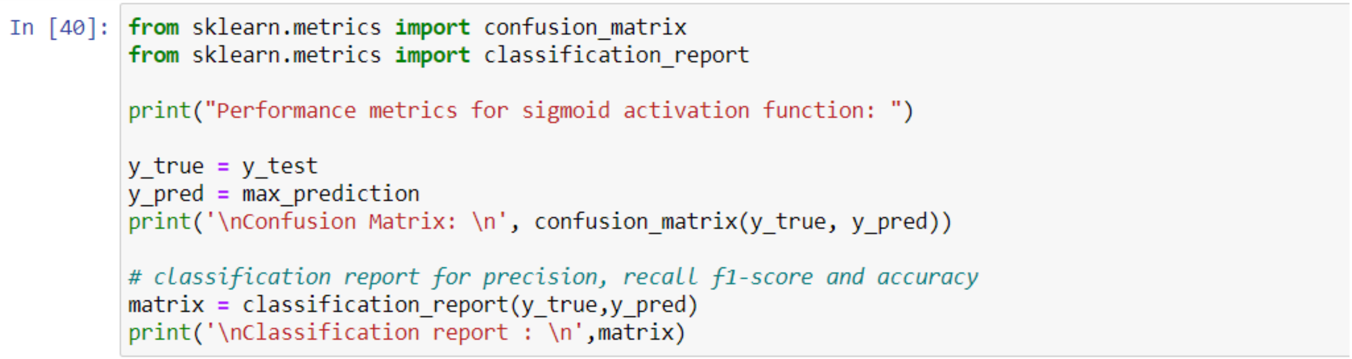


Number of neurons for maximum accuracy:

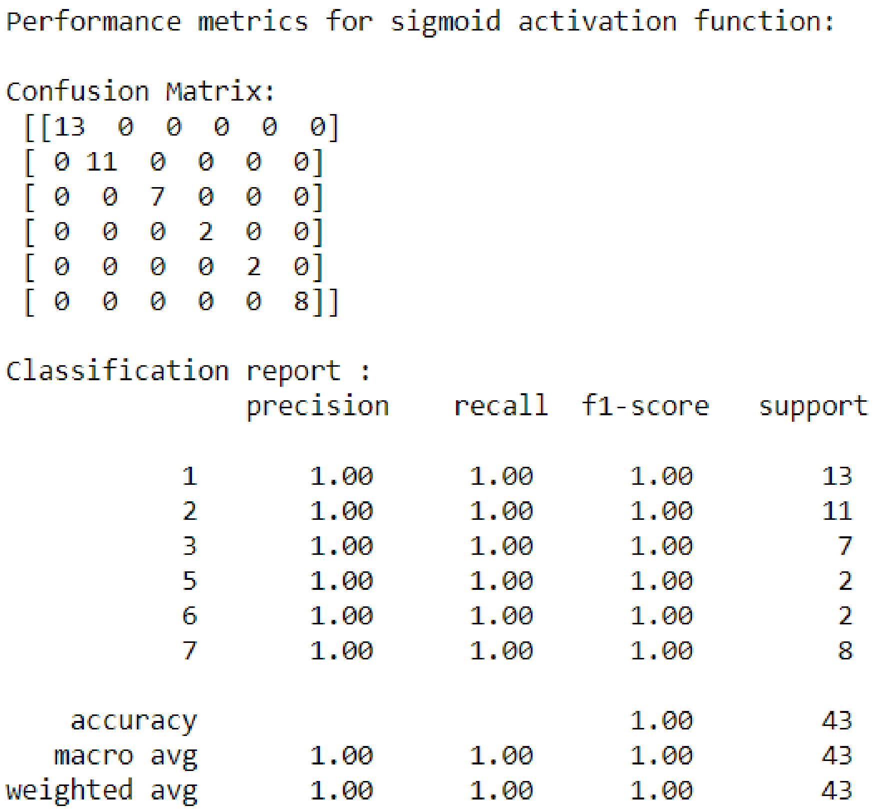
```
In [34]: n = np.argmax(acc)
         print("Number of neurons for maximum accuracy =", n)
         print("Accuracy = ", acc[n])
```

```
Number of neurons for maximum accuracy = 16
Accuracy =  1.0
```

Performance metrics for sigmoid activation function:

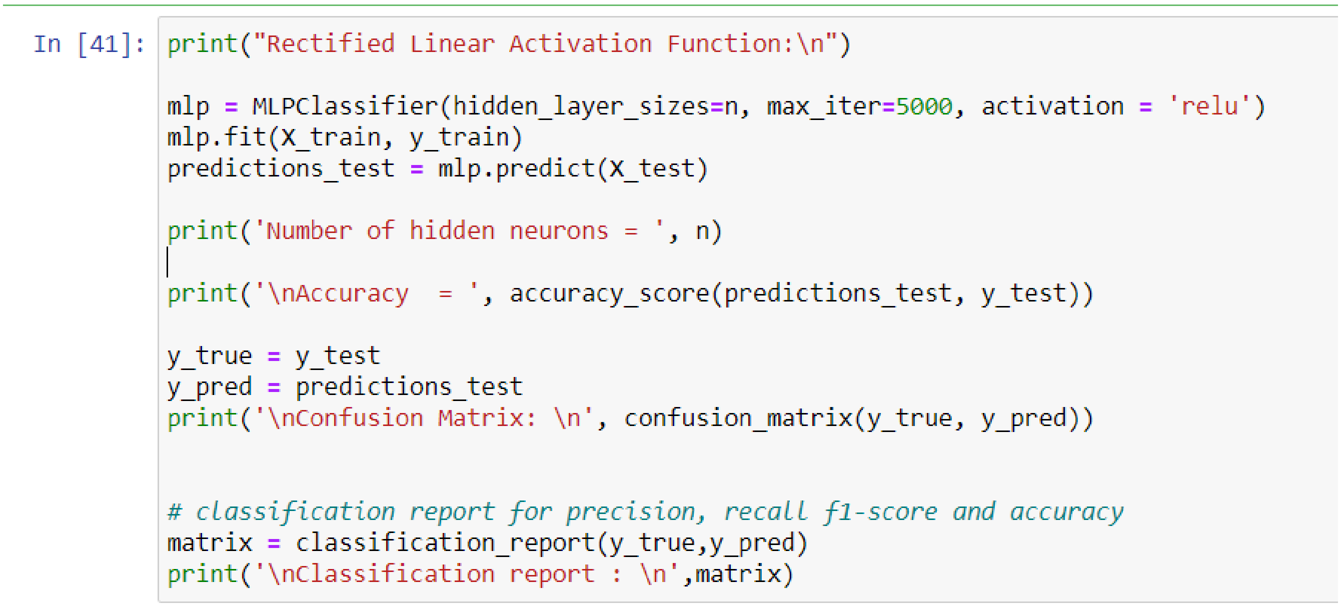


```
In [40]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report

         print("Performance metrics for sigmoid activation function: ")

         y_true = y_test
         y_pred = max_prediction
         print('\nConfusion Matrix: \n', confusion_matrix(y_true, y_pred))

         # classification report for precision, recall f1-score and accuracy
         matrix = classification_report(y_true,y_pred)
         print('\nClassification report : \n',matrix)
```

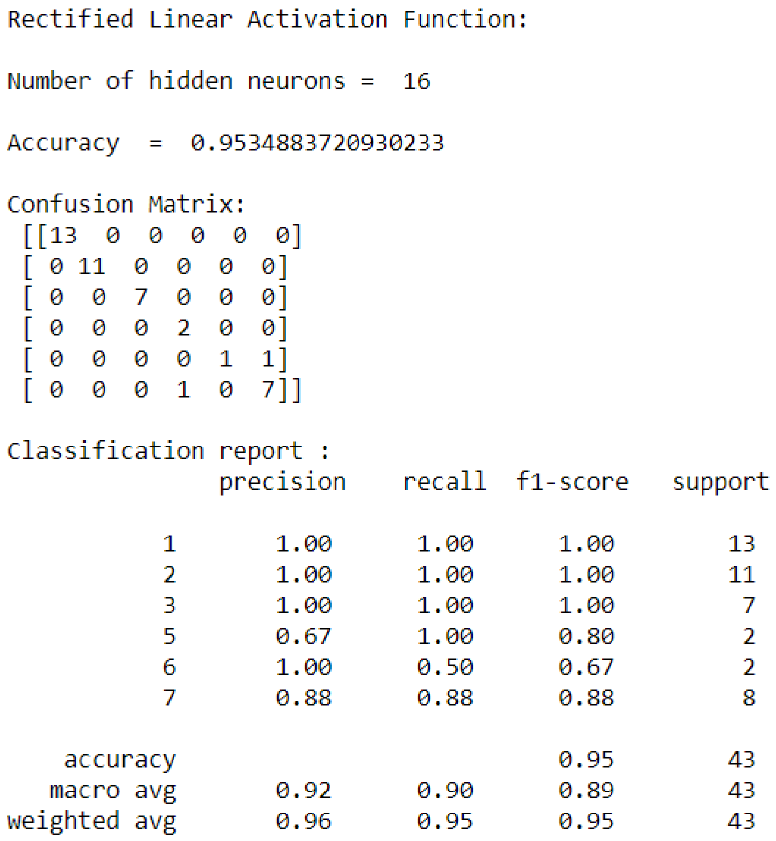


```
Performance metrics for sigmoid activation function: 

Confusion Matrix: 
 [[13  0  0  0  0  0]
 [ 0 11  0  0  0  0]
 [ 0  0  7  0  0  0]
 [ 0  0  0  2  0  0]
 [ 0  0  0  0  2  0]
 [ 0  0  0  0  0  8]]

Classification report : 
               precision    recall  f1-score   support

           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        11
           3       1.00      1.00      1.00         7
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         8

    accuracy                           1.00        43
   macro avg       1.00      1.00      1.00        43
weighted avg       1.00      1.00      1.00        43
```

Rectified linear activation function:

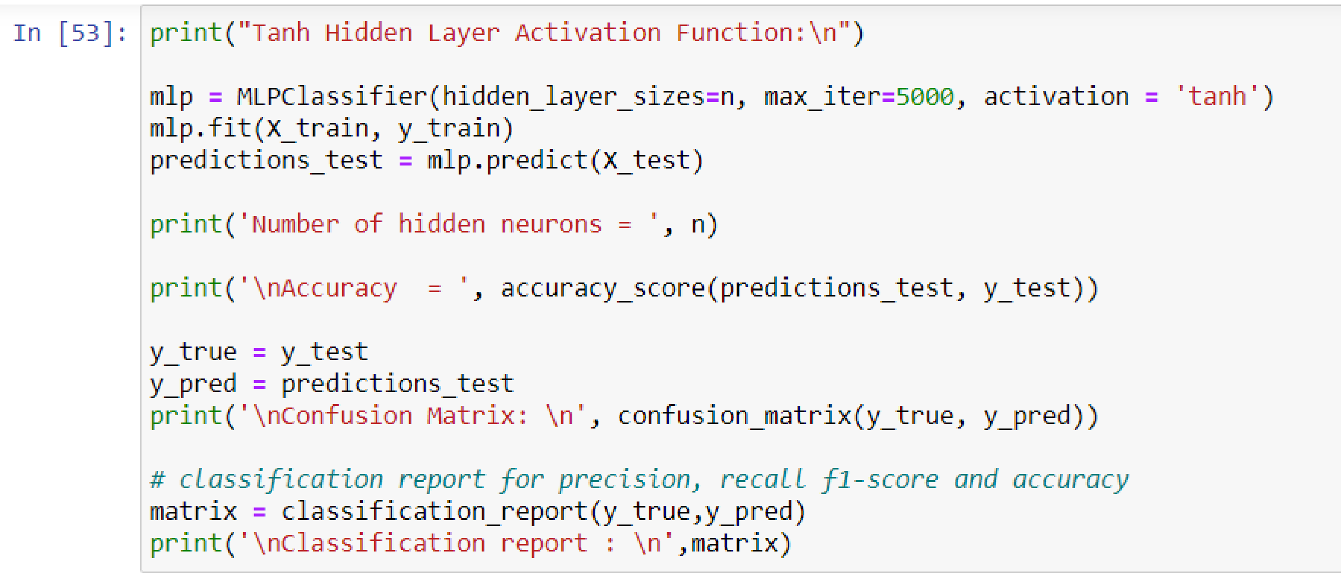


```
In [41]: print("Rectified Linear Activation Function:\n")

         mlp = MLPClassifier(hidden_layer_sizes=n, max_iter=5000, activation = 'relu')
         mlp.fit(X_train, y_train)
         predictions_test = mlp.predict(X_test)

         print('Number of hidden neurons = ', n)

         print('\nAccuracy  = ', accuracy_score(predictions_test, y_test))

         y_true = y_test
         y_pred = predictions_test
         print('\nConfusion Matrix: \n', confusion_matrix(y_true, y_pred))


         # classification report for precision, recall f1-score and accuracy
         matrix = classification_report(y_true,y_pred)
         print('\nClassification report : \n',matrix)
```

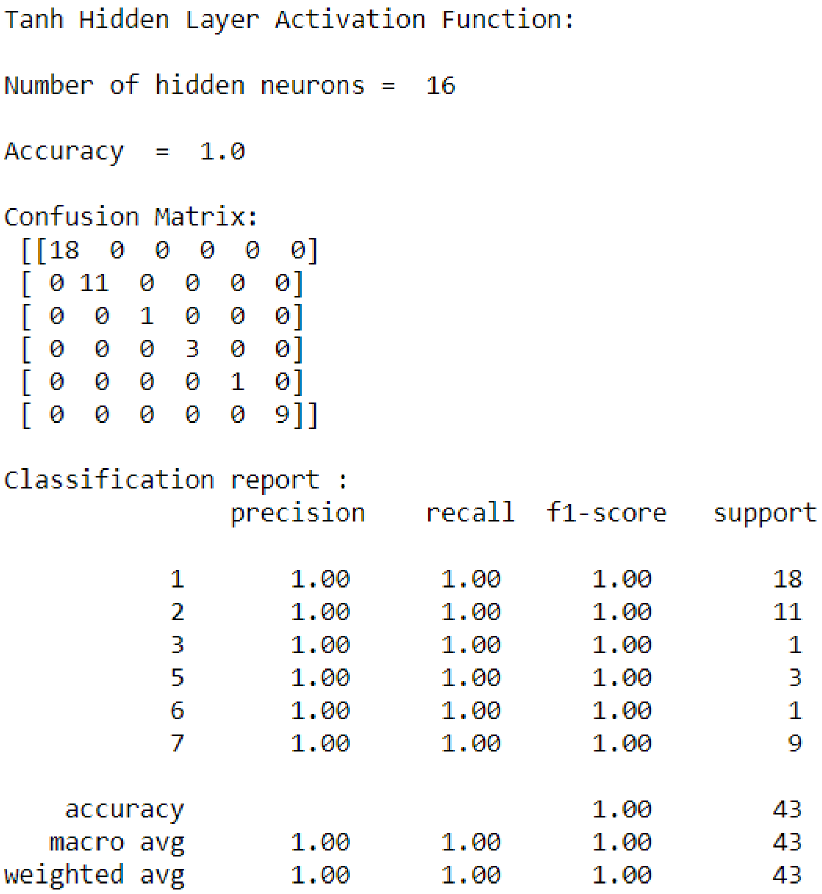


```
Rectified Linear Activation Function:

Number of hidden neurons =  16

Accuracy  =  0.9534883720930233

Confusion Matrix: 
 [[13  0  0  0  0  0]
 [ 0 11  0  0  0  0]
 [ 0  0  7  0  0  0]
 [ 0  0  0  2  0  0]
 [ 0  0  0  0  1  1]
 [ 0  0  0  1  0  7]]

Classification report : 
               precision    recall  f1-score   support

           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        11
           3       1.00      1.00      1.00         7
           5       0.67      1.00      0.80         2
           6       1.00      0.50      0.67         2
           7       0.88      0.88      0.88         8

    accuracy                           0.95        43
   macro avg       0.92      0.90      0.89        43
weighted avg       0.96      0.95      0.95        43
```

Hidden layer activation Function:

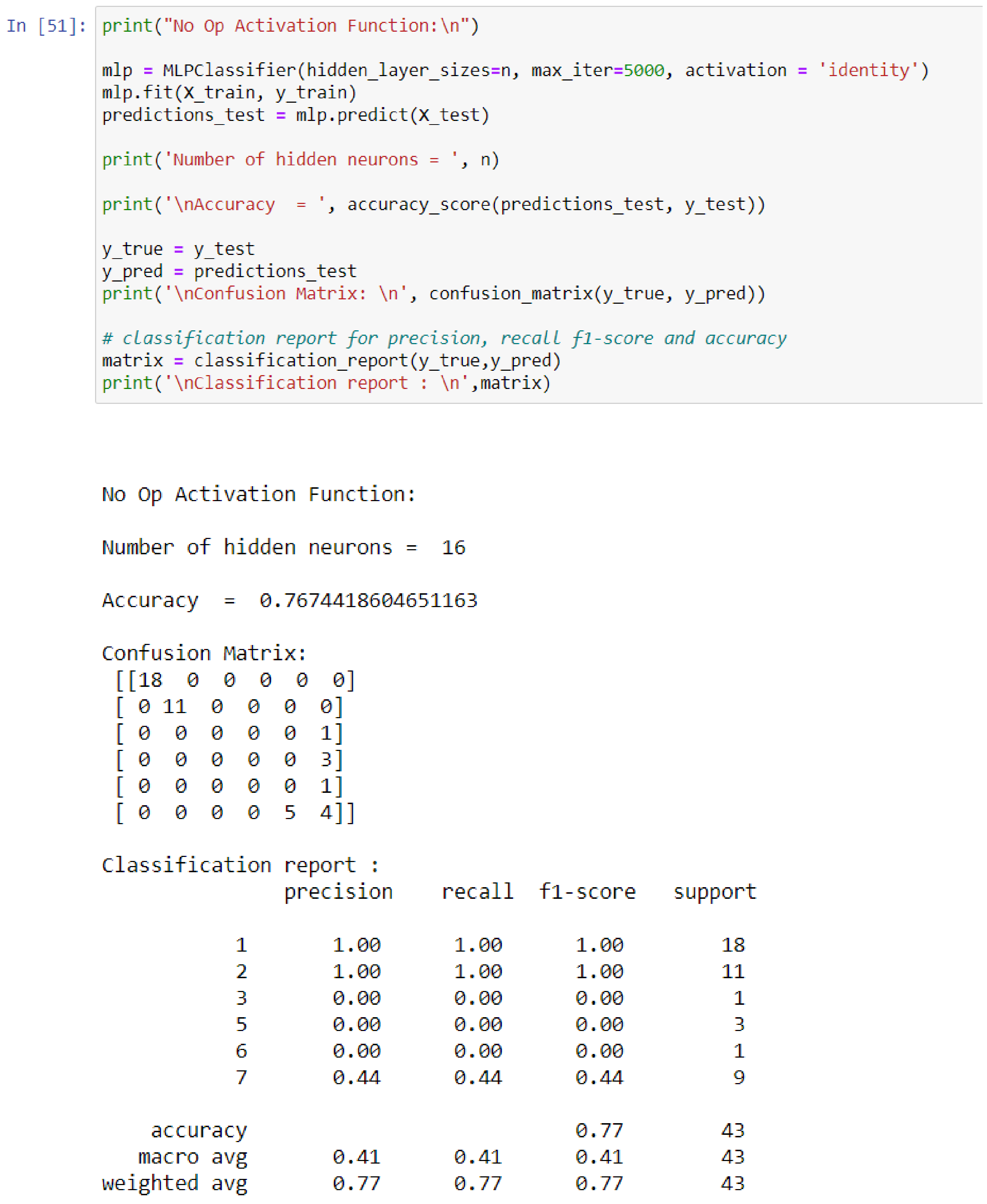


```
In [53]: print("Tanh Hidden Layer Activation Function:\n")

         mlp = MLPClassifier(hidden_layer_sizes=n, max_iter=5000, activation = 'tanh')
         mlp.fit(X_train, y_train)
         predictions_test = mlp.predict(X_test)

         print('Number of hidden neurons = ', n)

         print('\nAccuracy  = ', accuracy_score(predictions_test, y_test))

         y_true = y_test
         y_pred = predictions_test
         print('\nConfusion Matrix: \n', confusion_matrix(y_true, y_pred))

         # classification report for precision, recall f1-score and accuracy
         matrix = classification_report(y_true,y_pred)
         print('\nClassification report : \n',matrix)
```



```
Tanh Hidden Layer Activation Function:

Number of hidden neurons =  16

Accuracy  =  1.0

Confusion Matrix:
 [[18  0  0  0  0  0]
 [ 0 11  0  0  0  0]
 [ 0  0  1  0  0  0]
 [ 0  0  0  3  0  0]
 [ 0  0  0  0  1  0]
 [ 0  0  0  0  0  9]]

Classification report :
               precision    recall  f1-score   support

           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        11
           3       1.00      1.00      1.00         1
           5       1.00      1.00      1.00         3
           6       1.00      1.00      1.00         1
           7       1.00      1.00      1.00         9

    accuracy                           1.00        43
   macro avg       1.00      1.00      1.00        43
weighted avg       1.00      1.00      1.00        43
```

No OP activation function:



```
In [51]: print("No Op Activation Function:\n")

         mlp = MLPClassifier(hidden_layer_sizes=n, max_iter=5000, activation = 'identity')
         mlp.fit(X_train, y_train)
         predictions_test = mlp.predict(X_test)

         print('Number of hidden neurons = ', n)

         print('\nAccuracy  = ', accuracy_score(predictions_test, y_test))

         y_true = y_test
         y_pred = predictions_test
         print('\nConfusion Matrix: \n', confusion_matrix(y_true, y_pred))

         # classification report for precision, recall f1-score and accuracy
         matrix = classification_report(y_true,y_pred)
         print('\nClassification report : \n',matrix)
```

```
No Op Activation Function:

Number of hidden neurons =  16

Accuracy  =  0.7674418604651163

Confusion Matrix: 
 [[18  0  0  0  0  0]
 [ 0 11  0  0  0  0]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  0  3]
 [ 0  0  0  0  0  1]
 [ 0  0  0  0  5  4]]

Classification report : 
               precision    recall  f1-score   support

           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        11
           3       0.00      0.00      0.00         1
           5       0.00      0.00      0.00         3
           6       0.00      0.00      0.00         1
           7       0.44      0.44      0.44         9

    accuracy                           0.77        43
   macro avg       0.41      0.41      0.41        43
weighted avg       0.77      0.77      0.77        43
```