Multi-Layer Perceptron

Assignment 5_G4

Ahmed Yousry **Bassel Hamshary**
Mohamed El-Namoury

uOttawa
Faculté de génie
Faculty of Engineering

JULY 27, 2021
UNIVERSITY OF OTTAWA
Ottawa, Canada

Table of Contents

List of Figures

1. Implementation

Assumption: We assumed for the first question that the one layer used have **4** nodes in it and we built on that for the upcoming questions.

1.1. Apply MLP by using Activation Functions

```
act_fun = ["relu","logistic","tanh"]
for fun in act_fun:
    accuracy = []
    x_axis = []
    average = []
    for i in range(0,10):
        clf = MLPClassifier(random_state=i, max_iter=1000, activation =fun
                            ,hidden_layer_sizes = (4,),
                            learning_rate_init=0.1).fit(X_train, y_train)
        clf.predict(X_test)
        score = clf.score(X_test, y_test)
        accuracy.append(score)
        x_axis.append(i)
    fig, ax = plt.subplots()
    average_acc = sum(accuracy)/len(accuracy)
    average = [average_acc] * 10
    ax.plot(x_axis, accuracy, 'o-', label='Accuracy')
    ax.plot(x_axis ,average,color='orange',  label="Average")
    plt.title('Activation function: {}'.format(fun))
    plt.legend()
    plt.show()
    print('{} accuracy equals'.format(fun), average_acc)
```

1.1.1. Relu Activation Function

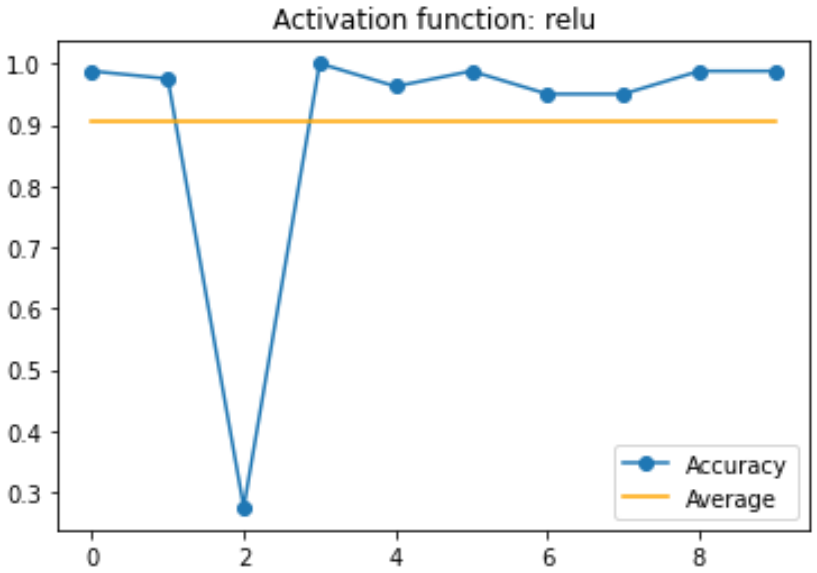

Figure 1: Relu activation function with average accuracy of 0.90625

1.1.2. Sigmoid Activation Function

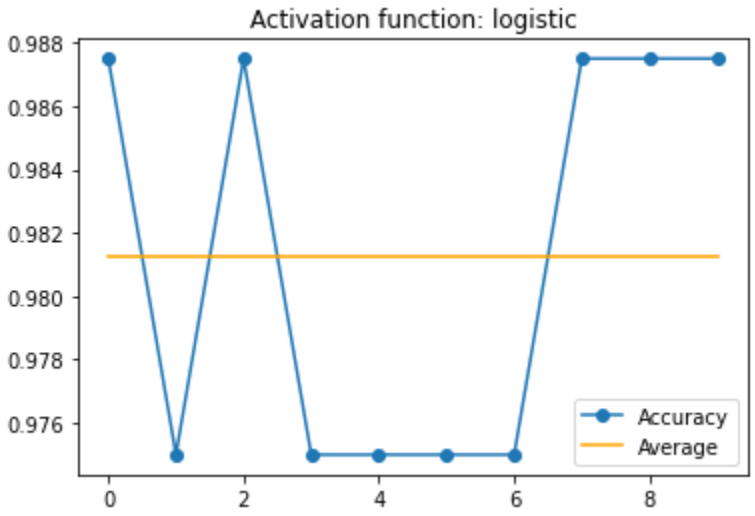

Figure 2:Sigmoid activation function with average accuracy of 0.98125

1.1.3. Tanh Activation Function

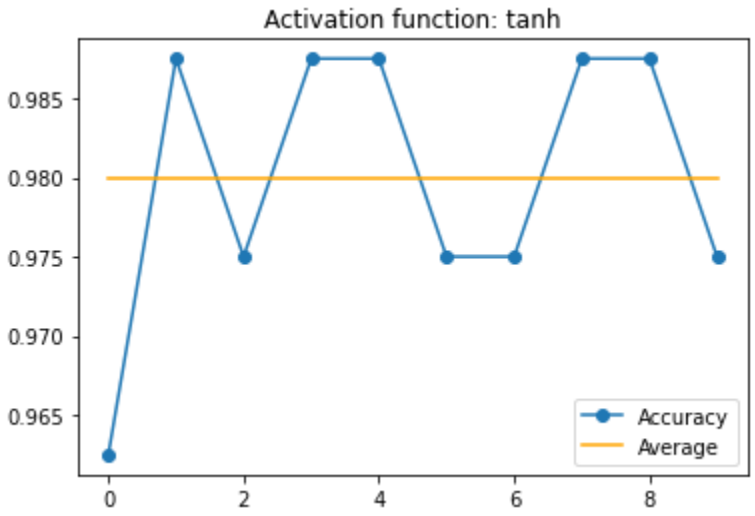

Figure 3: Sigmoid activation function with average accuracy of 0.97999

Deduction: After calculating the average accuracies for the three different activation functions, we found out that the champion one is logistic (sigmoid) with average accuracy **0.98125.**

1.2. Multiple Hidden Layers for the Highest Accuracy Activation Function

```
baseline = [0.98125] *10
for i in range(0,4):
    hidden_layers = [(4,),(6,4),(6,6,4),(6,6,6,4)]
    accuracy_2nd = []
    x_axis = []
    average_updated = []
    for j in range(0,10):
        clf = MLPClassifier(random_state=j, max_iter=1000,
                            activation ="logistic",
                            hidden_layer_sizes = hidden_layers[i],
                            learning_rate_init=0.1).fit(X_train, y_train)
        clf.predict(X_test)
        score = clf.score(X_test, y_test)
        accuracy_2nd.append(score)
        x_axis.append(j)
    fig, ax = plt.subplots()
    average_acc_upt = sum(accuracy_2nd)/len(accuracy_2nd)
    average_upt = [average_acc_upt] * 10
    ax.plot(x_axis, accuracy_2nd, 'o-', label='Accuracy')
    ax.plot(x_axis ,average_upt,color='orange',  label="Average")
    ax.plot(x_axis ,baseline,color='red',  label="Baseline")

    plt.title('Number of Hidden Layer:{}'.format(str(i+1)))
    plt.legend()
    plt.show()
    print('accuracy equals', average_acc_upt)
```

1.2.1. One Hidden layer

Hidden layer_1 **4** nodes.

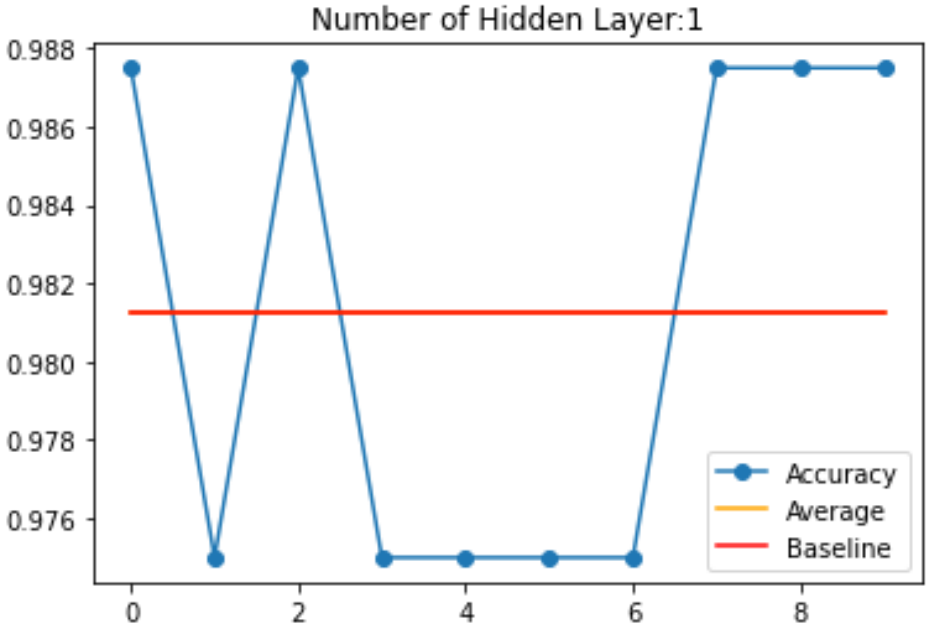

Figure 4: One hidden layer average accuracy is 0.98125

1.2.2. Two Hidden layers

Hidden layer_1 **6** nodes, and hidden layer_2 **4** nodes

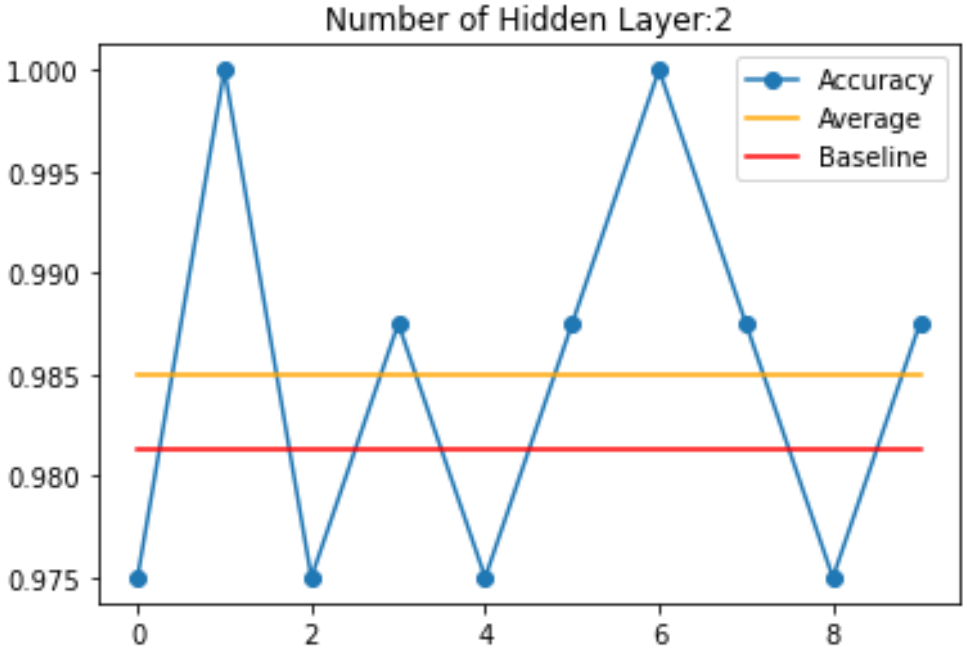

Figure 5: Two hidden layer average accuracy is 0.985

1.2.3. Three Hidden layers

Hidden layer_1 **6** nodes, hidden layer_2 **6** nodes, and hidden layer_3 **4** nodes.

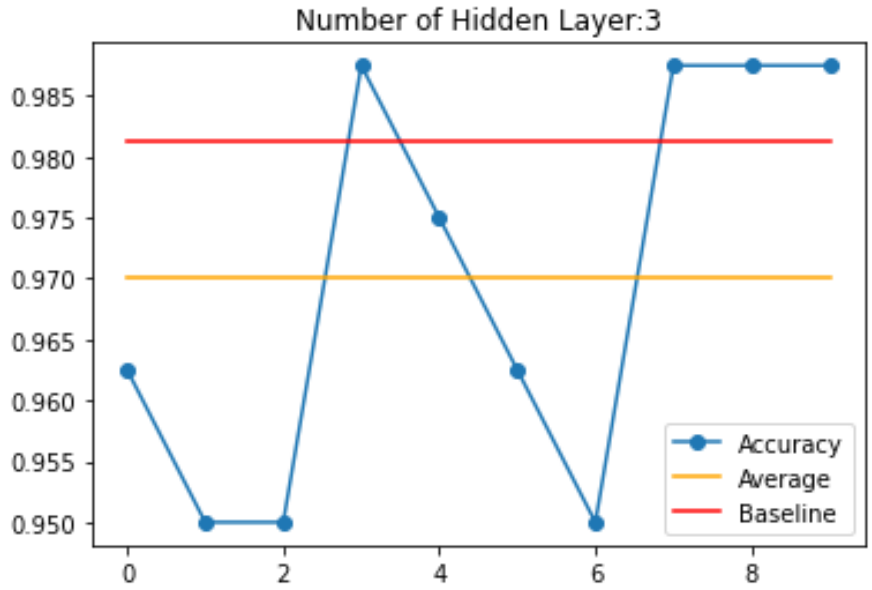

Figure 6: Three hidden layer average accuracy is 0.97

1.2.4. Four Hidden layers

Hidden layer_1 **6** nodes, hidden layer_2 **6** nodes, hidden layer_3 **6** nodes and hidden layer_4 **4** nodes.

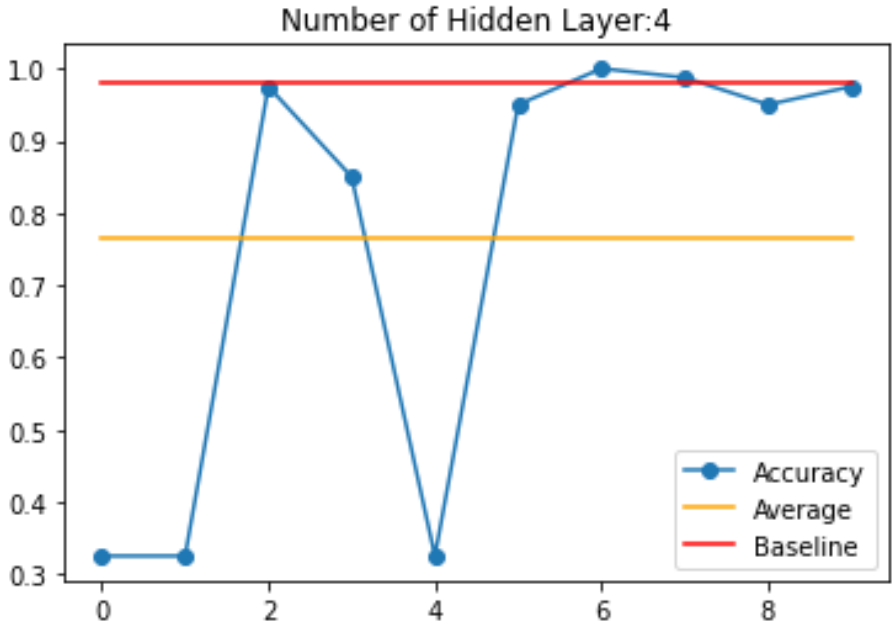

Figure 7: Four hidden layer average accuracy is 0.76625

Deduction: After measuring the average accuracies for different numbers of hidden layers we found that the network with 2 hidden layers and node size [6,4] with average accuracy of 0.985 is the best among the others.

1.3. Learning Rate Tuning

1.3.1. Learning Rate of 0.08

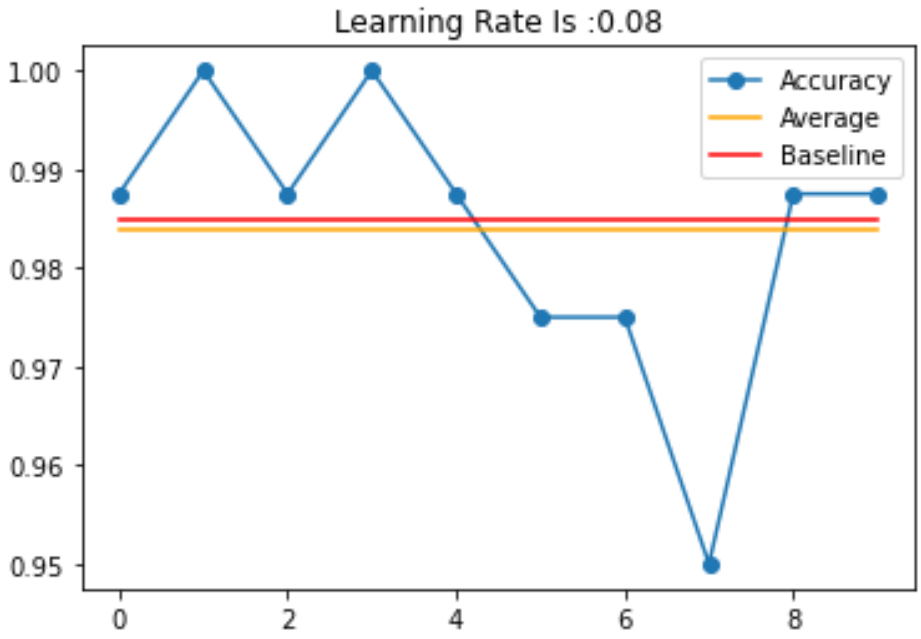

Figure 8: Two hidden layers with learning rate of 0.08 and average accuracy 0.98375

1.3.2. Learning Rate of 0.09

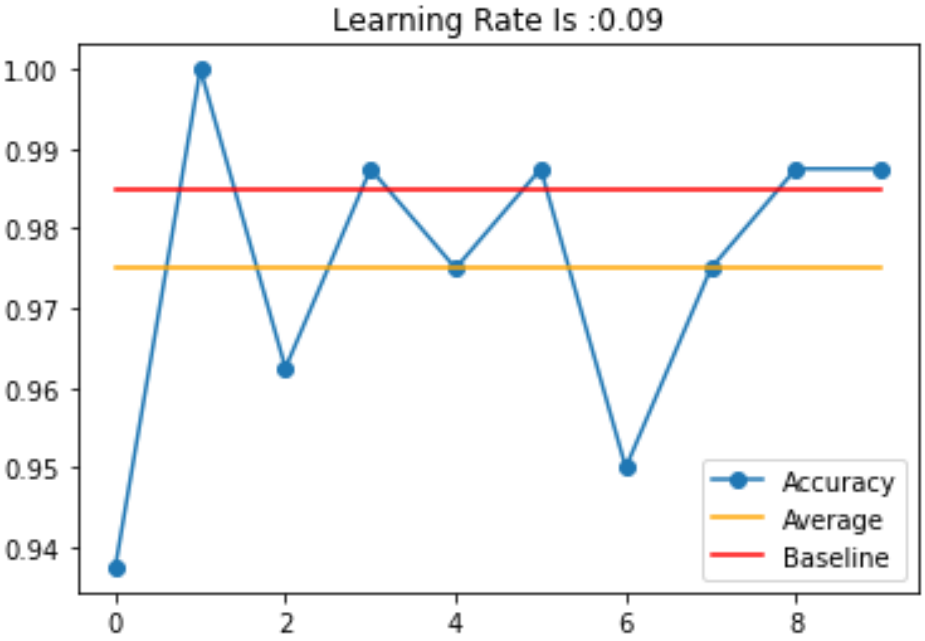

Figure 9: Two hidden layers with learning rate of 0.08 and average accuracy 0.975

1.3.3. Learning Rate of 0.1

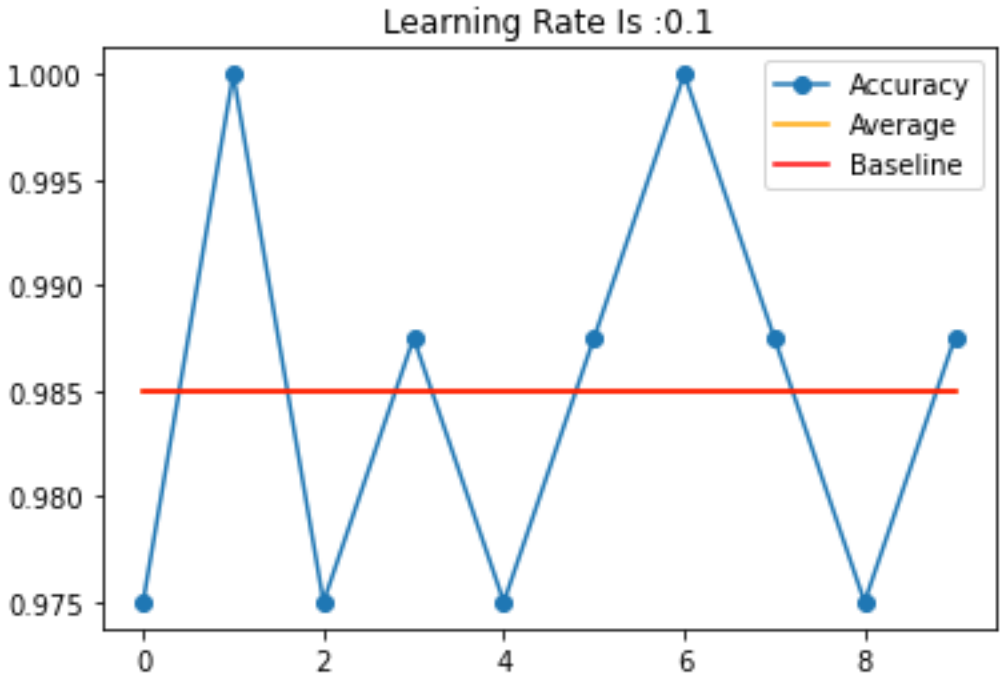

Figure 10: Two hidden layers with learning rate of 0.08 and average accuracy 0.985

Deduction: After tuning the learning rate, we found that the network with learning rate equals to 0.1 with average accuracy of 0.985 is the best tuning.
Then our final parameters:
activation function = logistic (Sigmoid)
Hidden layers = 2
Learning rate = 0.1

1.4. Final Model

```
fig, axes = plt.subplots(0, 1, figsize=(20, 30))

clf_final = MLPClassifier(random_state=10, max_iter=1000,
                          activation ="logistic" ,hidden_layer_sizes = 2,
                          learning_rate_init=0.1).fit(X_train, y_train)
clf_final.predict(X_test)
score = clf_final.score(X_test, y_test)
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
title= " learning curves for MLP"
plot_learning_curve(clf_final, title, X_train, y_train, ylim=(0.7, 1.01),
                    cv=cv, n_jobs=4)

plt.show()
```

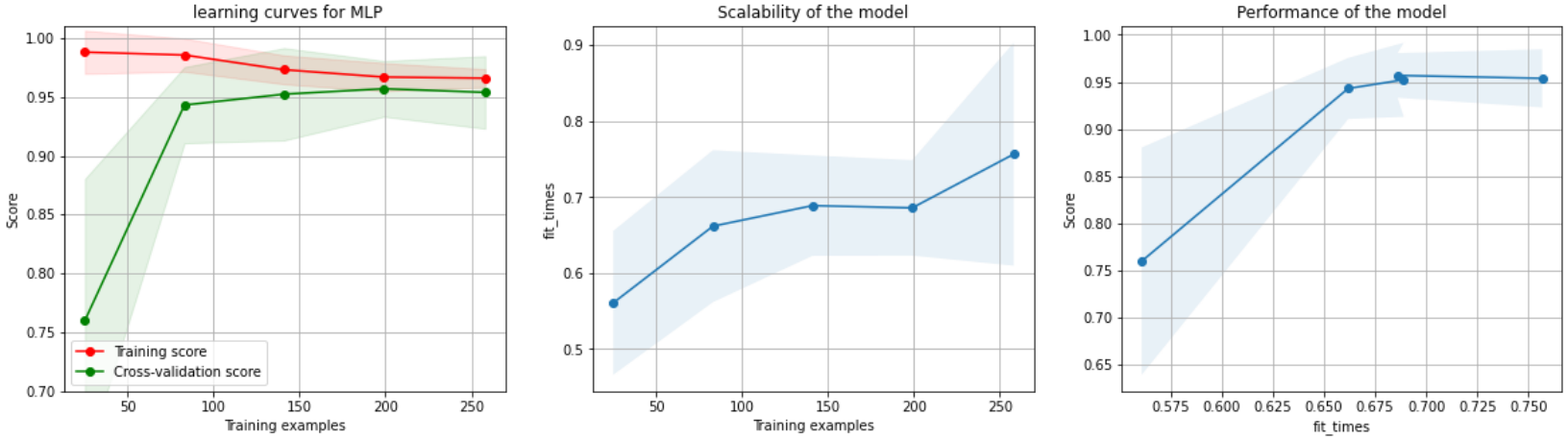

Figure 11: Final model curves

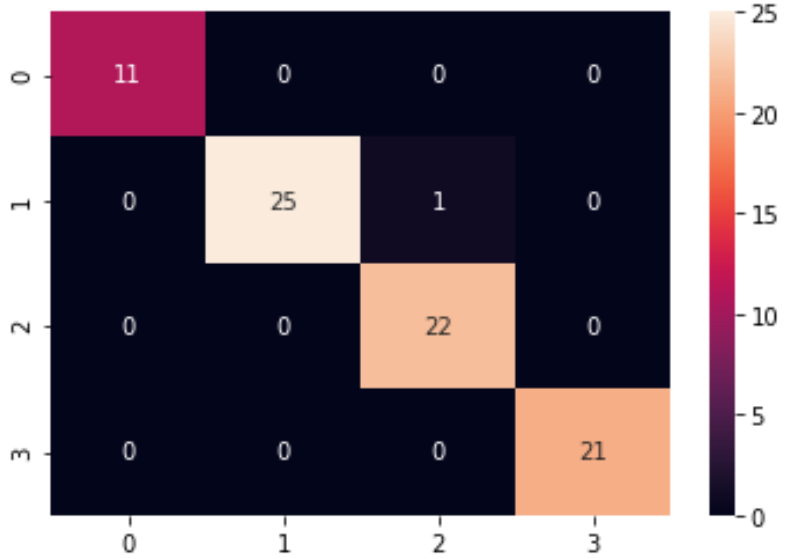

Figure 12: Final model Confusion Matrix and accuracy 0.9875