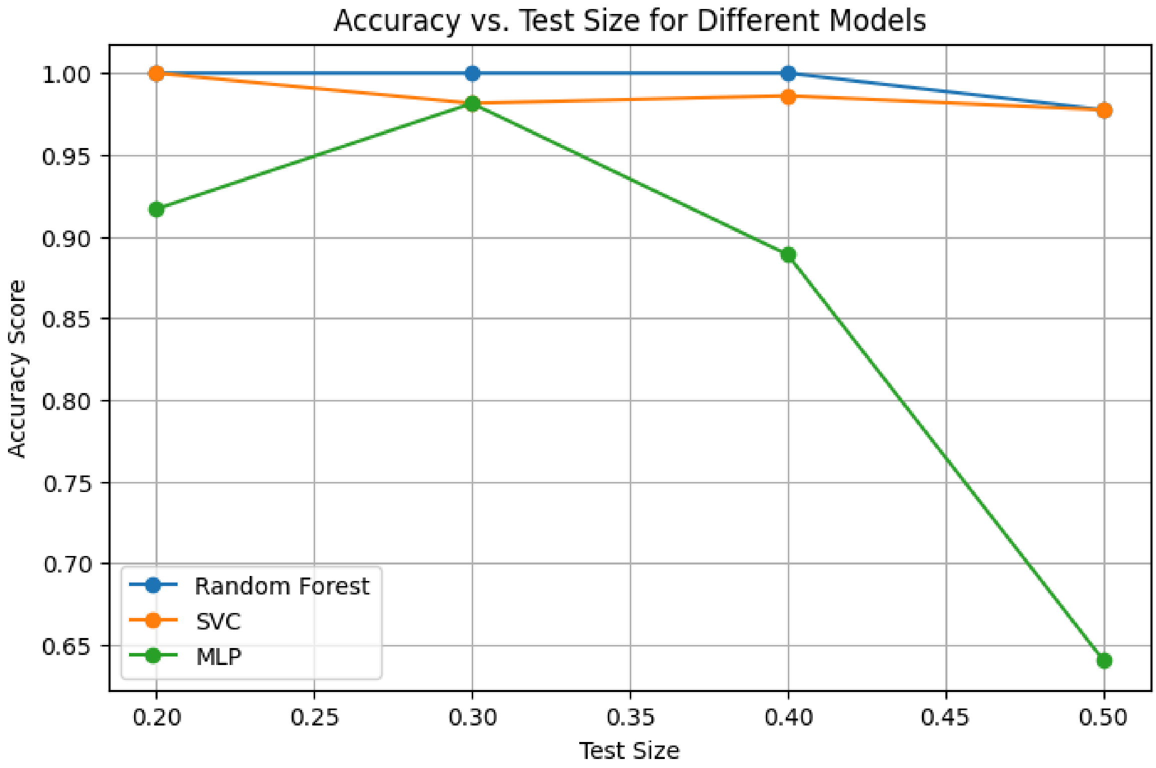
<u>Wine Data</u>

This data will use three models for the purpose of modelling the data classification: RandomForestClassifier, SupportVectorClassifier, MLPClassifier. We tested the performance of the three models for different test data sizes such as 0.2, 0.3 etc. The following is the observation:



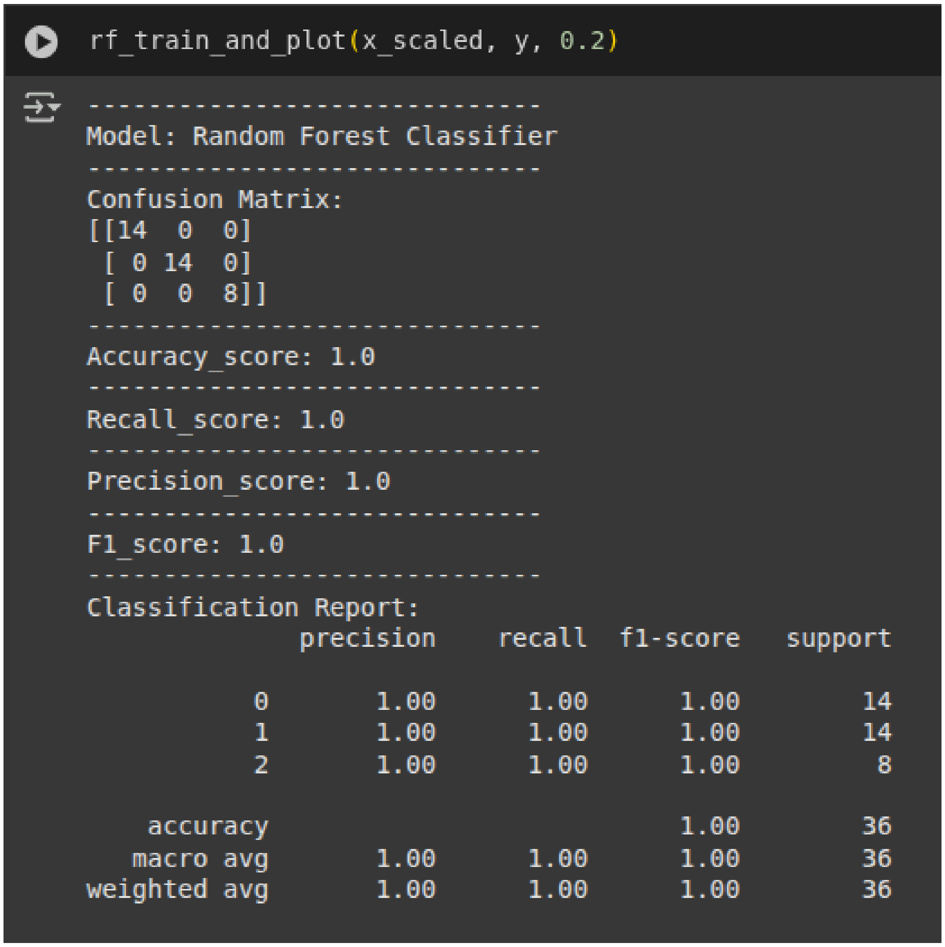
Thus it is observed that without any parameter tuning:

Model Name	Best Test Size	Approx. Accuracy Score
RandomForestClassifier	0.2	1.0
SupportVectorClassifier	0.2	1.0
MLPClassifier	0.3	0.98

Thus after training the models on the best observed test size the observation is as follows:

1. **<u>RandomForestClassifier</u>**

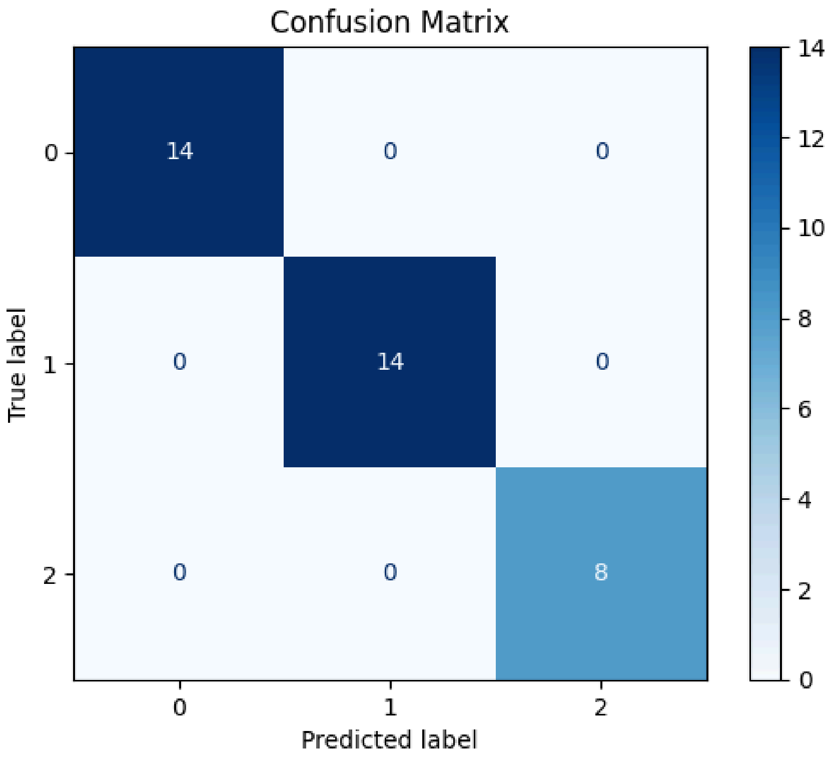
Since RandomForestClassifier performed best at test size= 0.5 thus we trained the model on the observed test size and below is the observed report:



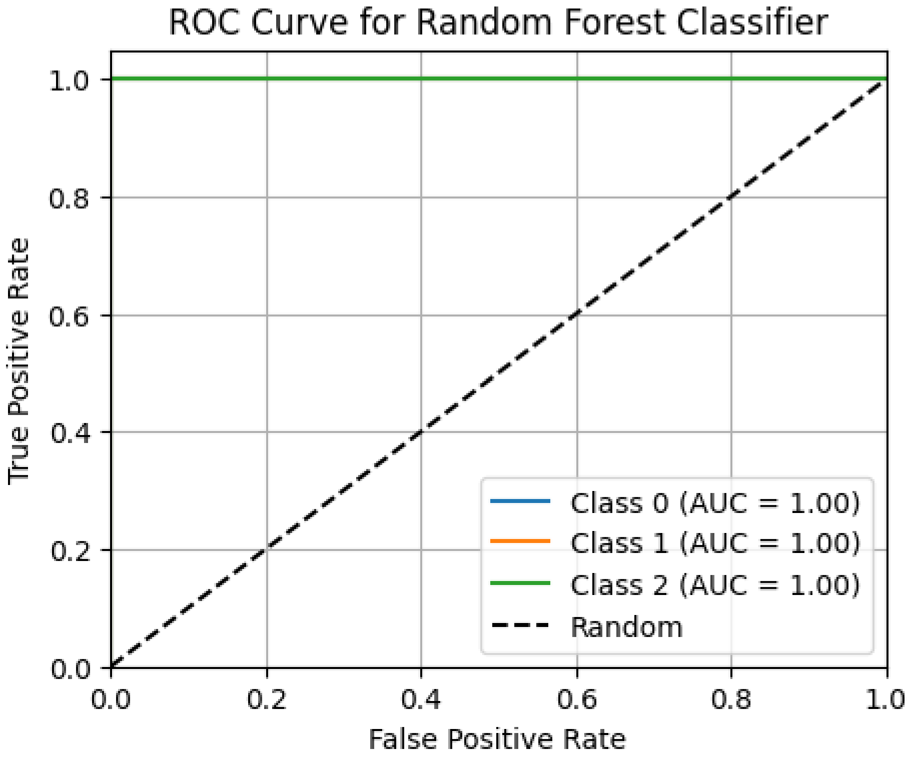
```
rf_train_and_plot(x_scaled, y, 0.2)
```

```
------------------------------
Model: Random Forest Classifier
------------------------------
Confusion Matrix:
[[14  0  0]
 [ 0 14  0]
 [ 0  0  8]]
------------------------------
Accuracy_score: 1.0
------------------------------
Recall_score: 1.0
------------------------------
Precision_score: 1.0
------------------------------
F1_score: 1.0
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```

ConfusionMatrix:

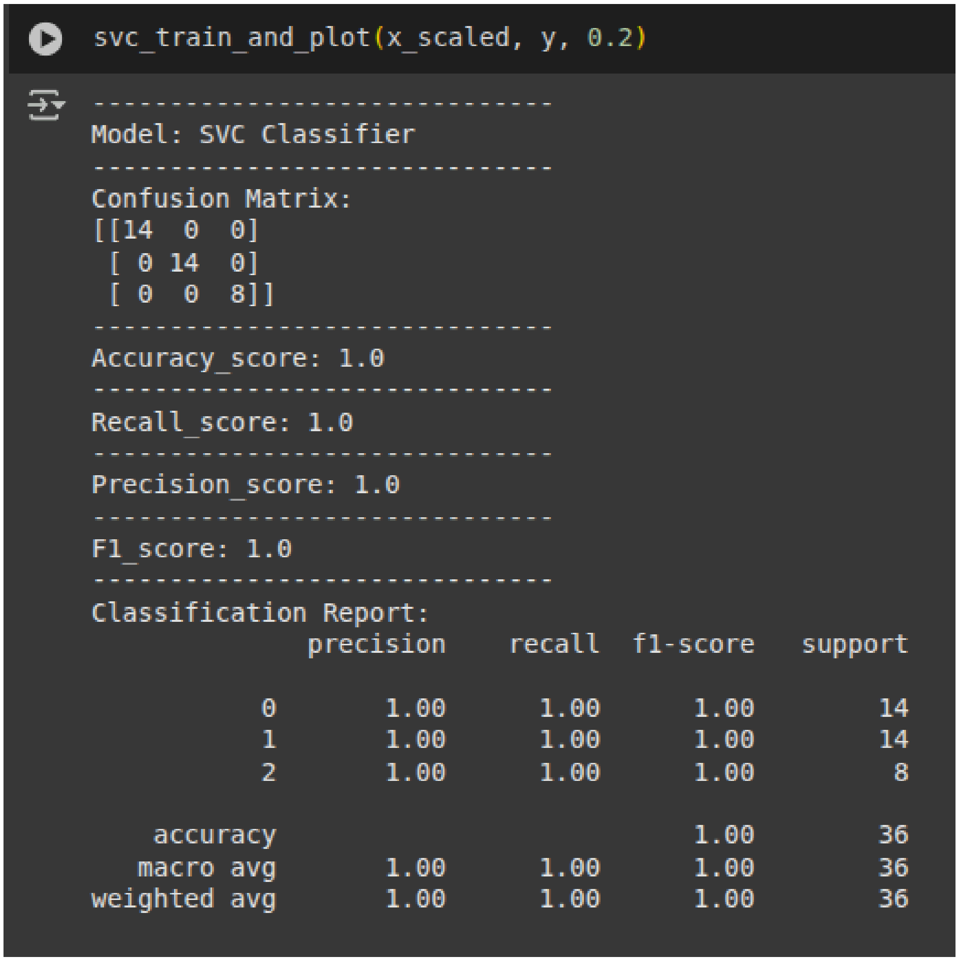


ROC Curve



2. **SupportVectorClassifier**

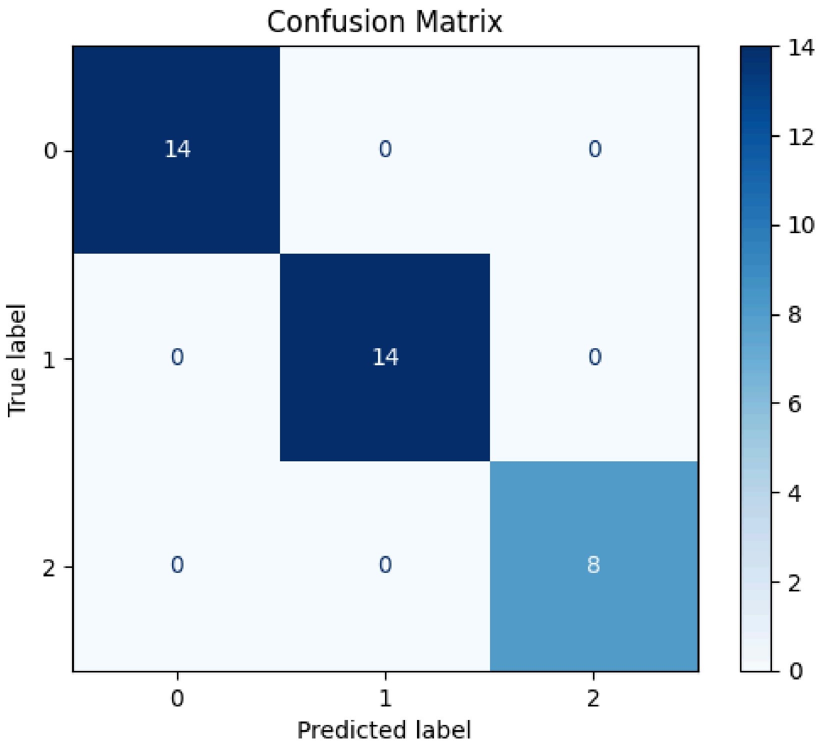
Since SupportVectorClassifier performed its best at test size= 0.2 so trained the model with the optimal performance test size. The observed report is as follows:



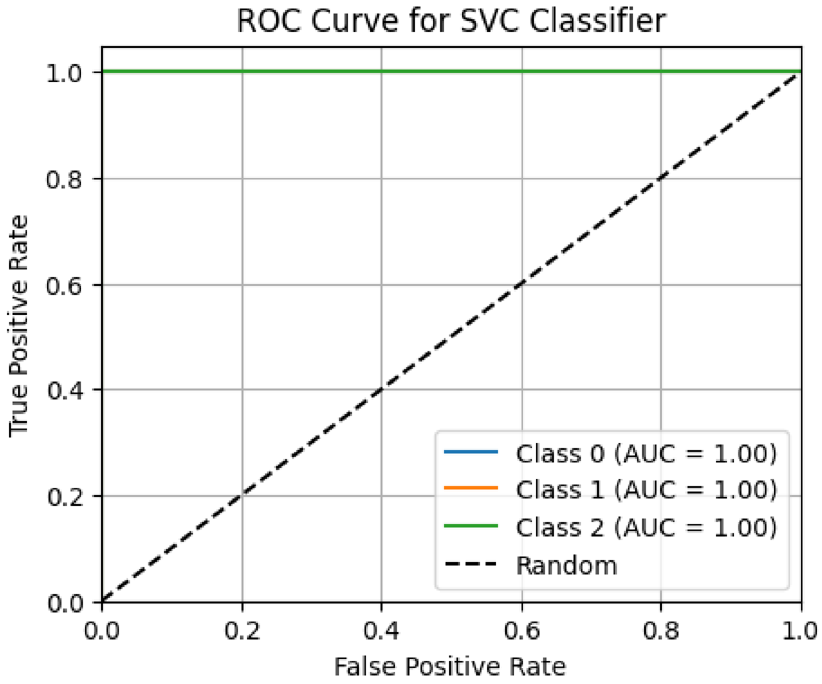
```
svc_train_and_plot(x_scaled, y, 0.2)
```

```
------------------------------
Model: SVC Classifier
------------------------------
Confusion Matrix:
[[14  0  0]
 [ 0 14  0]
 [ 0  0  8]]
------------------------------
Accuracy_score: 1.0
------------------------------
Recall_score: 1.0
------------------------------
Precision_score: 1.0
------------------------------
F1_score: 1.0
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```

ConfusionMatrix:

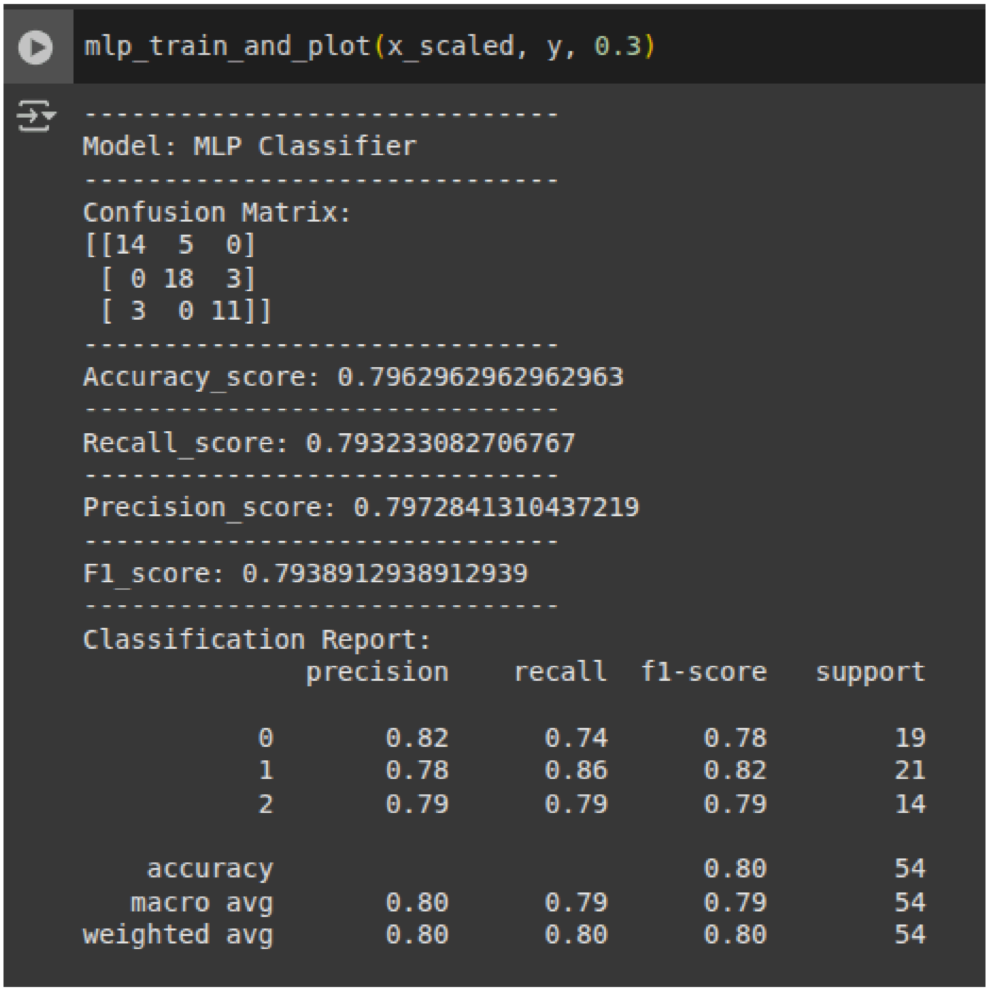


ROC Curve



3. **MLPClassifier**

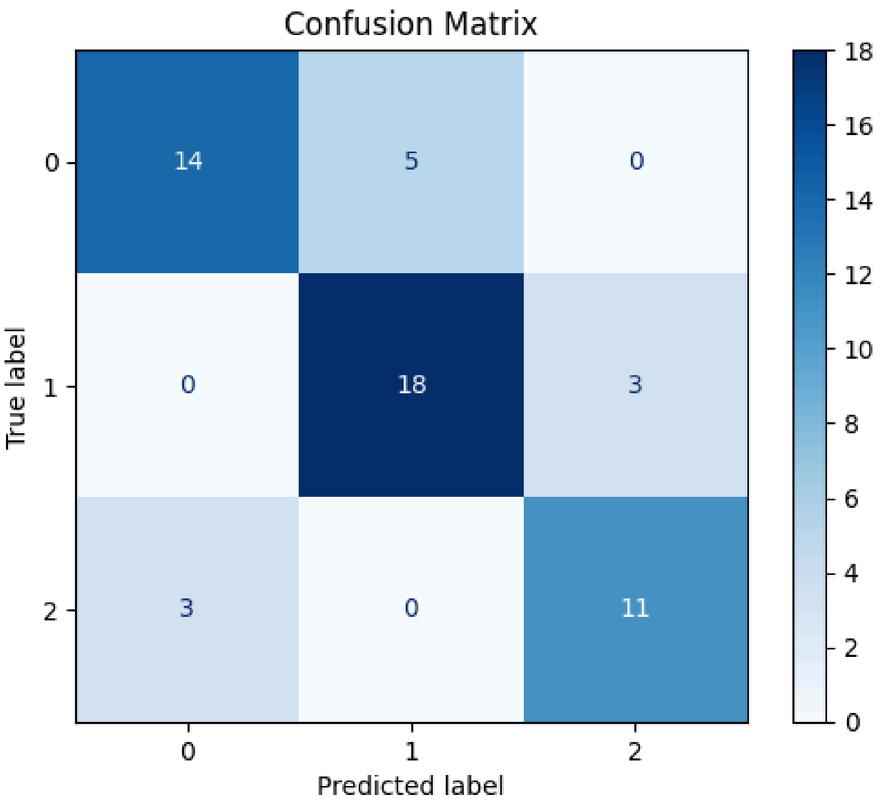
Since MLPClassifier performed optimally at test size= 0.3 so, trained the model on the optimal performing test size. The observed report is as follows:



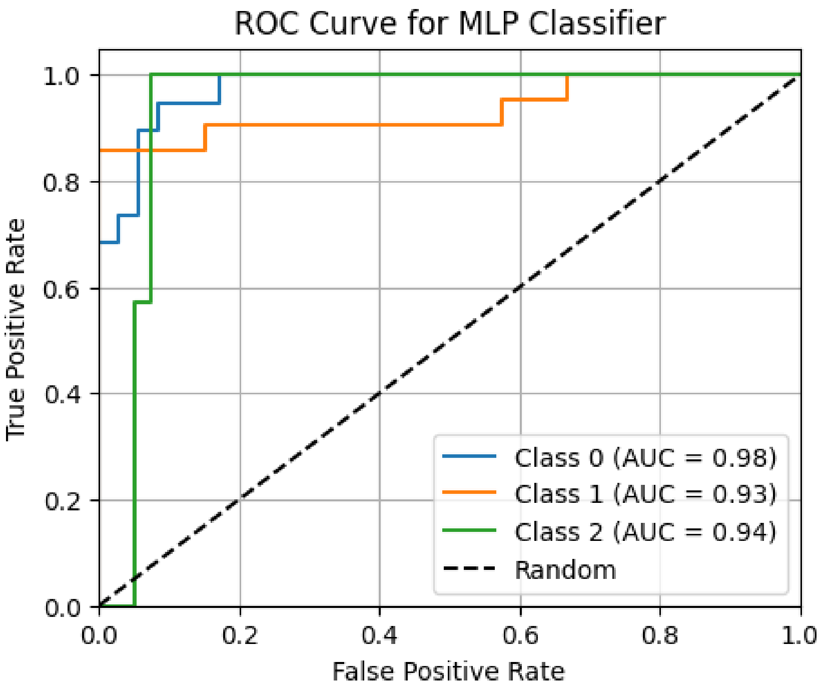
```
mlp_train_and_plot(x_scaled, y, 0.3)
```

```
------------------------------
Model: MLP Classifier
------------------------------
Confusion Matrix:
[[14  5  0]
 [ 0 18  3]
 [ 3  0 11]]
------------------------------
Accuracy_score: 0.7962962962962963
------------------------------
Recall_score: 0.793233082706767
------------------------------
Precision_score: 0.7972841310437219
------------------------------
F1_score: 0.7938912938912939
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.74      0.78        19
           1       0.78      0.86      0.82        21
           2       0.79      0.79      0.79        14

    accuracy                           0.80        54
   macro avg       0.80      0.79      0.79        54
weighted avg       0.80      0.80      0.80        54
```

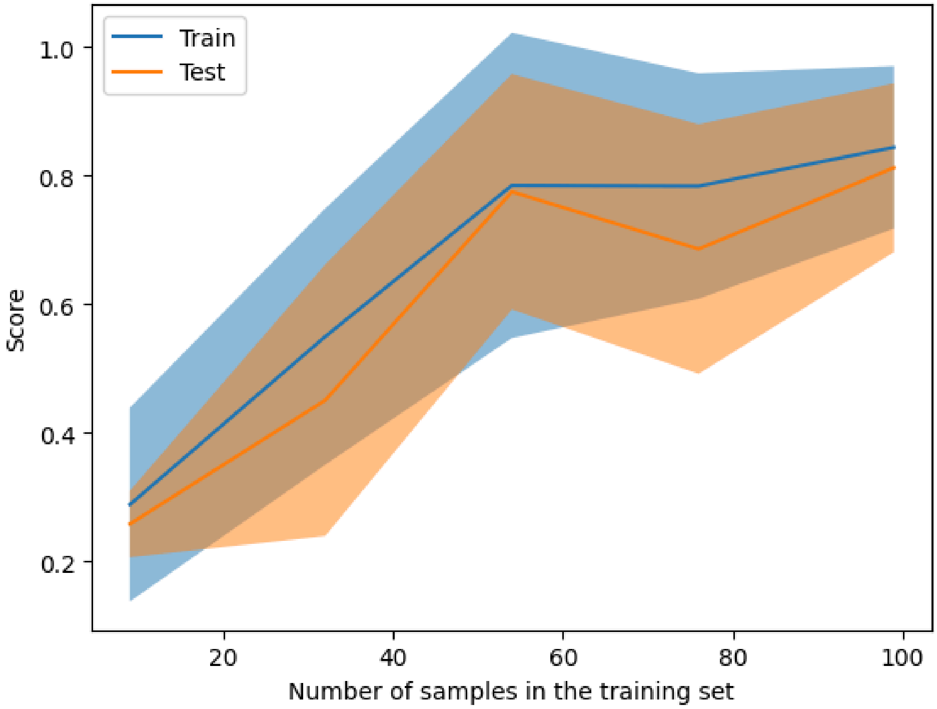
ConfusionMatrix:



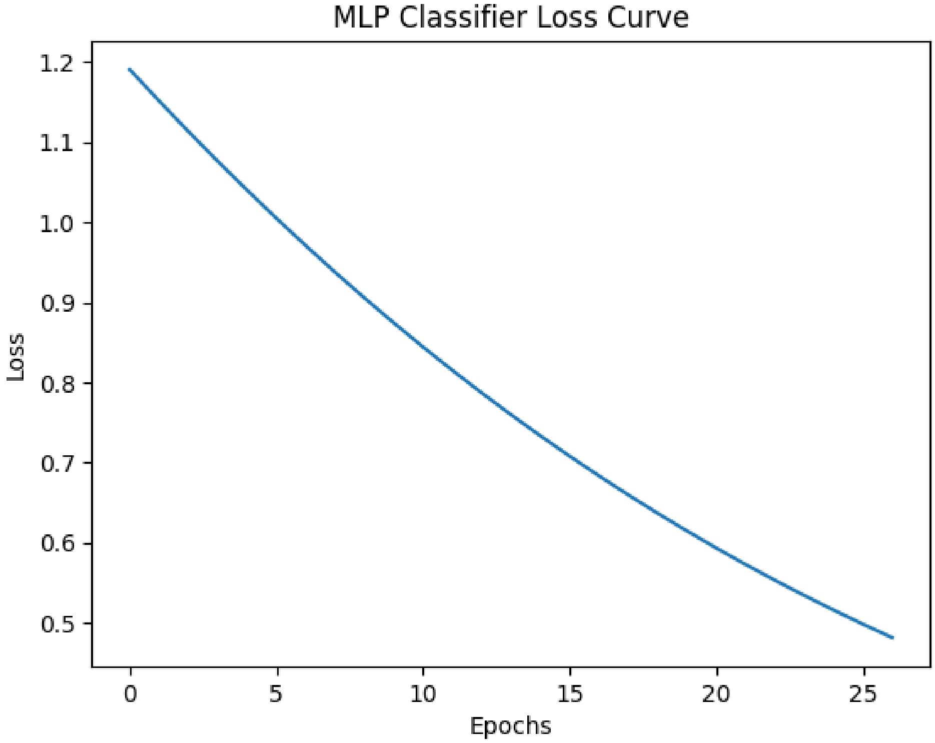
ROC Curve:



LearningCurve:



LossCurve:



After the preliminary testing lets move onto the next phase where we are going to apply PCA on the data and also use parameter tuning. For tuning the parameters, since the dataset is smaller, an exhaustive search to select the parameter where the model is performing optimally without overfitting, by trial-and-error of putting parameters values and measuring the metrics.

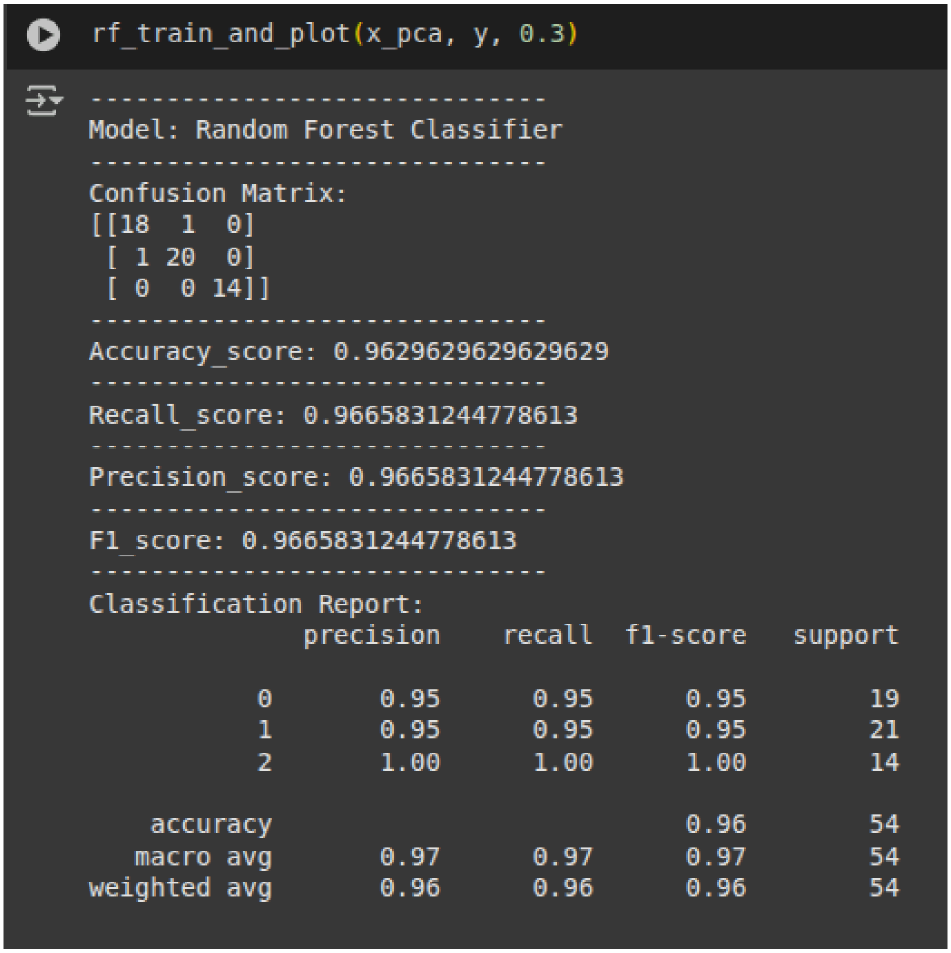
Post training the model with few parameter tuning here is the report:

RandomForestClassifier:

Here are few of the parameters registered for the model that helped squeeze the maximum optimal performance:

Parameter	Value Assigned
Test size	0.3
n_estimator	40
ccp_alpha	0.9
max_depth	10
min_samples	5

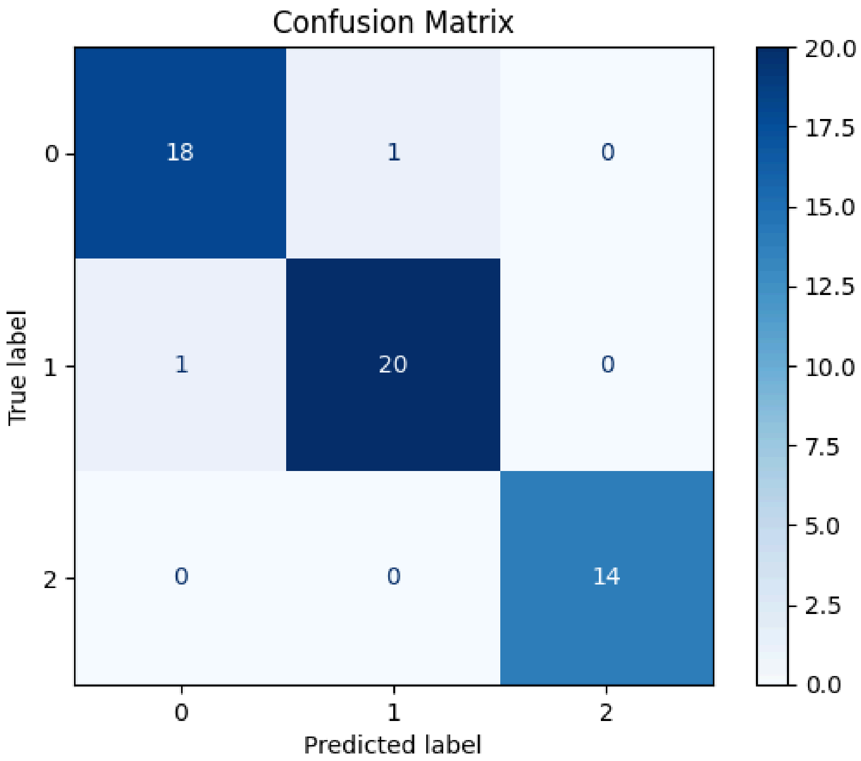
Performance Report:



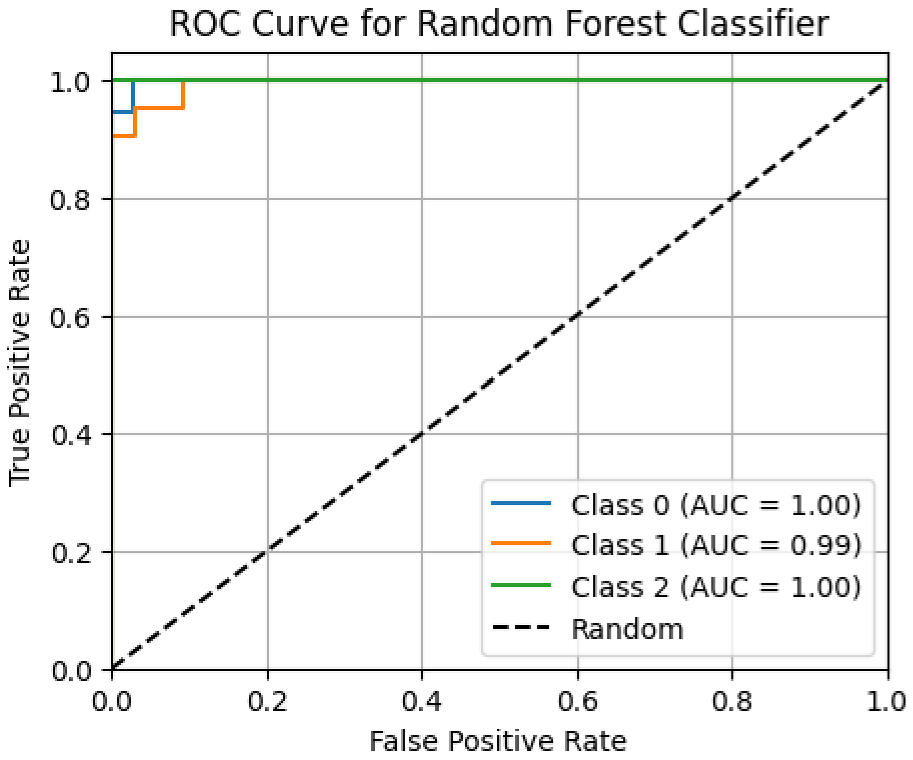
```
rf_train_and_plot(x_pca, y, 0.3)
```

```
------------------------------
Model: Random Forest Classifier
------------------------------
Confusion Matrix:
[[18  1  0]
 [ 1 20  0]
 [ 0  0 14]]
------------------------------
Accuracy_score: 0.9629629629629629
------------------------------
Recall_score: 0.9665831244778613
------------------------------
Precision_score: 0.9665831244778613
------------------------------
F1_score: 0.9665831244778613
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        19
           1       0.95      0.95      0.95        21
           2       1.00      1.00      1.00        14

    accuracy                           0.96        54
   macro avg       0.97      0.97      0.97        54
weighted avg       0.96      0.96      0.96        54
```

ConfusionMatrix:



ROC Curve:

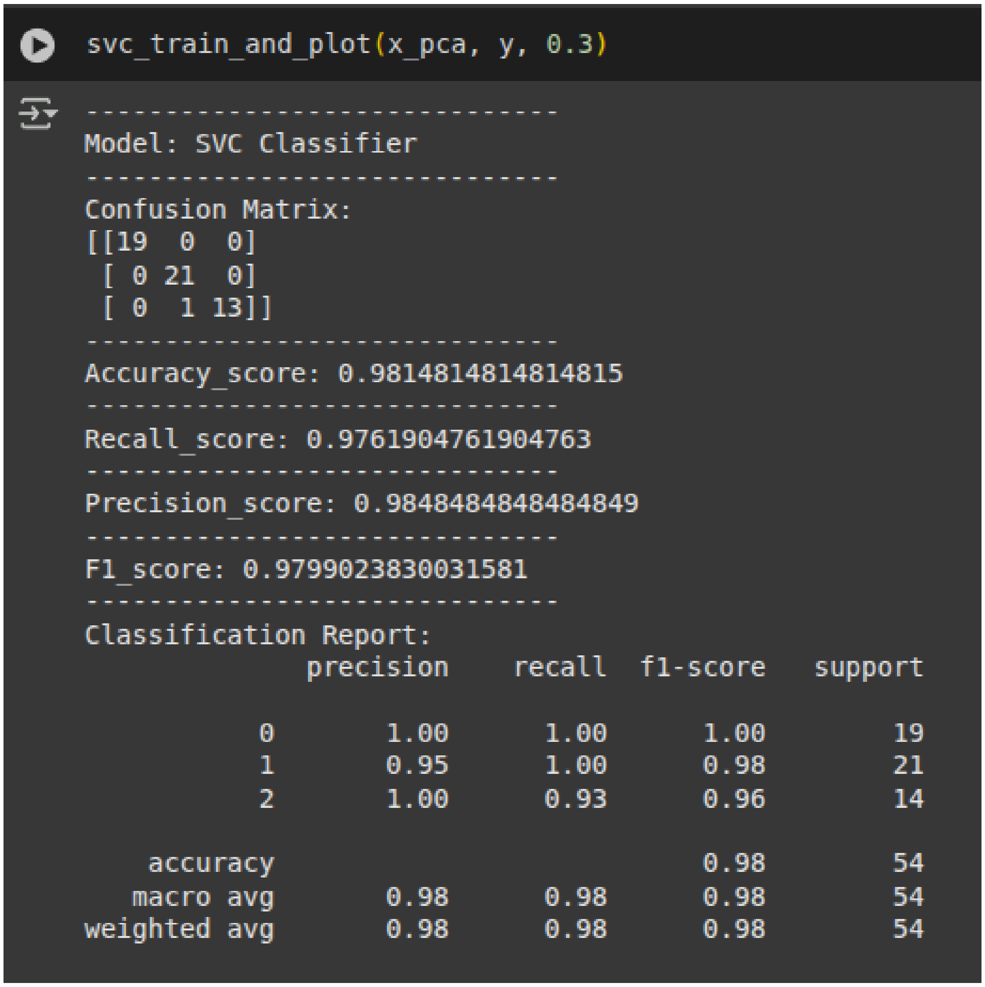


<u>SupportVectorClassifier:</u>

The few svc parameters which are tuned are as follows:

Parameter	Value Assigned
Test size	0.3
Kernel	Linear

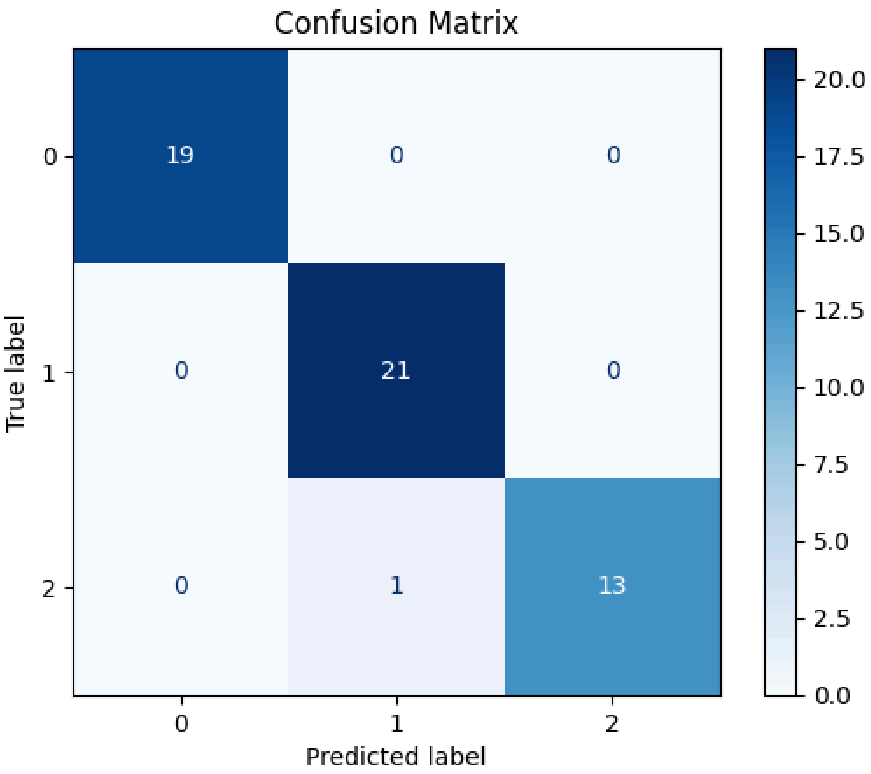
The performance report is as follows:



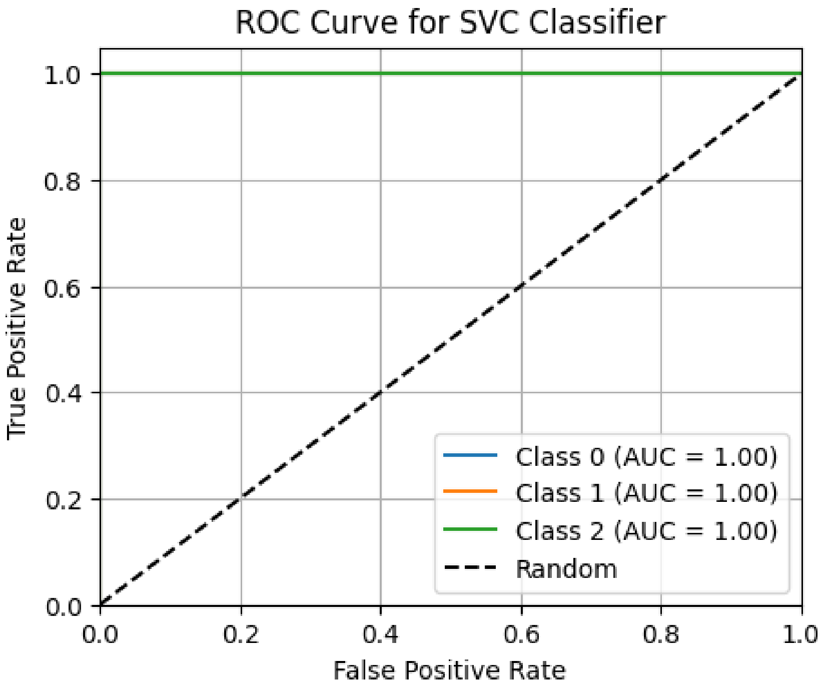
```
svc_train_and_plot(x_pca, y, 0.3)
```

```
------------------------------
Model: SVC Classifier
------------------------------
Confusion Matrix:
[[19  0  0]
 [ 0 21  0]
 [ 0  1 13]]
------------------------------
Accuracy_score: 0.9814814814814815
------------------------------
Recall_score: 0.9761904761904763
------------------------------
Precision_score: 0.9848484848484849
------------------------------
F1_score: 0.9799023830031581
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.95      1.00      0.98        21
           2       1.00      0.93      0.96        14

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

ConfusionMatrix:



ROC Curve:

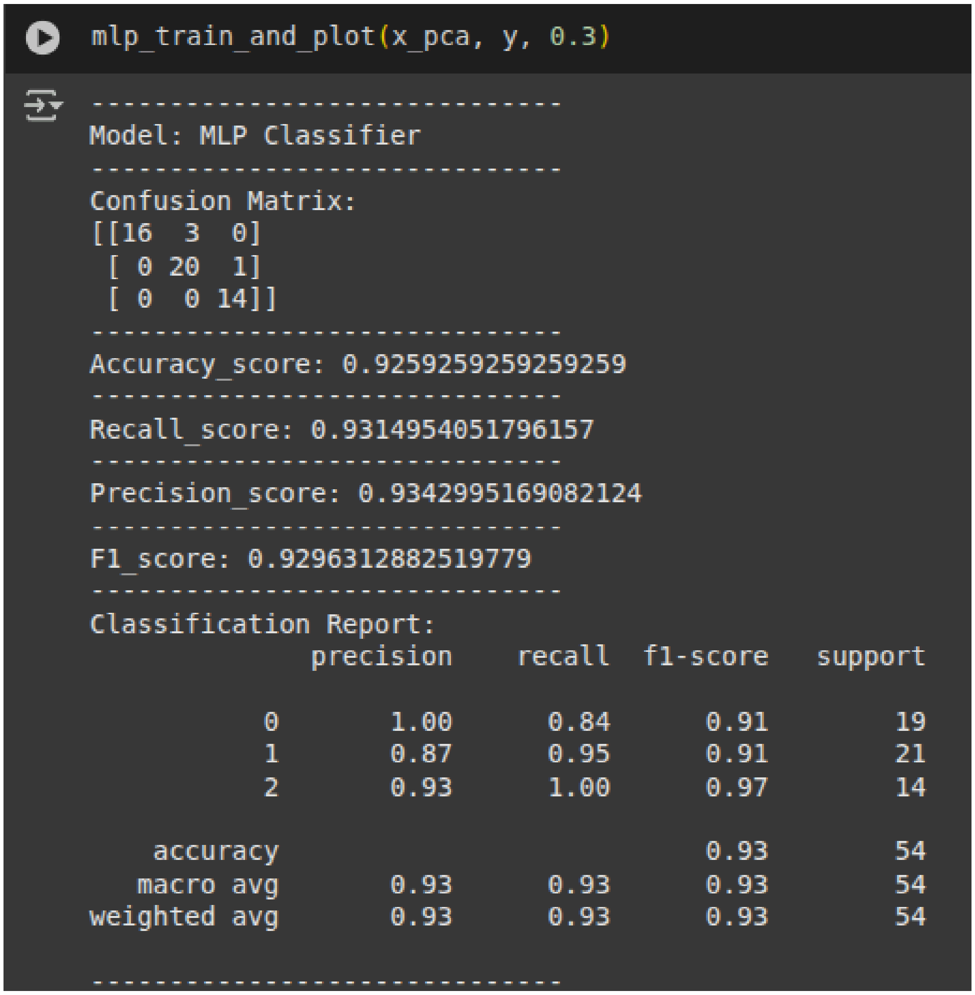


MLPClassifier:

The parameters that are tuned for MLPClassifier are as follows:

Parameter	Value Assigned
Test size	0.3
Max_iter	200 (default)
learning_rate	adaptive
momentum	0.9
early_stopping	True

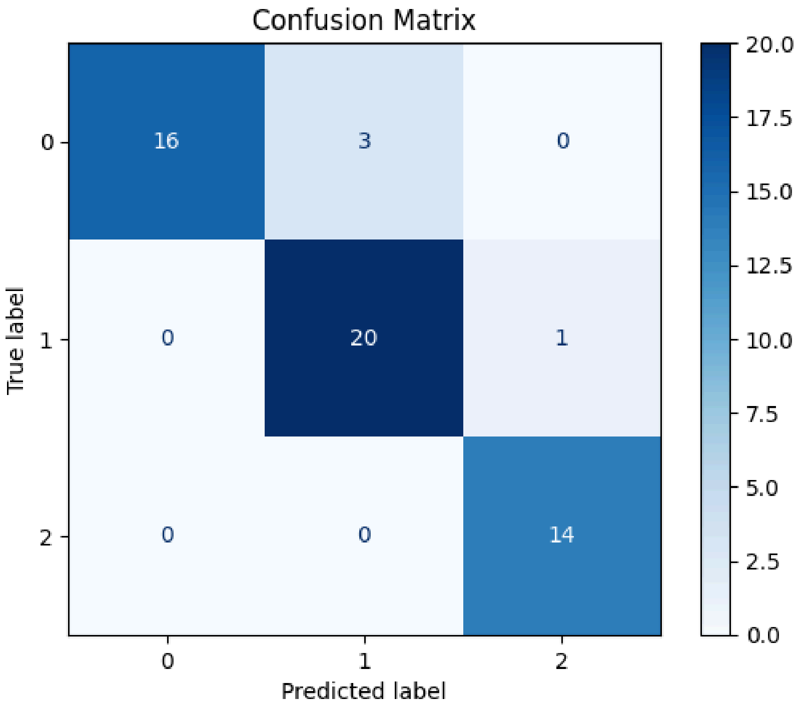
The performance report is as follows:



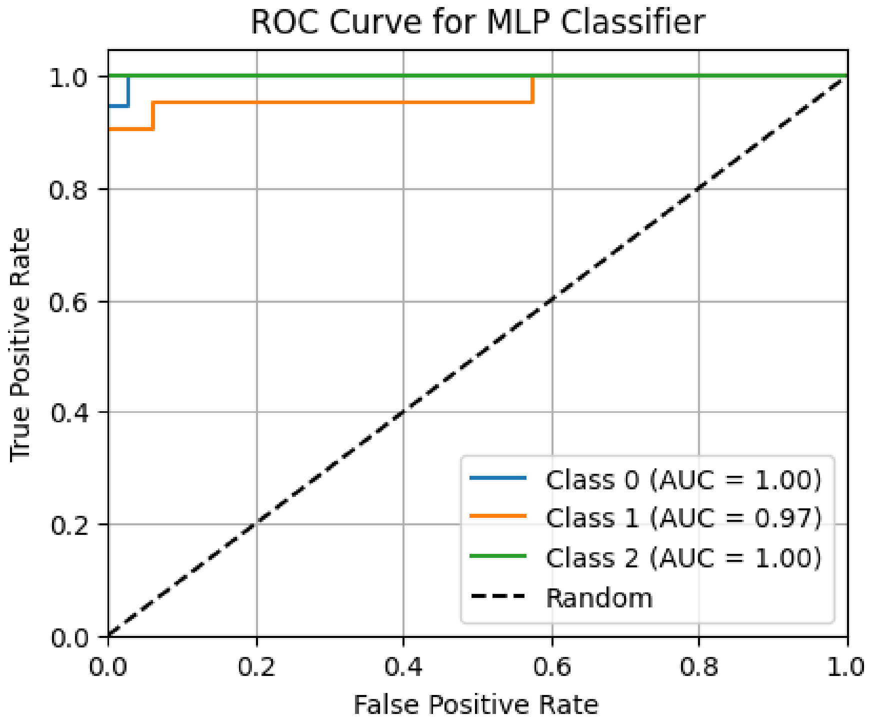
```
mlp_train_and_plot(x_pca, y, 0.3)
```

```
------------------------------
Model: MLP Classifier
------------------------------
Confusion Matrix:
[[16  3  0]
 [ 0 20  1]
 [ 0  0 14]]
------------------------------
Accuracy_score: 0.9259259259259259
------------------------------
Recall_score: 0.9314954051796157
------------------------------
Precision_score: 0.9342995169082124
------------------------------
F1_score: 0.9296312882519779
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.84      0.91        19
           1       0.87      0.95      0.91        21
           2       0.93      1.00      0.97        14

    accuracy                           0.93        54
   macro avg       0.93      0.93      0.93        54
weighted avg       0.93      0.93      0.93        54

------------------------------
```

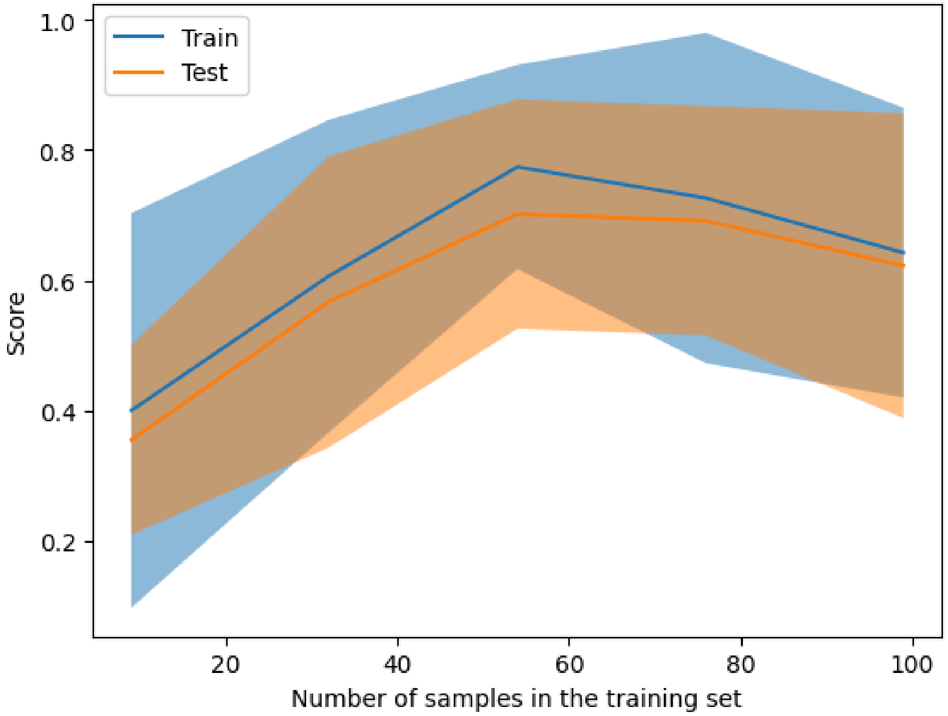
ConfusionMatrix:



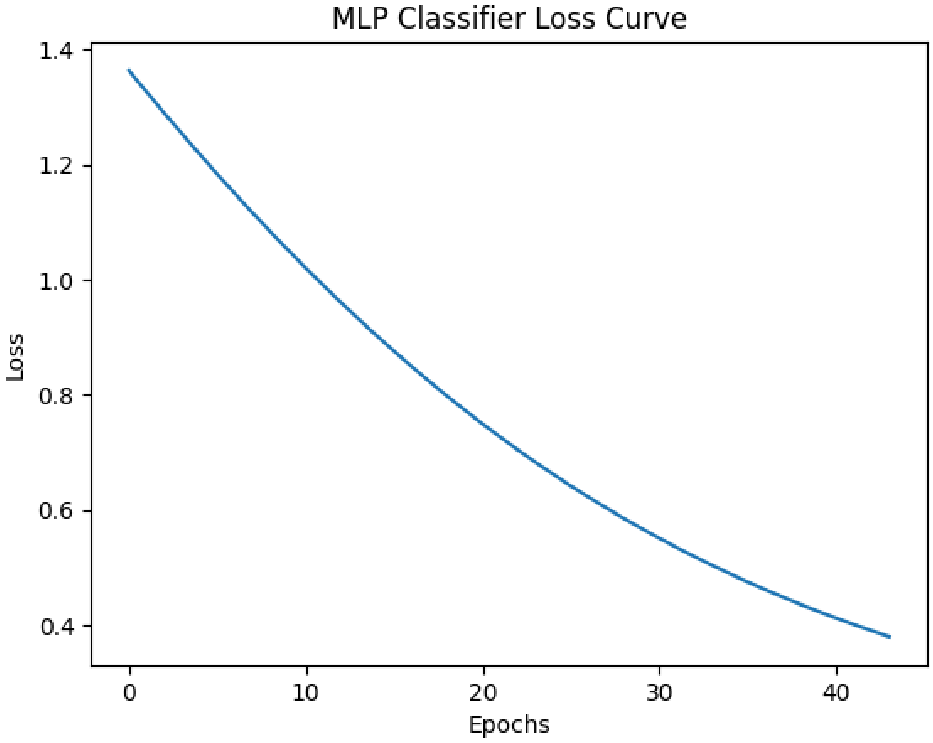
ROC Curve:



LearningCurve:

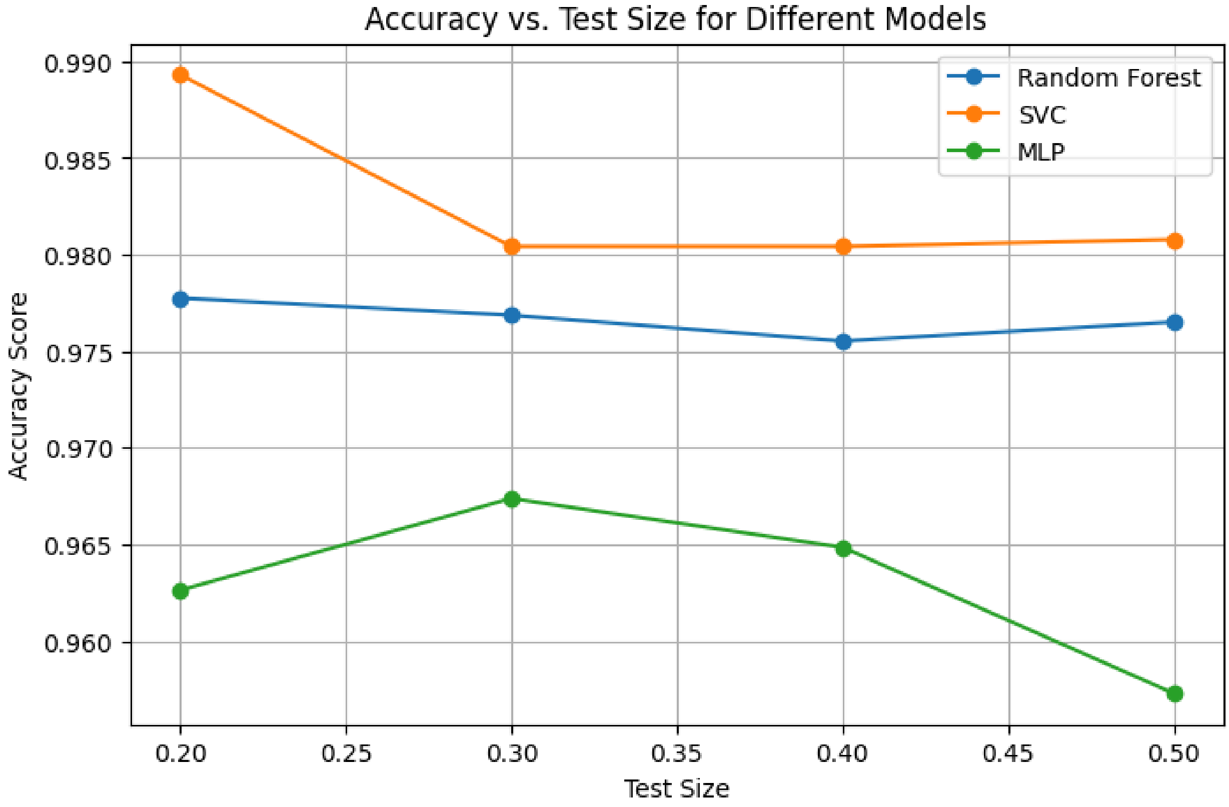


LossCurve:



Handwritten Digit Recognition

There are three models that are used for the purpose: RandomForestClassifier, Support Vector Classifier and MLPClassifier. We tested the performance of the three models for different test data sizes such as 0.2, 0.3 etc. The following is the observation:



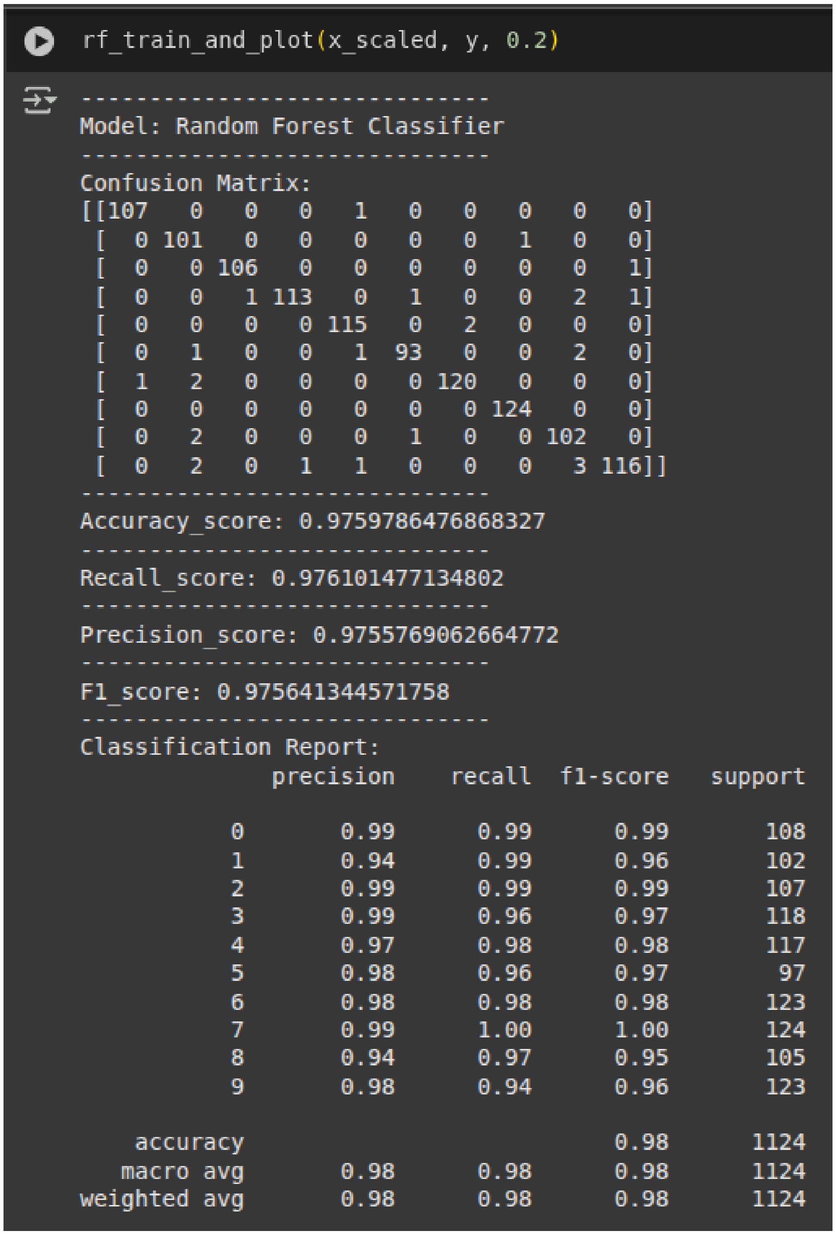
Thus it is observed that without any parameter tuning:

Model Name	Best Test Size	Best Approx. Accuracy Score
RandomForestClassifier	0.2	0.98
SupportVectorClassifier	0.2	0.99
MLPClassifier	0.3	0.975

Thus after training the models on the best observed test size the observation is as follows:

1. **RandomForestClassifier**

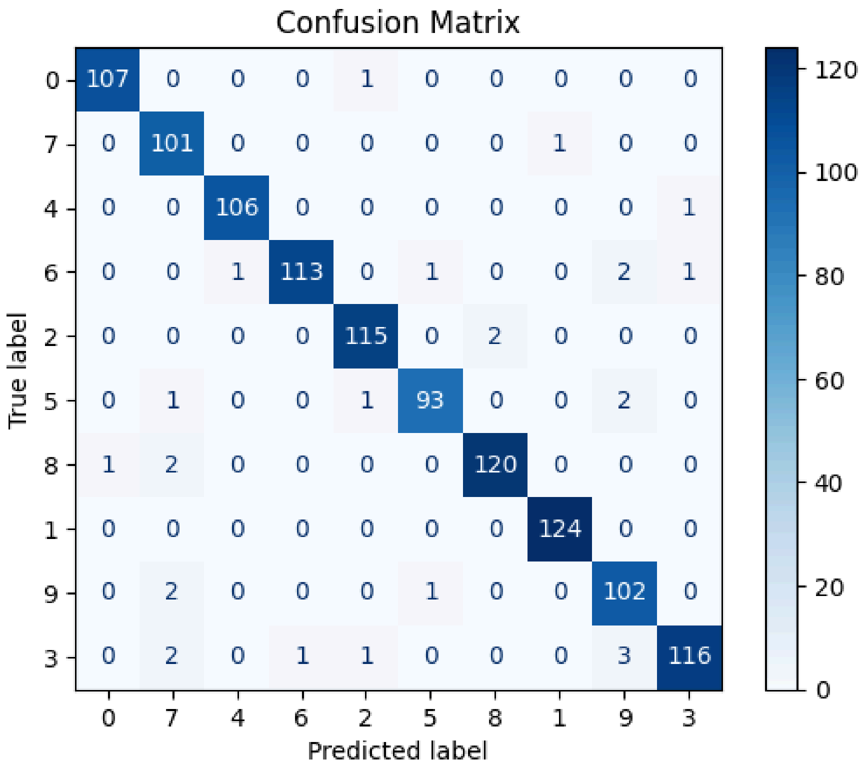
Since RandomForestClassifier performed best at test size= 0.2 thus we trained the model on the observed test size and below is the observed report:



```
rf_train_and_plot(x_scaled, y, 0.2)
```

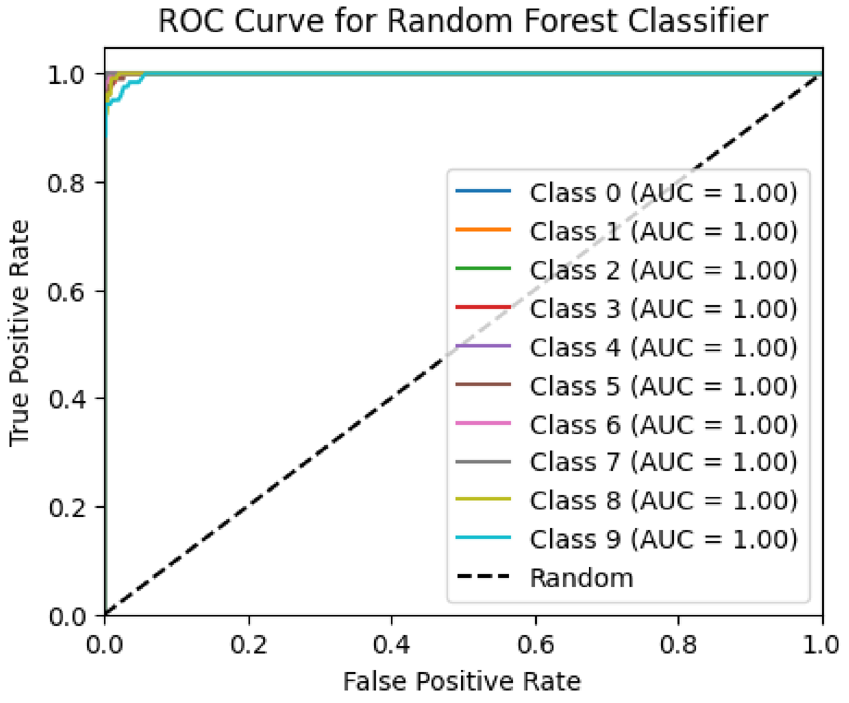
```
-----------------------------
Model: Random Forest Classifier
-----------------------------
Confusion Matrix:
[[107   0   0   0   1   0   0   0   0   0]
 [  0 101   0   0   0   0   0   1   0   0]
 [  0   0 106   0   0   0   0   0   0   1]
 [  0   0   1 113   0   1   0   0   2   1]
 [  0   0   0   0 115   0   2   0   0   0]
 [  0   1   0   0   1  93   0   0   2   0]
 [  1   2   0   0   0   0 120   0   0   0]
 [  0   0   0   0   0   0   0 124   0   0]
 [  0   2   0   0   0   1   0   0 102   0]
 [  0   2   0   1   1   0   0   0   3 116]]
-----------------------------
Accuracy_score: 0.9759786476868327
-----------------------------
Recall_score: 0.976101477134802
-----------------------------
Precision_score: 0.9755769062664772
-----------------------------
F1_score: 0.975641344571758
-----------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       108
           1       0.94      0.99      0.96       102
           2       0.99      0.99      0.99       107
           3       0.99      0.96      0.97       118
           4       0.97      0.98      0.98       117
           5       0.98      0.96      0.97        97
           6       0.98      0.98      0.98       123
           7       0.99      1.00      1.00       124
           8       0.94      0.97      0.95       105
           9       0.98      0.94      0.96       123

    accuracy                           0.98      1124
   macro avg       0.98      0.98      0.98      1124
weighted avg       0.98      0.98      0.98      1124
```

Confusion Matrix:



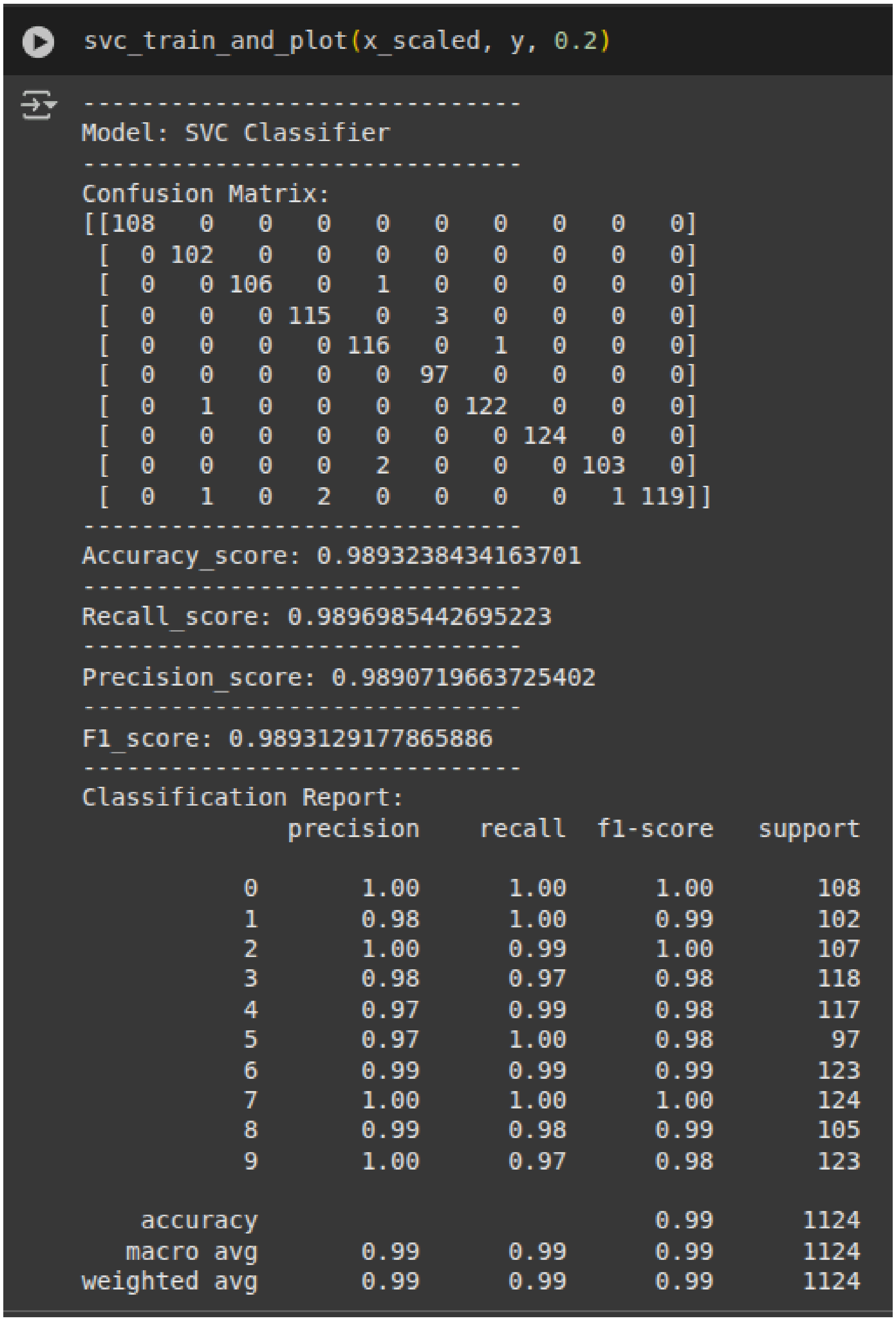
	0	7	4	6	2	5	8	1	9	3
0	107	0	0	0	1	0	0	0	0	0
7	0	101	0	0	0	0	0	1	0	0
4	0	0	106	0	0	0	0	0	0	1
6	0	0	1	113	0	1	0	0	2	1
2	0	0	0	0	115	0	2	0	0	0
5	0	1	0	0	1	93	0	0	2	0
8	1	2	0	0	0	0	120	0	0	0
1	0	0	0	0	0	0	0	124	0	0
9	0	2	0	0	0	1	0	0	102	0
3	0	2	0	1	1	0	0	0	3	116

ROC Curve



2. SupportVectorClassifier

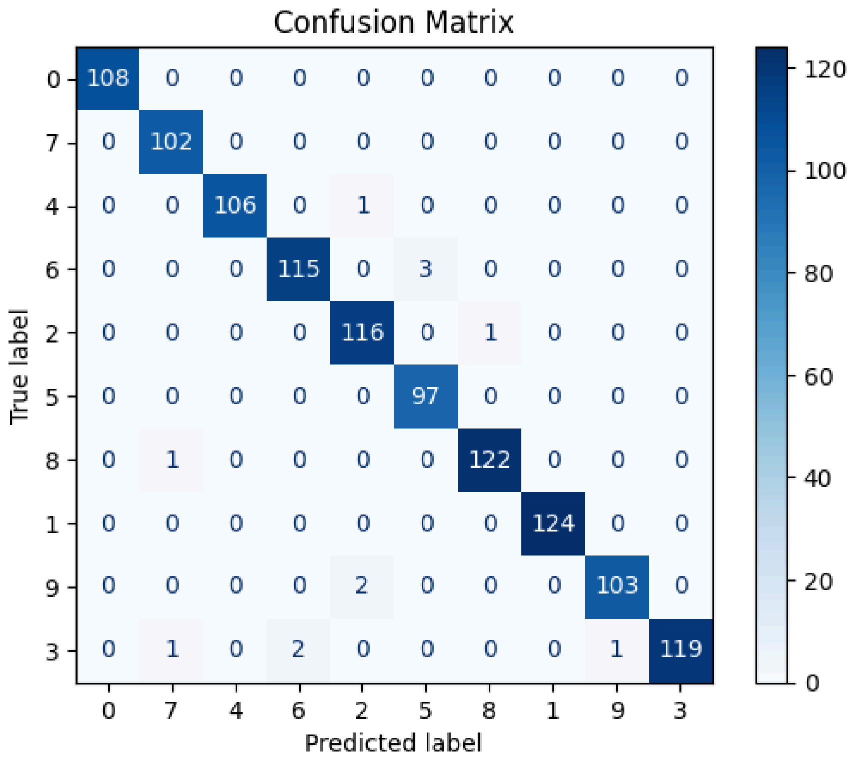
Since SupportVectorClassifier performed its best at test size= 0.2 so trained the model with the optimal performance test size. The observed report is as follows:



```
svc_train_and_plot(x_scaled, y, 0.2)
```

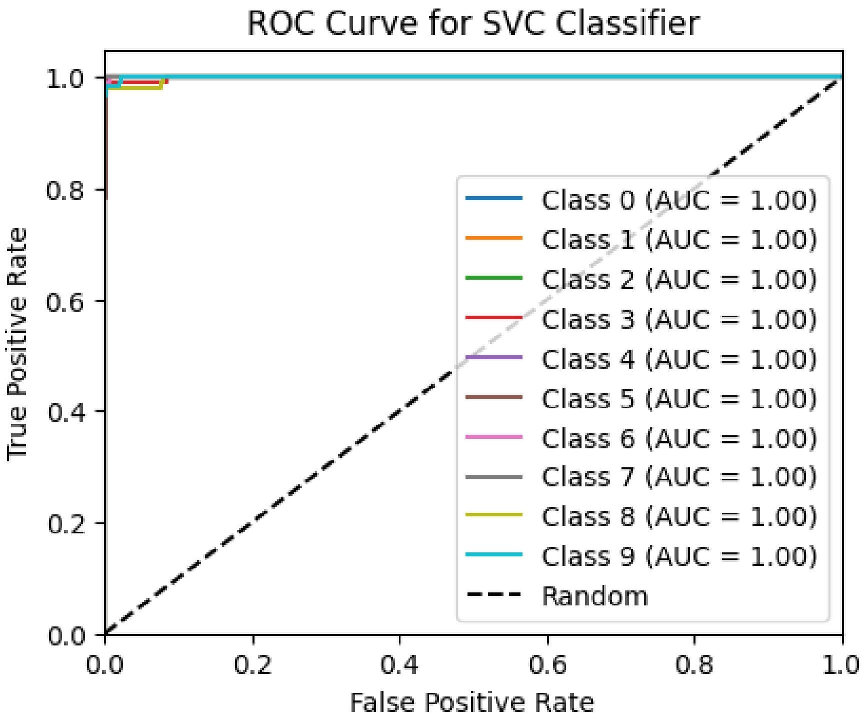
```
------------------------------
Model: SVC Classifier
------------------------------
Confusion Matrix:
[[108   0   0   0   0   0   0   0   0   0]
 [  0 102   0   0   0   0   0   0   0   0]
 [  0   0 106   0   1   0   0   0   0   0]
 [  0   0   0 115   0   3   0   0   0   0]
 [  0   0   0   0 116   0   1   0   0   0]
 [  0   0   0   0   0  97   0   0   0   0]
 [  0   1   0   0   0   0 122   0   0   0]
 [  0   0   0   0   0   0   0 124   0   0]
 [  0   0   0   0   2   0   0   0 103   0]
 [  0   1   0   2   0   0   0   0   1 119]]
------------------------------
Accuracy_score: 0.9893238434163701
------------------------------
Recall_score: 0.9896985442695223
------------------------------
Precision_score: 0.9890719663725402
------------------------------
F1_score: 0.9893129177865886
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       108
           1       0.98      1.00      0.99       102
           2       1.00      0.99      1.00       107
           3       0.98      0.97      0.98       118
           4       0.97      0.99      0.98       117
           5       0.97      1.00      0.98        97
           6       0.99      0.99      0.99       123
           7       1.00      1.00      1.00       124
           8       0.99      0.98      0.99       105
           9       1.00      0.97      0.98       123

    accuracy                           0.99      1124
   macro avg       0.99      0.99      0.99      1124
weighted avg       0.99      0.99      0.99      1124
```

Confusion Matrix:



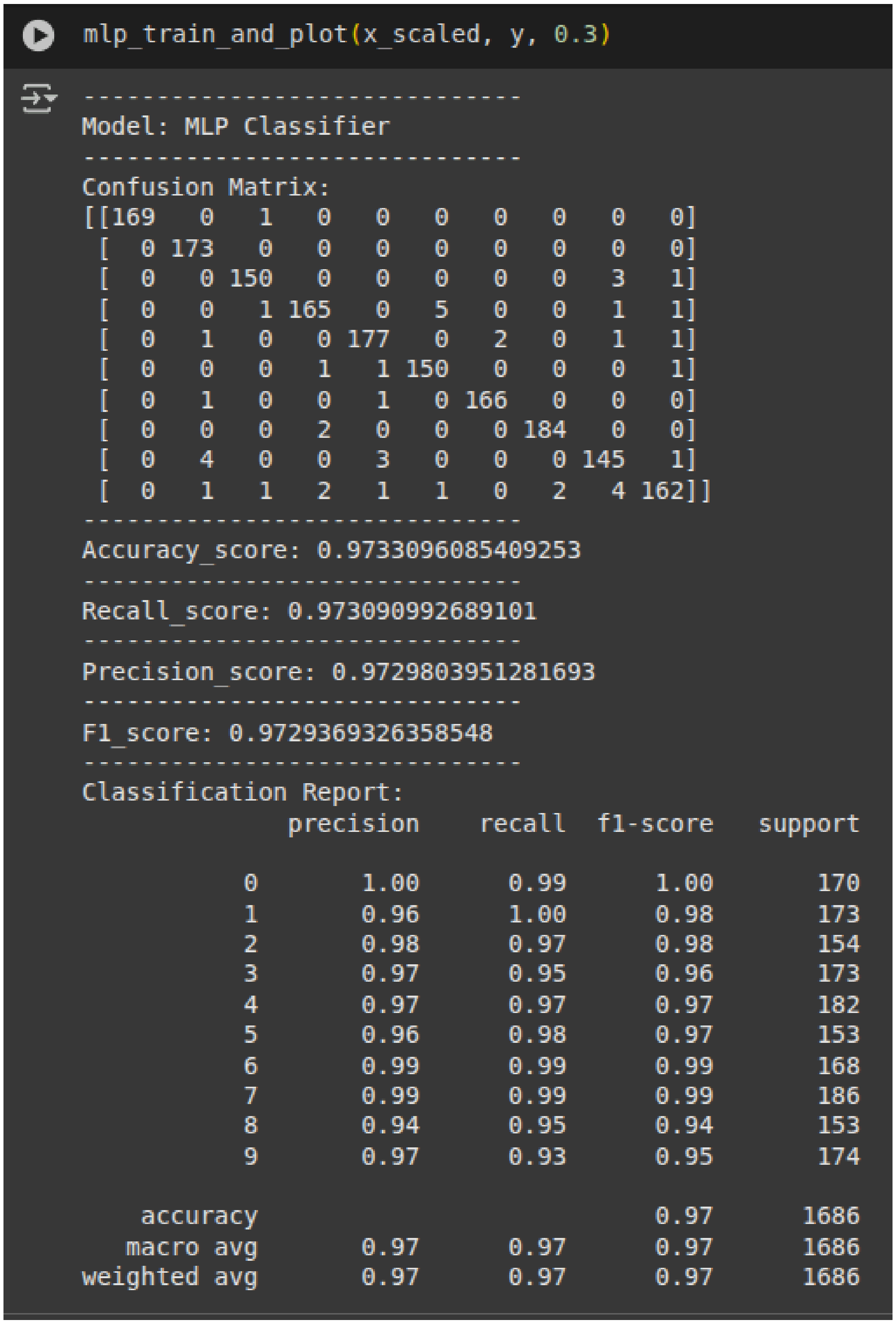
	0	7	4	6	2	5	8	1	9	3
0	108	0	0	0	0	0	0	0	0	0
7	0	102	0	0	0	0	0	0	0	0
4	0	0	106	0	1	0	0	0	0	0
6	0	0	0	115	0	3	0	0	0	0
2	0	0	0	0	116	0	1	0	0	0
5	0	0	0	0	0	97	0	0	0	0
8	0	1	0	0	0	0	122	0	0	0
1	0	0	0	0	0	0	0	124	0	0
9	0	0	0	0	2	0	0	0	103	0
3	0	1	0	2	0	0	0	0	1	119

ROC Curve:



3. **MLPClassifier**

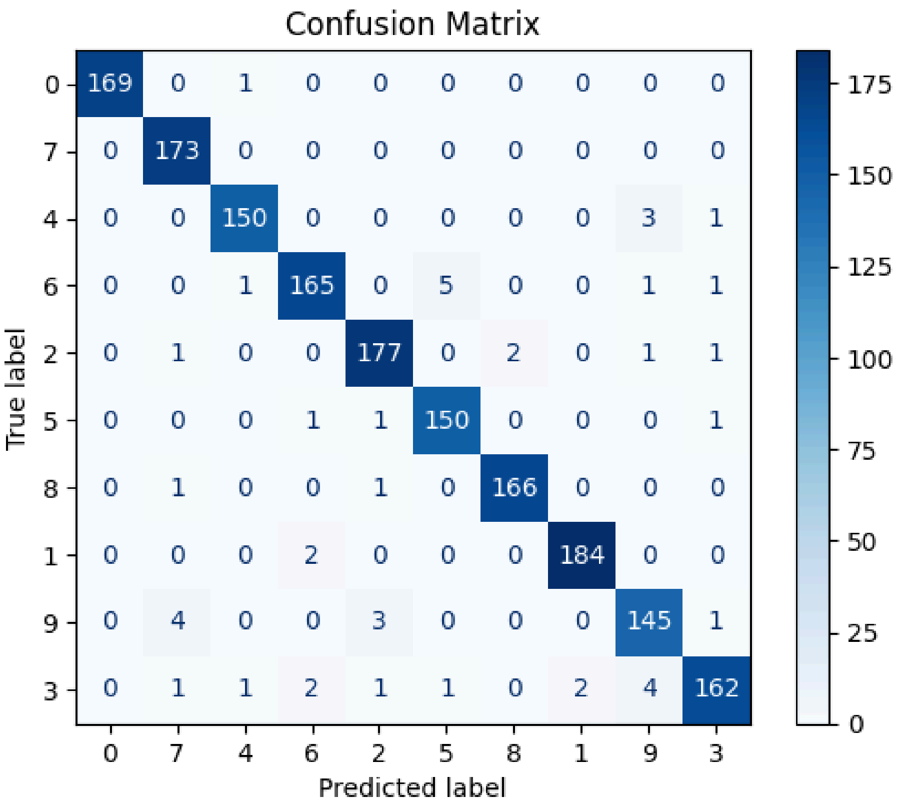
Since MLPClassifier performed optimally at test size= 0.3 so, trained the model on the optimal performing test size. The observed report is as follows:



```
mlp_train_and_plot(x_scaled, y, 0.3)
```

```
------------------------------
Model: MLP Classifier
------------------------------
Confusion Matrix:
[[169   0   1   0   0   0   0   0   0   0]
 [  0 173   0   0   0   0   0   0   0   0]
 [  0   0 150   0   0   0   0   0   3   1]
 [  0   0   1 165   0   5   0   0   1   1]
 [  0   1   0   0 177   0   2   0   1   1]
 [  0   0   0   1   1 150   0   0   0   1]
 [  0   1   0   0   1   0 166   0   0   0]
 [  0   0   0   2   0   0   0 184   0   0]
 [  0   4   0   0   3   0   0   0 145   1]
 [  0   1   1   2   1   1   0   2   4 162]]
------------------------------
Accuracy_score: 0.9733096085409253
------------------------------
Recall_score: 0.973090992689101
------------------------------
Precision_score: 0.9729803951281693
------------------------------
F1_score: 0.9729369326358548
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       170
           1       0.96      1.00      0.98       173
           2       0.98      0.97      0.98       154
           3       0.97      0.95      0.96       173
           4       0.97      0.97      0.97       182
           5       0.96      0.98      0.97       153
           6       0.99      0.99      0.99       168
           7       0.99      0.99      0.99       186
           8       0.94      0.95      0.94       153
           9       0.97      0.93      0.95       174

    accuracy                           0.97      1686
   macro avg       0.97      0.97      0.97      1686
weighted avg       0.97      0.97      0.97      1686
```

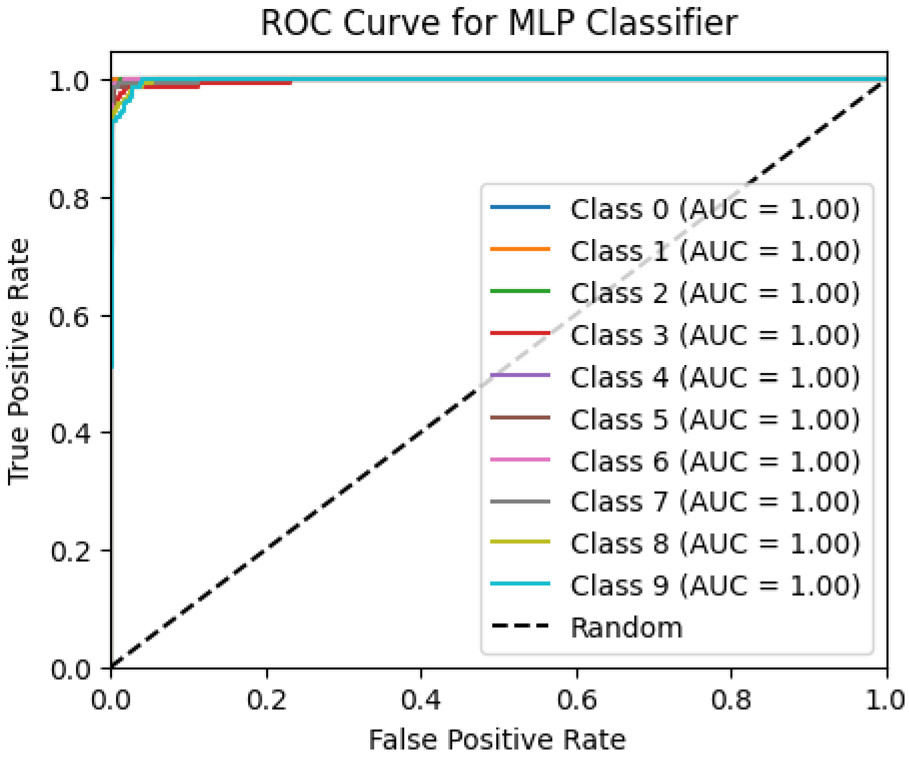
Confusion Matrix:



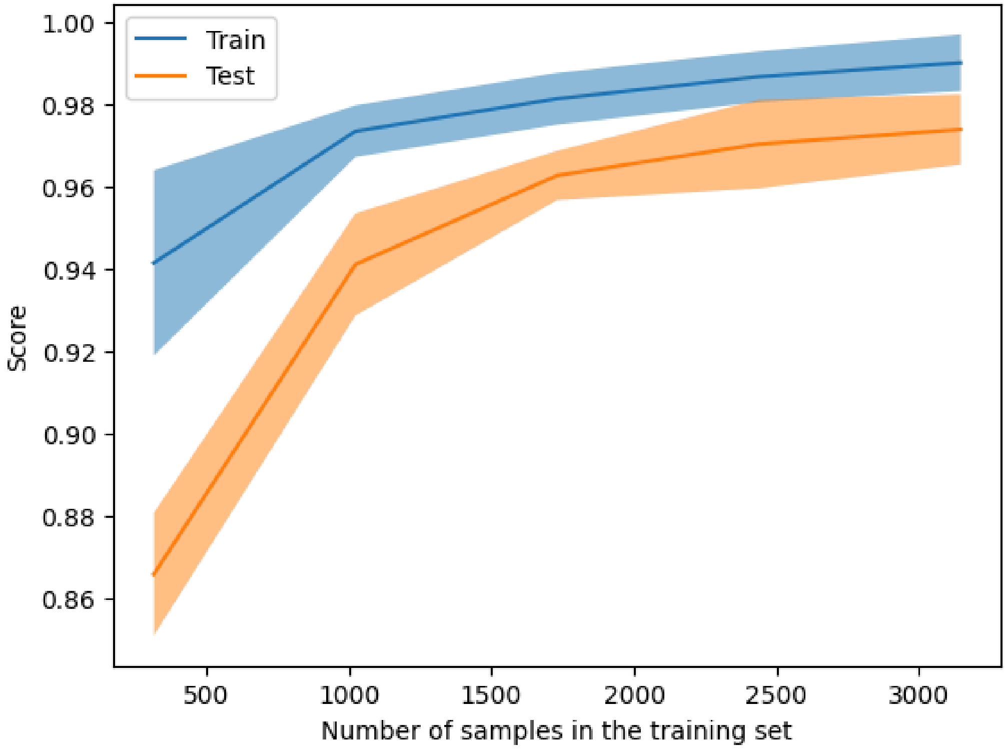
Confusion Matrix

True label \ Predicted label	0	7	4	6	2	5	8	1	9	3
0	169	0	1	0	0	0	0	0	0	0
7	0	173	0	0	0	0	0	0	0	0
4	0	0	150	0	0	0	0	0	3	1
6	0	0	1	165	0	5	0	0	1	1
2	0	1	0	0	177	0	2	0	1	1
5	0	0	0	1	1	150	0	0	0	1
8	0	1	0	0	1	0	166	0	0	0
1	0	0	0	2	0	0	0	184	0	0
9	0	4	0	0	3	0	0	0	145	1
3	0	1	1	2	1	1	0	2	4	162

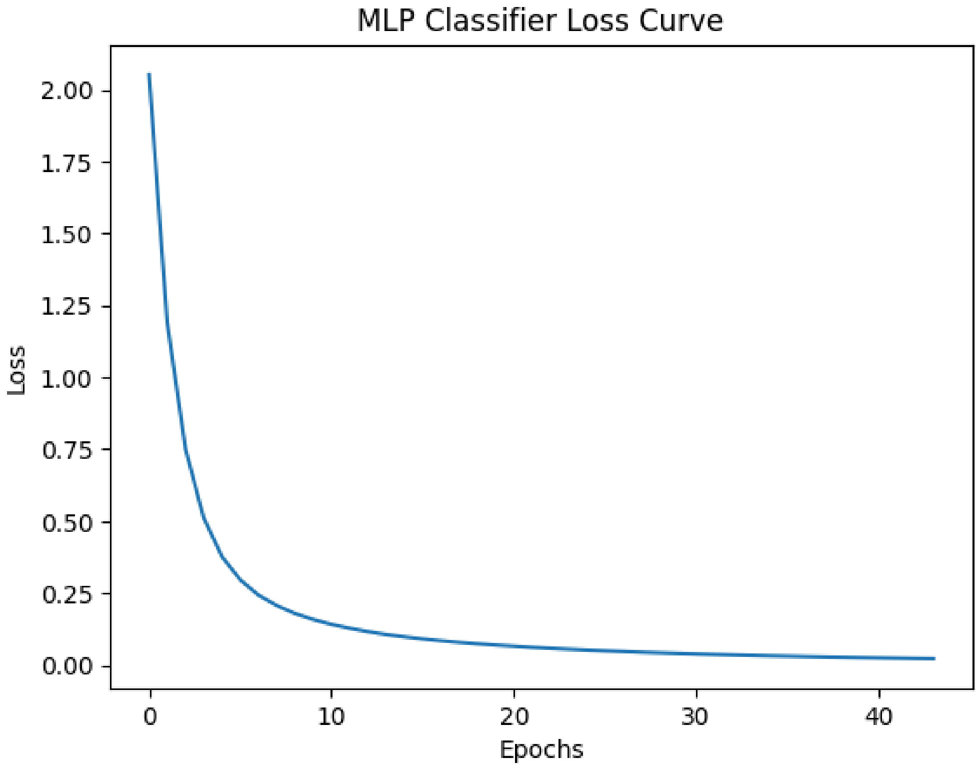
ROC Curve:



Learning Curve:



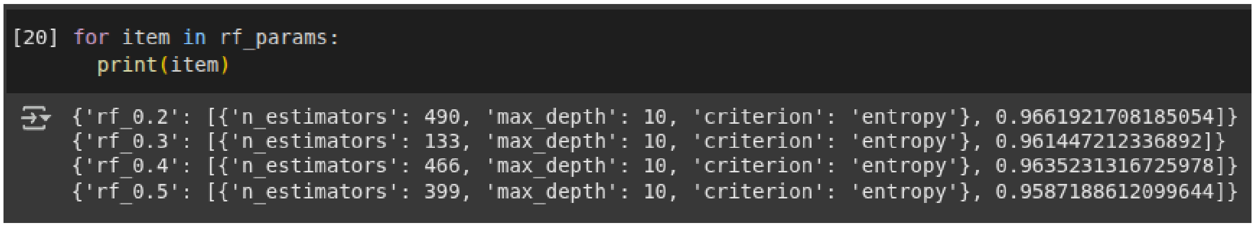
Loss Curve:



After the preliminary testing lets move onto the next phase where we are going to apply PCA on the data and also use parameter tuning. For parameter tuning in order to find the best parameter we are going to use Optuna for parameter tuning. Also we are going to find the test size that helps to maximize the performance of the model.

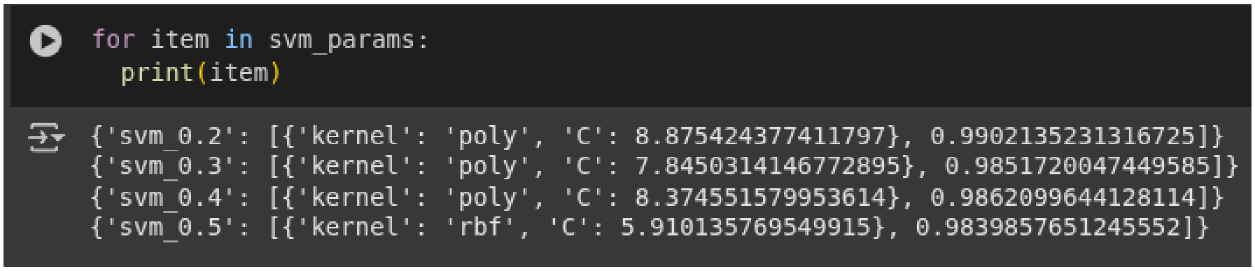
The best parameters obtained after parameter tuning:

- **RandomForestClassifier**



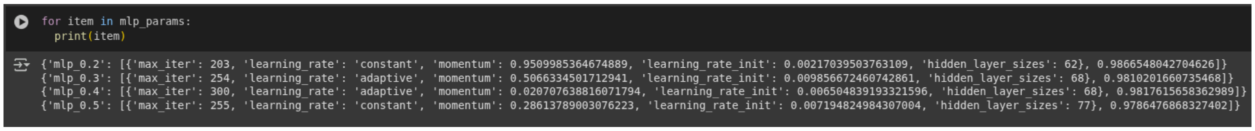
```
[20] for item in rf_params:
       print(item)

{'rf_0.2': [{'n_estimators': 490, 'max_depth': 10, 'criterion': 'entropy'}, 0.9661921708185054]}
{'rf_0.3': [{'n_estimators': 133, 'max_depth': 10, 'criterion': 'entropy'}, 0.961447212336892]}
{'rf_0.4': [{'n_estimators': 466, 'max_depth': 10, 'criterion': 'entropy'}, 0.9635231316725978]}
{'rf_0.5': [{'n_estimators': 399, 'max_depth': 10, 'criterion': 'entropy'}, 0.9587188612099644]}
```

- **SupportVectorClassifier**



```
for item in svm_params:
  print(item)

{'svm_0.2': [{'kernel': 'poly', 'C': 8.875424377411797}, 0.9902135231316725]}
{'svm_0.3': [{'kernel': 'poly', 'C': 7.8450314146772895}, 0.9851720047449585]}
{'svm_0.4': [{'kernel': 'poly', 'C': 8.374551579953614}, 0.9862099644128114]}
{'svm_0.5': [{'kernel': 'rbf', 'C': 5.910135769549915}, 0.9839857651245552]}
```

- **MLPClassifier**

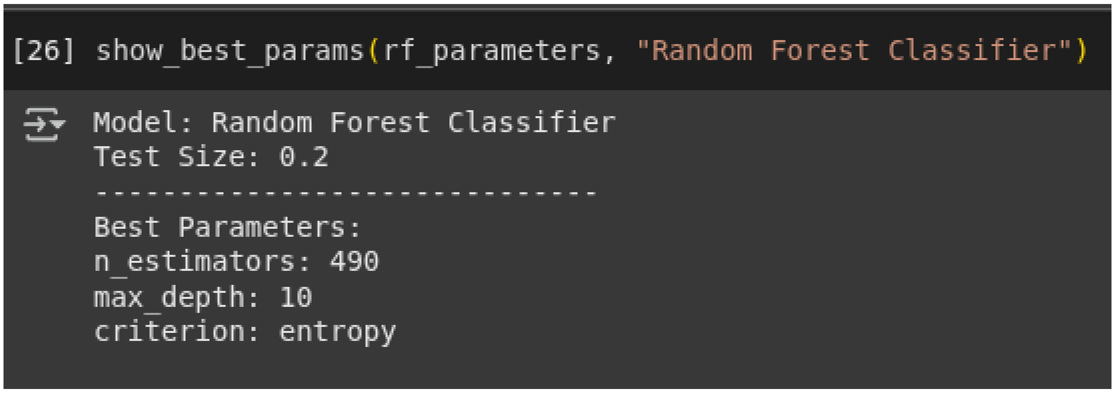


```
for item in mlp_params:
  print(item)

{'mlp_0.2': [{'max_iter': 203, 'learning_rate': 'constant', 'momentum': 0.9509985364674889, 'learning_rate_init': 0.00217039503763109, 'hidden_layer_sizes': 62}, 0.9866548042704626]}
{'mlp_0.3': [{'max_iter': 254, 'learning_rate': 'adaptive', 'momentum': 0.5066334501712941, 'learning_rate_init': 0.009856672460742861, 'hidden_layer_sizes': 68}, 0.9810201660735468]}
{'mlp_0.4': [{'max_iter': 300, 'learning_rate': 'adaptive', 'momentum': 0.020707638816071794, 'learning_rate_init': 0.006504839193321596, 'hidden_layer_sizes': 68}, 0.9817615658362989]}
{'mlp_0.5': [{'max_iter': 255, 'learning_rate': 'constant', 'momentum': 0.28613789003076223, 'learning_rate_init': 0.007194824984307004, 'hidden_layer_sizes': 77}, 0.9786476868327402]}
```

Post training the models using the best parameters the below is the observed report:

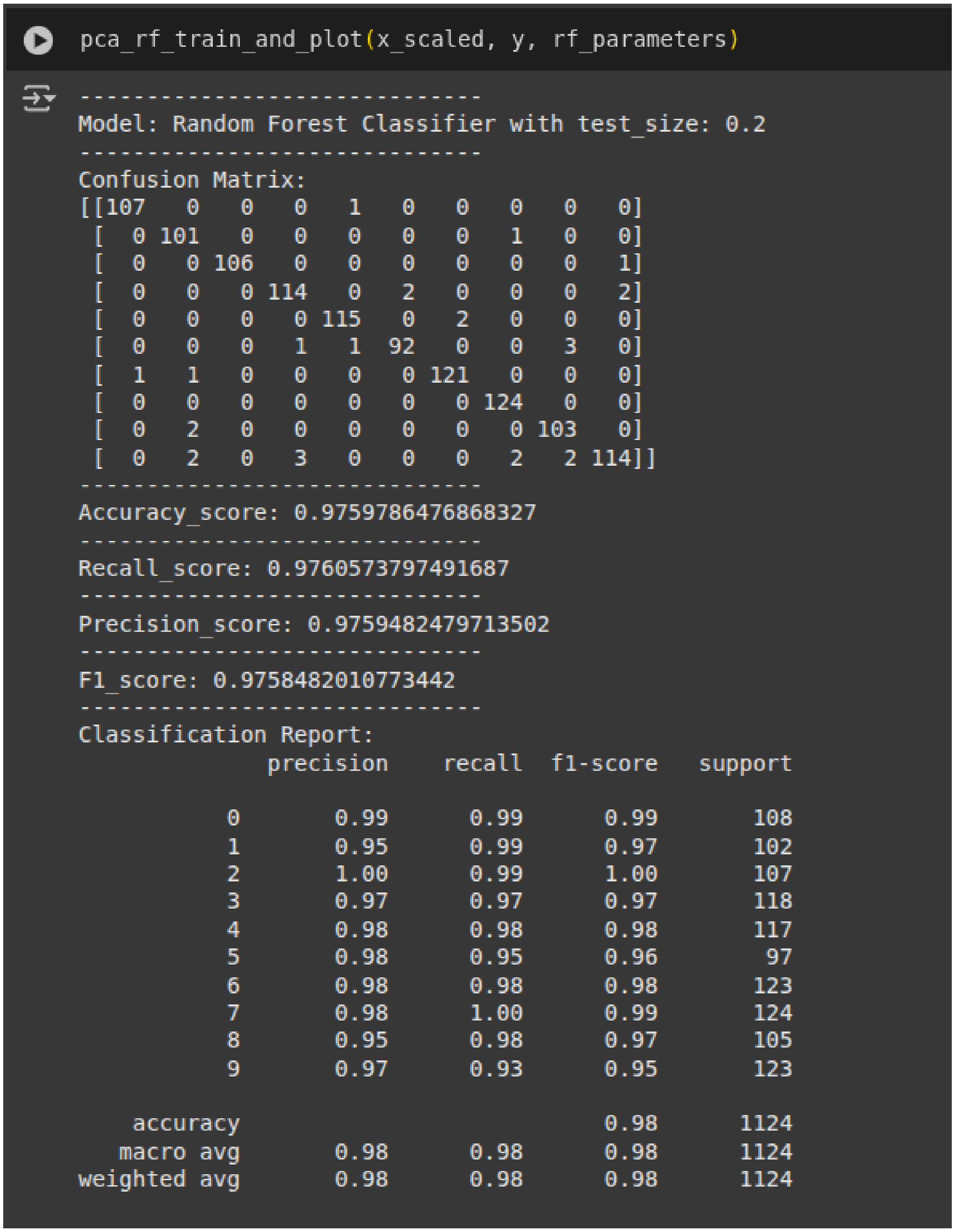
RandomForestClassifier

The best parameters after parameter tuning RandomForestClassifier:



```
[26] show_best_params(rf_parameters, "Random Forest Classifier")

     Model: Random Forest Classifier
     Test Size: 0.2
     ------------------------------
     Best Parameters:
     n_estimators: 490
     max_depth: 10
     criterion: entropy
```

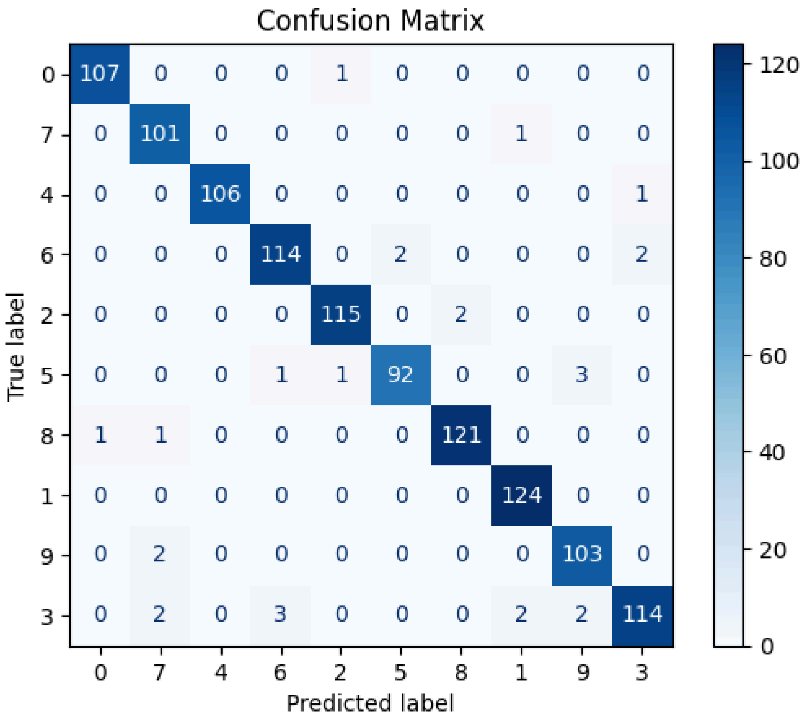
The performance report are:



```
pca_rf_train_and_plot(x_scaled, y, rf_parameters)
```

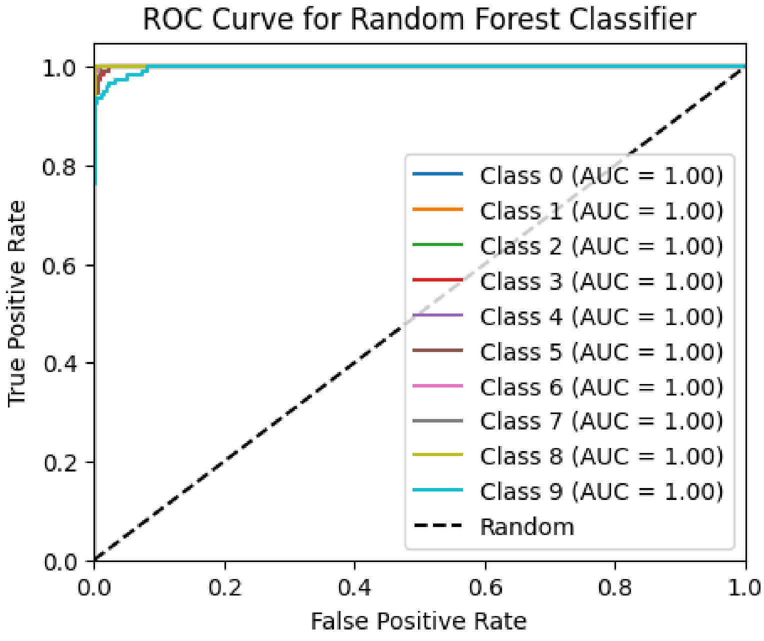
```
------------------------------
Model: Random Forest Classifier with test_size: 0.2
------------------------------
Confusion Matrix:
[[107   0   0   0   1   0   0   0   0   0]
 [  0 101   0   0   0   0   0   1   0   0]
 [  0   0 106   0   0   0   0   0   0   1]
 [  0   0   0 114   0   2   0   0   0   2]
 [  0   0   0   0 115   0   2   0   0   0]
 [  0   0   0   1   1  92   0   0   3   0]
 [  1   1   0   0   0   0 121   0   0   0]
 [  0   0   0   0   0   0   0 124   0   0]
 [  0   2   0   0   0   0   0   0 103   0]
 [  0   2   0   3   0   0   0   2   2 114]]
------------------------------
Accuracy_score: 0.9759786476868327
------------------------------
Recall_score: 0.9760573797491687
------------------------------
Precision_score: 0.9759482479713502
------------------------------
F1_score: 0.9758482010773442
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       108
           1       0.95      0.99      0.97       102
           2       1.00      0.99      1.00       107
           3       0.97      0.97      0.97       118
           4       0.98      0.98      0.98       117
           5       0.98      0.95      0.96        97
           6       0.98      0.98      0.98       123
           7       0.98      1.00      0.99       124
           8       0.95      0.98      0.97       105
           9       0.97      0.93      0.95       123

    accuracy                           0.98      1124
   macro avg       0.98      0.98      0.98      1124
weighted avg       0.98      0.98      0.98      1124
```

Confusion Matrix:



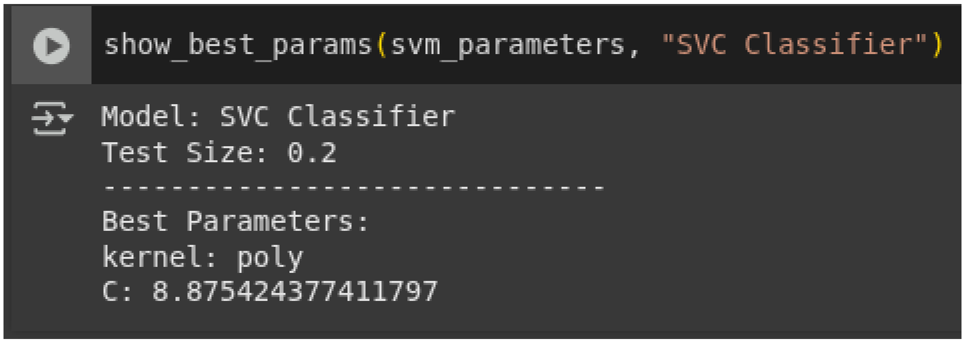
True label \ Predicted label	0	7	4	6	2	5	8	1	9	3
0	107	0	0	0	1	0	0	0	0	0
7	0	101	0	0	0	0	0	1	0	0
4	0	0	106	0	0	0	0	0	0	1
6	0	0	0	114	0	2	0	0	0	2
2	0	0	0	0	115	0	2	0	0	0
5	0	0	0	1	1	92	0	0	3	0
8	1	1	0	0	0	0	121	0	0	0
1	0	0	0	0	0	0	0	124	0	0
9	0	2	0	0	0	0	0	0	103	0
3	0	2	0	3	0	0	0	2	2	114

ROC Curve:



SupportVectorClassifier

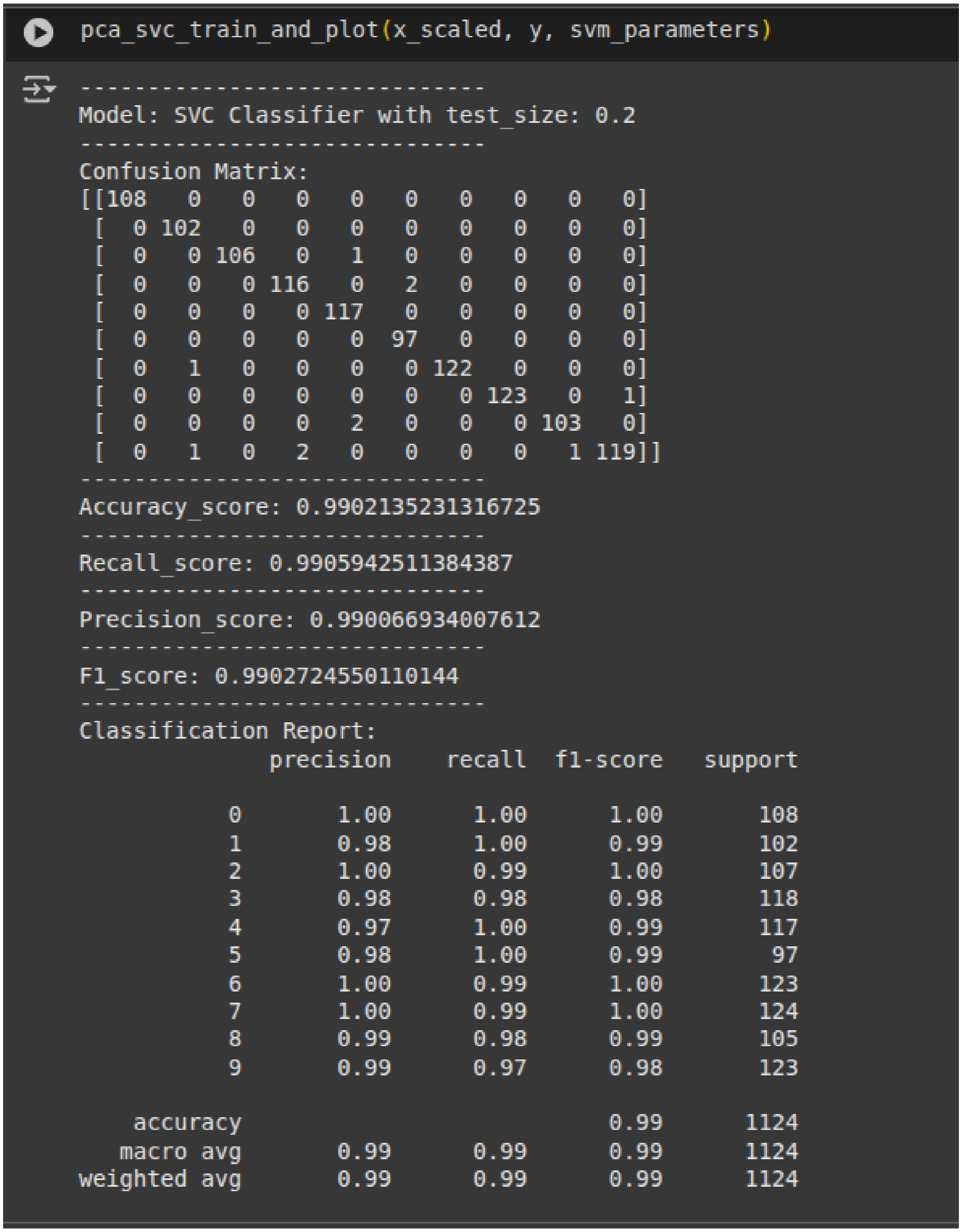
The parameters obtained after parameter tuning the SVC classifier:



```
show_best_params(svm_parameters, "SVC Classifier")
```

```
Model: SVC Classifier
Test Size: 0.2
------------------------------
Best Parameters:
kernel: poly
C: 8.875424377411797
```

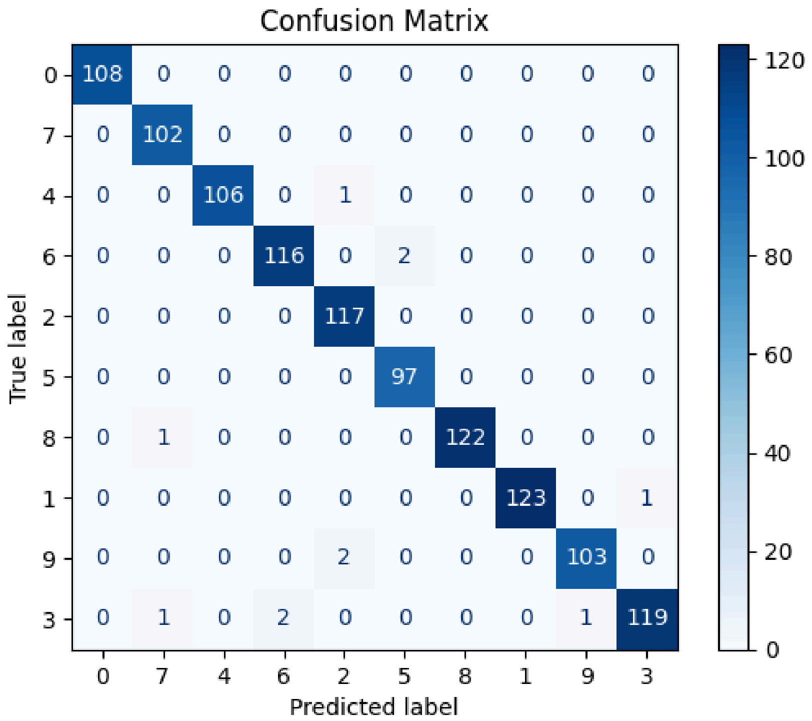
The performance metrics are:



```
pca_svc_train_and_plot(x_scaled, y, svm_parameters)
```

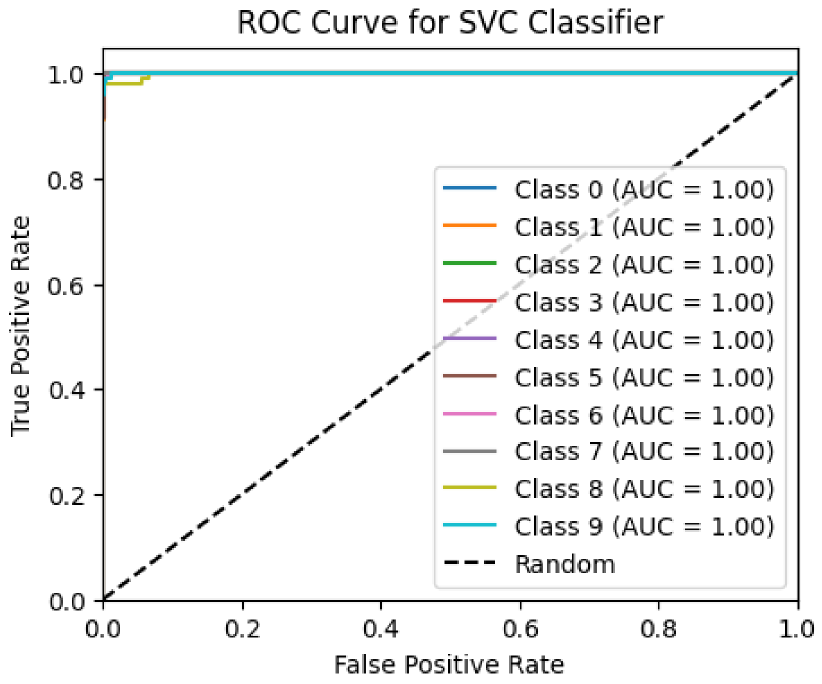
```
------------------------------
Model: SVC Classifier with test_size: 0.2
------------------------------
Confusion Matrix:
[[108   0   0   0   0   0   0   0   0   0]
 [  0 102   0   0   0   0   0   0   0   0]
 [  0   0 106   0   1   0   0   0   0   0]
 [  0   0   0 116   0   2   0   0   0   0]
 [  0   0   0   0 117   0   0   0   0   0]
 [  0   0   0   0   0  97   0   0   0   0]
 [  0   1   0   0   0   0 122   0   0   0]
 [  0   0   0   0   0   0   0 123   0   1]
 [  0   0   0   0   2   0   0   0 103   0]
 [  0   1   0   2   0   0   0   0   1 119]]
------------------------------
Accuracy_score: 0.9902135231316725
------------------------------
Recall_score: 0.9905942511384387
------------------------------
Precision_score: 0.990066934007612
------------------------------
F1_score: 0.9902724550110144
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       108
           1       0.98      1.00      0.99       102
           2       1.00      0.99      1.00       107
           3       0.98      0.98      0.98       118
           4       0.97      1.00      0.99       117
           5       0.98      1.00      0.99        97
           6       1.00      0.99      1.00       123
           7       1.00      0.99      1.00       124
           8       0.99      0.98      0.99       105
           9       0.99      0.97      0.98       123

    accuracy                           0.99      1124
   macro avg       0.99      0.99      0.99      1124
weighted avg       0.99      0.99      0.99      1124
```

ConfusionMatrix:



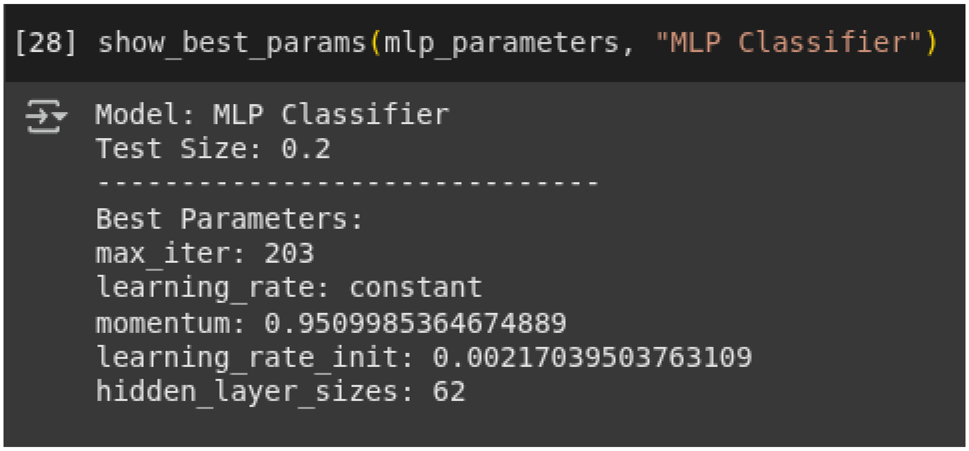
	0	7	4	6	2	5	8	1	9	3
0	108	0	0	0	0	0	0	0	0	0
7	0	102	0	0	0	0	0	0	0	0
4	0	0	106	0	1	0	0	0	0	0
6	0	0	0	116	0	2	0	0	0	0
2	0	0	0	0	117	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0
8	0	1	0	0	0	0	122	0	0	0
1	0	0	0	0	0	0	0	123	0	1
9	0	0	0	0	2	0	0	0	103	0
3	0	1	0	2	0	0	0	0	1	119

ROC Curve:



MLPClassifier

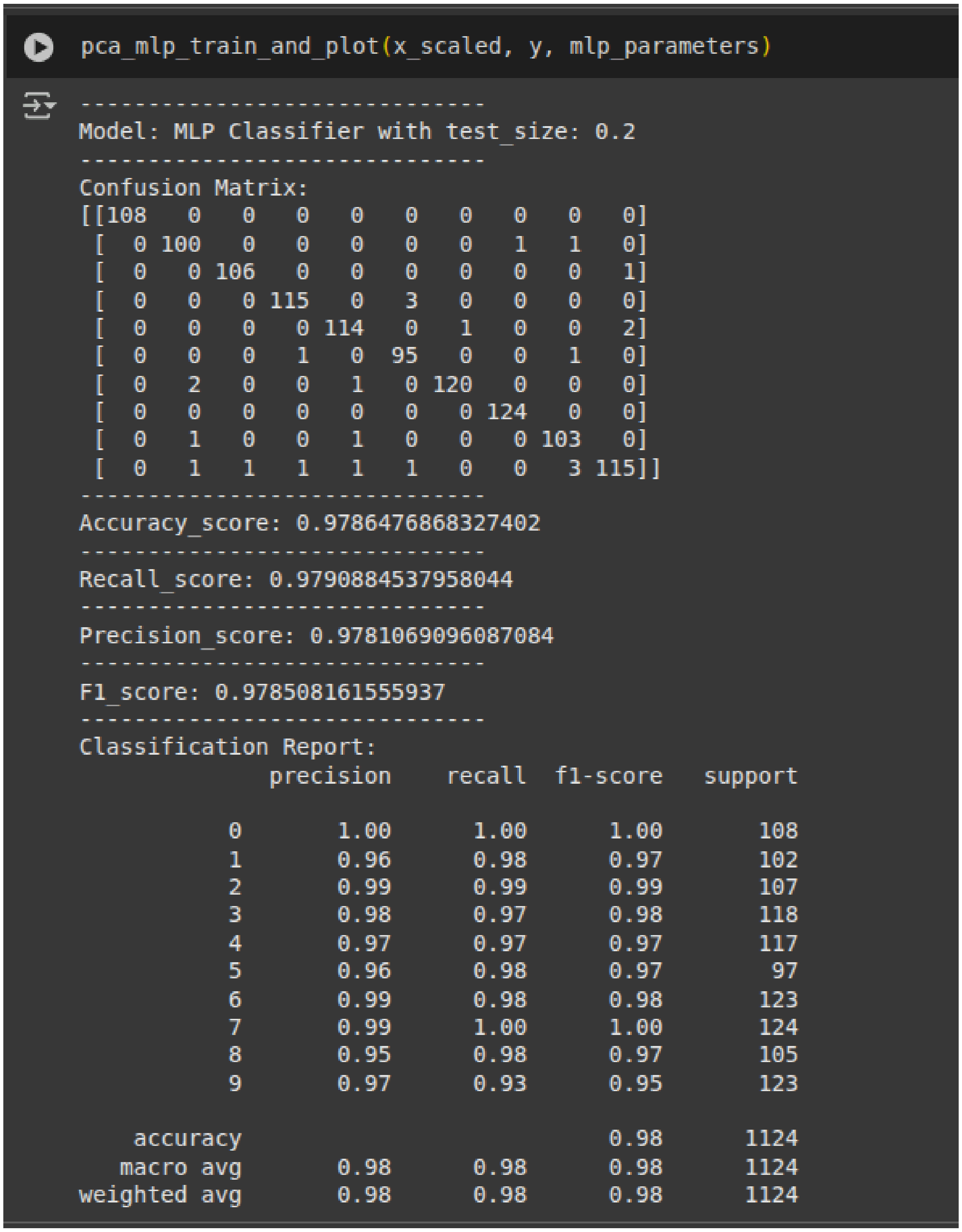
The parameters obtained after parameter tuning the MLP Classifier:



```
[28] show_best_params(mlp_parameters, "MLP Classifier")
```

```
Model: MLP Classifier
Test Size: 0.2
------------------------------
Best Parameters:
max_iter: 203
learning_rate: constant
momentum: 0.9509985364674889
learning_rate_init: 0.00217039503763109
hidden_layer_sizes: 62
```

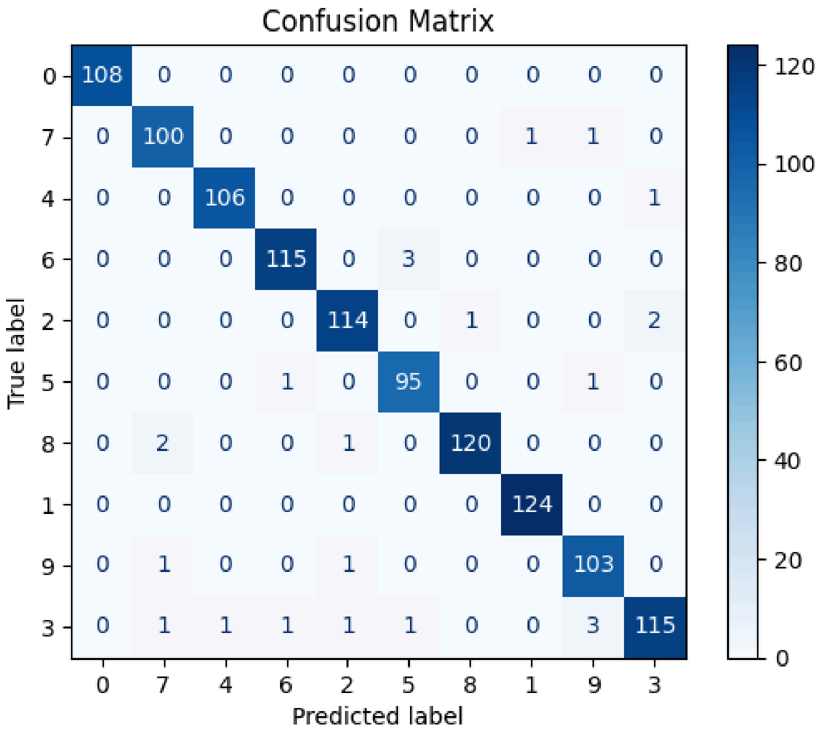
The performance reports are:



```
pca_mlp_train_and_plot(x_scaled, y, mlp_parameters)
```

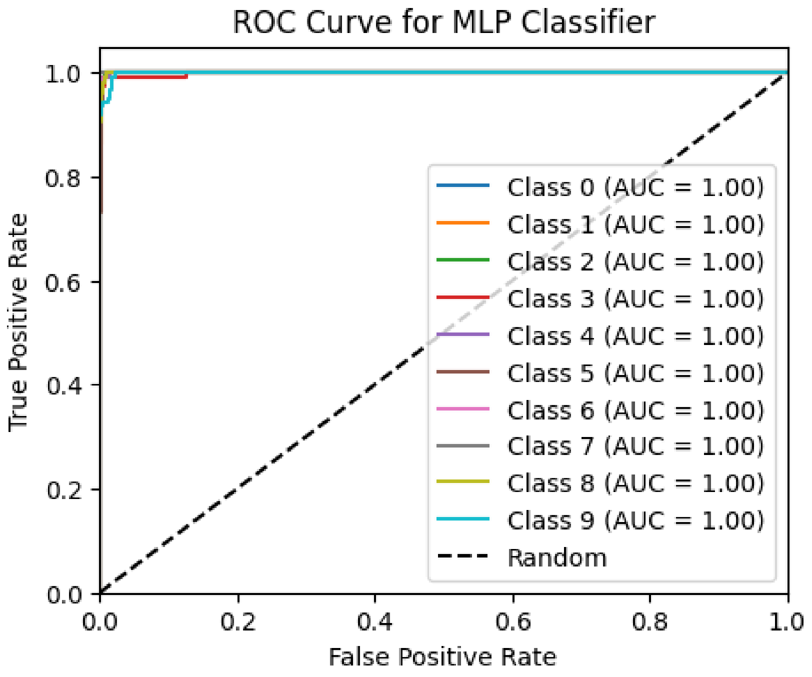
```
------------------------------
Model: MLP Classifier with test_size: 0.2
------------------------------
Confusion Matrix:
[[108   0   0   0   0   0   0   0   0   0]
 [  0 100   0   0   0   0   0   1   1   0]
 [  0   0 106   0   0   0   0   0   0   1]
 [  0   0   0 115   0   3   0   0   0   0]
 [  0   0   0   0 114   0   1   0   0   2]
 [  0   0   0   1   0  95   0   0   1   0]
 [  0   2   0   0   1   0 120   0   0   0]
 [  0   0   0   0   0   0   0 124   0   0]
 [  0   1   0   0   1   0   0   0 103   0]
 [  0   1   1   1   1   1   0   0   3 115]]
------------------------------
Accuracy_score: 0.9786476868327402
------------------------------
Recall_score: 0.9790884537958044
------------------------------
Precision_score: 0.9781069096087084
------------------------------
F1_score: 0.978508161555937
------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       108
           1       0.96      0.98      0.97       102
           2       0.99      0.99      0.99       107
           3       0.98      0.97      0.98       118
           4       0.97      0.97      0.97       117
           5       0.96      0.98      0.97        97
           6       0.99      0.98      0.98       123
           7       0.99      1.00      1.00       124
           8       0.95      0.98      0.97       105
           9       0.97      0.93      0.95       123

    accuracy                           0.98      1124
   macro avg       0.98      0.98      0.98      1124
weighted avg       0.98      0.98      0.98      1124
```

Confusion Matrix:

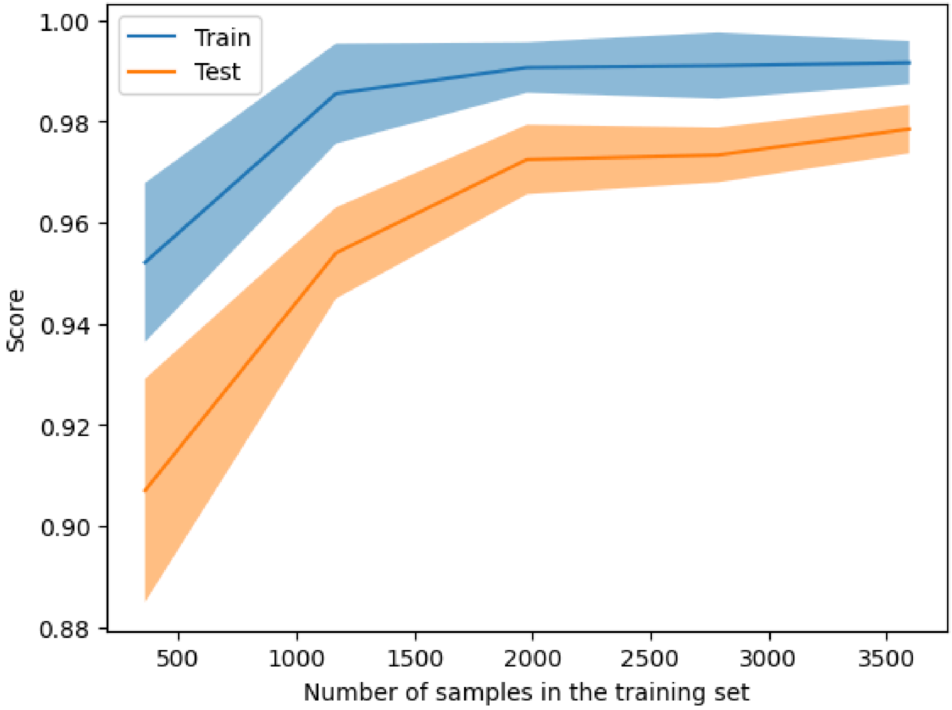


True label \ Predicted label	0	7	4	6	2	5	8	1	9	3
0	108	0	0	0	0	0	0	0	0	0
7	0	100	0	0	0	0	0	1	1	0
4	0	0	106	0	0	0	0	0	0	1
6	0	0	0	115	0	3	0	0	0	0
2	0	0	0	0	114	0	1	0	0	2
5	0	0	0	1	0	95	0	0	1	0
8	0	2	0	0	1	0	120	0	0	0
1	0	0	0	0	0	0	0	124	0	0
9	0	1	0	0	1	0	0	0	103	0
3	0	1	1	1	1	1	0	0	3	115

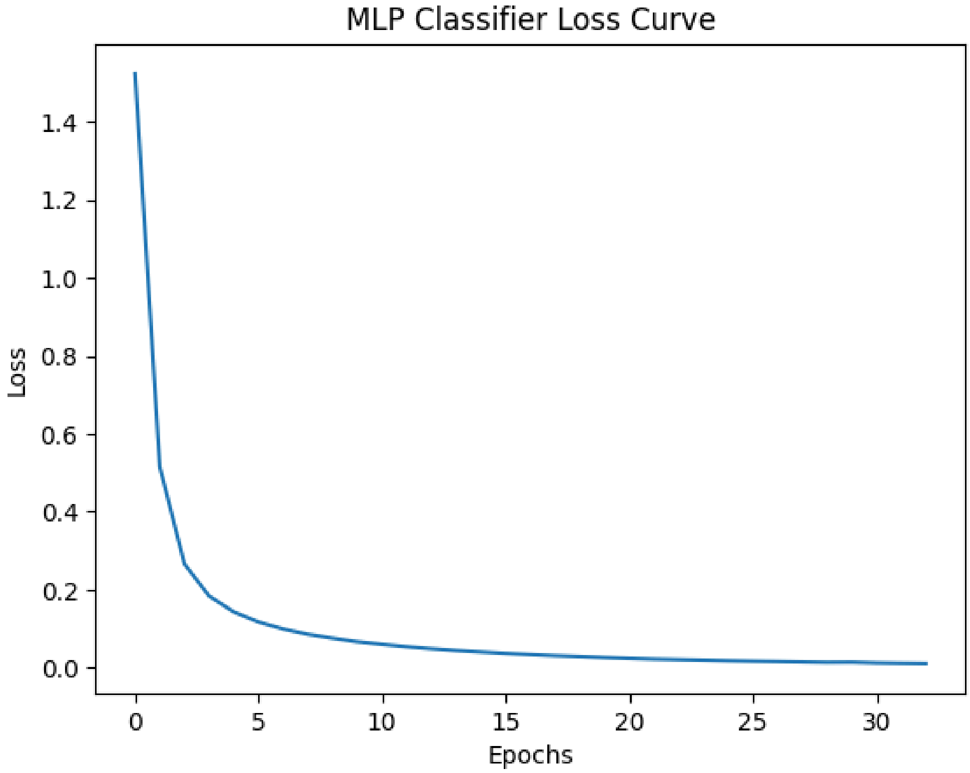
ROC Curve:



Learning Curve:

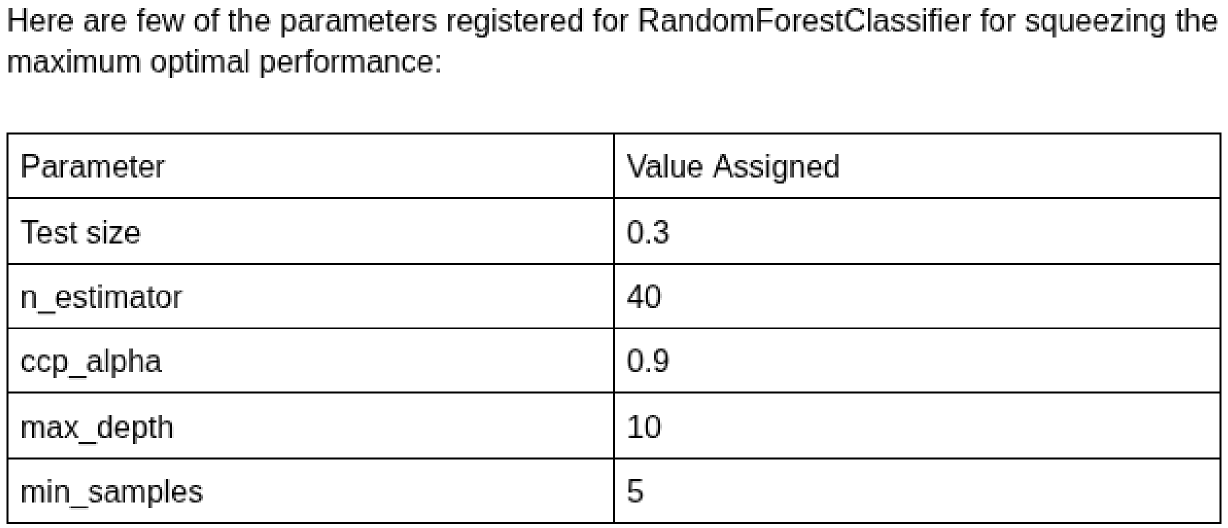


Loss Curve:



Discussion:

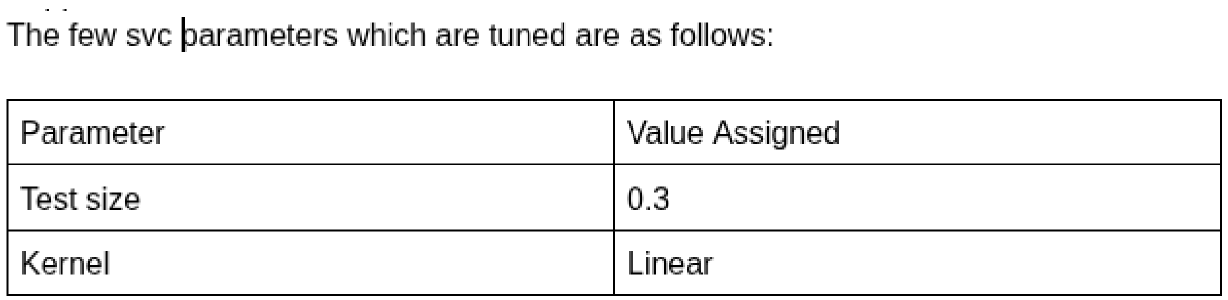
Let’s revisit the model training and how the parameters helped achieve the optimal performance in the models. For the Wine dataset we used three models: RandomForestClassifier, SupportVectorClassifier and MLPClassifier. All three are excellent models for modelling classification problems. Since the dataset size was smaller with only 178 data elements, using a hyperparameter tuning library would be an overkill and only increase the variance in the model’s generalization i.e. overfit. Thus, performed some parameter tuning with manual exhaustive search on the parameters with some trial-and-error.



Here are few of the parameters registered for RandomForestClassifier for squeezing the maximum optimal performance:

Parameter	Value Assigned
Test size	0.3
n_estimator	40
ccp_alpha	0.9
max_depth	10
min_samples	5

This helped achieve an accuracy of 0.96.



The few svc parameters which are tuned are as follows:

Parameter	Value Assigned
Test size	0.3
Kernel	Linear

This helped achieve an accuracy score of 0.98 reducing overfitting and enforcing generalization by the model.

The parameters that are tuned for MLPClassifier are as follows:

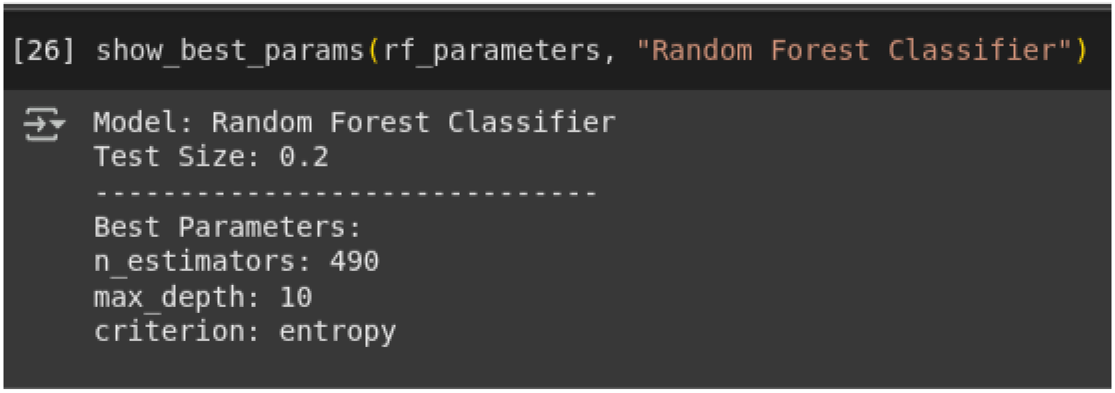
Parameter	Value Assigned
Test size	0.3
Max_iter	200 (default)
learning_rate	adaptive
momentum	0.9
early_stopping	True

This helped achieve an accuracy score of 0.93.

Thus, on the Wine dataset SupportVectorClassifier helped achieve the maximum accuracy score out of the three models, while influencing the least parameters.

Now, for the Handwritten Digit classification problem where we have a dataset which gave pixel values of each of the 64*64 pixels of an image and target value as the digit which it classifies itself to. Since the dataset was larger, parameter tuning was done with the help of Optuna after reducing the data dimension using PCA while maximizing the variance present in the data to 95%. The parameters tuned and result obtained are as follows:

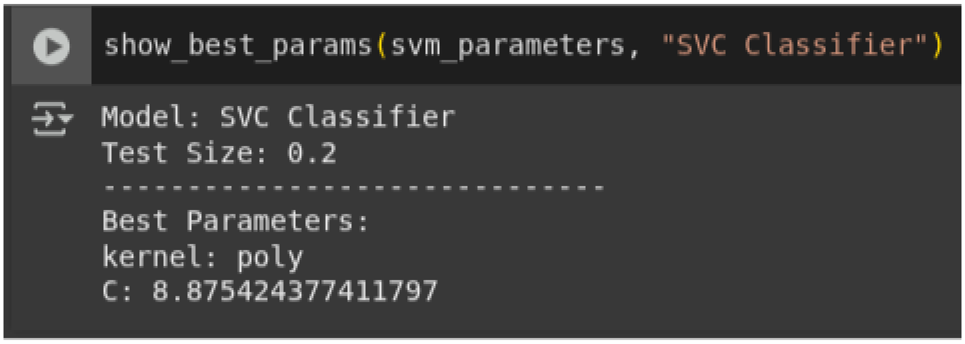
The best parameters after parameter tuning RandomForestClassifier:



```
[26] show_best_params(rf_parameters, "Random Forest Classifier")

     Model: Random Forest Classifier
     Test Size: 0.2
     ------------------------------
     Best Parameters:
     n_estimators: 490
     max_depth: 10
     criterion: entropy
```

These parameters helped achieve an accuracy score of 0.98.

The parameters obtained after parameter tuning the SVC classifier:

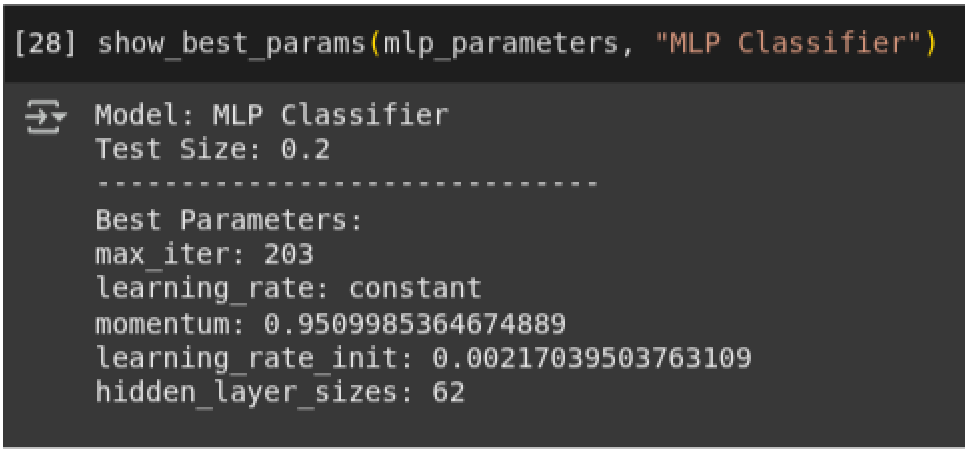


```
show_best_params(svm_parameters, "SVC Classifier")
```

```
Model: SVC Classifier
Test Size: 0.2
------------------------------
Best Parameters:
kernel: poly
C: 8.875424377411797
```

These parameters helped achieve an accuracy score of 0.99.

The parameters obtained after parameter tuning the MLP Classifier:



```
[28] show_best_params(mlp_parameters, "MLP Classifier")
```

```
Model: MLP Classifier
Test Size: 0.2
------------------------------
Best Parameters:
max_iter: 203
learning_rate: constant
momentum: 0.9509985364674889
learning_rate_init: 0.00217039503763109
hidden_layer_sizes: 62
```

These parameters helped achieve an accuracy score of 0.98.