Statistical Methods in AI (CSE/ECE 471) Spring-2020

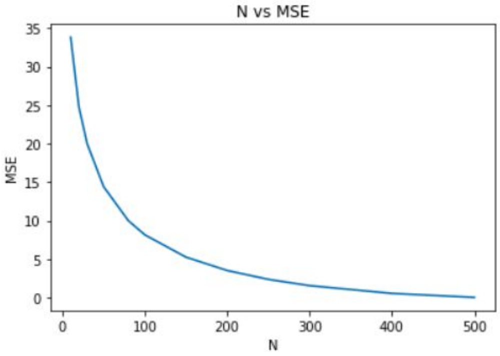
Assignment-3

Submitted By : Jyoti Gambhir (2019201032)

1. PCA

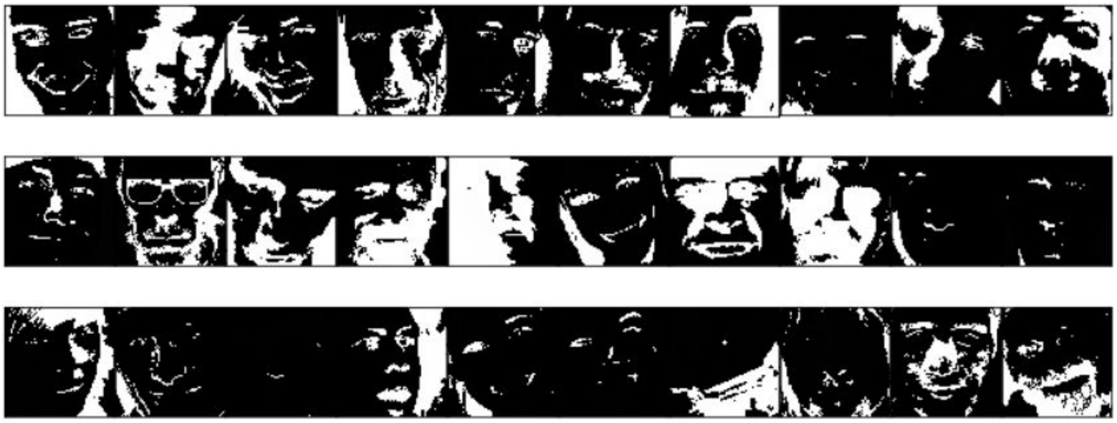
Original Images

MSE vs N Plot



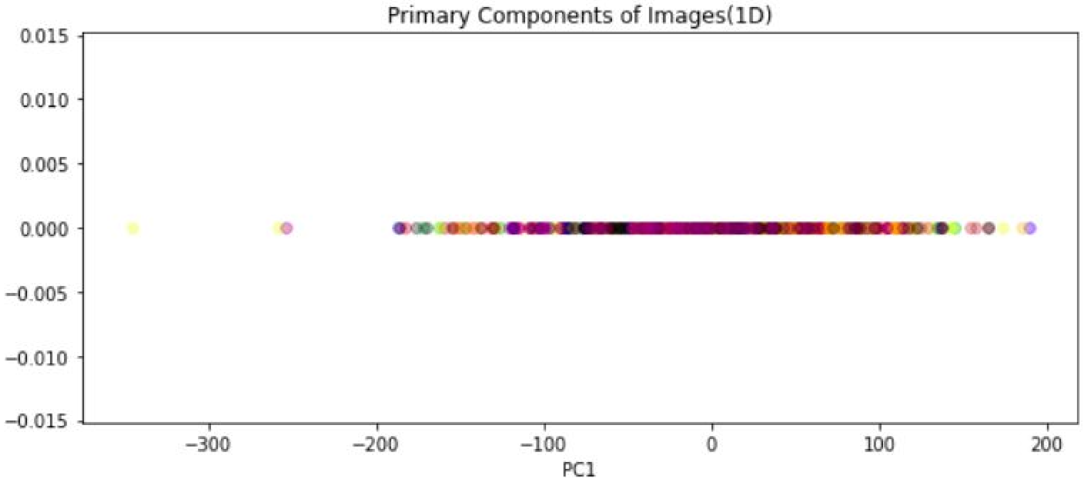
MSE is less than 20% beyond N=300

Images After PCA (N = 400)

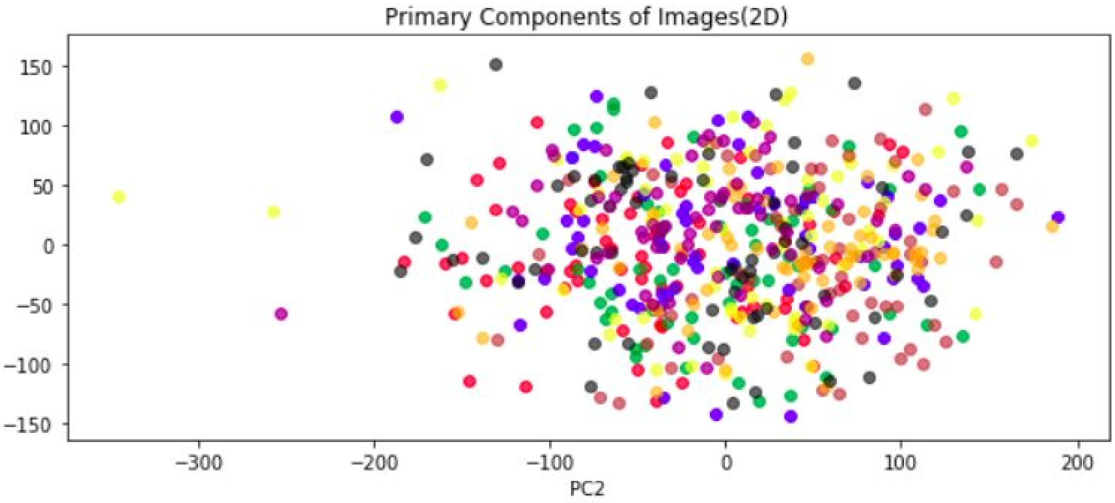


Scatterplots of Images

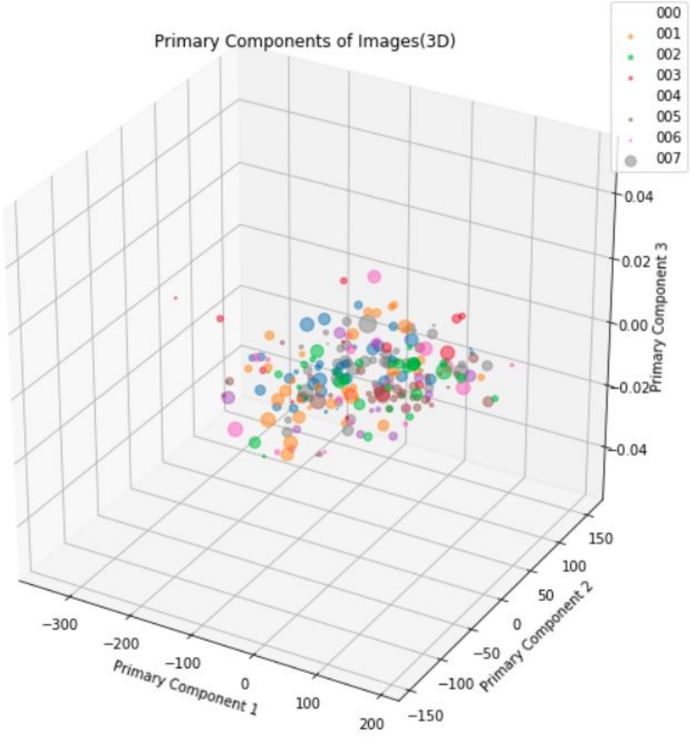
Scatterplots of Images in 1D space



Scatterplots of Images in 2D space

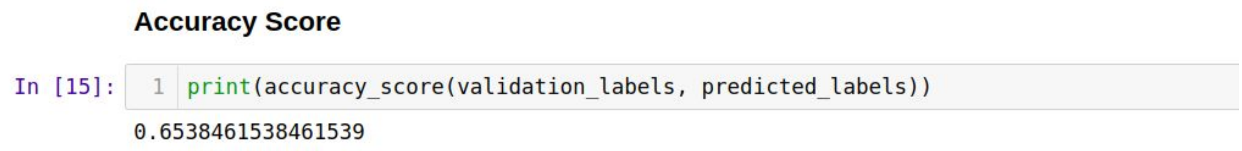


Scatterplots of Images in 3D space



2. Logistic Regression

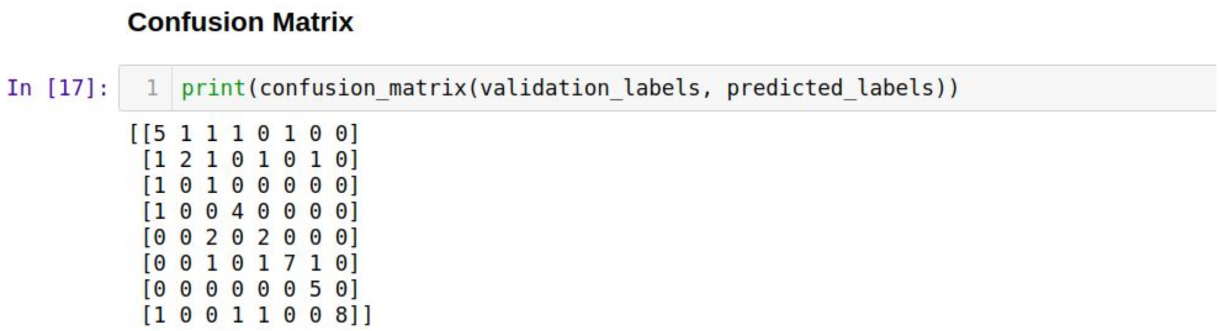
Accuracy Score



Accuracy Score

```
In [15]:  print(accuracy_score(validation_labels, predicted_labels))
0.6538461538461539
```

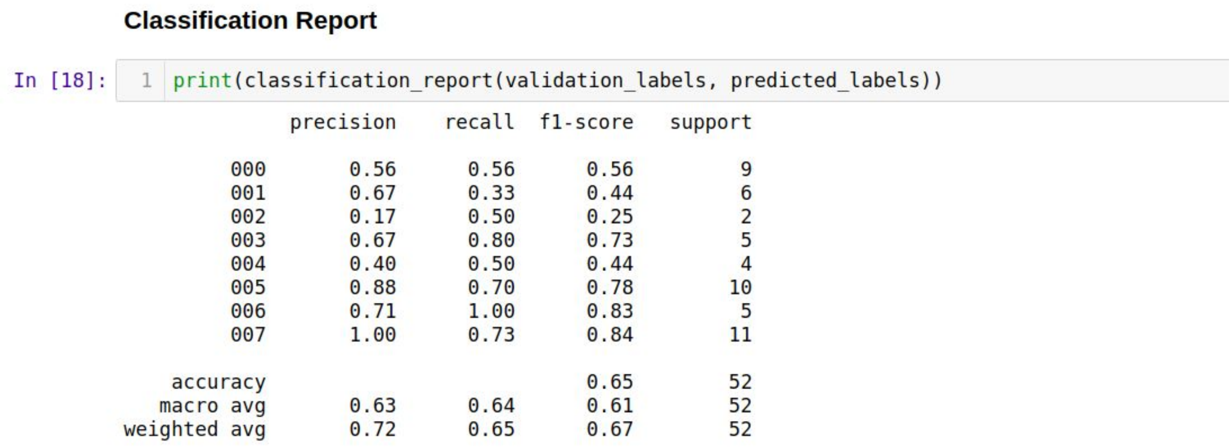
Confusion Matrix



Confusion Matrix

```
In [17]:  print(confusion_matrix(validation_labels, predicted_labels))
[[5 1 1 1 0 1 0 0]
 [1 2 1 0 1 0 1 0]
 [1 0 1 0 0 0 0 0]
 [1 0 0 4 0 0 0 0]
 [0 0 2 0 2 0 0 0]
 [0 0 1 0 1 7 1 0]
 [0 0 0 0 0 0 5 0]
 [1 0 0 1 1 0 0 8]]
```

Classification Report



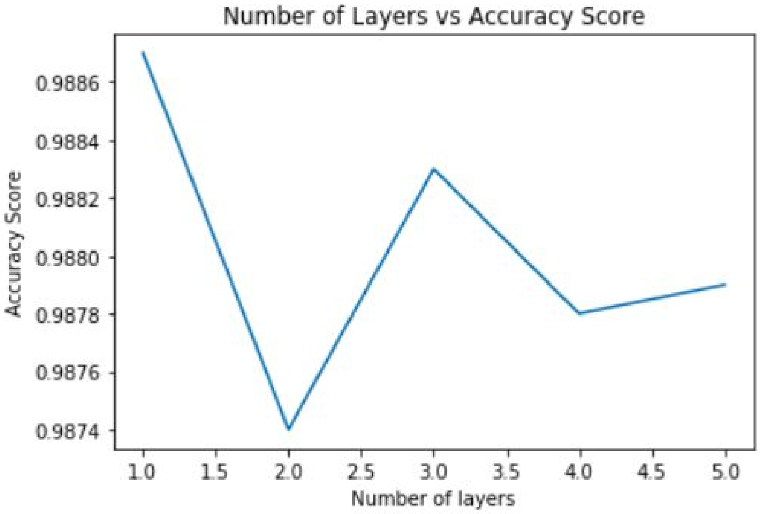
Classification Report

```
In [18]:  print(classification_report(validation_labels, predicted_labels))
              precision    recall  f1-score   support

         000       0.56      0.56      0.56         9
         001       0.67      0.33      0.44         6
         002       0.17      0.50      0.25         2
         003       0.67      0.80      0.73         5
         004       0.40      0.50      0.44         4
         005       0.88      0.70      0.78        10
         006       0.71      1.00      0.83         5
         007       1.00      0.73      0.84        11

    accuracy                           0.65        52
   macro avg       0.63      0.64      0.61        52
weighted avg       0.72      0.65      0.67        52
```

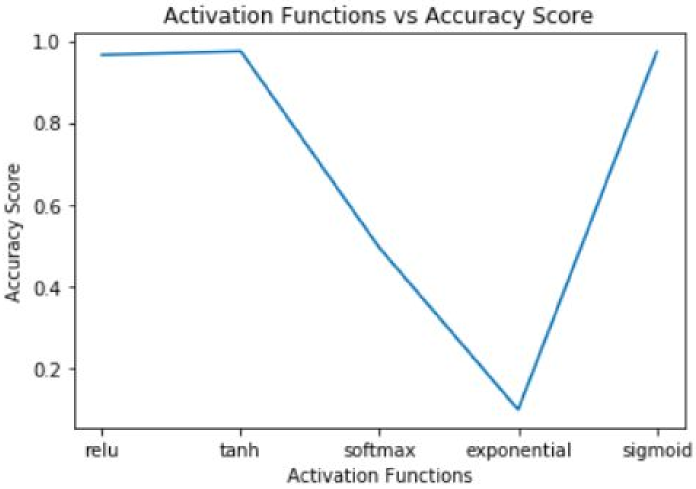
3. MNIST Classification

Observations By CNN

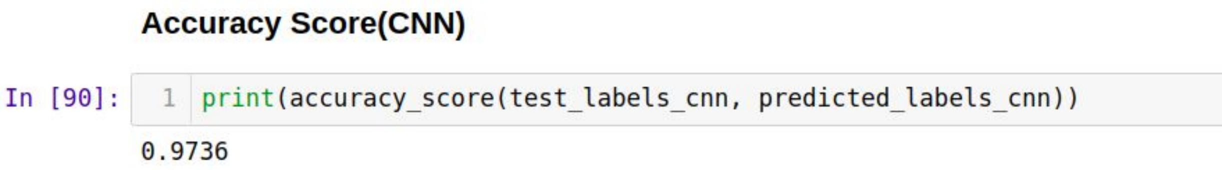
Number of Layers vs Accuracy Score



Activation Functions vs Accuracy Score



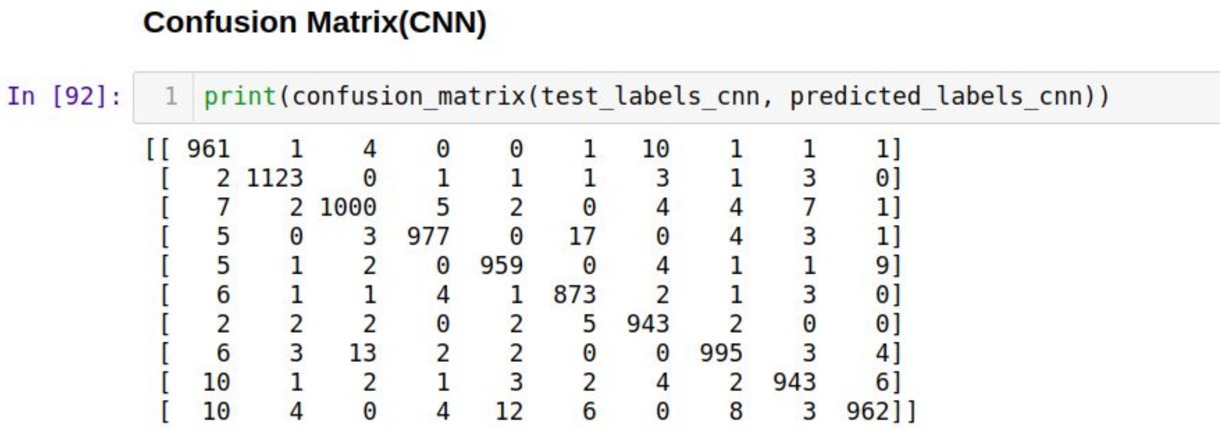
Accuracy Score



Accuracy Score(CNN)

```
In [90]: print(accuracy_score(test_labels_cnn, predicted_labels_cnn))

0.9736
```

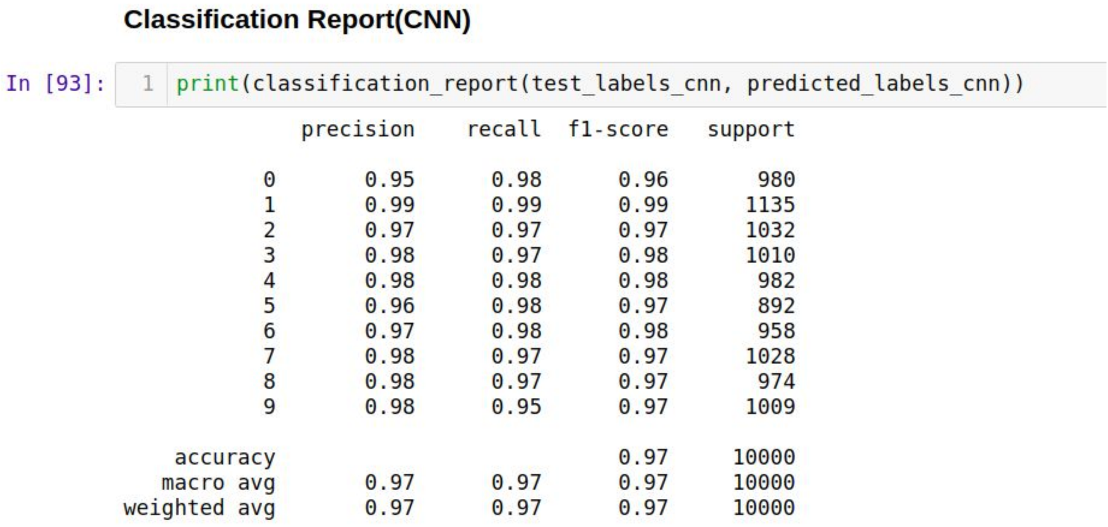
Confusion Matrix



Confusion Matrix(CNN)

```
In [92]: print(confusion_matrix(test_labels_cnn, predicted_labels_cnn))

[[ 961    1    4    0    0    1   10    1    1    1]
 [   2 1123    0    1    1    1    3    1    3    0]
 [   7    2 1000    5    2    0    4    4    7    1]
 [   5    0    3  977    0   17    0    4    3    1]
 [   5    1    2    0  959    0    4    1    1    9]
 [   6    1    1    4    1  873    2    1    3    0]
 [   2    2    2    0    2    5  943    2    0    0]
 [   6    3   13    2    2    0    0  995    3    4]
 [  10    1    2    1    3    2    4    2  943    6]
 [  10    4    0    4   12    6    0    8    3  962]]
```

Classification Report

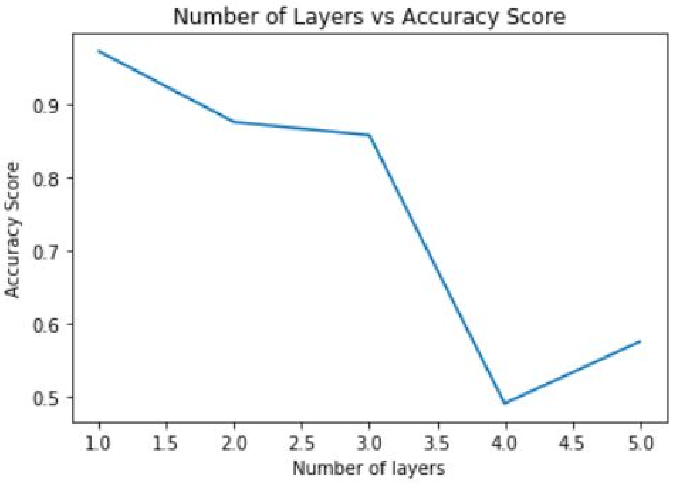


Classification Report(CNN)

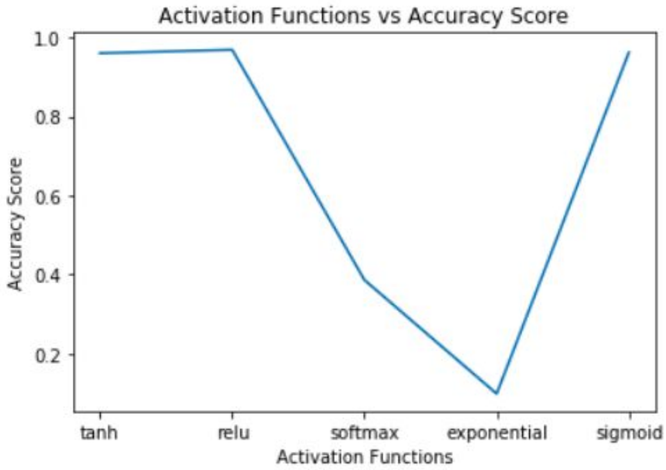
```
In [93]: print(classification_report(test_labels_cnn, predicted_labels_cnn))

              precision    recall  f1-score   support

           0       0.95      0.98      0.96       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.97      0.97      1032
           3       0.98      0.97      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.96      0.98      0.97       892
           6       0.97      0.98      0.98       958
           7       0.98      0.97      0.97      1028
           8       0.98      0.97      0.97       974
           9       0.98      0.95      0.97      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

Observations By Multi Layer Perceptron

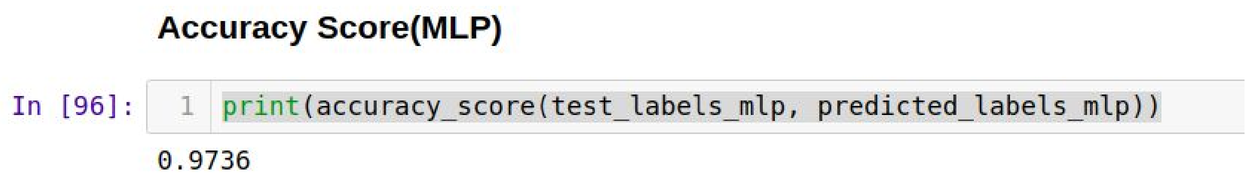
Number of Layers vs Accuracy Score



Activation Functions vs Accuracy Score



Accuracy Score

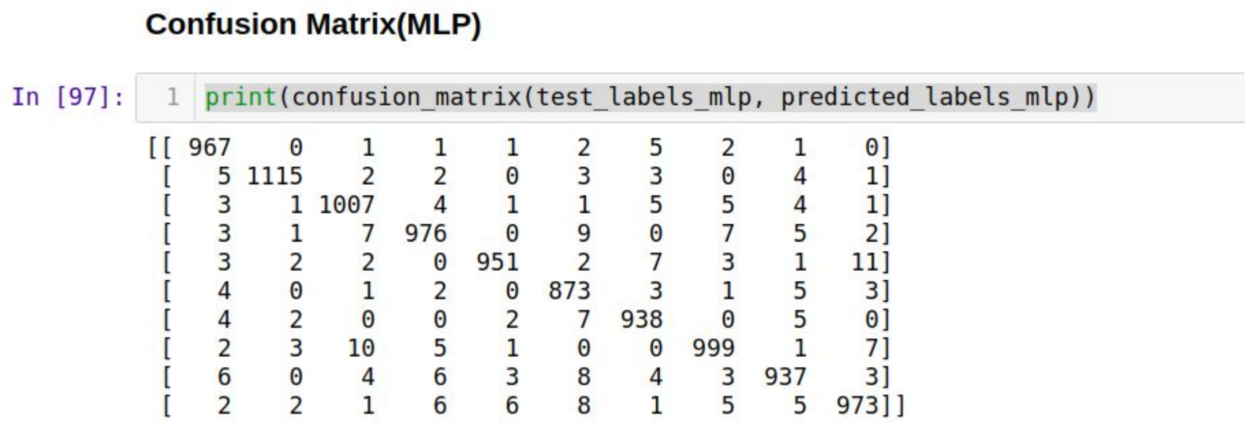


Accuracy Score(MLP)

In [96]:
```
print(accuracy_score(test_labels_mlp, predicted_labels_mlp))
```

```
0.9736
```

Confusion Matrix

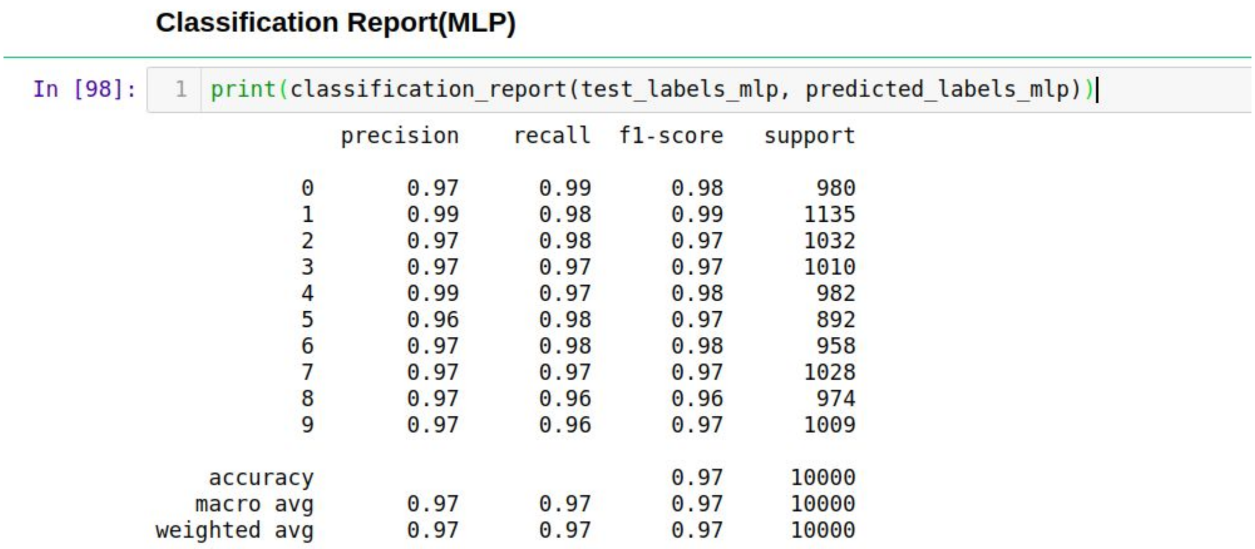


Confusion Matrix(MLP)

In [97]:
```
print(confusion_matrix(test_labels_mlp, predicted_labels_mlp))
```

```
[[ 967    0    1    1    1    2    5    2    1    0]
 [    5 1115    2    2    0    3    3    0    4    1]
 [    3    1 1007    4    1    1    5    5    4    1]
 [    3    1    7  976    0    9    0    7    5    2]
 [    3    2    2    0  951    2    7    3    1   11]
 [    4    0    1    2    0  873    3    1    5    3]
 [    4    2    0    0    2    7  938    0    5    0]
 [    2    3   10    5    1    0    0  999    1    7]
 [    6    0    4    6    3    8    4    3  937    3]
 [    2    2    1    6    6    8    1    5    5  973]]
```

Classification Report



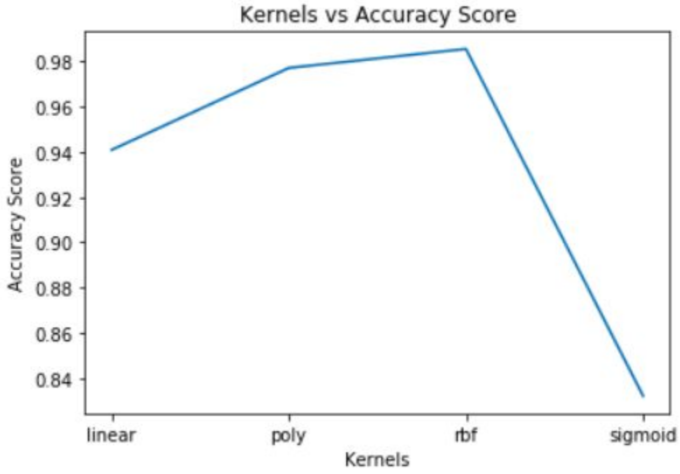
Classification Report(MLP)

In [98]:
```
print(classification_report(test_labels_mlp, predicted_labels_mlp))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       980
           1       0.99      0.98      0.99      1135
           2       0.97      0.98      0.97      1032
           3       0.97      0.97      0.97      1010
           4       0.99      0.97      0.98       982
           5       0.96      0.98      0.97       892
           6       0.97      0.98      0.98       958
           7       0.97      0.97      0.97      1028
           8       0.97      0.96      0.96       974
           9       0.97      0.96      0.97      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

Observations By SVM

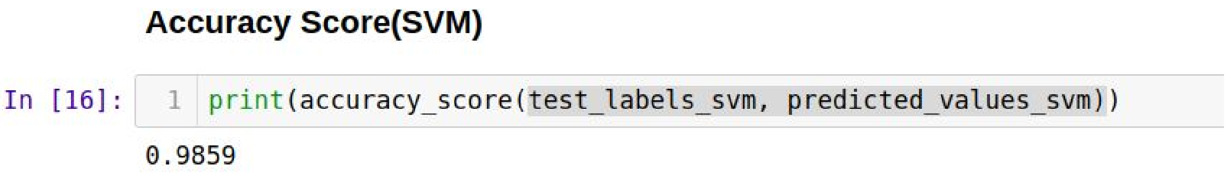
Kernels vs Accuracy Score



Best accuracy score is given by rbf model.

Observations rbf model for SVM:

Accuracy Score

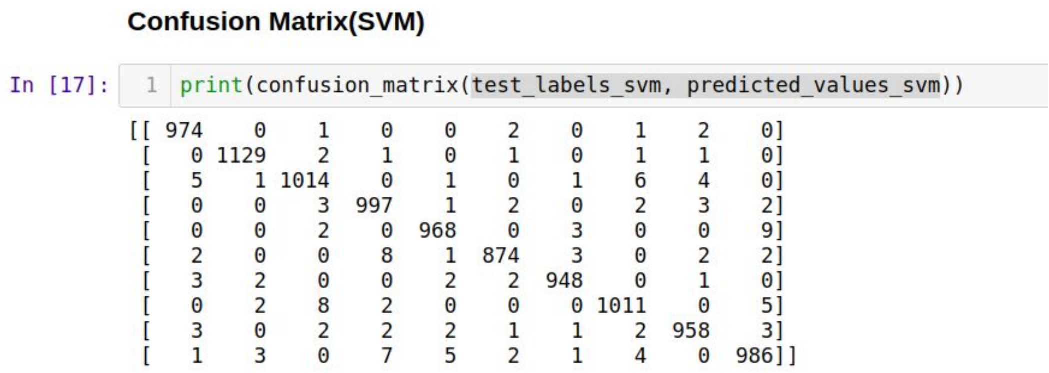


Accuracy Score(SVM)

```
In [16]: print(accuracy_score(test_labels_svm, predicted_values_svm))
```

```
0.9859
```

Confusion Matrix

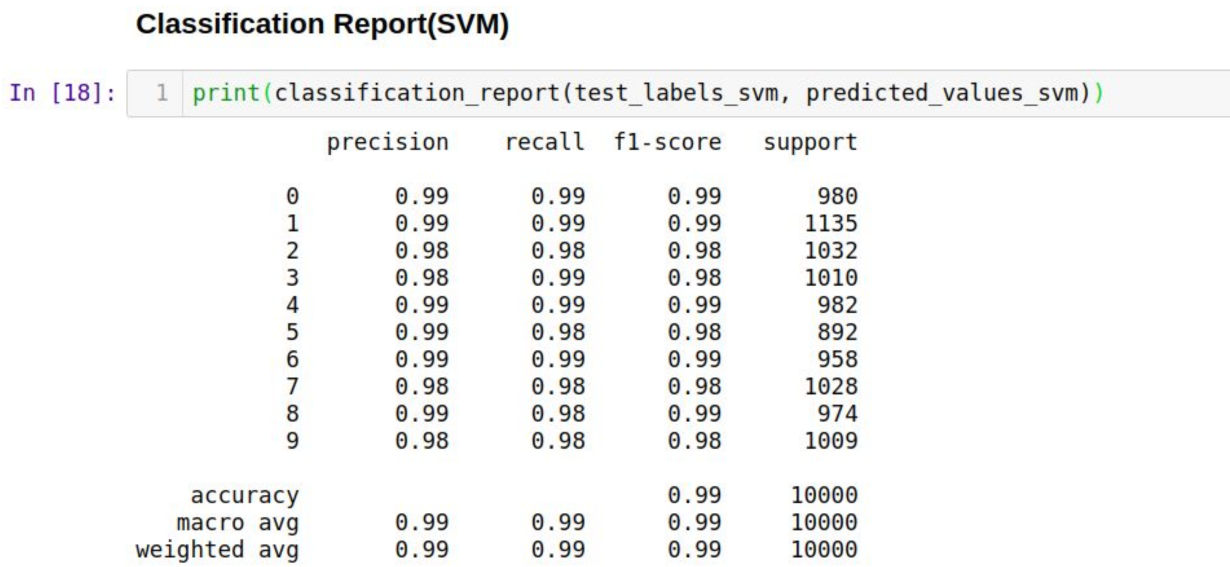


Confusion Matrix(SVM)

```
In [17]:  print(confusion_matrix(test_labels_svm, predicted_values_svm))
```

```
[[ 974    0    1    0    0    2    0    1    2    0]
 [    0 1129    2    1    0    1    0    1    1    0]
 [    5    1 1014    0    1    0    1    6    4    0]
 [    0    0    3  997    1    2    0    2    3    2]
 [    0    0    2    0  968    0    3    0    0    9]
 [    2    0    0    8    1  874    3    0    2    2]
 [    3    2    0    0    2    2  948    0    1    0]
 [    0    2    8    2    0    0    0 1011    0    5]
 [    3    0    2    2    2    1    1    2  958    3]
 [    1    3    0    7    5    2    1    4    0  986]]
```

Classification Report



Classification Report(SVM)

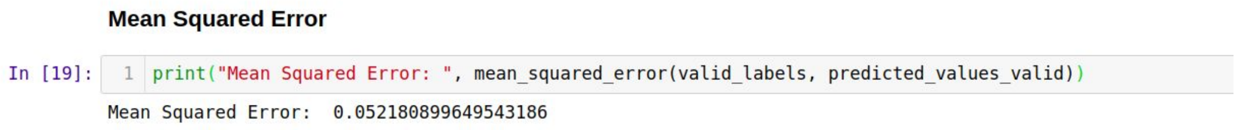
```
In [18]:  print(classification_report(test_labels_svm, predicted_values_svm))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.98      0.98      1032
           3       0.98      0.99      0.98      1010
           4       0.99      0.99      0.99       982
           5       0.99      0.98      0.98       892
           6       0.99      0.99      0.99       958
           7       0.98      0.98      0.98      1028
           8       0.99      0.98      0.99       974
           9       0.98      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

4. Regression

Observations By Multi Layer Perceptron

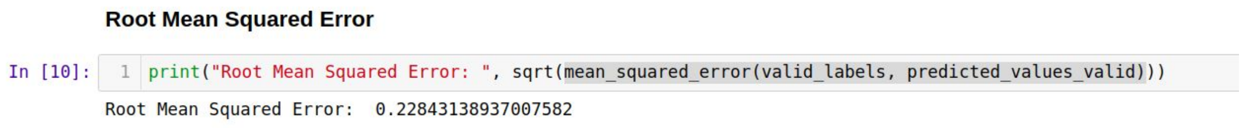
Mean Squared Error



Mean Squared Error

```
In [19]: print("Mean Squared Error: ", mean_squared_error(valid_labels, predicted_values_valid))
Mean Squared Error:  0.052180899649543186
```

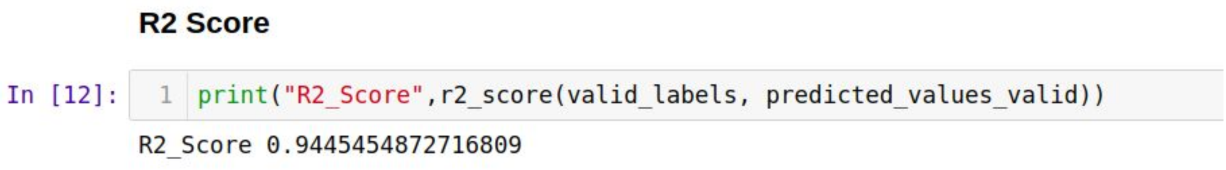
Root Mean Squared Error



Root Mean Squared Error

```
In [10]: print("Root Mean Squared Error: ", sqrt(mean_squared_error(valid_labels, predicted_values_valid)))
Root Mean Squared Error:  0.22843138937007582
```

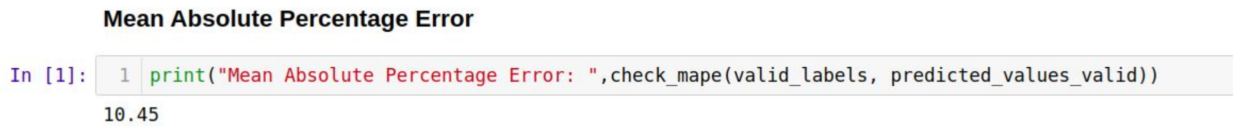
R2 Score



R2 Score

```
In [12]: print("R2_Score",r2_score(valid_labels, predicted_values_valid))
R2_Score 0.9445454872716809
```

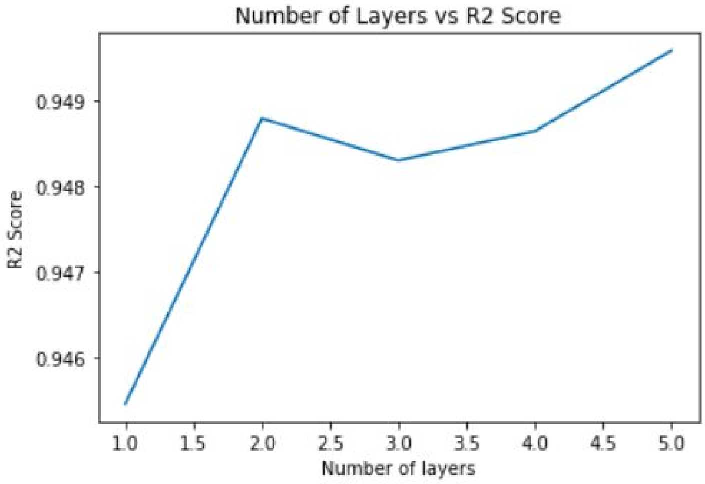
Mean Absolute Percentage Error



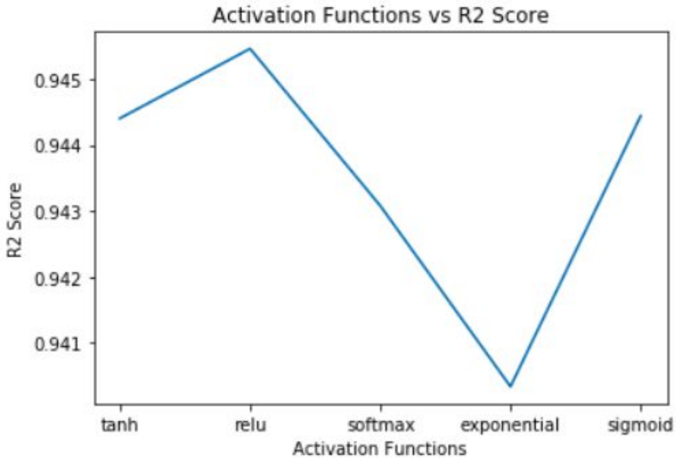
Mean Absolute Percentage Error

```
In [1]: print("Mean Absolute Percentage Error: ",check_mape(valid_labels, predicted_values_valid))
10.45
```

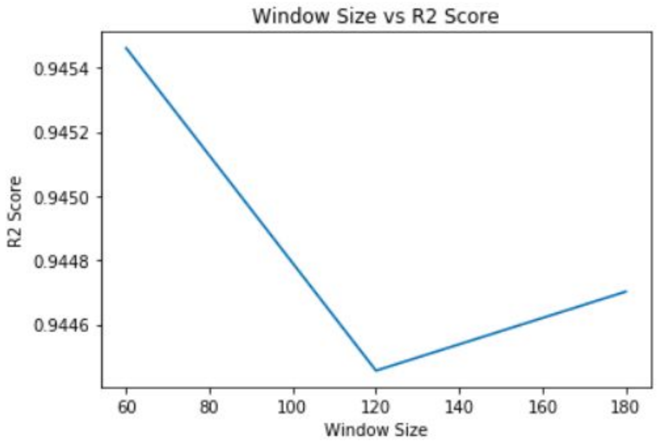
Number of Layers vs R2 Score



Activation Functions vs R2 Score

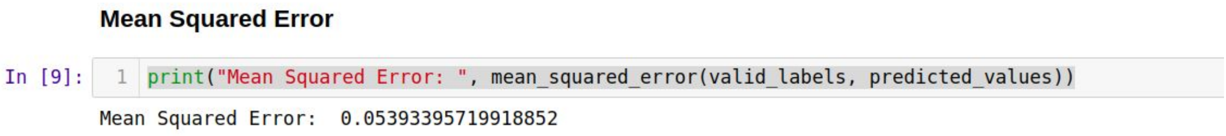


Window Size vs R2 Score



Observations By Linear Regression

Mean Squared Error

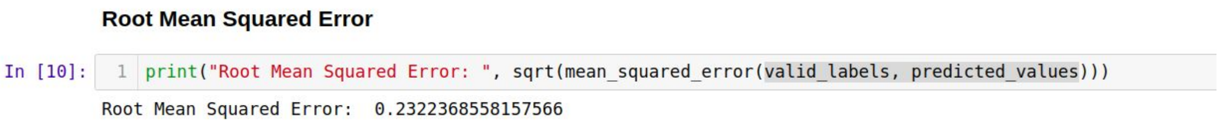


Mean Squared Error

```
In [9]: print("Mean Squared Error: ", mean_squared_error(valid_labels, predicted_values))
```

```
Mean Squared Error:  0.05393395719918852
```

Root Mean Squared Error

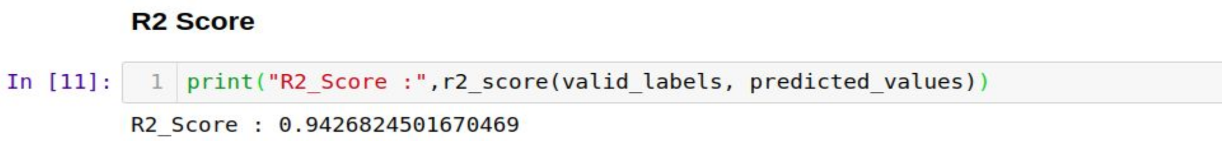


Root Mean Squared Error

```
In [10]: print("Root Mean Squared Error: ", sqrt(mean_squared_error(valid_labels, predicted_values)))
```

```
Root Mean Squared Error:  0.2322368558157566
```

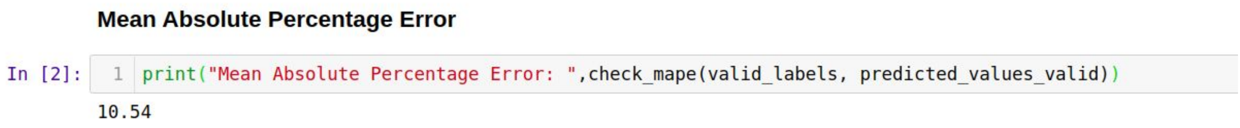
R2 Score



R2 Score

```
In [11]: print("R2_Score :",r2_score(valid_labels, predicted_values))
R2_Score : 0.9426824501670469
```

Mean Absolute Percentage Error



Mean Absolute Percentage Error

```
In [2]: print("Mean Absolute Percentage Error: ",check_mape(valid_labels, predicted_values_valid))
10.54
```