Lab -9

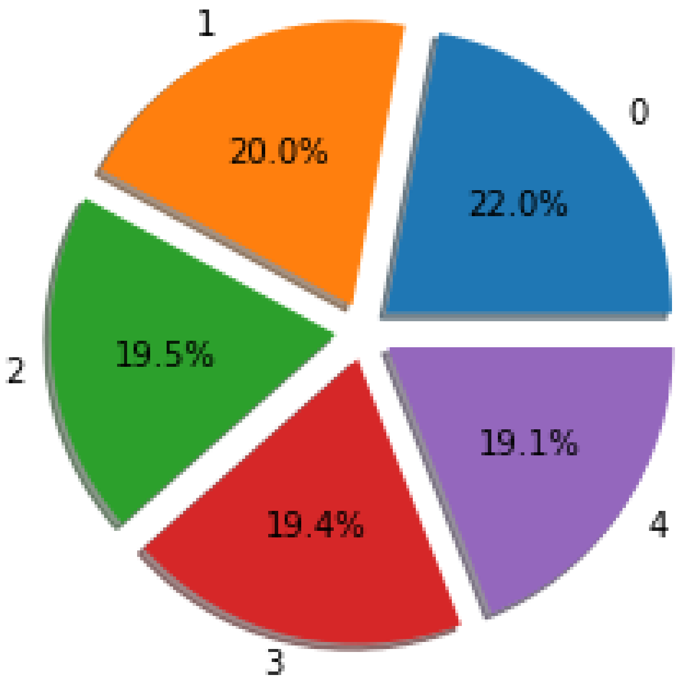
PRML
AY 2020-21 Trimester - III

Support vector machine (SVM) **J Jahnavi(B19CSE109)**

Handwritten Digit Classification

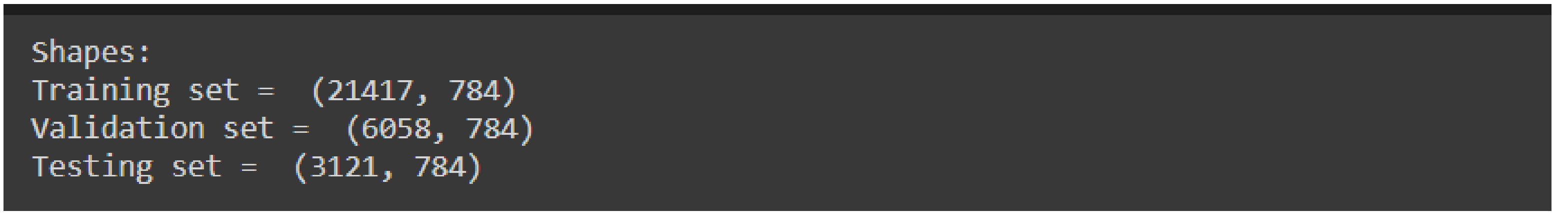
This model will take input as an image of a Handwritten Digit and classify it to one of the five classes {0,1,2,3,4}

1.Distribution of the Classes in the dataset:



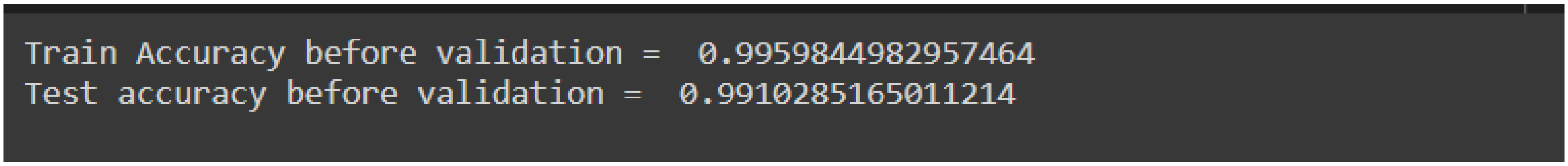
2. Split of the dataset

Having the dataset split into training, validation, and testing sets in the ratio 70:20:10.
Shapes of the datasets:

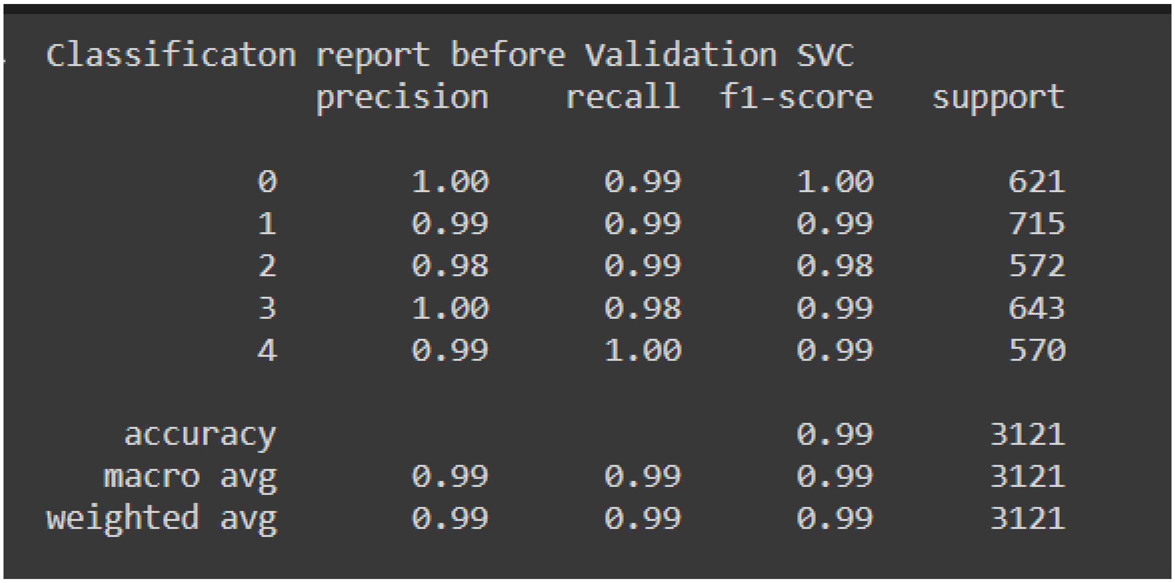


```
Shapes:
Training set =  (21417, 784)
Validation set =  (6058, 784)
Testing set =  (3121, 784)
```

3.SVC Classification before Validation
Training the model before hyperparameter tuning by doing a grid search.
Following is the model performance:



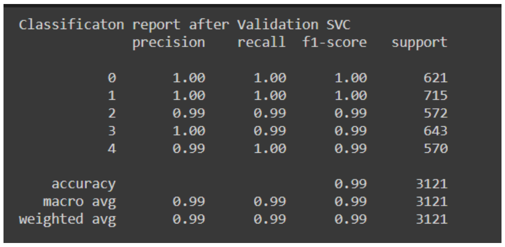
```
Train Accuracy before validation =  0.9959844982957464
Test accuracy before validation =  0.9910285165011214
```



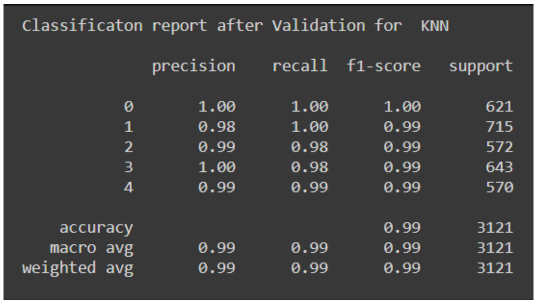
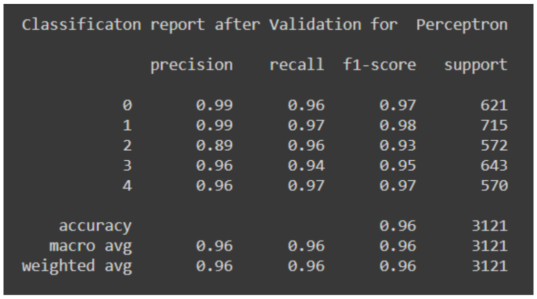
```
Classificaton report before Validation SVC
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       621
           1       0.99      0.99      0.99       715
           2       0.98      0.99      0.98       572
           3       1.00      0.98      0.99       643
           4       0.99      1.00      0.99       570

    accuracy                           0.99      3121
   macro avg       0.99      0.99      0.99      3121
weighted avg       0.99      0.99      0.99      3121
```

4. Using the validation set for hyperparameter tuning by doing a grid search and using nearest neighbor, perceptron, and SVM classifiers for classifying handwritten digits of MNIST

Comparison of the model performance:

Model	**Best Parameters**	**Score**	**Classification Report**
SVM	{'C': 10, 'kernel': 'rbf'}	0.9948734380006409	

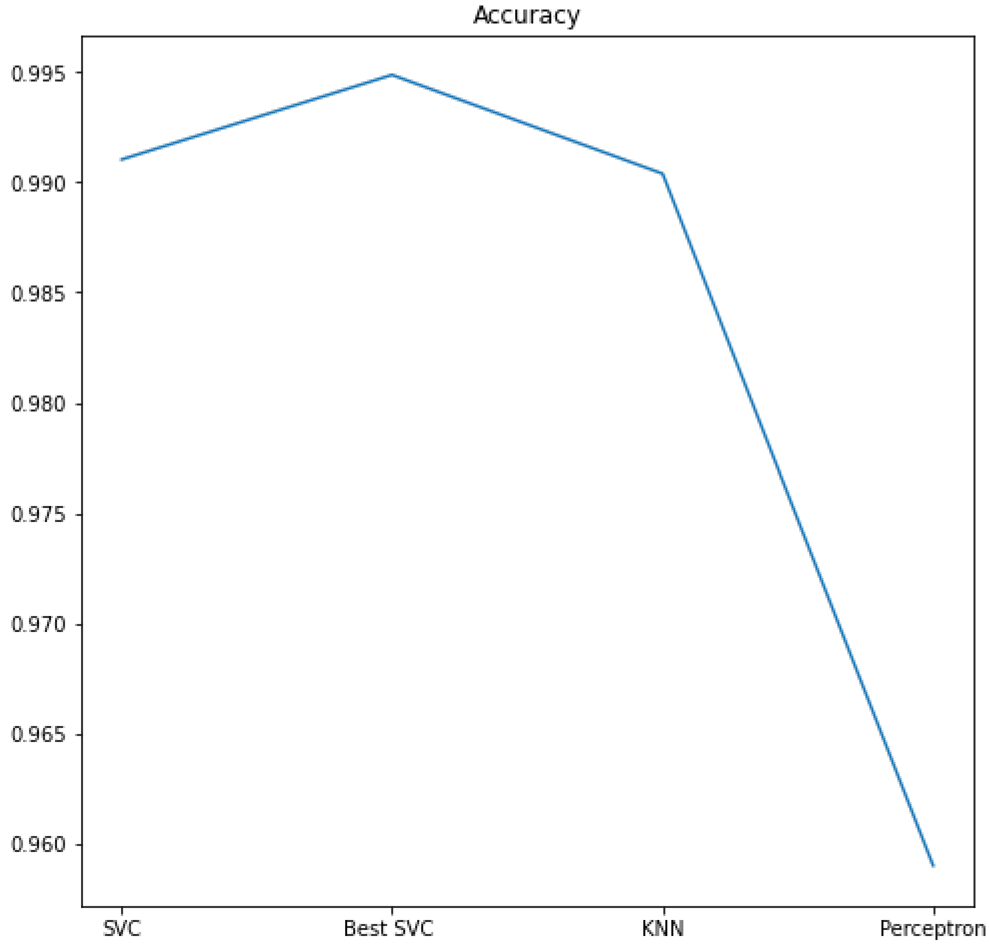
```
Classificaton report after Validation SVC
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       621
           1       1.00      1.00      1.00       715
           2       0.99      0.99      0.99       572
           3       1.00      0.99      0.99       643
           4       0.99      1.00      0.99       570

    accuracy                           0.99      3121
   macro avg       0.99      0.99      0.99      3121
weighted avg       0.99      0.99      0.99      3121
```

KNN	{'n_neighbors': 5}	0.9903876962512015	
Perceptron	{'alpha': 0.0001, 'eta0': 1, 'max_iter': 10}	0.9589875040051266	

```
Classificaton report after Validation for  KNN

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       621
           1       0.98      1.00      0.99       715
           2       0.99      0.98      0.99       572
           3       1.00      0.98      0.99       643
           4       0.99      0.99      0.99       570

    accuracy                           0.99      3121
   macro avg       0.99      0.99      0.99      3121
weighted avg       0.99      0.99      0.99      3121
```

```
Classificaton report after Validation for  Perceptron

              precision    recall  f1-score   support

           0       0.99      0.96      0.97       621
           1       0.99      0.97      0.98       715
           2       0.89      0.96      0.93       572
           3       0.96      0.94      0.95       643
           4       0.96      0.97      0.97       570

    accuracy                           0.96      3121
   macro avg       0.96      0.96      0.96      3121
weighted avg       0.96      0.96      0.96      3121
```

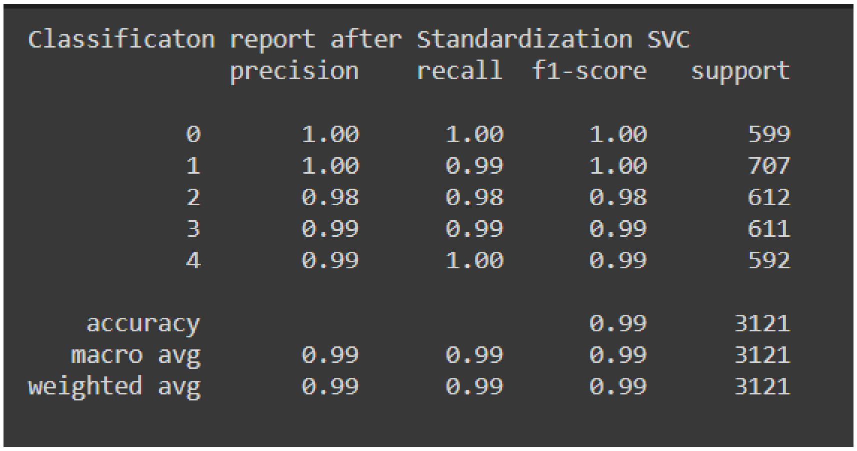
Plot for Performance Comparison before Standardization:



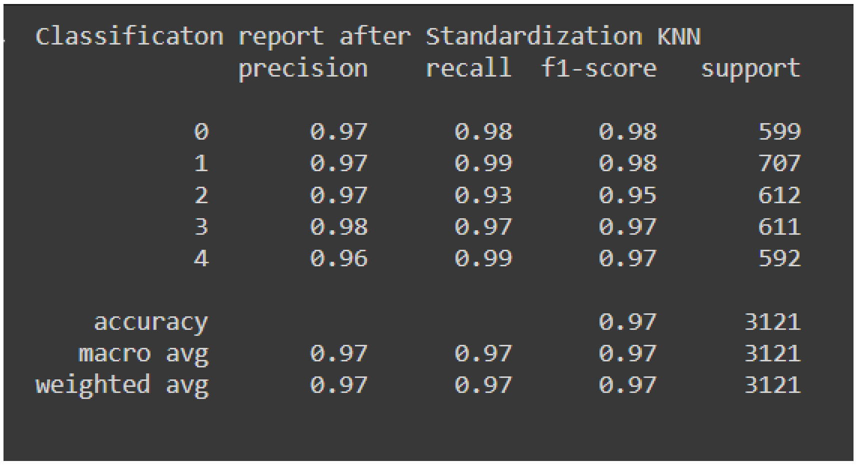
5. Standardization of the data

Normalized the data by mean subtraction followed by standard deviation division.

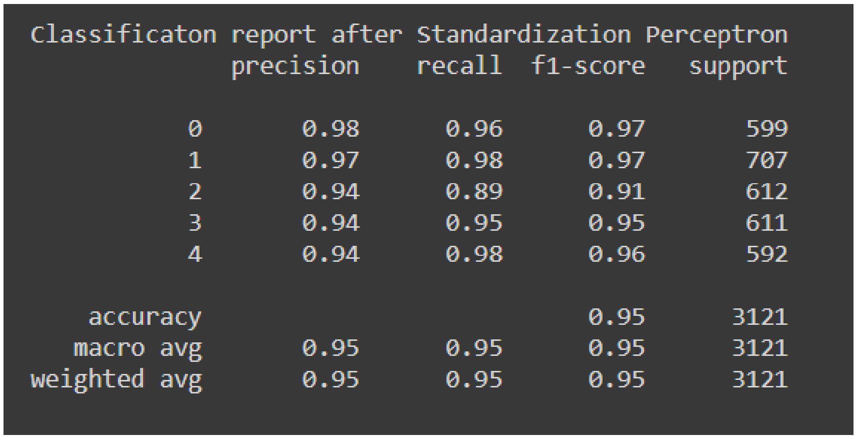
Comparison of the model performance after Standardization:

<table>
<tr><th>Model</th><th>Score</th><th>Classification Report</th></tr>
<tr><td>SVM</td><td>0.9919897468760013</td><td>
<pre>
Classificaton report after Standardization SVC
              precision   recall f1-score   support

           0   1.00   1.00   1.00   599
           1   1.00   0.99   1.00   707
           2   0.98   0.98   0.98   612
           3   0.99   0.99   0.99   611
           4   0.99   1.00   0.99   592

    accuracy   0.99   3121
   macro avg   0.99   0.99   0.99   3121
weighted avg   0.99   0.99   0.99   3121
</pre></td></tr>
<tr><td>KNN</td><td>0.9727651393784044</td><td>
<pre>
Classificaton report after Standardization KNN
              precision   recall f1-score   support

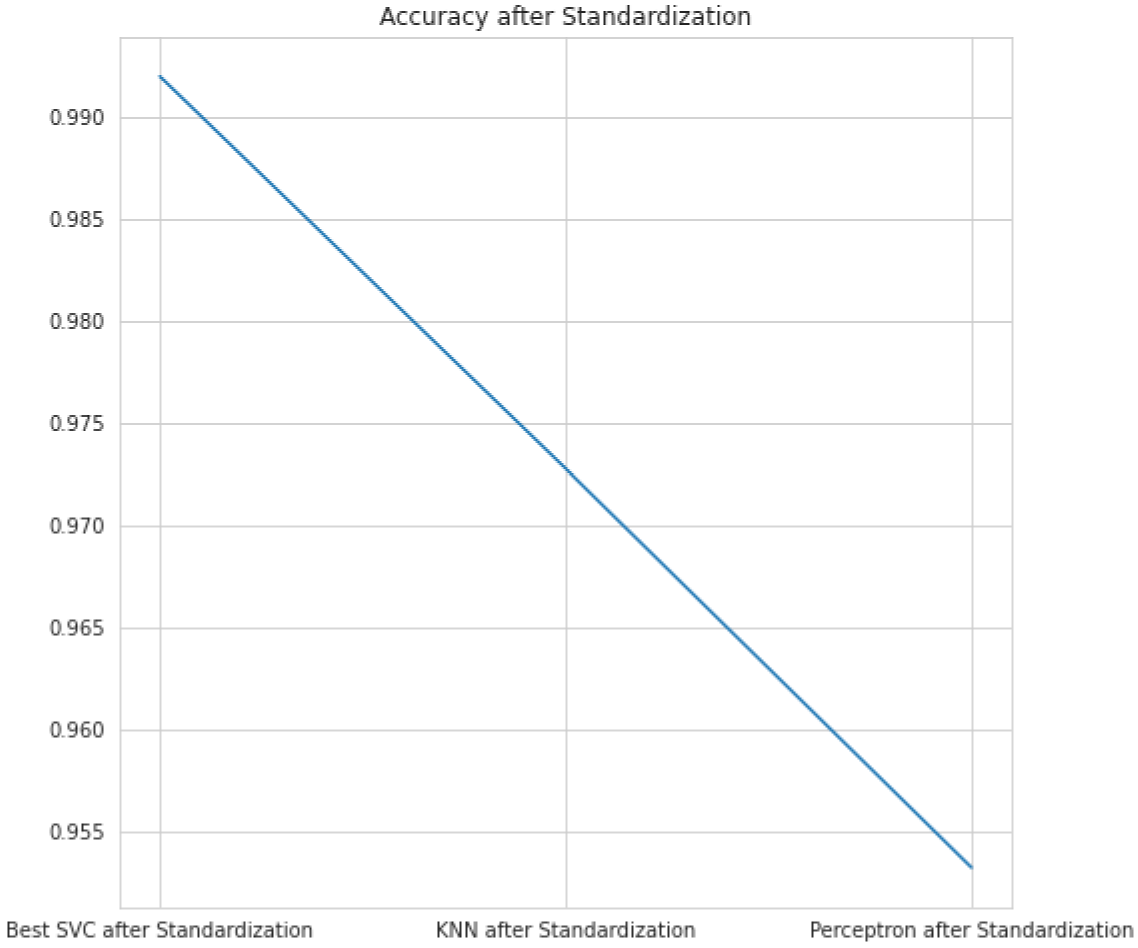
           0   0.97   0.98   0.98   599
           1   0.97   0.99   0.98   707
           2   0.97   0.93   0.95   612
           3   0.98   0.97   0.97   611
           4   0.96   0.99   0.97   592

    accuracy   0.97   3121
   macro avg   0.97   0.97   0.97   3121
weighted avg   0.97   0.97   0.97   3121
</pre></td></tr>
<tr><td>Perceptron</td><td>0.9532201217558475</td><td>
<pre>
Classificaton report after Standardization Perceptron
              precision   recall f1-score   support

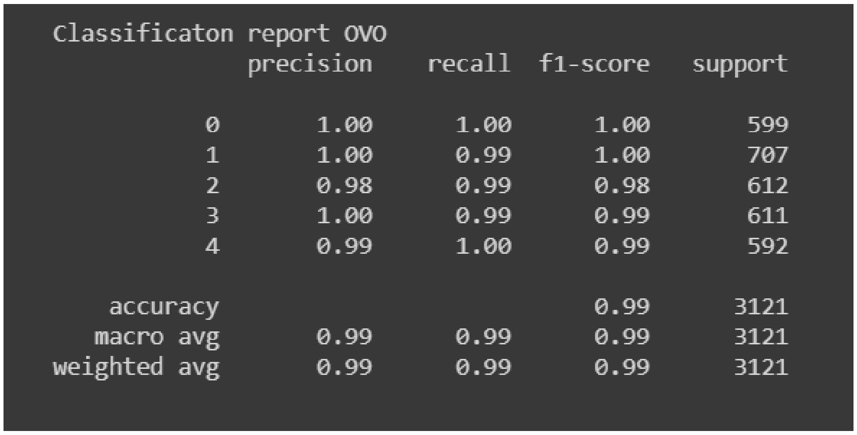
           0   0.98   0.96   0.97   599
           1   0.97   0.98   0.97   707
           2   0.94   0.89   0.91   612
           3   0.94   0.95   0.95   611
           4   0.94   0.98   0.96   592

    accuracy   0.95   3121
   macro avg   0.95   0.95   0.95   3121
weighted avg   0.95   0.95   0.95   3121
</pre></td></tr>
</table>

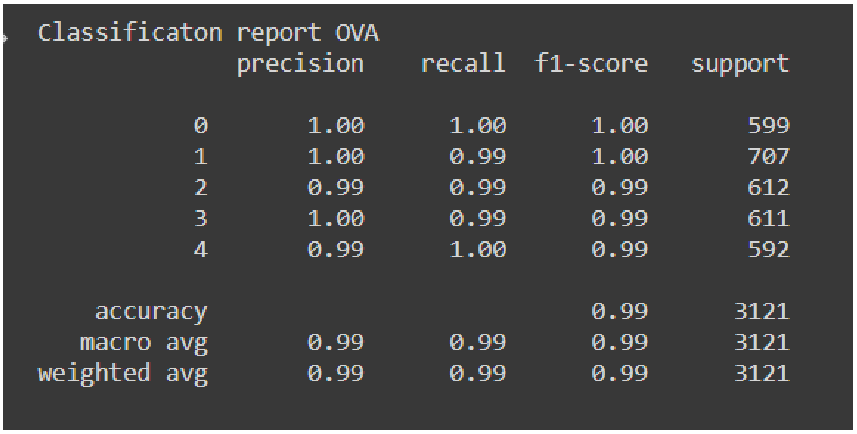
Plot for Performance Comparison after Standardization:



6. Implementing OVO and OVA
Comparison:

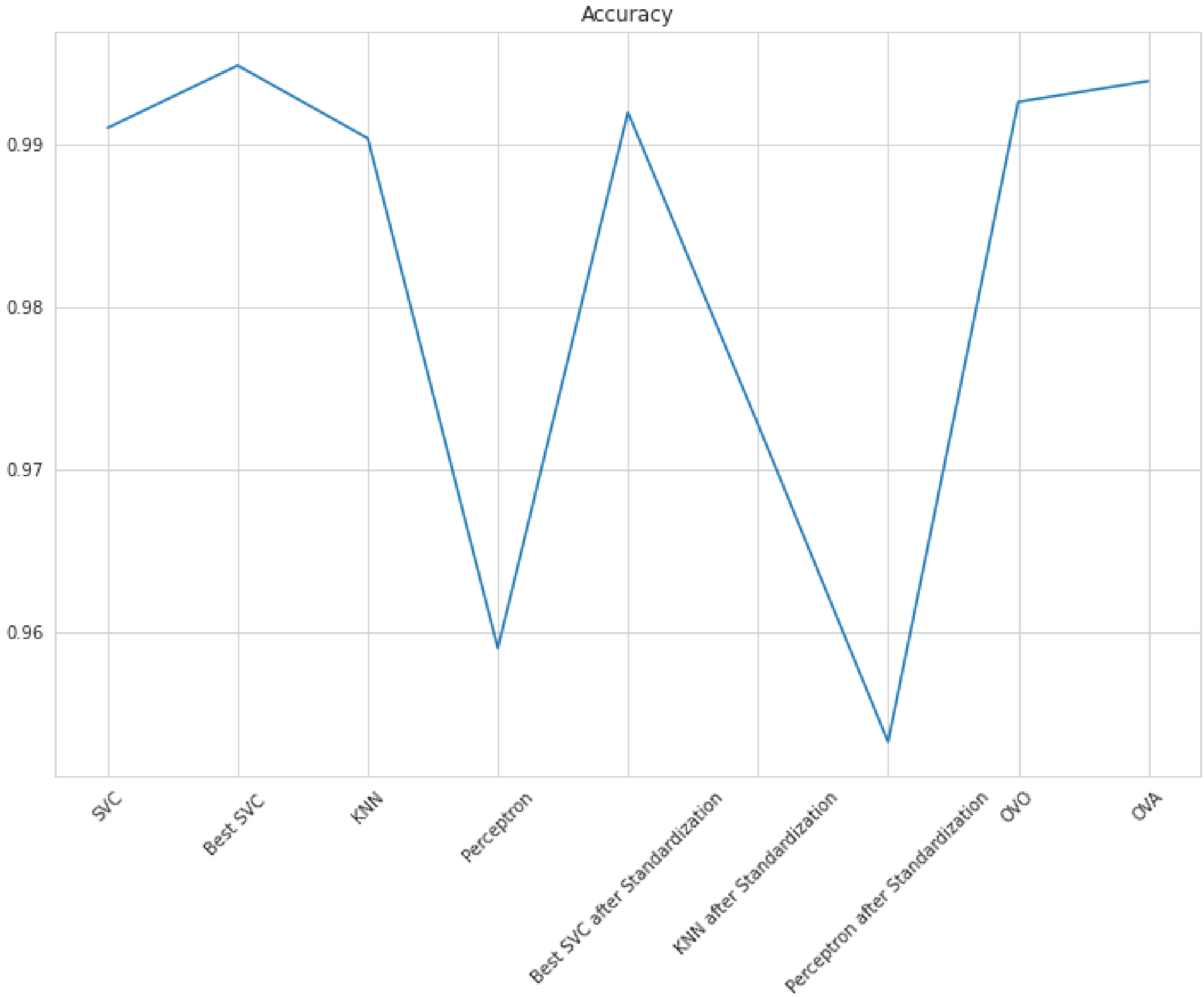
Classifier	**Score**	**Classification Report**
OVO	0.9926305671259211	

```
Classificaton report OVO
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       599
           1       1.00      0.99      1.00       707
           2       0.98      0.99      0.98       612
           3       1.00      0.99      0.99       611
           4       0.99      1.00      0.99       592

    accuracy                           0.99      3121
   macro avg       0.99      0.99      0.99      3121
weighted avg       0.99      0.99      0.99      3121
```

OVA	0.9939122076257609	

```
Classificaton report OVA
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       599
           1       1.00      0.99      1.00       707
           2       0.99      0.99      0.99       612
           3       1.00      0.99      0.99       611
           4       0.99      1.00      0.99       592

    accuracy                           0.99      3121
   macro avg       0.99      0.99      0.99      3121
weighted avg       0.99      0.99      0.99      3121
```

Overall Plot for Performance Comparison:

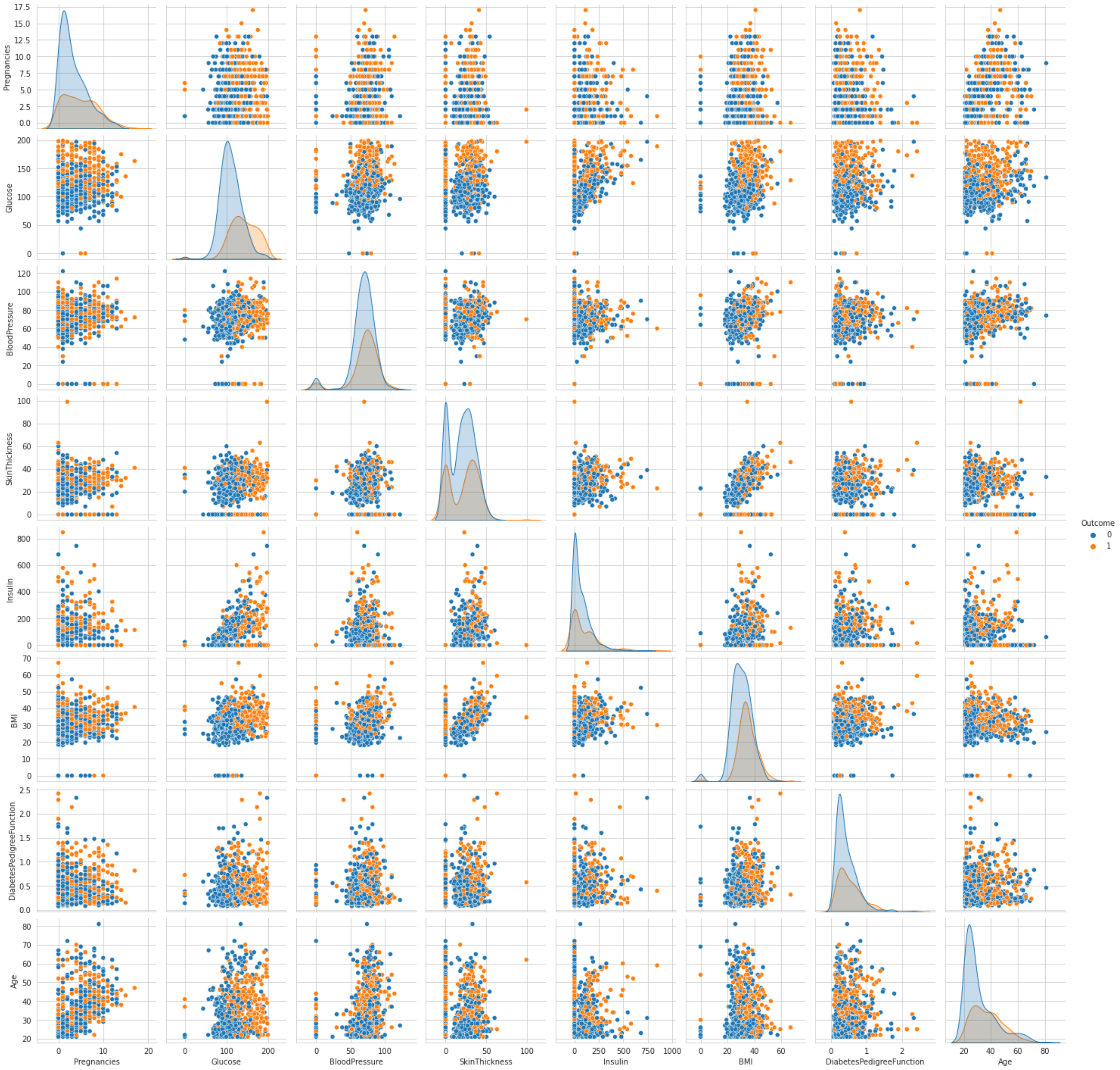


The above plot shows that the SVM model trained after making use of the validation set for hyperparameter tuning by doing grid search had the highest accuracy as it has been tuned with the best parameters among the given parameters for SVM.

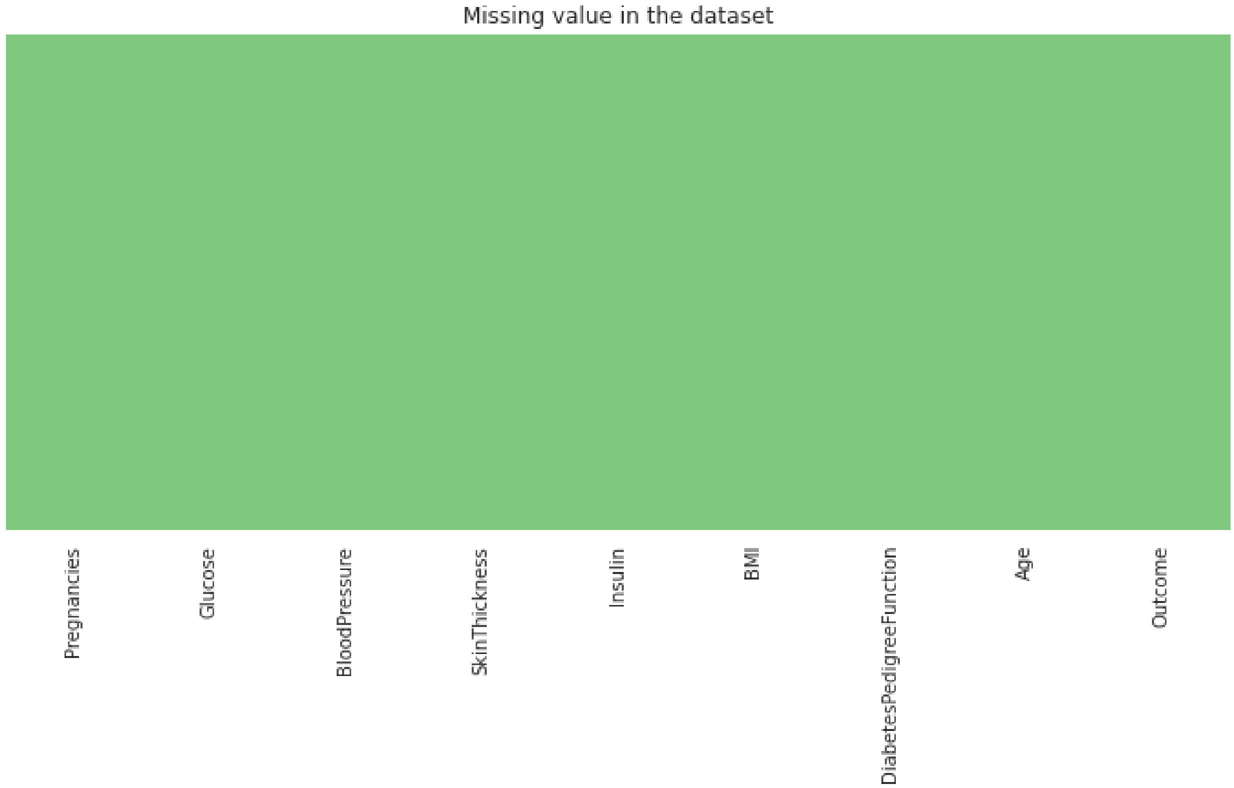
Comparing SVM Classifiers performance on the diabetes dataset

1. Exploratory Data Analysis

Data Visualization-

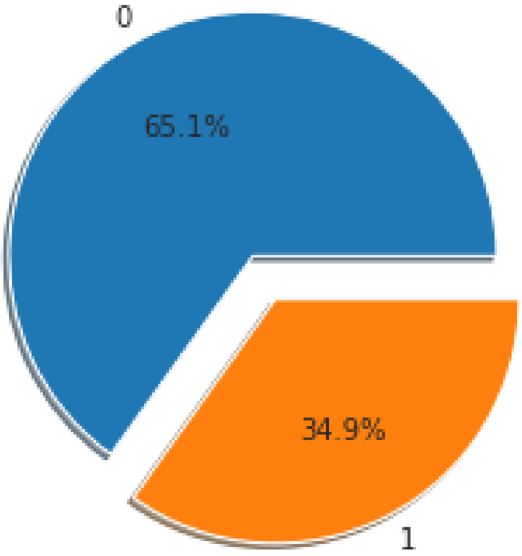


Checking for missing data-

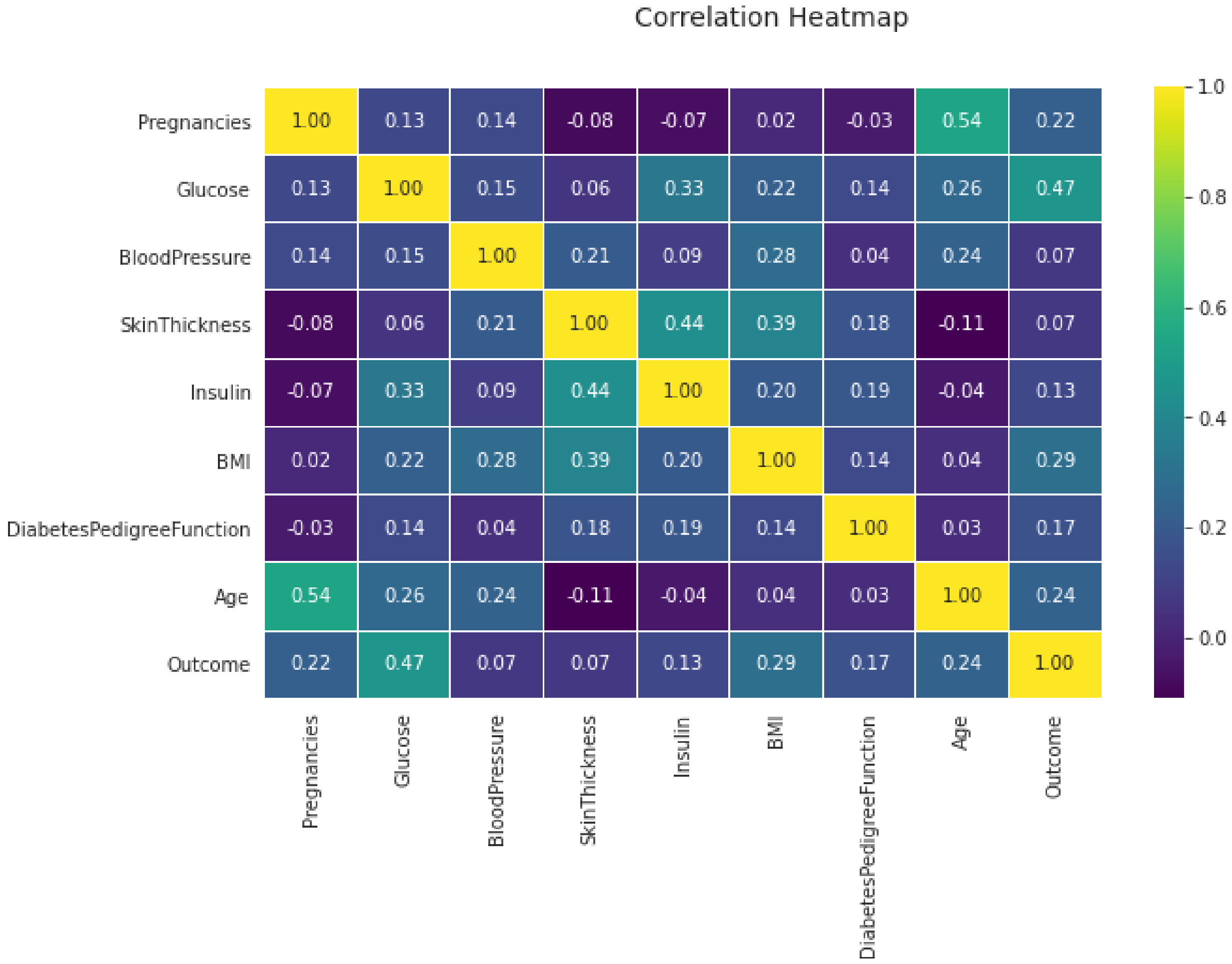


The above plot shows that there is no missing data in the given dataset.

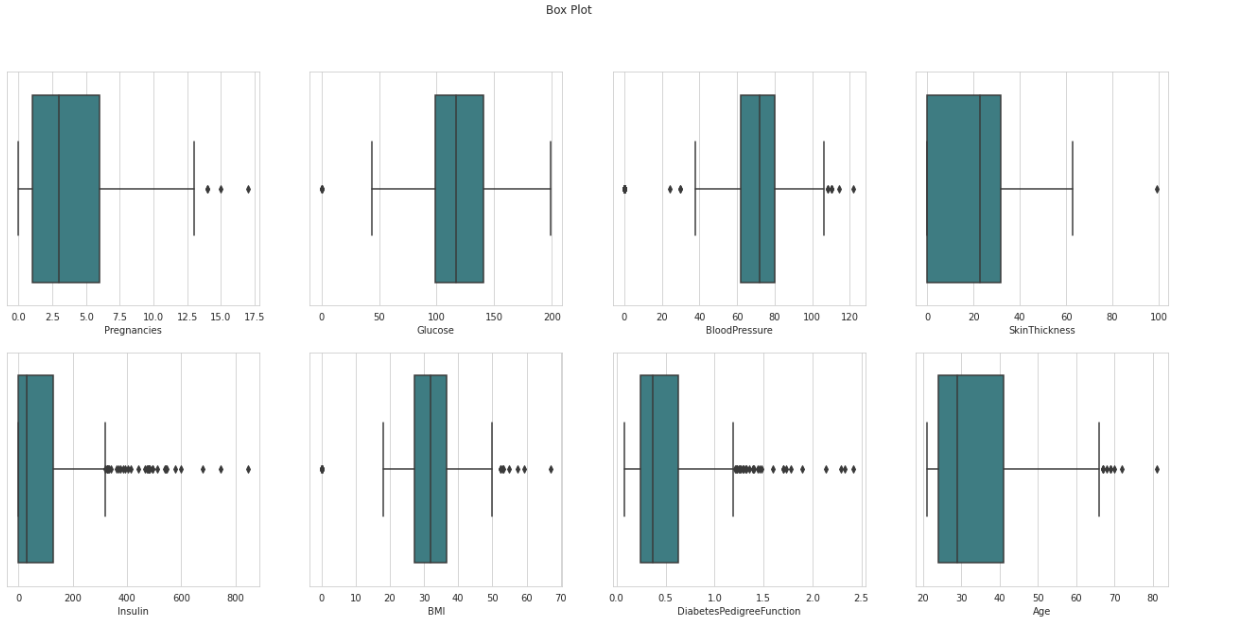
Distribution of the Classes in the dataset-



Plot for correlation between features-

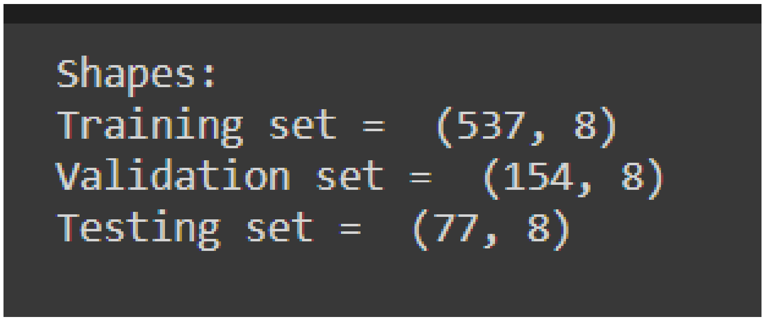


Box plot visualization for the features in the dataset-



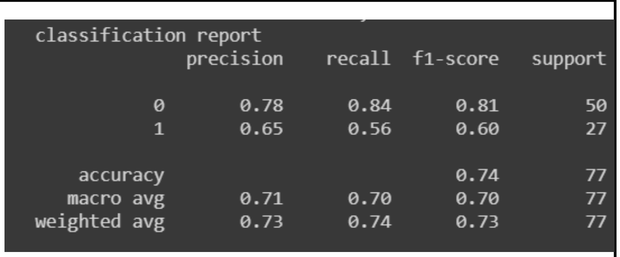
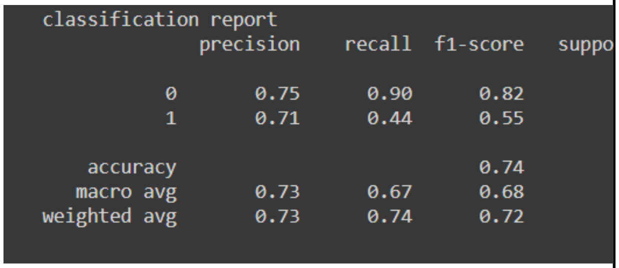
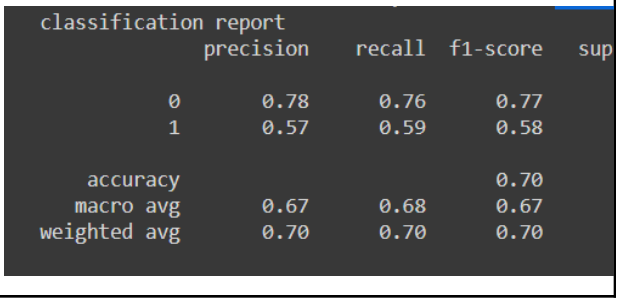
2. Split of the dataset

Having the dataset split into training, validation, and testing sets in the ratio 70:20:10.
Shapes of the datasets:

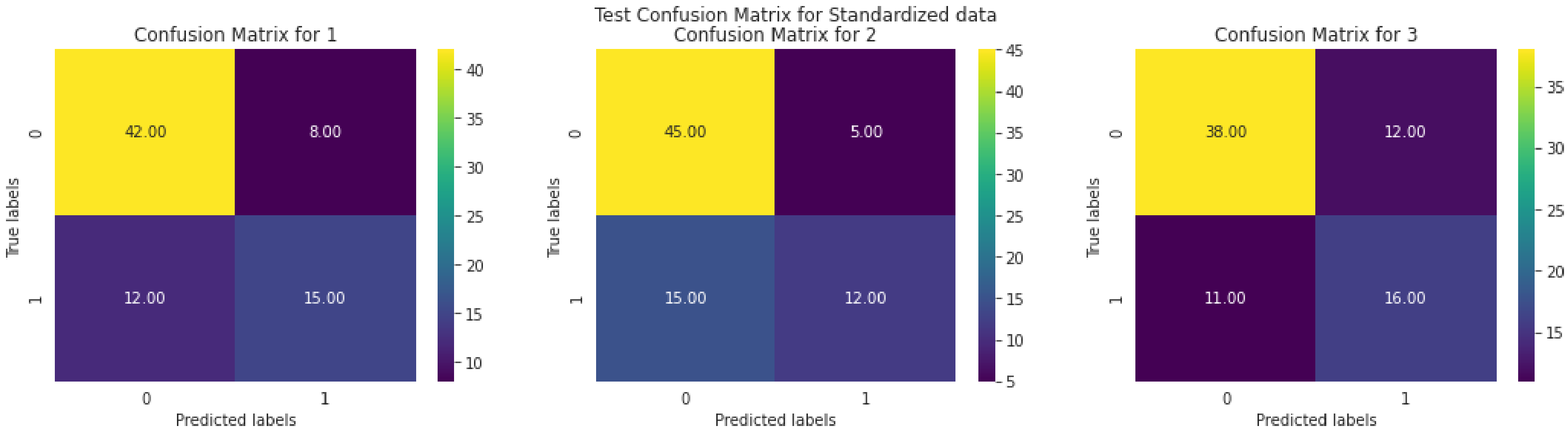


```
Shapes:
Training set =  (537, 8)
Validation set =  (154, 8)
Testing set =  (77, 8)
```

3.Performance Comparison for different SVM models

S.no	Model	Accuracy Scores	Max Score	Min Score	Classification Report
1	SVC linear	0.7279569892473119	0.8387096774193549	0.5806451612903226	 classification report precision recall f1-score support 0 0.78 0.84 0.81 50 1 0.65 0.56 0.60 27 accuracy 0.74 77 macro avg 0.71 0.70 0.70 77 weighted avg 0.73 0.74 0.73 77
2	SVC poly	0.701505376344086	0.8064516129032258	0.5806451612903226	 classification report precision recall f1-score suppo 0 0.75 0.90 0.82 1 0.71 0.44 0.55 accuracy 0.74 macro avg 0.73 0.67 0.68 weighted avg 0.73 0.74 0.72
3	SVC Gaussian	0.7150537634408602	0.8333333333333334	0.6451612903225806	 classification report precision recall f1-score sup 0 0.78 0.76 0.77 1 0.57 0.59 0.58 accuracy 0.70 macro avg 0.67 0.68 0.67 weighted avg 0.70 0.70 0.70

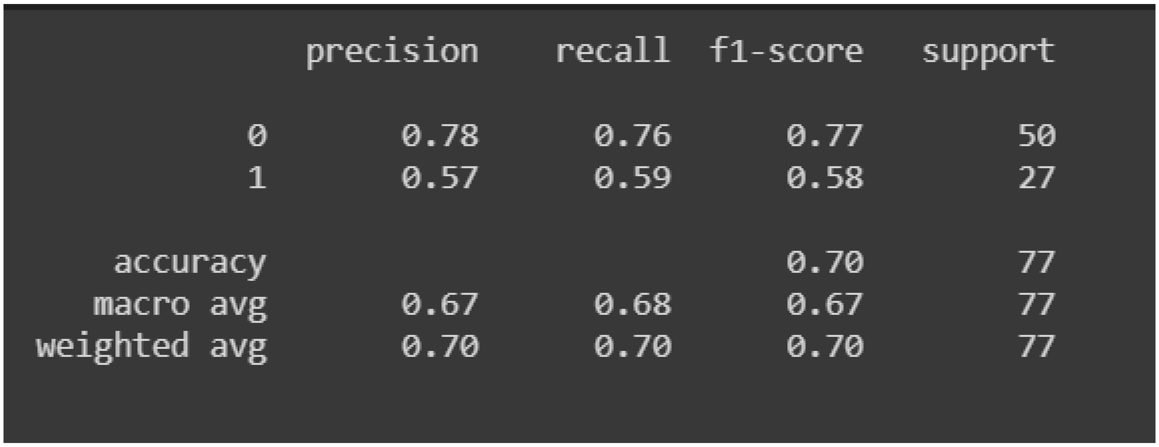
4.Confusion Matrices for the 3 models:



5.Best Model after validation with grid search:

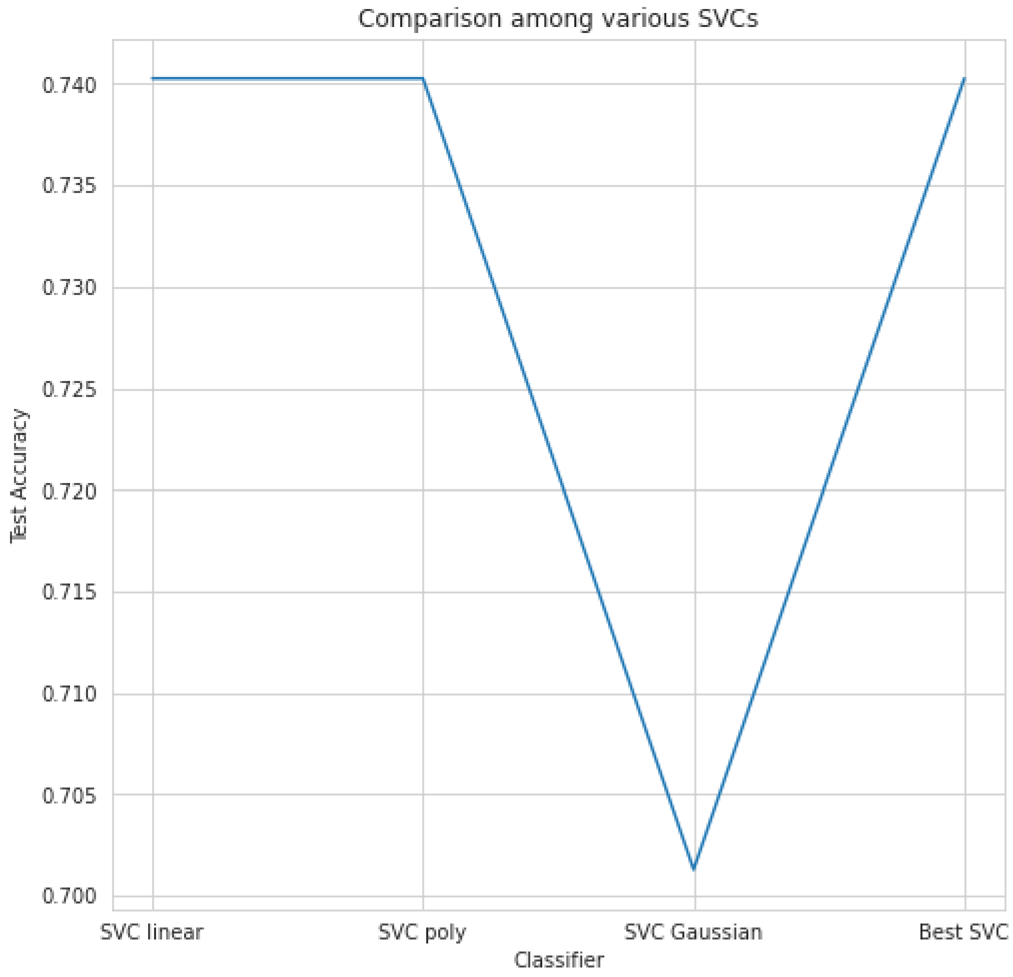
Best parameters : {'C': 0.1, 'kernel': 'linear'}

Classification Report:

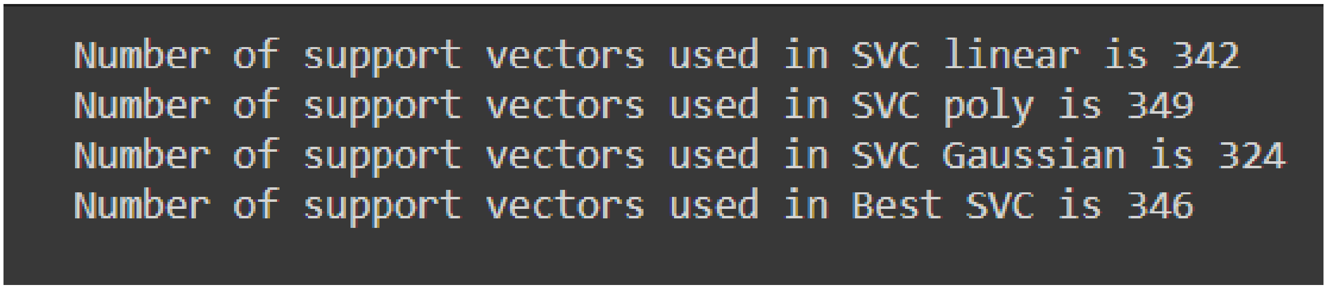


```
              precision    recall  f1-score   support

           0       0.78      0.76      0.77        50
           1       0.57      0.59      0.58        27

    accuracy                           0.70        77
   macro avg       0.67      0.68      0.67        77
weighted avg       0.70      0.70      0.70        77
```

Overall Plot for Performance Comparison:



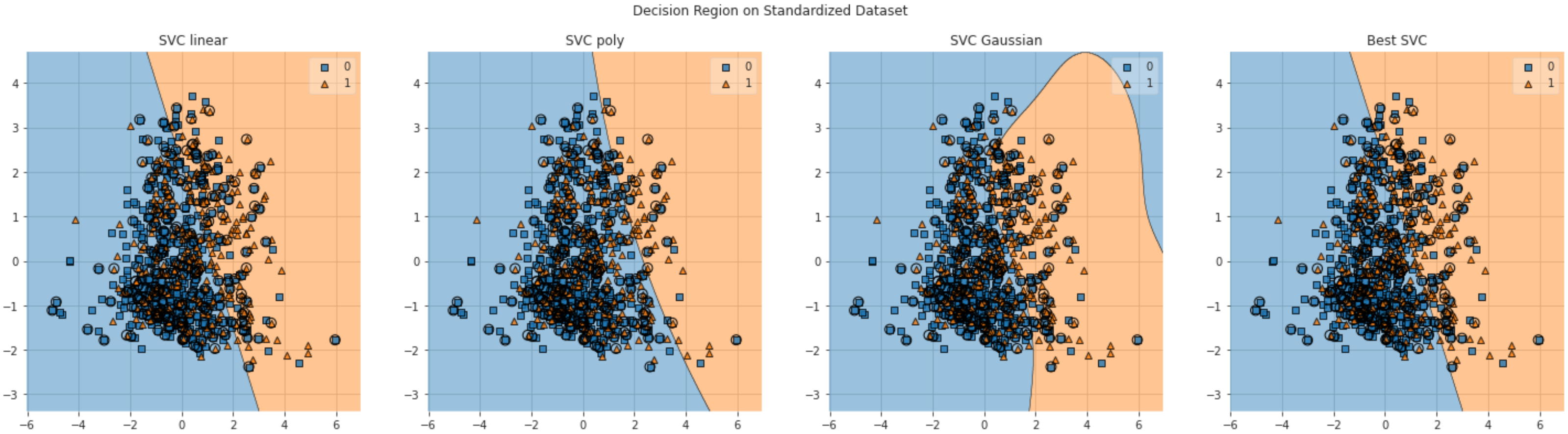
6.Number of support vectors obtained in the final model in each case



```
Number of support vectors used in SVC linear is 342
Number of support vectors used in SVC poly is 349
Number of support vectors used in SVC Gaussian is 324
Number of support vectors used in Best SVC is 346
```

7. Visualizing the separating hyper-plane/decision region

Hyperplane - A hyperplane in an n-dimensional Euclidean space is a flat, n-1 dimensional subset of that space that divides the space into two disconnected parts.


Decision Region on Standardized Dataset
SVC linear
SVC poly
SVC Gaussian
Best SVC