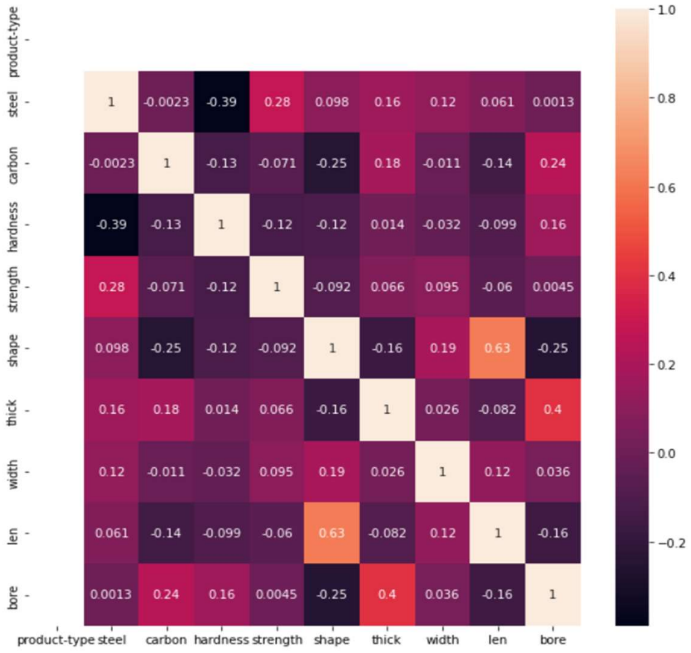
PRML LAB 7

Name: - Shalin Jain
Roll No.: - B21CS070

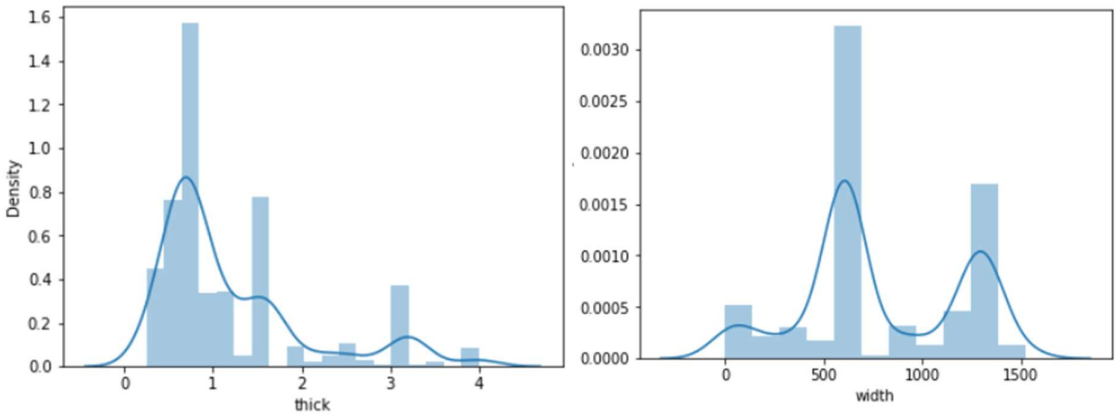
Problem 1

Part 1

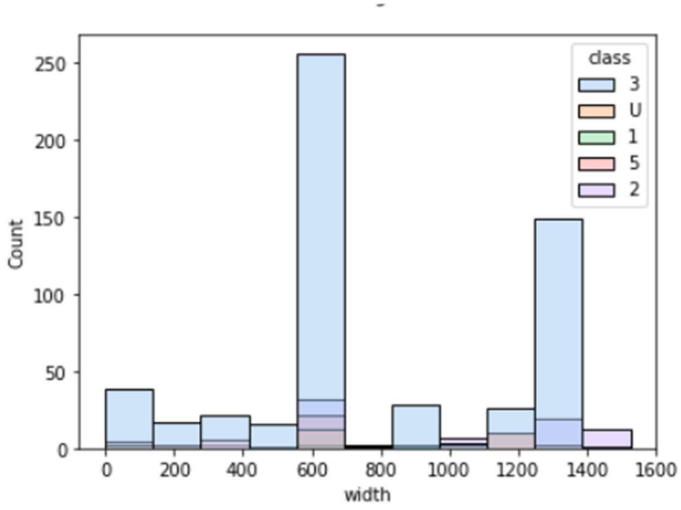
In this part we, downloaded the Annealing dataset and some exploratory analysis is done such as plotting the heatmap to see the relation and dependencies of features of the dataset.

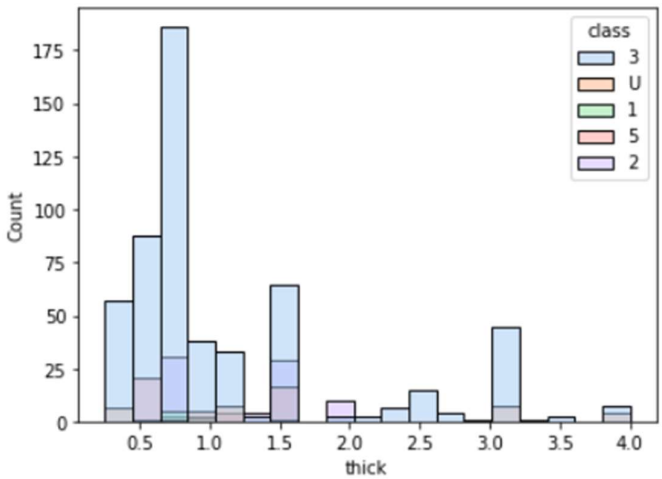


Now we plotted the distplot of all the features to see their distribution.



Now we plotted the count plot of different features to see which feature affect which class the most.





Part 2

In this part including the part 1 we pre-process the data like dropping the columns which have less than 75% of useful data and that dropping the particular rows which have null values. Then we split the scaled the dataset and stored in a new variable.

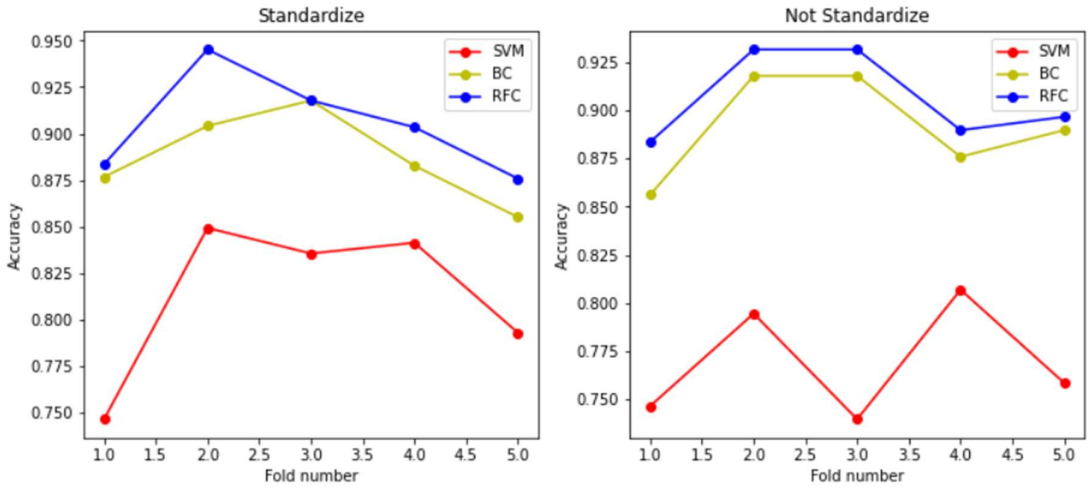
Part 3

Now we trained the model using SVM classifier, Bagging Classifier and Random Forest Classifier on scaled as well as unscaled dataset. Now we performed the cross-validation score (5-fold) on the dataset using these three classifiers and obtained the following graph.

SVM

Bagging Classifier

Random Forest Classifier




```
Standardize [0.74657534 0.84931507 0.83561644 0.84137931 0.79310345]
Not Standardize [0.74657534 0.79452055 0.73972603 0.80689655 0.75862069]
```

From the graph we can say that the

```
Standardize [0.87671233 0.90410959 0.91780822 0.88275862 0.85517241]
Not Standardize [0.85616438 0.91780822 0.91780822 0.87586207 0.88965517]
```

cross_val_score for

```
Standardize [0.88356164 0.94520548 0.91780822 0.90344828 0.87586207]
Not Standardize [0.88356164 0.93150685 0.93150685 0.88965517 0.89655172]
```

the

standardize version is more stable that the non – standardize version of the dataset. This is because the non – standardize or non – scaled might give a higher preference to data with large values.

Part 4

In this part I have made a class of PCA with the following functions:

```
__init__(self,dataset,n_components,standardize = True):
```

In this function we are taking the dataset to be reduced or transformed as an input. n_components i.e. the number of components in which dataset is to be reduced.

```
_Centrailizing(self):
```

This function scales or centralize the input dataset by subtracting mean and dividing each value of the column by its standard deviation.

```
_covariance_mat(self):
```

This function finds the covariance matrix for the given dataset

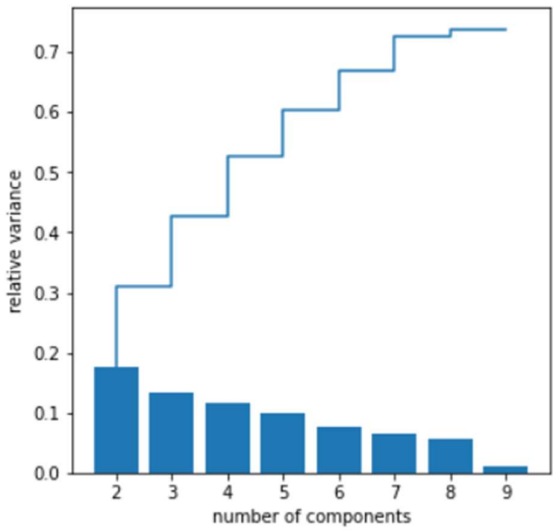
```
_eigen_val_vec(self):
```

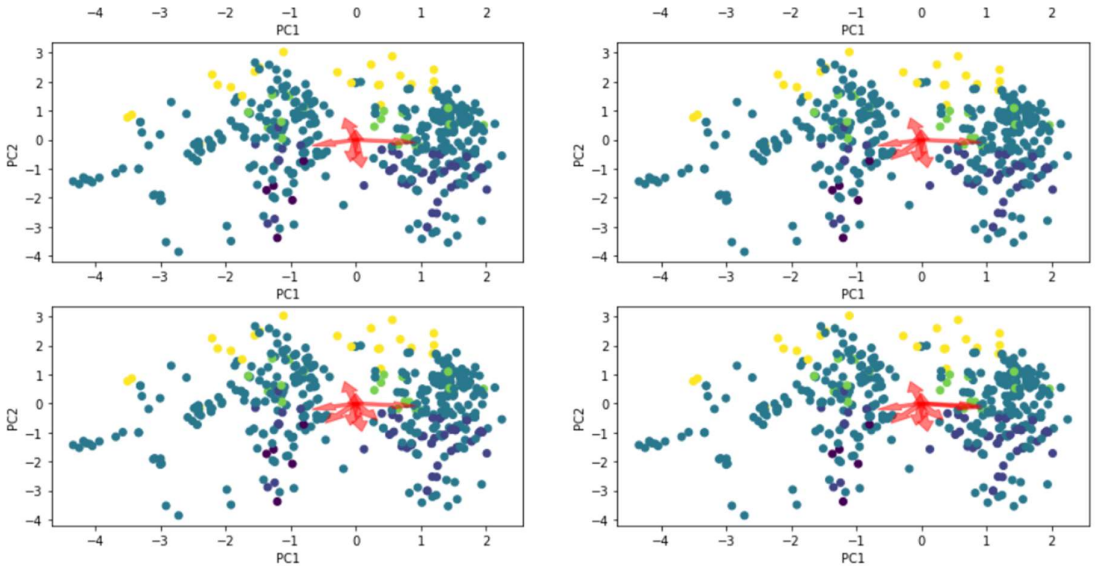
This function finds the eigen values and the eigen vectors of the covariance and the sort the vector, value pair in reverse order.

```
fit(self):
```

In this function we call all the above functions and transform the dataset by taking dot product with the eigen vectors till the n_components value.

Part 5

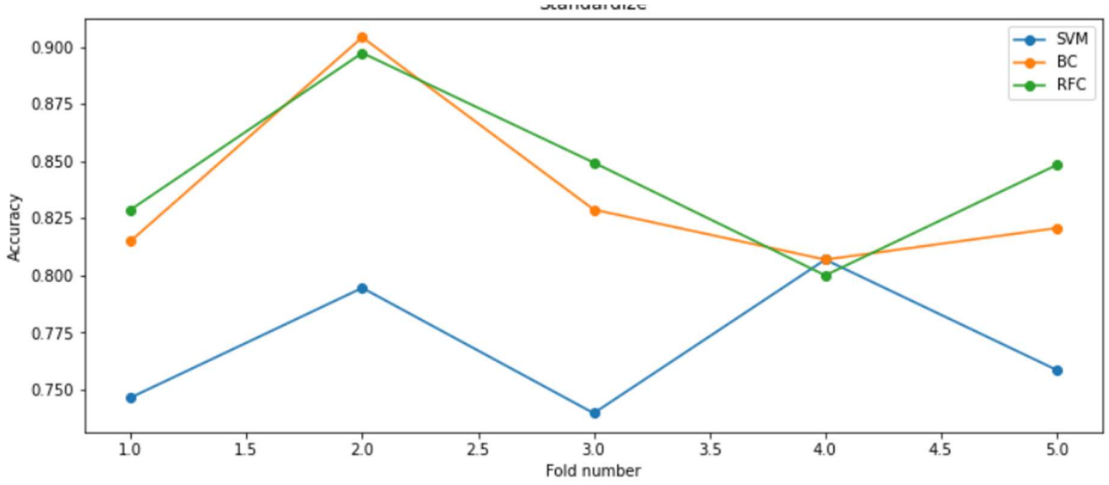




From the figures we can see that with increase in n_components there is increase in explained variance ratio as show in the step bar plot in figure.

Part 6

In the part 6 we trained the model using the PCA and find the cross validation score as shown in the figure.



We have used the SVM, Bagging and Random Forest Classifiers. We also found the accuracy and the f1 score of the model. Classifiers used are the same as above but done in two parts. The first is reduced dataset using PCA and in second case we have not reduced the dataset.

For SVM Classifier: -

```
Reduced --> accuracy_score 0.7803921568627451
Reduced --> f1-score 0.6841323313466355

UnReduced --> accuracy_score 0.8117647058823529
UnReduced --> f1-score 0.7463473336186737
```

For Bagging Classifier: -

```
Reduced --> accuracy_score 0.8352941176470589
Reduced --> f1_score 0.8151792498541726

UnReduced --> accuracy_score 0.8705882352941177
UnReduced --> f1_score 0.8546682899739416
```

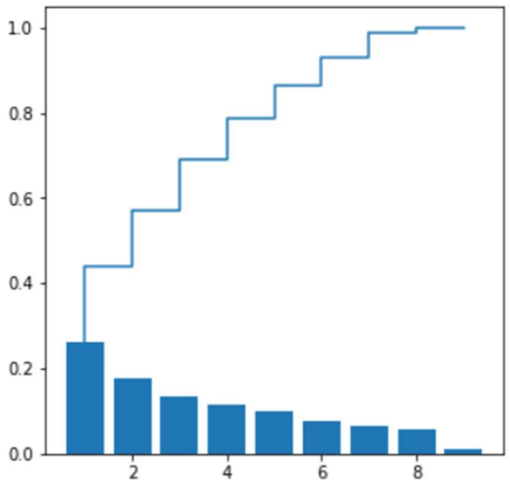
For Random Forest Classifier: -

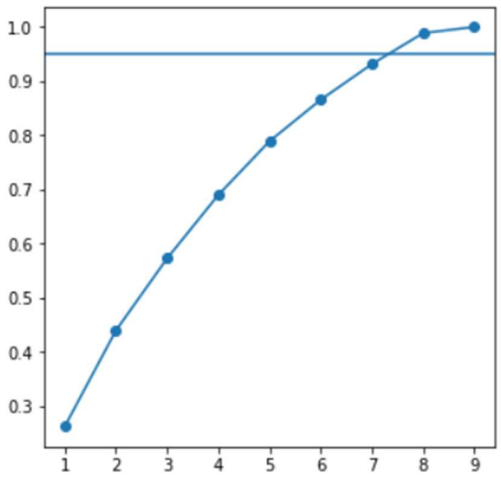
```
Reduced --> accuracy_score 0.8470588235294118
Reduced --> f1-score 0.8344097895568482

UnReduced --> accuracy_score 0.8901960784313725
UnReduced --> f1-score 0.8806928726128469
```

We can without reducing the dataset and on reducing the dataset the accuracy and f1 -score hasn't reduced much. So, we can conclude that the 2 features are only dominating in the classification task and the rest features are not much contributing in the classification.

Part 7

We can see from the part 6 that there only 2 which affect the decision of classification. So yes, there were changes in the dataset when PCA is applied. The dataset is reduced and the it easy to separate the classes in the new vector space.



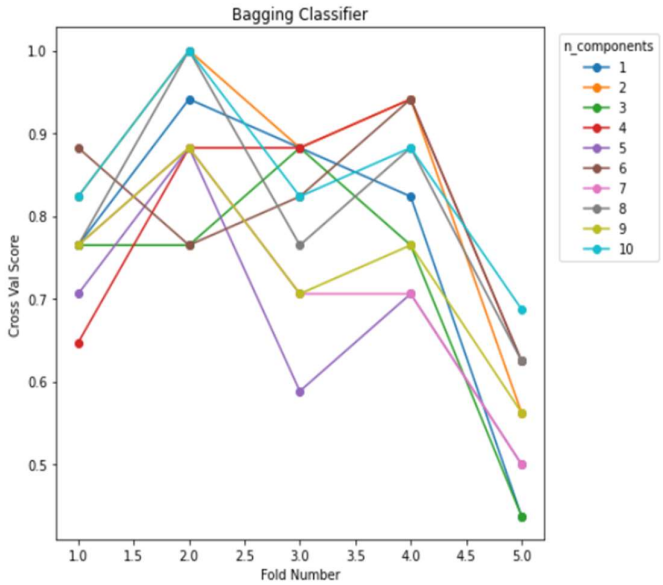


From the graph we can also say that after 7 components there is not much increase in the variance, but for the particular dataset we can also say that since the magnitude of the eigenvalues beyond 7 values does not affect the variance cumulative ratio. Hence optimal number of components can be decided there as 7.

Bonus

In this part we have made a slight change in the class of PCA. The change is that we made the non-diagonal values of the covariance matrix as 0 to inculcate the effect of independency among the features. We see that we got the high accuracy on the number of components at 5 as shown below: -

```
Number of componets 1 Accuracy obtained => 0.7058823529411765
Number of componets 2 Accuracy obtained => 0.8235294117647058
Number of componets 3 Accuracy obtained => 0.6470588235294118
Number of componets 4 Accuracy obtained => 0.8235294117647058
Number of componets 5 Accuracy obtained => 0.8235294117647058
Number of componets 6 Accuracy obtained => 0.7058823529411765
Number of componets 7 Accuracy obtained => 0.6470588235294118
Number of componets 8 Accuracy obtained => 0.8235294117647058
Number of componets 9 Accuracy obtained => 0.6470588235294118
Number of componets 10 Accuracy obtained => 0.7058823529411765
```



One advantage of applying the naïve bayes is that we have to calculate only variance of feature columns instead of calculating the whole matrix and since it has similar accuracy and n_components = 2. We can say that it reduces the computing time for high dimensional data.

Problem 2

Part 1

In this part we have made the class LDA i.e., Linear Discriminant Analysis with the following functions: -

```
__init__(self,per_var = 0.90):
```

This function acts as a constructor for the class which takes the percentage of variance that is need to be conserved.

```
fit(self,X,y):
```

This function takes X and y as the input and calculate the scatter matrix between and within the class using the give formula: -

Within class = $\sum_j \sum_{xj \in \Delta j} (x-u)(x-u).T$

Between Class = $\sum_j n(uj-u)(uj-u).T$

We have also calculated the eigenvalues and the eigen vectors of the matrix in this function itself.

```
select_components(self):
```

In this function we have used the variance ratio to select the number of components for the reduction of the dataset. The relation used is as follows: -

```
self.n_components = np.sum(self.cumsum_var < self.per_var) + 1
```

```
transform(self,X):
```

In this function we have only transformed the data using the eigen vectors taking the dot product of the dataset and transpose of the eigen vector array.

```
predict(self,X,transform = True):
```

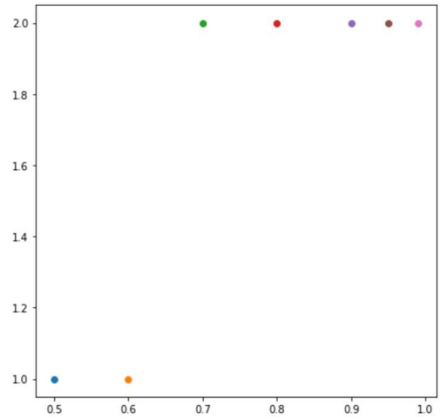
In this function we have transform the data if not transformed and predicted the class of the input by calculating the distance from classes means.

```
predict_proba(self,X):
```

In this function we have calculated the probability of input data to fall in a particular class by normalizing the distances found in the predict function.

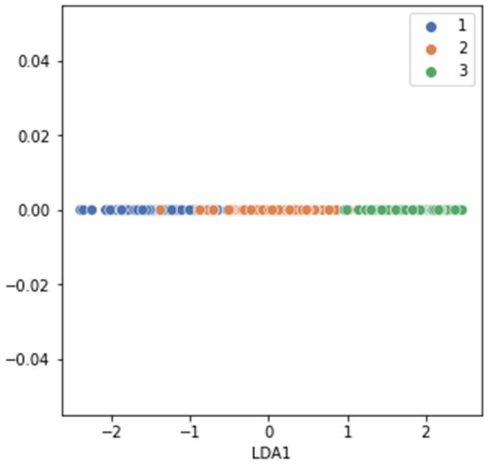
Part 2

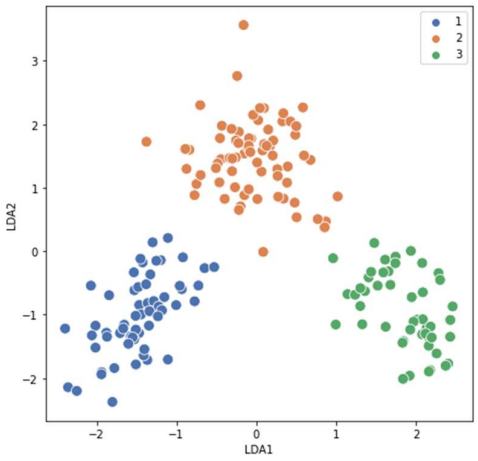
Now I have varied the variance as shown in the graph and calculated the number of linear discriminants: -




```
Number of High imapct features 1
Number of High imapct features 1
Number of High imapct features 2
Number of High imapct features 2
Number of High imapct features 2
Number of High imapct features 2
Number of High imapct features 2
```

Now I have plotted the separation of classes using these linear discriminants as follows: -





Part 3

In this part I have trained the bagging classifier and the KNN classifier for classification purpose and used LDA and PCA for reduction of dataset.

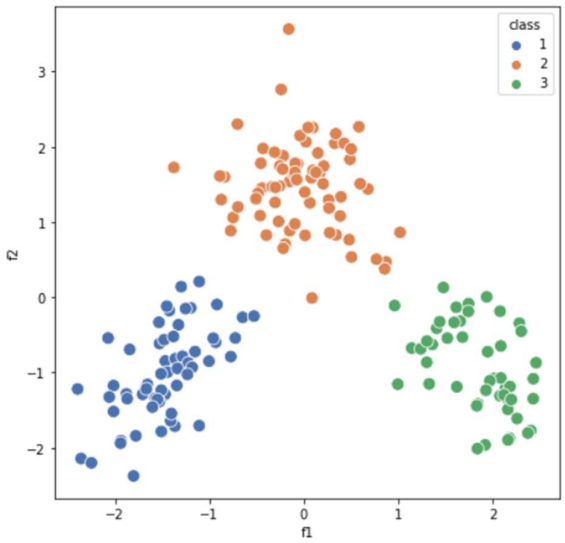
Part 4

The obtained results of the Part 3 as stored in table as shown in the figure: -

	Classification Technique	Reduction Technique	Accuracy
0	Bagging Classifier	LDA	0.814815
1	KNN Classifier	LDA	0.740741
2	Bagging Classifier	PCA	0.962963
3	KNN Classifier	PCA	0.981481

We can also make the same table in word.

The obtained graph separating the classes using 2 linear discriminants is as follows:

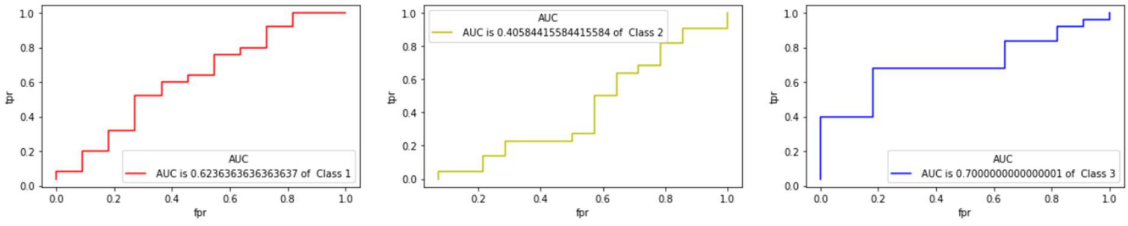


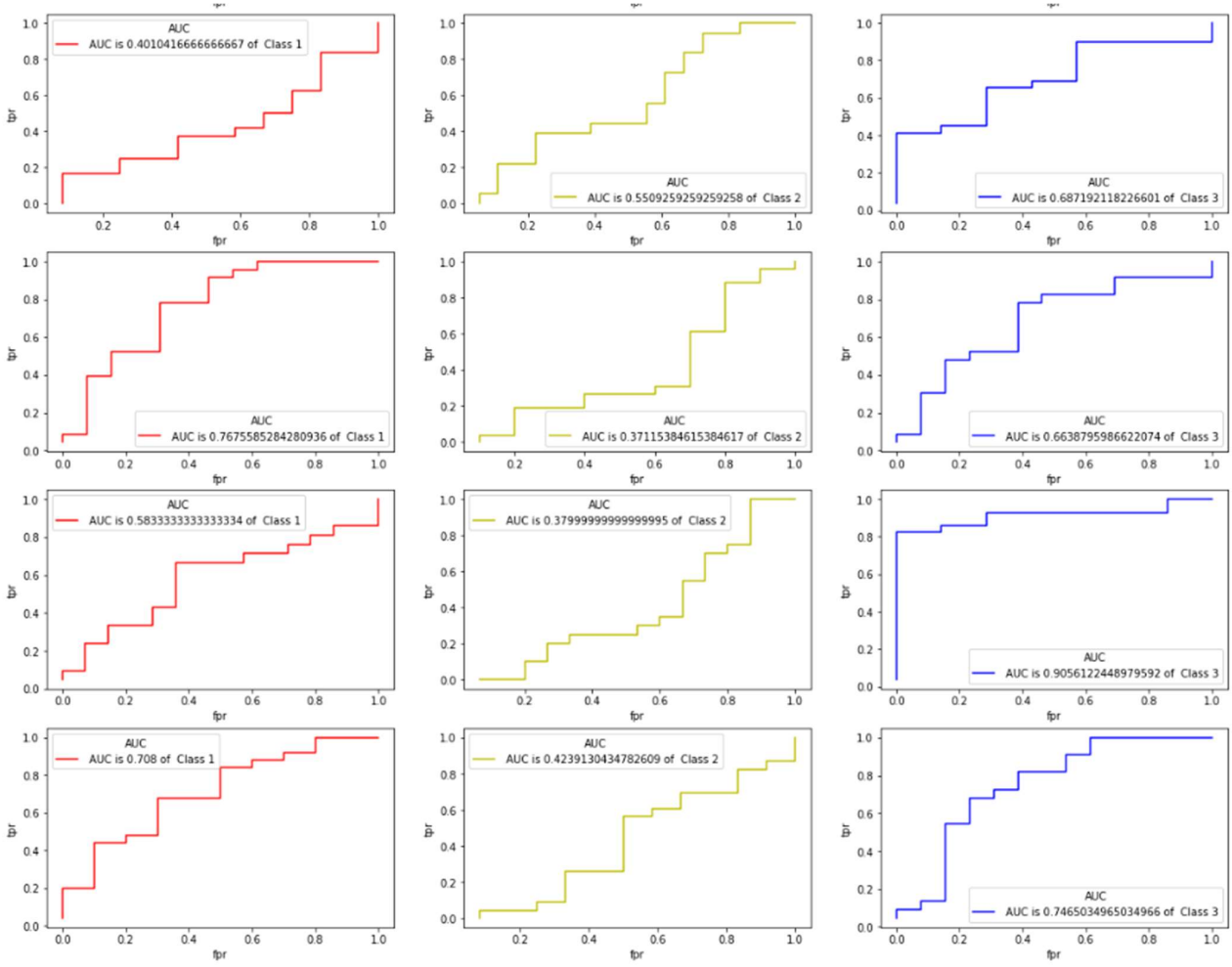
Part 5

In this part we have used LDA as classifier and used the predict function which for coded in the part 1 of the question. We used this predict function to predict the class of the input. Since the tpr and fpr are mainly calculated for the binary classification so we have assumed correct classification as 1 and incorrect classification for the multiclass as 0. We have formed a function roc_curve to find the tpr and fpr and auc to calculate the area of the curve using the trapezoidal formula.

The obtained roc auc curves and the five-fold cross-validation scores are as follows: -

```
[1.0, 0.9722222222222222, 1.0, 0.9142857142857143, 0.9714285714285714]
```




AUC
AUC is 0.4010416666666667 of Class 1
AUC
AUC is 0.5509259259259258 of Class 2
AUC
AUC is 0.687192118226601 of Class 3
AUC
AUC is 0.7675585284280936 of Class 1
AUC
AUC is 0.37115384615384617 of Class 2
AUC
AUC is 0.6638795986622074 of Class 3
AUC
AUC is 0.5833333333333334 of Class 1
AUC
AUC is 0.37999999999999995 of Class 2
AUC
AUC is 0.9056122448979592 of Class 3
AUC
AUC is 0.708 of Class 1
AUC
AUC is 0.4239130434782609 of Class 2
AUC
AUC is 0.7465034965034966 of Class 3
tpr
fpr