Pattern Recognition and Machine Learning

Lab - 7 Assignment

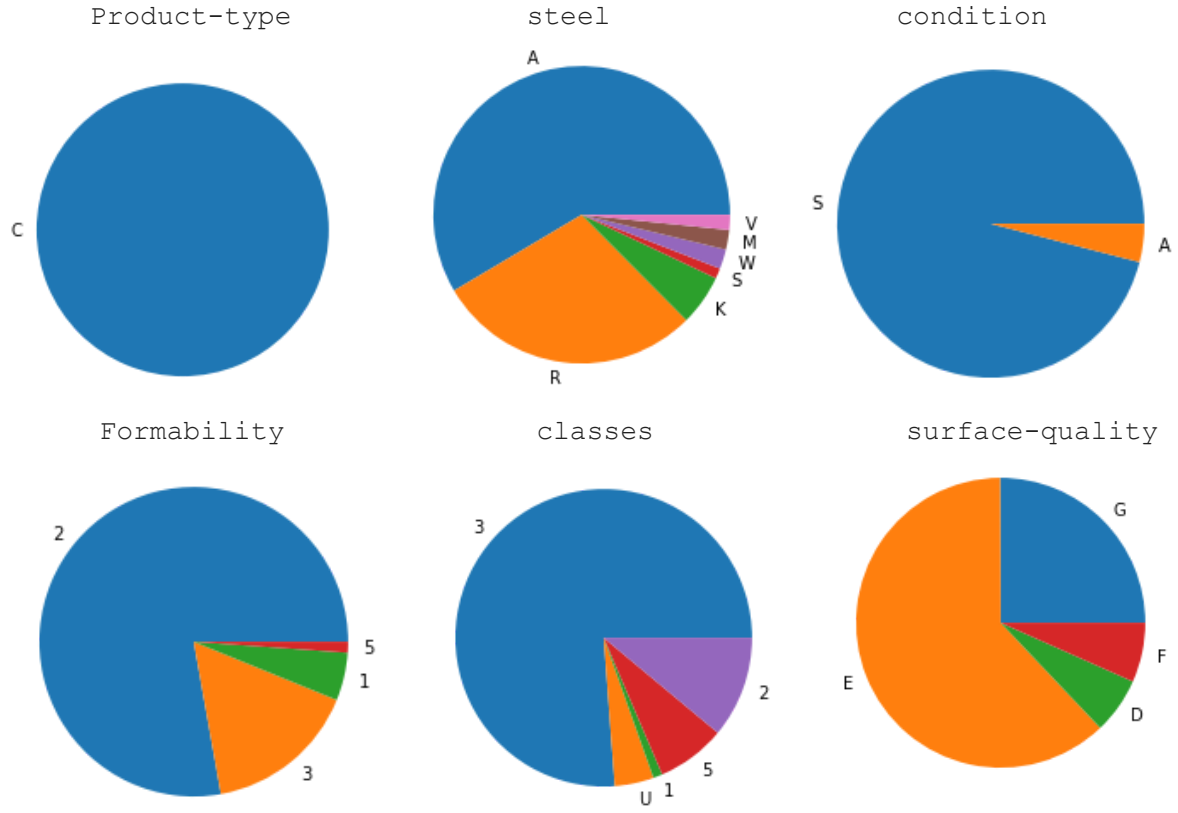
Akriti Gupta
B21AI005

Question 01:

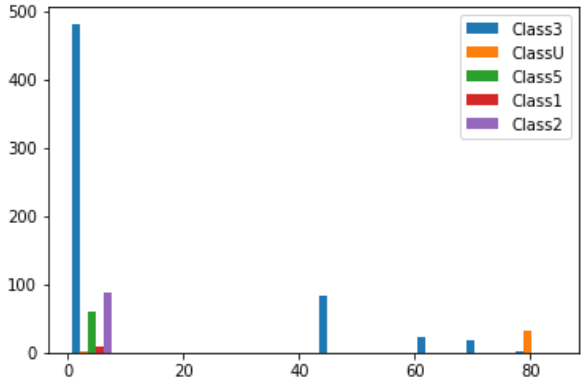
Dataset: Annealing Dataset

Part 1:

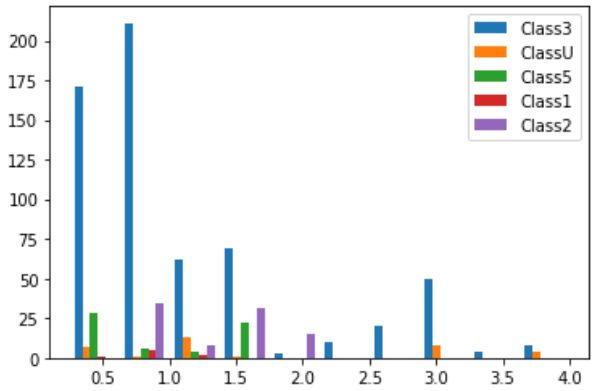
We start by iporting all the necessary libraries and then import the anneal.data and anneal.test files by adding a dummy column on the top row of these files before importing them in order to preserve the top most row. Now we get all the column names from anneal.names file and apply them to both of our training and testing dataframes. Next, we drop the column having more than 75% of the data as NaN values i.e '?' in this case as these do not provide sufficient information. For others we replace Nan values by the most frequent occurring value. Now, we do exploratory analysis by plotting pie charts for categorical features and histogram for numerical features. The plots observed are as follows:



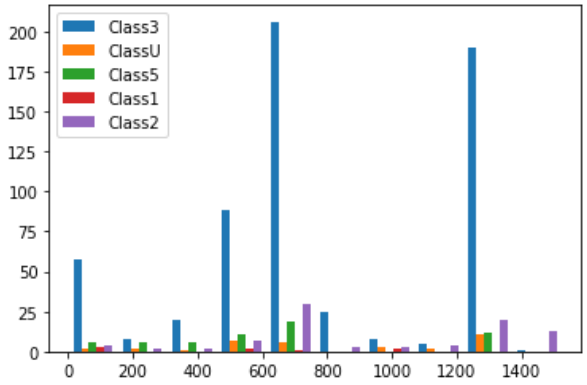
hardness



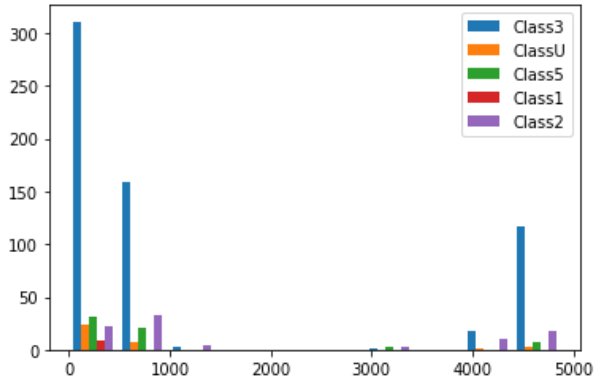
thick



Width



len



Other plots are in the ipynb file.

Part 2:

In order to preprocess the dataset, we start by analyzing the error in the dataset by applying counter to each column then we drop the ‘product-type’ column as its value is same for all rows. Next, we apply label encoding to the categorical features. We the create two datasets- one with continuous features that are standardized, and another with continuous features that are not standardized. To standardize the dataset, we use StandardScalar() module of sklearn. We then split both the standardized and non-standardized data into training and testing sets using a 65:35 ratio and keeping the random_state consistent for analysis. Finally, we preprocess the testing dataset using the same steps followed by us to preprocess the he training data.

Part 3:

The 3 classification models we chose are as follows:

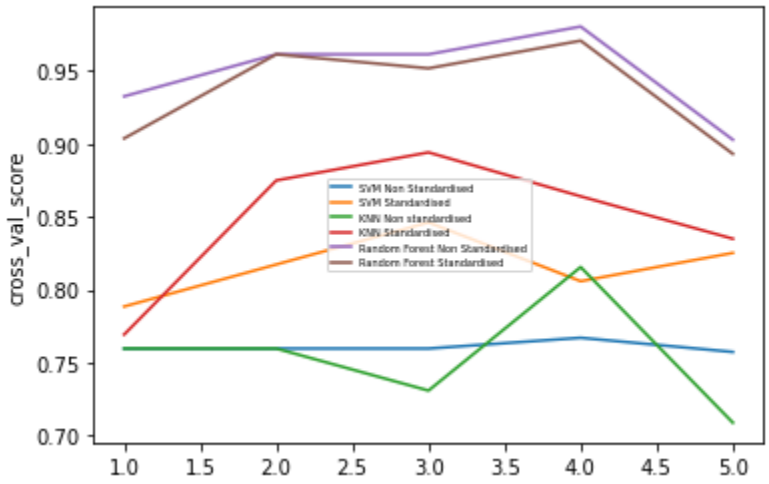
1. SVM: The reason for using SVMs is because they are effective in handling multi-class problems, which is exactly what our dataset is.
   ```
   Cross Validation Score for SVM with non standardized data is
   0.7606235997012696.
   Cross Validation Score for SVM with standardized data is
   0.816598207617625.
   ```

2. KNN: We chose this because it works nicely for categorical as well as continuous data and can accommodate dataset that is of average size like our dataset.
   ```
   Cross Validation Score for KNN with non standardized data is
   0.7548543689320388.
   Cross Validation Score for KNN with standardized data is
   0.8474981329350261.
   ```

3. Random Forest: This classification model also works very nicely for dataset with both categorical and continuous attributes. Also as it is an ensemble of decision trees, it reduces overfitting and gives good results.
   ```
   Cross Validation Score for Random Forest with non standardized data
   is 0.9478528752800598.
   Cross Validation Score for Random Forest with standardized data is
   0.936277072442121.
   ```

So, the average cross validation score with standardization is better for all the 3 models.

We plot the cross validation score of each model and obtain the below graph:



From this, we observe that random_forest model gives better score than the other two models i.e knn and svm models.

Part 4:

In order to implement PCA fro scratch we create a class named PCA_scratch. The attributes of the class are as follows:

1. df: It contains the dataset we need to fit in the model.
2. comp: It contains the value of n_components i.e the no. of columns we we get after PCA reduction.
3. Eigenval: It contains the eigenvalues of covariance matrix of columns of df.
4. Eigenvector: It contains the eigenvectors corresponding to the eigenvalues of the covariance matrix.

The methods of the class are as follows:

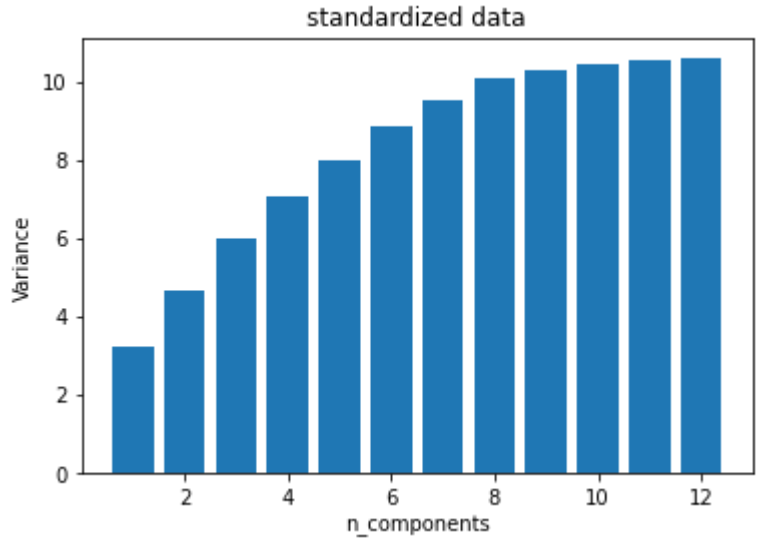
1. centralize_data: it gives the centralized data via feature-wise means and standard deviations.
2. cov_mat: It gives the covariance matrix of the attributes in the dataset.
3. eigen: It gives the eigenvalues and corresponding eigenvectors.
4. principal_comp: Here, we firstly sort the eigenvalues in decreasing order along with corresponding eigenvectors and then get the \eigenvectors equal to the number of n_components and return them as principal components.
5. Fit_transform: This gives the transformed dataset by using the principal components.

Part 5:

Now we reduce both the standardized data and the non standardized data by using our PCA class by choosing n_components i.e principal components=9. So, we get the shapes of datasets as `(518, 9)`.

Bar graph to show the change in variance as we increase the no. of components is obtained as follows:

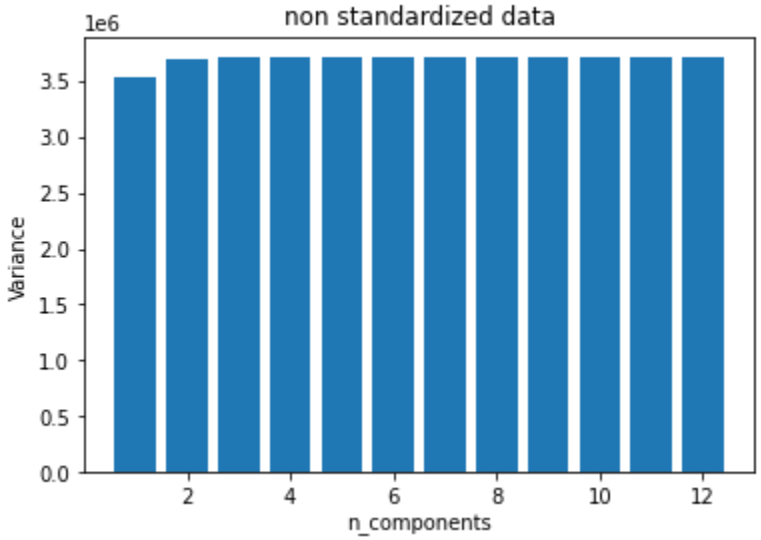
For standardized dataset:



We observe that after n_components=10, the value of variance doesn't increase much that is for n_components=10 , we get the variance=98%. Hence, it is the most optimal value. We also get this result by using a for loop to get the value of least value n_components for which variance>=98%. We get the below result:

```
best value of n_components: 10
```

For non-standardized dataset:



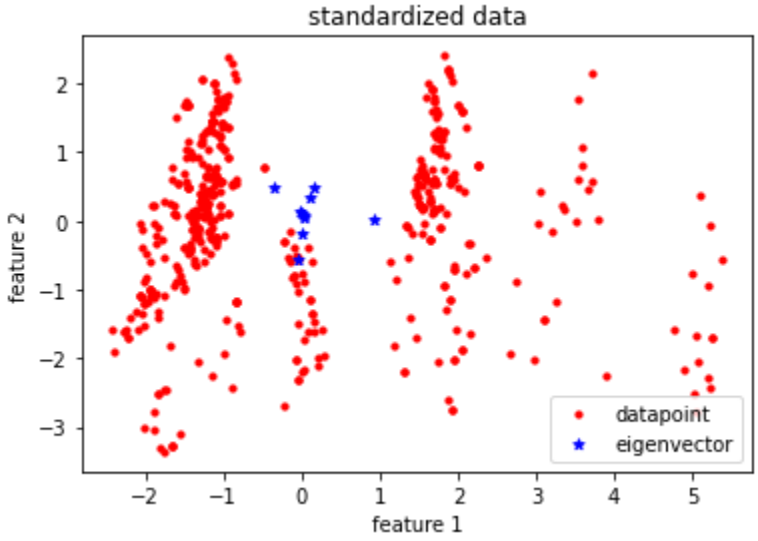
Here, we observe that after n_components=2, we get variance>=99%. Hence it is the most optimal value. We also get this result by using a for loop to get the

value of least value n_components for which variance>=98%. We get the below result:

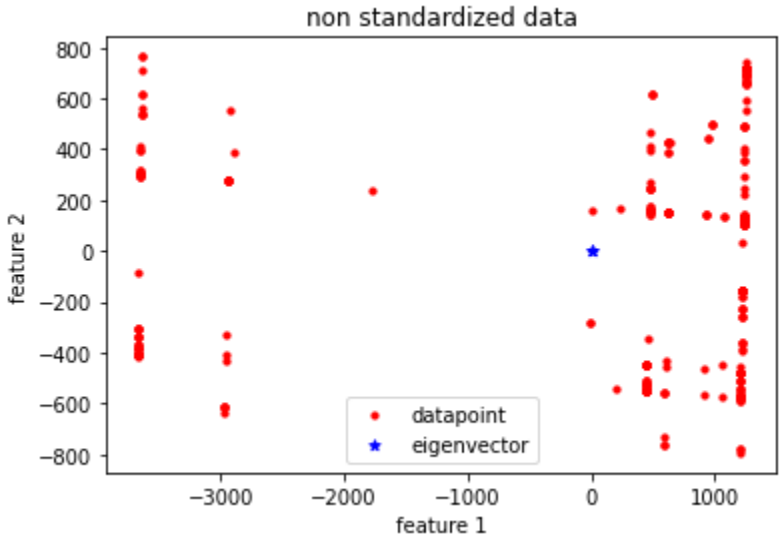
```
best value of n_components: 2
```

Hence, we train the datasets with most optimal values of n_components i.e 10 for standardized and 2 of non-standardized.

The scatter plot to show the direction of eigenvectirs along with datapoints are observed as below:
For standardized dataset:



For non-standardized dataset:



Part 6:

Now, we use the below mentioned classification models to train the datasets and get the scores as follows:

1. SVM:

   ```
   Cross Validation Score for SVM with standardized data with pca is
   0.8224234503360718.
   Cross Validation Score for SVM with non standardized data with pca
   is 0.7606235997012696.
   ```

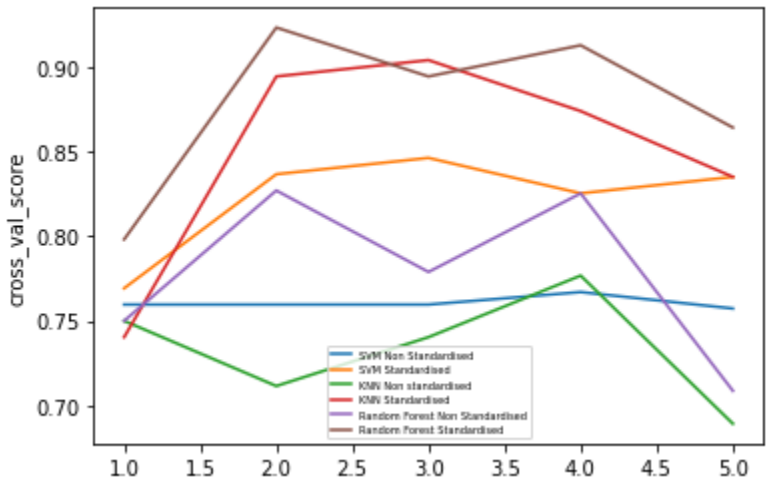
2. Random Forest:

   ```
   Cross Validation Score for Random Forest with standardized data with
   pca is 0.8784167289021658.
   Cross Validation Score for Random Forest with non standardized data
   with pca is 0.7779499626587005.
   ```

3. KNN:

   ```
   Cross Validation Score for KNN with standardized data with pca is
   0.8494398805078417.
   Cross Validation Score for KNN with non standardized data with pca
   is 0.7335884988797611.
   ```

We plot the cross validation score of each model and obtain the below graph:



We observe that we still get best score for random forest but it is less than what we got befor pca.

Part 7:

We choose the below given evaluation metrics to test our classification models:

1. F1_score: The advantage of using it is that it considers both precision and recall, making it particularly helpful in situations where there is class imbalance.
2. accuracy_score: gives the average score and hence tells us how good a model is performing.
3. recall_score: Recall score is used to measure the model performance in terms of measuring the count of true positives in a correct manner out of all the actual positive values. Precision-Recall score is a useful measure of success of prediction when the classes are very imbalanced.

SVM Model:

Before PCA:

```
f1 score for standardized data before pca of svm model:
0.856873926137426
accuracy_score score for standardized data before pca of svm model:
0.8783783783783784
recall_score for standardized data before pca of svm model:
0.8783783783783784

f1 score for non_standardized data before pca of svm model:
0.657200433516223
accuracy_score score for non_standardized data before pca of svm
model: 0.7606177606177607
recall_score for non_standardized data before pca of svm model:
0.7606177606177607
```

After PCA:

```
f1 score for standardized data after pca of svm model:
0.833663497438156
accuracy_score score for standardized data after pca of svm model:
0.8706563706563707
recall_score for standardized data after pca of svm model:
0.8706563706563707

f1 score for non standardized data after pca of svm model:
0.657200433516223
accuracy_score score for non standardized data after pca of svm
model: 0.7606177606177607
recall_score for non standardized data after pca of svm model:
0.7606177606177607
```

Here, we observe that standardization through PCA resulted in improved scores. This is likely due to SVM's sensitivity to feature scaling. Interestingly, for non-standardized data, the scores remained largely unchanged before and after PCA. However, for standardized data, there was a decrease in score after PCA was applied. This could be due to the fact that the transformation to a lower dimension resulted in the loss of necessary relevant information required for classification from the original dataset.

Random Forest Model:

Before PCA:

```
f1 score for standardized data before pca of Random Forest model:
0.9941989417482934
accuracy_score score for standardized data before pca of Random
Forest model: 0.9942084942084942
recall_score for standardized data before pca of Random Forest
model: 0.9942084942084942

f1 score for non standardized data before pca of Random Forest
model: 0.9941723857655339
accuracy_score score for non standardized data before pca of Random
Forest model: 0.9942084942084942
recall_score for non standardized data before pca of Random Forest
model: 0.9942084942084942
```

After PCA:

```
f1 score for standardized data after pca of random_forest_model:
0.9942176847386505
accuracy_score score for standardized data after pca of
random_forest_model: 0.9942084942084942
recall_score for standardized data after pca of random_forest_model:
0.9942084942084942

f1 score for non standardized data after pca of Random Forest model:
0.9727444747750239
accuracy_score score for non standardized data after pca of Random
Forest model: 0.972972972972973
recall_score for non standardized data after pca of Random Forest
model: 0.972972972972973
```

Here, we observe that standardization through PCA resulted in improved scores for standardized data as the random forest model focuses on the important

attributes to calculate the class.But, for non-standaridized dataset the score decreased.

KNN Model:

Before PCA:

```
f1 score for standardized data before pca of KNN_model:
0.9017532804733915
accuracy_score score for standardized data before pca of KNN_model:
0.9034749034749034
recall_score for standardized data before pca of KNN_model:
0.9034749034749034

f1 score for non standardized data before pca of KNN_model:
0.8494259863309176
accuracy_score score for non standardized data before pca of
KNN_model: 0.859073359073359
recall_score for non standardized data before pca of KNN_model:
0.859073359073359
```

After PCA:

```
f1 score for standardized data after pca of KNN_model:
0.9055067916141887
accuracy_score score for standardized data after pca of KNN_model:
0.9073359073359073
recall_score for standardized data after pca of KNN_model:
0.9073359073359073

f1 score for non standardized data after pca of KNN_model:
0.8054094792437857
accuracy_score score for non standardized data after pca of
KNN_model: 0.8301158301158301
recall_score for non standardized data after pca of KNN_model:
0.8301158301158301
```

For, this model there is not much change on applying PCA. So, there is only drawback that is applying PCA requires additional preprocessing steps and parameter tuning, which can increase the complexity of the overall pipeline.

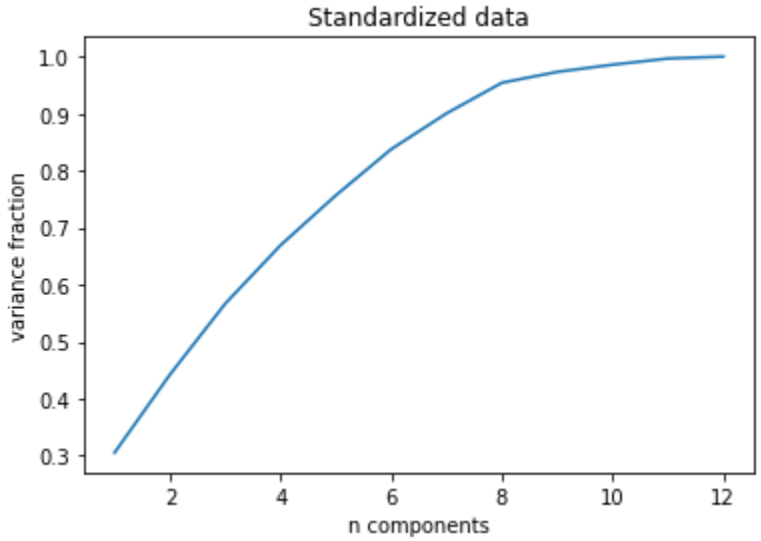
Part 8:

As previously noted, the performance of the Random Forest model exhibited notable differences before and after PCA, whereas the SVM and KNN models

demonstrated some variability based on the application of standardization and PCA. Detailed results and explanations can be found in the preceding subsection.
To determine the optimal number of n_components, PCA was performed using n_components=the original number of components in the dataset. The percentage of variance accounted for by each component was then calculated and plotted for both standardized and non-standardized data, as depicted in the below curves:

Standardized:

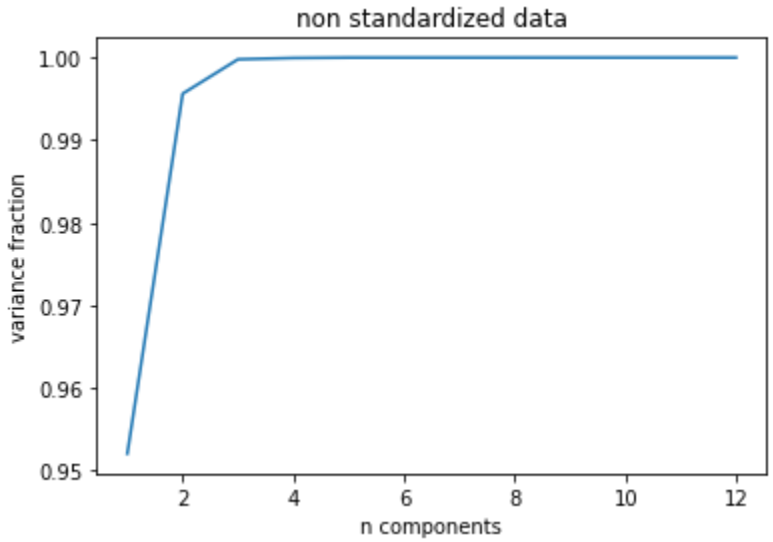
```
variance fraction for n_components= 1 is 0.30523955819054793
variance fraction for n_components= 1 is 0.30523955819054793
variance fraction for n_components= 2 is 0.44214042004725834
variance fraction for n_components= 3 is 0.5666395066289425
variance fraction for n_components= 4 is 0.6691311779670623
variance fraction for n_components= 5 is 0.7567060676990104
variance fraction for n_components= 6 is 0.8377018860694614
variance fraction for n_components= 7 is 0.9004439451178392
variance fraction for n_components= 8 is 0.9541778589478769
variance fraction for n_components= 9 is 0.9731919372751564
variance fraction for n_components= 10 is 0.9858201429605171
variance fraction for n_components= 11 is 0.9966680095519989
variance fraction for n_components= 12 is 1.0
```



Non-Standardized:

```
variance fraction for n_components= 1 is 0.9520618683752533
```

```
variance fraction for n_components= 1 is 0.9520618683752533
variance fraction for n_components= 2 is 0.9956303380948849
variance fraction for n_components= 3 is 0.9997933570852808
variance fraction for n_components= 4 is 0.9999537652179782
variance fraction for n_components= 5 is 0.9999988184708178
variance fraction for n_components= 6 is 0.9999994611241584
variance fraction for n_components= 7 is 0.9999996800763475
variance fraction for n_components= 8 is 0.9999998648860298
variance fraction for n_components= 9 is 0.9999999197750351
variance fraction for n_components= 10 is 0.999999958528975
variance fraction for n_components= 11 is 0.9999999904492084
variance fraction for n_components= 12 is 1.0
```



Bonus:

The assumption made by naiive Bayes is that the features are independent of one another, which results in the covariance matrix being a diagonal matrix.
So we make another class for PCA with covariance matrix as diagonal matrix.
Everything is same as the previous PCA class.
Now we do part 6 with the new principal components:

For standardized data:

We train it with n_components=10.
We get below scores for different classification models:

```
Cross Validation Score for SVM with standardized data with pca is
0.8166168782673637
f1 score for standardized data after pca of svm model: 0.833663497438156
```

```
accuracy_score for standardized data after pca of svm model:
0.8706563706563707
recall_score for standardized data after pca of svm model:
0.8706563706563707


Cross Validation Score for rondom forest with standardized data with pca
is 0.9343539955190441
f1 score for standardized data after pca of random_forest_model:
0.9941989417482934
accuracy_score score for standardized data after pca of
random_forest_model: 0.9942084942084942
recall_score for standardized data after pca of random_forest_model:
0.9942084942084942


Cross Validation Score for knn with standardized data with pca is
0.8435959671396563
f1 score for standardized data after pca of KNN_model: 0.8998539243270564
accuracy_score score for standardized data after pca of KNN_model:
0.9015444015444015
recall_score for standardized data after pca of KNN_model:
0.9015444015444015
```

For non-standardized data:

We train it with n_components=2.

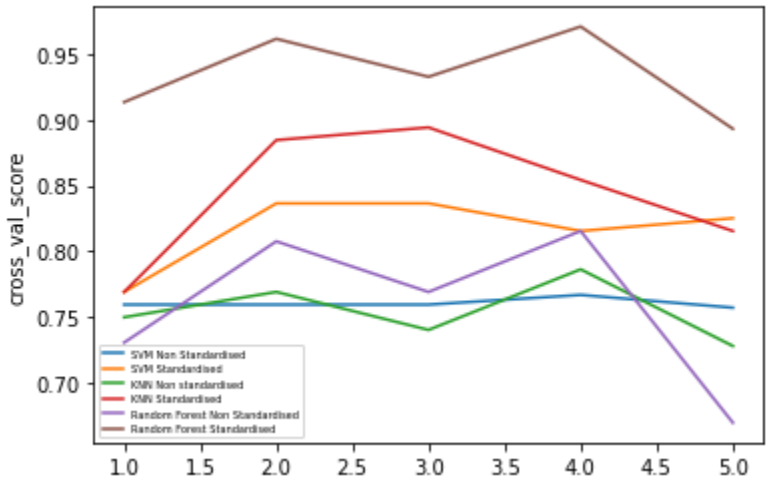
We get below scores for different classification models:

```
Cross Validation Score for SVM with non standardized data with pca is
0.7606235997012696
f1 score for non standardized data after pca of svm model:
0.657200433516223
accuracy_score score for non standardized data after pca of svm model:
0.7606177606177607
recall_score for non standardized data after pca of svm model:
0.7606177606177607


Cross Validation Score for Random Forest with non standardized data with
pca is 0.7586258401792383
f1 score for non standardized data after pca of Random Forest model:
0.8076783602916008
accuracy_score score for non standardized data after pca of Random Forest
model: 0.8359073359073359
recall_score for non standardized data after pca of Random Forest model:
0.8359073359073359
```

```
Cross Validation Score for KNN with non standardized data with pca is
0.7548356982823002
f1 score for non standardized data after pca of KNN_model:
0.7149365847783353
accuracy_score score for non standardized data after pca of KNN_model:
0.752895752895753
recall_score for non standardized data after pca of KNN_model:
0.752895752895753
```

The cross validation plot observed is as follows:



Consequently, it can be concluded that assuming independence between features, PCA resulted in significantly improved outcomes for standardized datasets compared to those obtained prior to the assumption. However, in the case of non-standardized data, the accuracy remained nearly unchanged.

Question 02:

Dataset: Wine Classification Dataset

Part 1:

We import the dataset and check if any NaN valuesare present in it. Now, we scale the attributes using standardScalar() and then split data into training and testing.

Next, in order to implement LDA from scratch. We make a class LDA_scratch. The attributes present in the class are as follows:

1. x_t: it contains the dataset we need to fit.

2. y_t: it contains the output attribute.
3. eigenval: It contains the eigenvalues of covariance matrix.
4. eigenvec: It contains the eigenvectors corresponding to the eigenvalues of the covariance matrix.
5. lin_dis: It contains the linear discrimanants.
6. per_variance: It contains the percentage of variance that needs to be conserved and is passed by the user.

The methods of the class are as follows:

1. within_between_class_mat: This method returns the within class matrix i.e Sw and between class matrix i.e Sb by calculating them by utilizing the data given by the user.
2. fit: this takes Sw and Sb from the above method and then obtains eigenvalues and eigenvectors. The eigenvectors and eigenvalues are sorted based on the eigenvalues, and the eigenvectors are added to the linear_discriminants attribute until the cumulative variance matches the variance provided by the user. This then reuters linear discriminants.
3. Transform: gives the transformed dataset by applying linear discriminants.

The other methods in this class are to use LDA as a classifier. For this we implement naive bayes to calculate probability of data points for different classes. These methods are as follows:

1. prob: it gives a dataframe after applying the probability formula to calculate independent likelyhood probabilities.
2. post: calculates posterior probabilities.
3. cross_val_score: gives cross validation score by using actual and predicted values.
4. y_pred: predicts class of the datapoints.
5. confusion_mat: gives the confusion matrix of datapoints.
6. roc_val: gives the true positive rate and false positive rate to get roc curve.

Part 2:

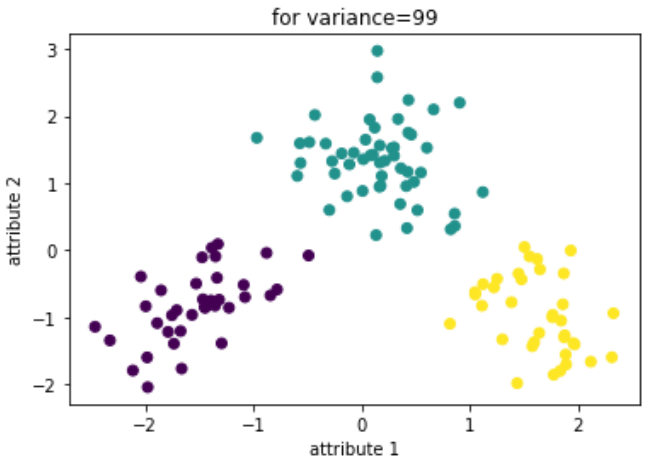
We train LDA model and get linear discriminant from which we get the most optimal attributes using argsort.

```
[[11  8  4  5 10  7  1  0  3  6  9  2 12]
 [ 4  1  7  8  2 10  0  5  3  9 11 12  6]]
```

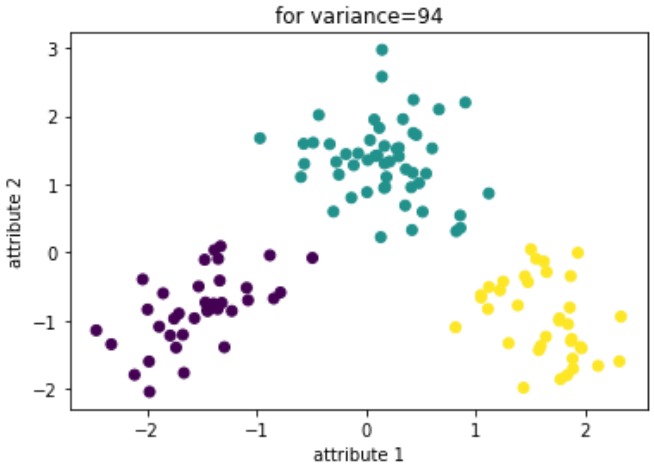
Therefore, we observe that features with index=11,8,4,5 and 10 i.e. 'OD280/OD315 of diluted wines' , 'Proanthocyanins', 'Magnesium', 'Total phenols' and 'Hue' are the top 5 impactful features.

Now,we alter the variance and conduct training of our LDA model across various variance values. Following this, an argsort() function is executed on the linear_discriminants, enabling us to determine the index of the features, corresponding to the most prominent linear discriminant. This index signifies the direction with the most substantial bending of the eigenvectors/linear discriminants. In conclusion, we obtain the ensuing results:

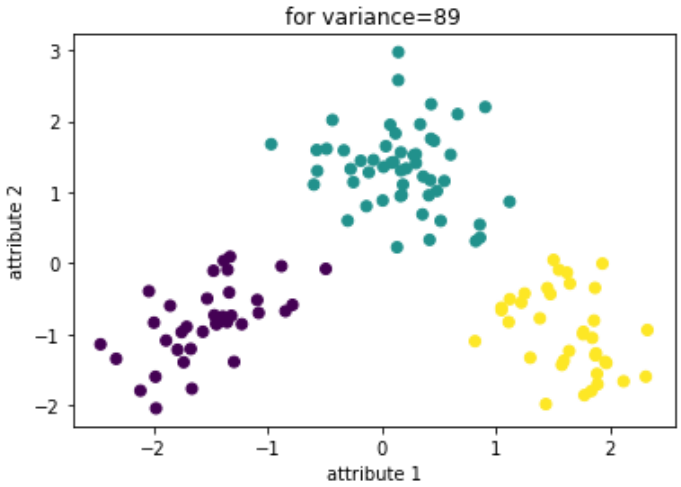
```
For variance_percentage= 99 %
The most impactful features indices:
[[12  2  9  0  1 11  8  4  5 10  7  3  6]
 [ 6 12 11  0 10  2  7  4  1  8  5  3  9]]
The top 5 most impactful features are: ['Proline ', 'Ash', 'Color
intensity', 'Alcohol', 'Malic acid', 'OD280/OD315 of diluted wines',
'Proanthocyanins', 'Magnesium', 'Total phenols', 'Hue', 'Nonflavanoid
phenols', 'Alcalinity of ash', 'Flavanoids']
```




```
For variance_percentage= 94 %
The most impactful features indices:
[[12  2  9  0  1 11  8  4  5 10  7  3  6]
 [ 6 12 11  0 10  2  7  4  1  8  5  3  9]]
The top 5 most impactful features are: ['Proline ', 'Ash', 'Color
intensity', 'Alcohol', 'Malic acid', 'OD280/OD315 of diluted wines',
'Proanthocyanins', 'Magnesium', 'Total phenols', 'Hue', 'Nonflavanoid
phenols', 'Alcalinity of ash', 'Flavanoids']
```



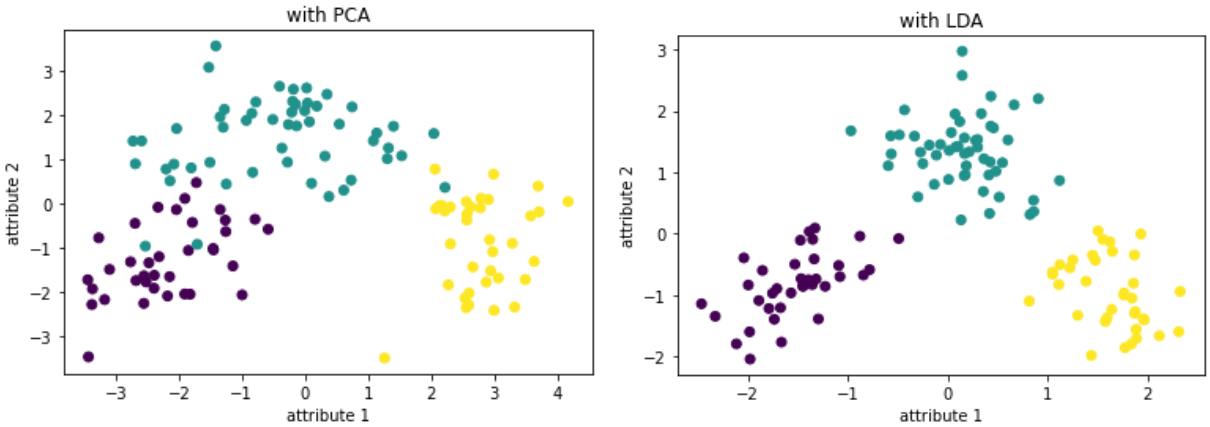

```
For variance_percentage= 89 %
The most impactful features indices:
[[12  2  9  0  1 11  8  4  5 10  7  3  6]
 [ 6 12 11  0 10  2  7  4  1  8  5  3  9]]
The top 5 most impactful features are: ['Proline ', 'Ash', 'Color
intensity', 'Alcohol', 'Malic acid', 'OD280/OD315 of diluted wines',
'Proanthocyanins', 'Magnesium', 'Total phenols', 'Hue', 'Nonflavanoid
phenols', 'Alcalinity of ash', 'Flavanoids']
```



It is evident from our analysis that data points with distinct classes are appropriately grouped together, and for all variance values, we can achieve a clear separation between them.

Part 3:

In order to compare PCA and LDA, we plot scatterplots. From these we observe that for PCA, the data points belonging to the 3 classes are overlapping. However, in the case of LDA, there are no such overlaps, and data points from different classes are significantly separated.



Now, we compare them by using 2 classification models:

1. KNN:

   For PCA:

   ```
   f1 score after PCA of KNN_model: 0.9594622382781447
   accuracy_score score after PCA of KNN_model: 0.9596774193548387
   recall_score after PCA of KNN_model: 0.9596774193548387
   ```

   For LDA:

   ```
   f1 score after LDA of KNN_model: 0.7758452670003138
   accuracy_score score after LDA of KNN_model: 0.782258064516129
   recall_score after LDA of KNN_model: 0.782258064516129
   ```

   It gives good result for pca but not lda.

2. Decision_trees

   For PCA:

   ```
   f1 score after PCA of DecisionTree_model: 1.0
   accuracy_score score after PCA of DecisionTree_model: 1.0
   recall_score after PCA of DecisionTree_model: 1.0
   ```

   For LDA:

   ```
   f1 score after LDA of DecisionTree_model: 1.0
   accuracy_score score after LDA of DecisionTree_model: 1.0
   ```

```
recall_score after LDA of DecisionTree_model: 1.0
```

This model gives good result for both pca and lda.

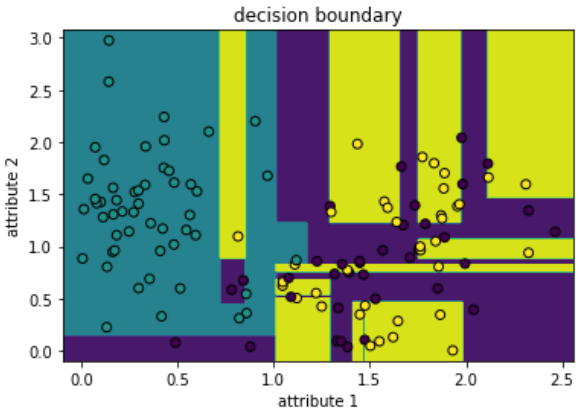
Part 4:

Table to note down the accuracies in case of each classifier and the corresponding reduction technique.

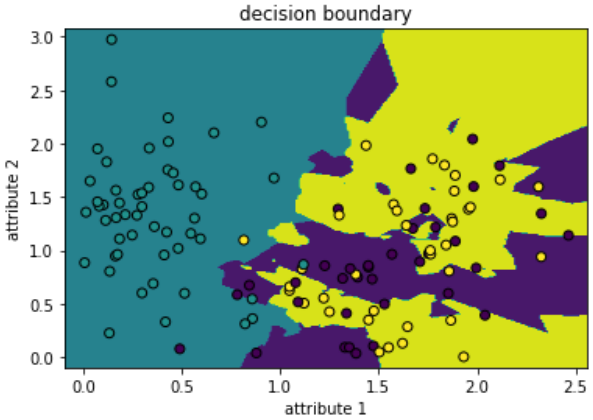
Classifier	**Decision tree**	**KNN**
PCA	f1_score=1	f1_SCORE=0.9594622382781447
	accuracy =1	accuracy =0.9596774193548387
	recall_score=1	recall_score=0.9596774193548387
LDA	f1_SCORE=1	f1_SCORE=0.7758452670003138
	accuracy =1	accuracy =0.782258064516129
	recall_score=1	recall_score=0.782258064516129

Decision boundaries:

For decision tree:



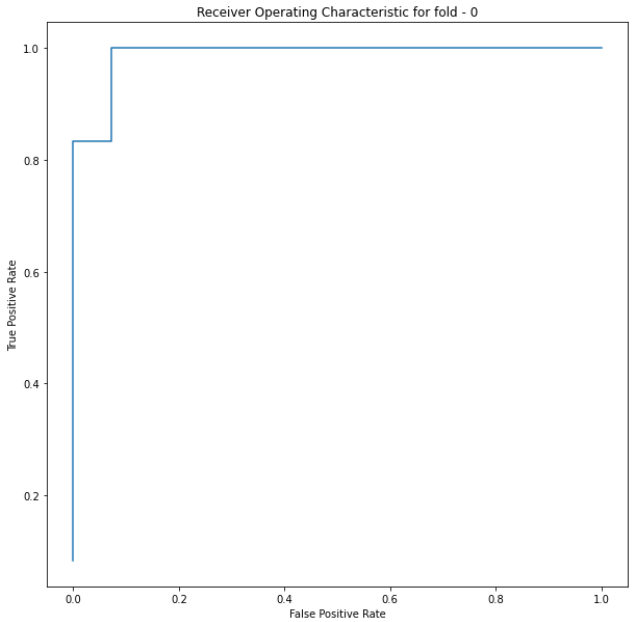
For KNN:



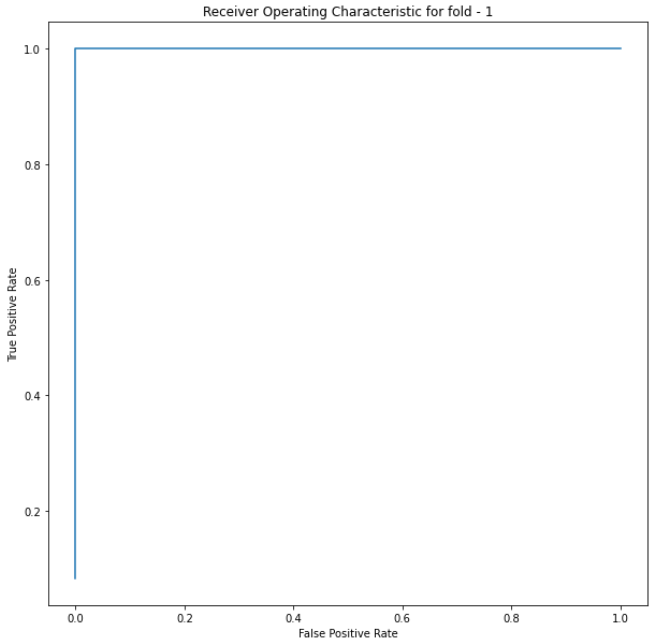
From here we can observe both are doing a great job in classifying classes of the datapoints.

Part 5:

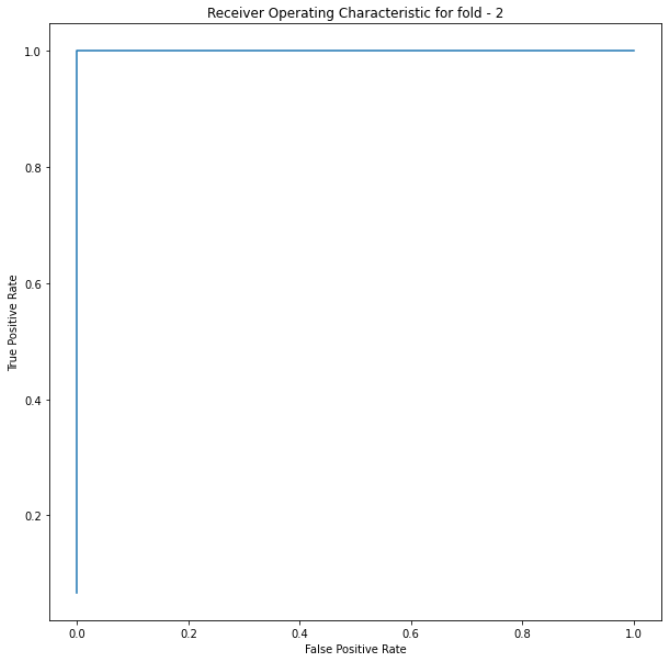
To implement k-fold from scratch, we divide our dataset into 5 parts and use 4 of them for training while using the remaining 1 for testing. This process is repeated 5 times and the score obtained from running LDA from scratch is printed along with the roc curve which gives us tpr vs fpr graph. We also compute the auc by using np.trapz.
We get the following results:



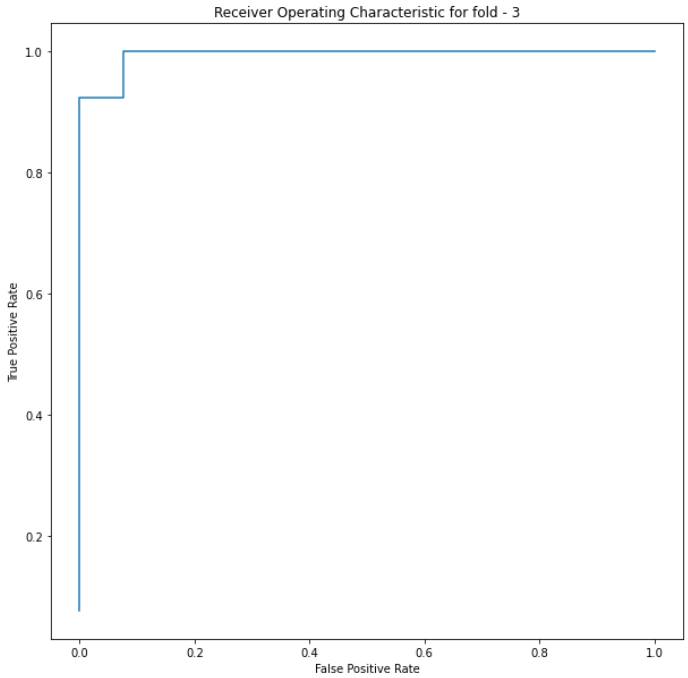

```
cross_val_score for fold 1 is:  0.8461538461538461
AUC_score for fold 1 is:  0.9880952380952381
```



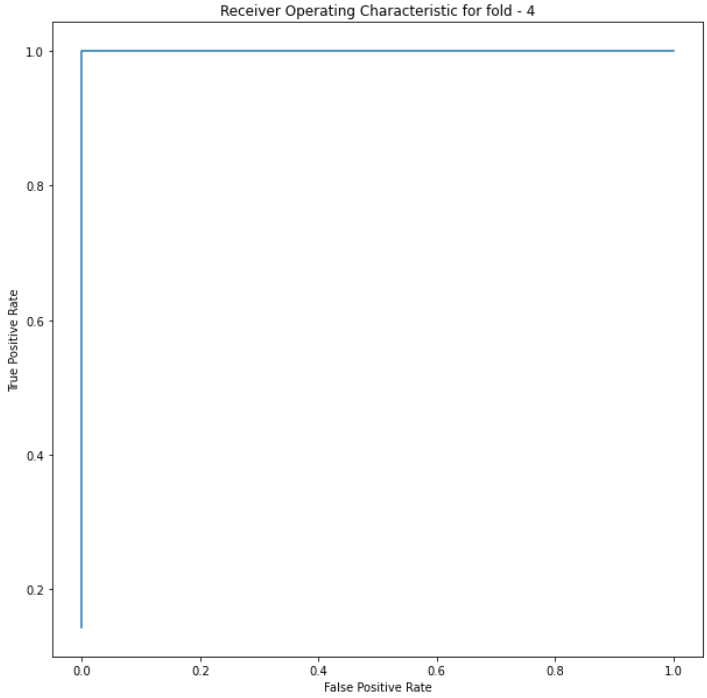

```
cross_val_score for fold 2 is:  0.8846153846153846
AUC_score for fold 2 is:  1.0
```




```
cross_val_score for fold 3 is:  1.0
AUC_score for fold 3 is:  1.0
```




```
cross_val_score for fold 4 is:  0.9615384615384616
AUC_score for fold 4 is:  0.9940828402366865
```




```
cross_val_score for fold 5 is:  0.9615384615384616
AUC_score for fold 5 is:  1.0
```