Pattern Recognition & Machine Learning
Lab-7 Assignment

Name: Tanish Pagaria
Roll No.: B21AI040

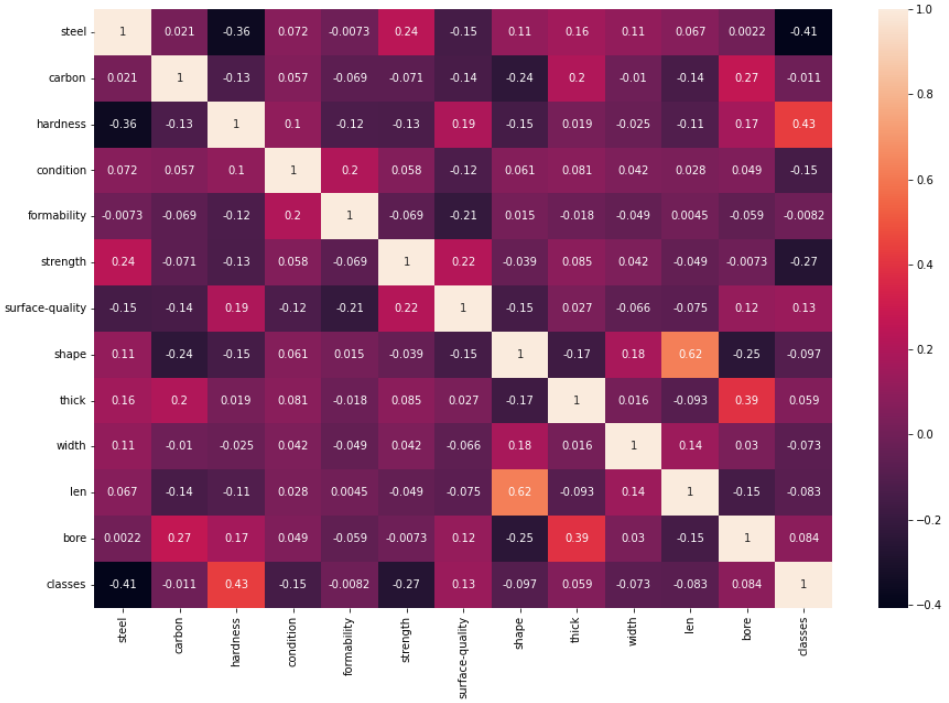
Question-1

Loading the data

The Annealing Dataset was divided into two files, ‘anneal.data’ and ‘anneal.test’. These files were concatenated to make the complete dataset. The files were first converted into .csv format. The dataset had many missing values, represented by "?" These were replaced manually by "NaN" before loading the CSV file in the notebook.

Performing exploratory analysis and Preprocessing

The dataset had many columns, with most of the values being "NaN". The columns with more than half the values as "NaN" were dropped. The rest of the columns were imputed with the mode of the corresponding column.
After imputation of the dataframe, it was observed that the column "product-type" had the same value for each data entry. Hence, it was also dropped.



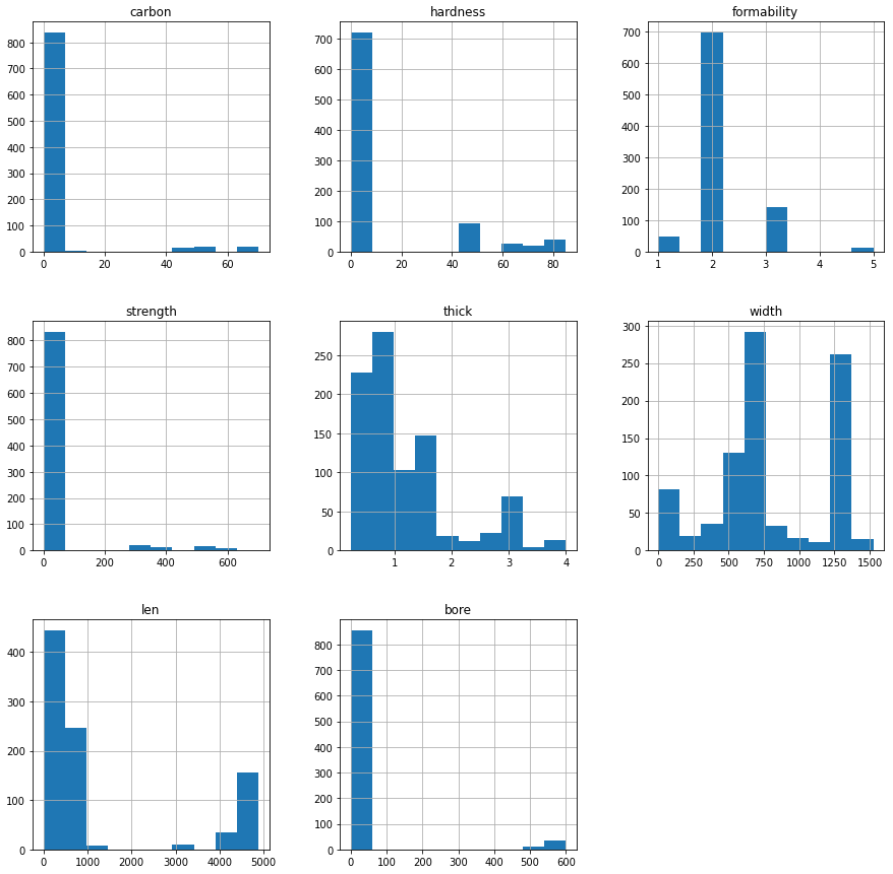
Correlation heatmap of the data

Also, in the dataset, the classes were [1, 2, 3, 5, and "U"]. We replaced the "U" with the number "6". The categorical features were encoded.

	steel	carbon	hardness	condition	formability	strength	surface-quality	shape	thick	width	len	bore	classes
0	0	8	0	1	2.0	0	3	0	0.700	610.0	0	0	2
1	3	0	0	1	2.0	0	1	0	3.200	610.0	0	0	2
2	3	0	0	1	2.0	0	1	1	0.700	1300.0	762	0	2
3	0	0	60	1	2.0	0	3	0	2.801	385.1	0	0	2
4	0	0	60	1	2.0	0	3	1	0.801	255.0	269	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	3	0	0	1	3.0	0	1	1	1.599	610.0	762	0	1
96	3	0	0	1	3.0	0	1	1	1.601	830.0	880	0	1
97	5	0	0	1	2.0	0	1	1	1.599	150.0	762	0	1
98	0	0	85	1	2.0	0	3	0	0.400	20.0	0	0	4
99	0	0	85	1	2.0	0	3	0	4.000	610.0	0	500	4

898 rows × 13 columns

Final dataframe after the preprocessing



Histogram plot of the preprocessed dataframe

Train-test split and Standardization

Two copies of the dataset were made, and feature standardization was performed on one of the copies using the `sklearn.preprocessing.StandardScaler` function, while the other was left untouched. Each copy was split into a [65:35] ratio using the `sklearn.model_selection.train_test_split` function.

The standardized and the unstandardized data were used separately for all the subsequent tasks, and observations were made comparatively.

Training classification models and Cross-validation

Two classification models were chosen along with the SVM classifier from the `sklearn` library. These classifiers were trained on the standardized and non-standardized data sets separately, and 5-fold cross-validation plots were plotted.
The classifiers chosen included: RandomForestClassifier and BaggingClassifier.

Reasoning for choosing the classifiers

Random Forest and Bagging classifiers are powerful and flexible machine learning models that can handle a wide range of classification problems. They are robust against overfitting, can handle high-dimensional data and missing values.

Cross-validation scores

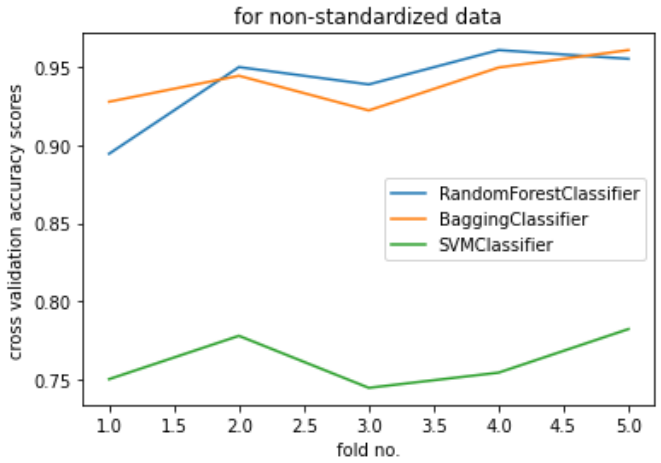
For non-standardized data:-

```
5-fold-cross-validation scores for RandomForestClassifier: [0.89444444 0.95       0.93888889 0.96089385 0.95530726]
5-fold-cross-validation scores for BaggingClassifier: [0.92777778 0.94444444 0.92222222 0.94972067 0.96089385]
5-fold-cross-validation scores for SVMClassifier: [0.75       0.77777778 0.74444444 0.75418994 0.78212291]
```

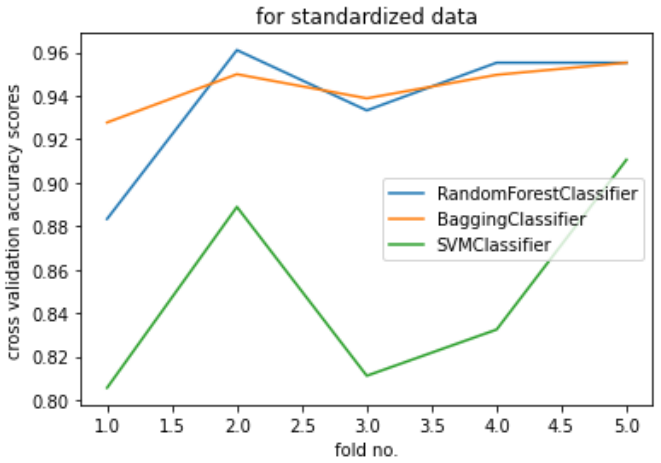
For standardized data:-

```
5-fold-cross-validation scores for RandomForestClassifier: [0.88333333 0.96111111 0.93333333 0.95530726 0.95530726]
5-fold-cross-validation scores for BaggingClassifier: [0.92777778 0.95       0.93888889 0.94972067 0.95530726]
5-fold-cross-validation scores for SVMClassifier: [0.80555556 0.88888889 0.81111111 0.83240223 0.91061453]
```

Cross-validation plots



(for non-standardized data)



(for standardized data)

Principal Component Analysis implementation from scratch

We defined a class called PCA. The constructor for the class took n_components as the input (let the value be n), which defined the number of principal components (features or dimensions) required in the transformed dataset or dataframe.

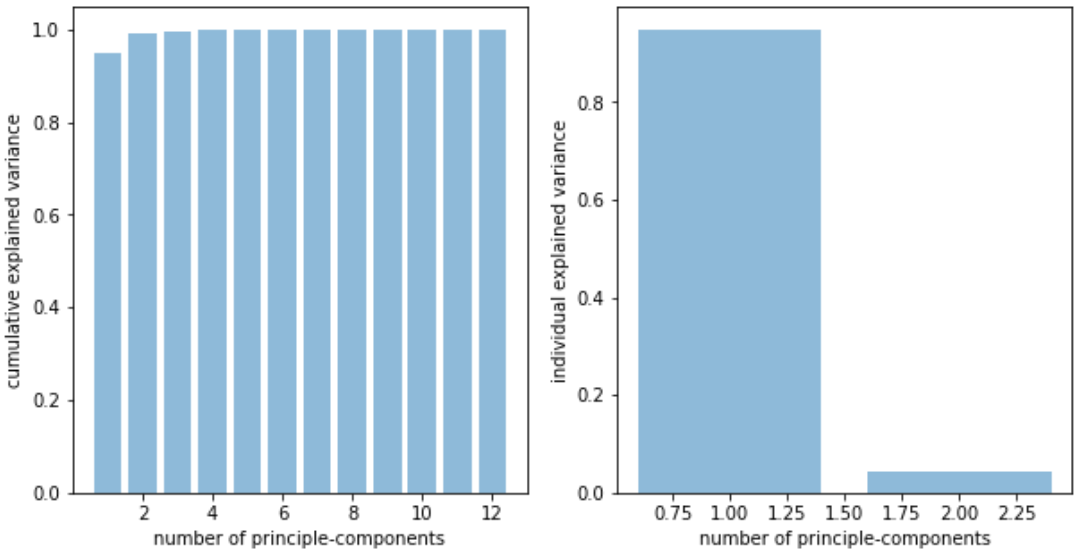
The `fit` method took the training data as input and trained the model using the helper methods. Helper methods were defined in order to calculate the mean and the standard deviation of the feature dataset. A method was defined to calculate the covariance matrix of the dataset from scratch.
Another method was defined to calculate the eigenvectors and the eigenvalues of the covariance matrix. It was performed using the `numpy.linalg.eig` function. The eigenvalues were sorted in descending order, and the corresponding eigenvectors were also sorted accordingly.
The eigen-vector matrix was sliced for the first n components. The feature vectors or the dataset that were given to the `fit` method could be then reduced to the chosen principal components by calling the `transform` method. This method would also be used to reduce the dimensions of the test dataset. The

transform method would return the transformed feature vector by performing the dot product of the feature vector with the sliced eigen-vector matrix.

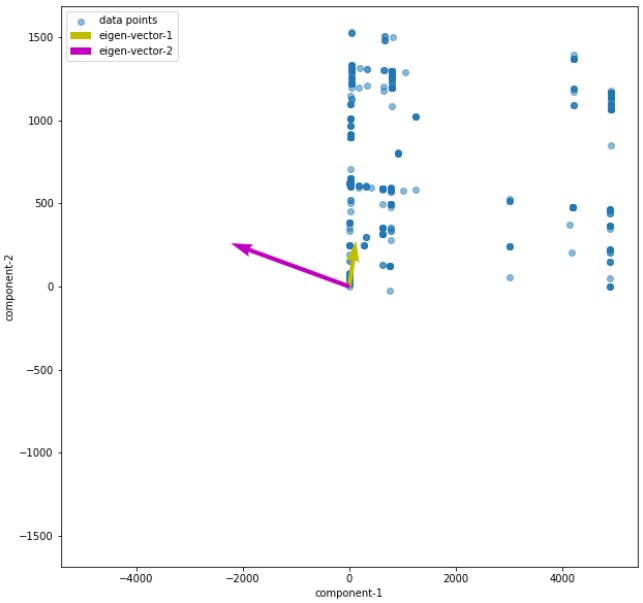
A method was defined to show the change in variance with the increase in the number of components. It showed the cumulative and individual variances in two subplots. The variance was calculated by dividing the cumulative sum of eigenvalues by the sum of all the eigenvalues.

Another method was defined for creating a scatter plot in order to show the direction of the eigenvectors along with the data points (the two best features from the reduced dataset).
The plots are shown below for the non-standardized and standardized datasets.

Non-standardized data

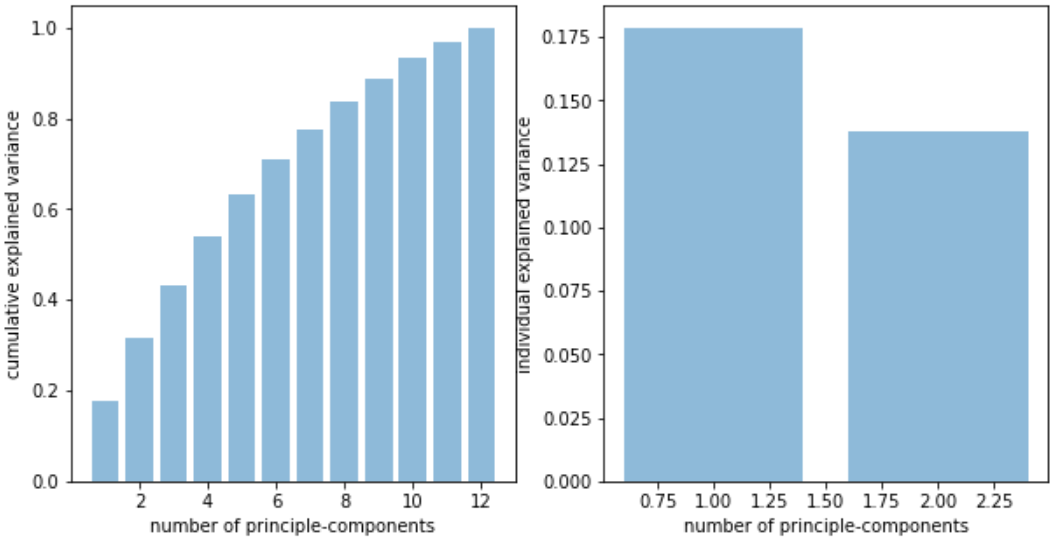


Change in variance with change in the number of components
(also used in identifying the optimal number of components)

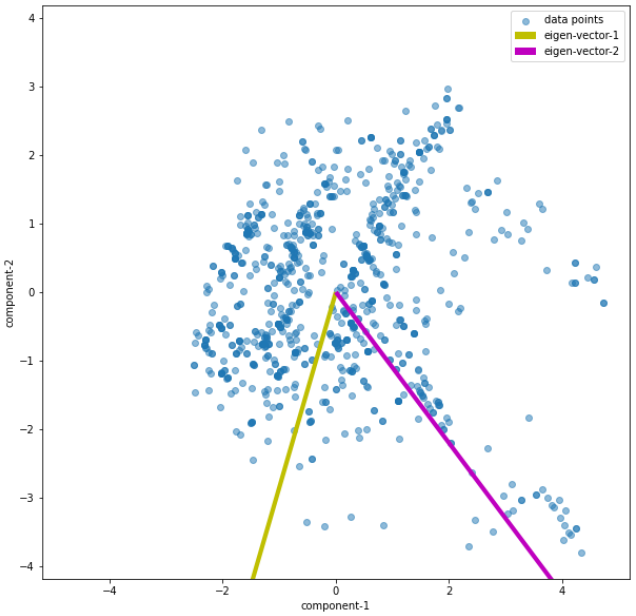


Scatter plot to show the direction of the eigenvectors along with the data points

Standardized data



Change in variance with change in the number of components
(also used in identifying the optimal number of components)



Scatter plot to show the direction of the eigenvectors along with the data points

Performing classification on the reduced datasets *(before and after PCA)*
The same classification models as used previously: RandomForestClassifier, BaggingClassifier, and SVMClassifier—were used to train and test the reduced dataset for both standardized and non-standardized cases.

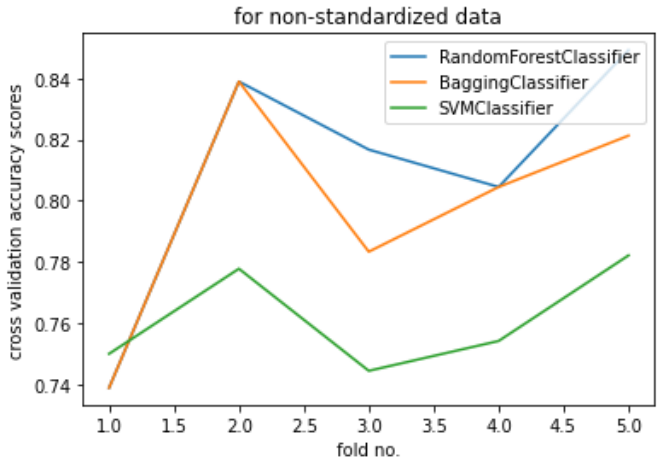
Cross-validation scores
For non-standardized case:-

```
5-fold-cross-validation scores for RandomForestClassifier: [0.73888889 0.83888889 0.81666667 0.80446927 0.84916201]
5-fold-cross-validation scores for BaggingClassifier: [0.73888889 0.83888889 0.78333333 0.80446927 0.82122905]
5-fold-cross-validation scores for SVMClassifier: [0.75       0.77777778 0.74444444 0.75418994 0.78212291]
```

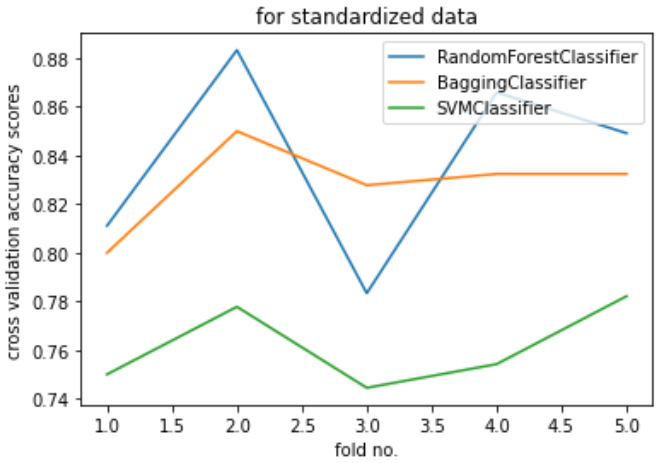
For standardized case:-

```
5-fold-cross-validation scores for RandomForestClassifier: [0.81111111 0.88333333 0.78333333 0.86592179 0.84916201]
5-fold-cross-validation scores for BaggingClassifier: [0.8        0.85       0.82777778 0.83240223 0.83240223]
5-fold-cross-validation scores for SVMClassifier: [0.75       0.77777778 0.74444444 0.75418994 0.78212291]
```

Cross-validation Plots



(for non-standardized)



(for standardized data)

Testing of the classification models and evaluation metrics

The evaluation metrics used in showing the test results for the classification tasks include `accuracy_score` and `classification_report` (which incorporate precision, recall, and f1-score), which are built-in functions in the `sklearn.metrics` module.

Non-standardized data

```
for RandomForestClassifier:-
Accuracy score: 0.9396825396825397
Classification-report:-
              precision    recall  f1-score   support

     class-0       1.00      1.00      1.00         2
     class-1       0.69      0.77      0.73        26
     class-2       0.98      0.95      0.96       251
     class-3       0.83      1.00      0.90        19
     class-4       1.00      1.00      1.00        17

    accuracy                           0.94       315
   macro avg       0.90      0.94      0.92       315
weighted avg       0.94      0.94      0.94       315

------------------------------------------------------------
for BaggingClassifier:-
Accuracy score: 0.926984126984127
Classification-report:-
              precision    recall  f1-score   support

     class-0       1.00      1.00      1.00         2
     class-1       0.59      0.88      0.71        26
     class-2       0.99      0.92      0.95       251
     class-3       0.83      1.00      0.90        19
     class-4       1.00      1.00      1.00        17

    accuracy                           0.93       315
   macro avg       0.88      0.96      0.91       315
weighted avg       0.95      0.93      0.93       315

------------------------------------------------------------
for SVMClassifier:-
Accuracy score: 0.7968253968253968
Classification-report:-
              precision    recall  f1-score   support

     class-0       0.00      0.00      0.00         2
     class-1       0.00      0.00      0.00        26
     class-2       0.80      1.00      0.89       251
     class-3       0.00      0.00      0.00        19
     class-4       0.00      0.00      0.00        17

    accuracy                           0.80       315
   macro avg       0.16      0.20      0.18       315
weighted avg       0.63      0.80      0.71       315
```

(before PCA)

```
for RandomForestClassifier:-
Accuracy score: 0.8253968253968254
Classification-report:-
              precision    recall  f1-score   support

     class-0       0.50      0.50      0.50         2
     class-1       0.49      0.65      0.56        26
     class-2       0.90      0.88      0.89       251
     class-3       0.63      0.63      0.63        19
     class-4       0.67      0.47      0.55        17

    accuracy                           0.83       315
   macro avg       0.64      0.63      0.63       315
weighted avg       0.83      0.83      0.83       315

------------------------------------------------------------
for BaggingClassifier:-
Accuracy score: 0.7841269841269841
Classification-report:-
              precision    recall  f1-score   support

     class-0       0.33      1.00      0.50         2
     class-1       0.38      0.62      0.47        26
     class-2       0.89      0.84      0.86       251
     class-3       0.60      0.63      0.62        19
     class-4       0.67      0.35      0.46        17

    accuracy                           0.78       315
   macro avg       0.57      0.69      0.58       315
weighted avg       0.81      0.78      0.79       315

------------------------------------------------------------
for SVMClassifier:-
Accuracy score: 0.7968253968253968
Classification-report:-
              precision    recall  f1-score   support

     class-0       0.00      0.00      0.00         2
     class-1       0.00      0.00      0.00        26
     class-2       0.80      1.00      0.89       251
     class-3       0.00      0.00      0.00        19
     class-4       0.00      0.00      0.00        17

    accuracy                           0.80       315
   macro avg       0.16      0.20      0.18       315
weighted avg       0.63      0.80      0.71       315
```

(after PCA)

Observations

The accuracy scores get worse in the cases of RandomForestClassifier and BaggingClassifier. However, in the case of the SVMClassifier, the accuracy score is the same. The same is true for the other evaluation metrics as well, as visible in the classification reports above.

Standardized data

```
for RandomForestClassifier:-
Accuracy score: 0.9396825396825397
Classification-report:-
              precision    recall  f1-score   support

     class-0       1.00      1.00      1.00         2
     class-1       0.69      0.77      0.73        26
     class-2       0.98      0.95      0.96       251
     class-3       0.83      1.00      0.90        19
     class-4       1.00      1.00      1.00        17

    accuracy                           0.94       315
   macro avg       0.90      0.94      0.92       315
weighted avg       0.94      0.94      0.94       315

------------------------------------------------------------
for BaggingClassifier:-
Accuracy score: 0.9206349206349206
Classification-report:-
              precision    recall  f1-score   support

     class-0       1.00      1.00      1.00         2
     class-1       0.57      0.77      0.66        26
     class-2       0.97      0.92      0.95       251
     class-3       0.83      1.00      0.90        19
     class-4       1.00      1.00      1.00        17

    accuracy                           0.92       315
   macro avg       0.87      0.94      0.90       315
weighted avg       0.93      0.92      0.93       315

------------------------------------------------------------
for SVMClassifier:-
Accuracy score: 0.8507936507936508
Classification-report:-
              precision    recall  f1-score   support

     class-0       0.33      0.50      0.40         2
     class-1       0.67      0.38      0.49        26
     class-2       0.89      0.92      0.91       251
     class-3       0.58      0.79      0.67        19
     class-4       0.91      0.59      0.71        17

    accuracy                           0.85       315
   macro avg       0.68      0.64      0.64       315
weighted avg       0.85      0.85      0.85       315
```

(before PCA)

```
for RandomForestClassifier:-
Accuracy score: 0.8571428571428571
Classification-report:-
              precision    recall  f1-score   support

     class-0       1.00      0.50      0.67         2
     class-1       0.64      0.62      0.63        26
     class-2       0.89      0.93      0.91       251
     class-3       0.62      0.68      0.65        19
     class-4       1.00      0.35      0.52        17

    accuracy                           0.86       315
   macro avg       0.83      0.62      0.68       315
weighted avg       0.86      0.86      0.85       315

------------------------------------------------------------
for BaggingClassifier:-
Accuracy score: 0.8476190476190476
Classification-report:-
              precision    recall  f1-score   support

     class-0       1.00      0.50      0.67         2
     class-1       0.59      0.85      0.70        26
     class-2       0.91      0.90      0.91       251
     class-3       0.57      0.68      0.62        19
     class-4       0.83      0.29      0.43        17

    accuracy                           0.85       315
   macro avg       0.78      0.64      0.66       315
weighted avg       0.86      0.85      0.84       315

------------------------------------------------------------
for SVMClassifier:-
Accuracy score: 0.8031746031746032
Classification-report:-
              precision    recall  f1-score   support

     class-0       0.00      0.00      0.00         2
     class-1       0.80      0.15      0.26        26
     class-2       0.81      0.98      0.89       251
     class-3       0.40      0.11      0.17        19
     class-4       0.00      0.00      0.00        17

    accuracy                           0.80       315
   macro avg       0.40      0.25      0.26       315
weighted avg       0.74      0.80      0.74       315
```

(after PCA)

Observations

After standardization, the loss in accuracy due to PCA was smaller in comparison to the non-standardized data. Hence, it can be concluded that standardization was beneficial for the dataset in this case before applying PCA.

It can also be observed that the accuracy score for SVM in case of standardized dataset was higher than that in the non-standardized dataset while that of other classifiers was same (before applying PCA)

Finding optimal number of principal components

Using a previously implemented helper method, we get a graph based on the cumulative sum of eigenvalues divided by the sum of all eigenvalues. The graph can be used to analyze and find out the optimal value of the number of components when the change in variance becomes negligible.

(plot shown previously)

Changes observed before and after implementing PCA

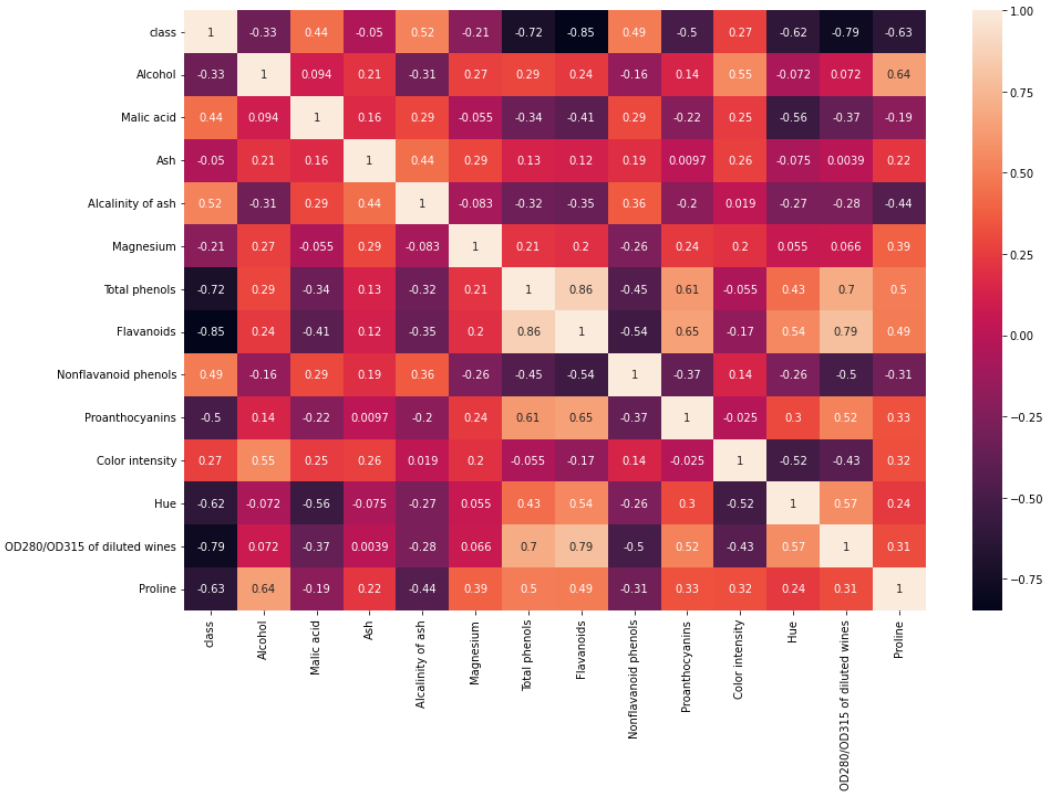
The dimensionality of the dataset was reduced after applying PCA. The data was linearly transformed into a new coordinate system where (most of) the variation in the data could be described with fewer

dimensions than the initial data. However, after implementing PCA, the accuracy of the models dropped heavily.

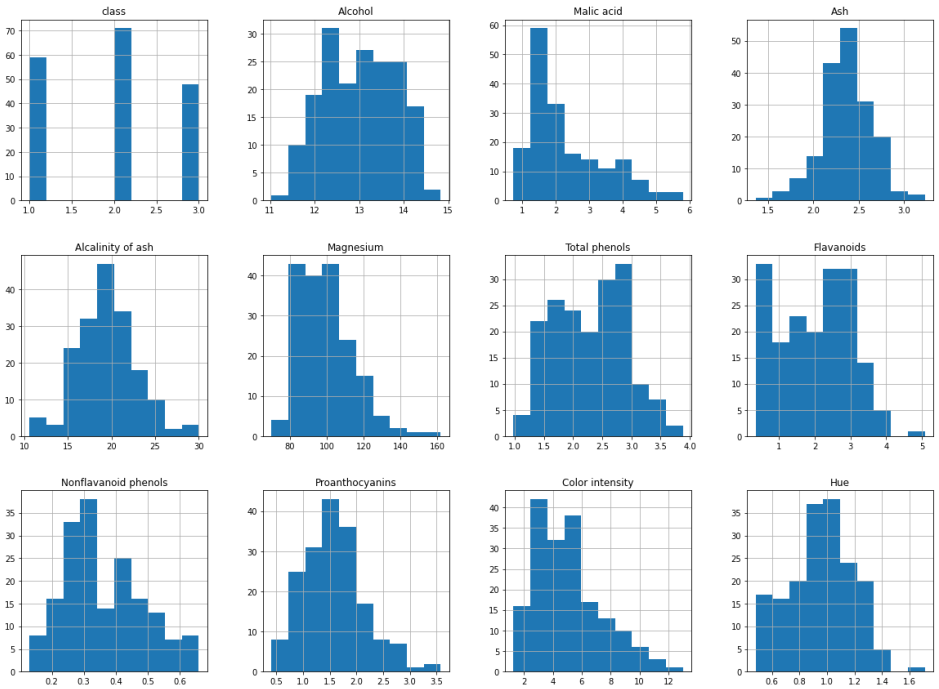
Question-2

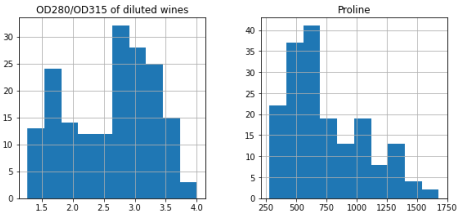
Loading the data

The Wines Classification dataset was loaded as a pandas dataframe. Upon analysis, it was observed that the dataset did not contain any NaN values or NULL entries. However, it required standardization. The data was standardized using `sklearn.preprocessing.StandardScaler`.



Correlation heatmap of the dataset





Data visualization using histogram

Implementation of Linear Discriminant Analysis from scratch

We defined a class called LDA. The constructor method took in the value of `n_components` (if the user wants to give custom input for the number of dimensions or components). The default value of `n_components` was set to be **None** (since this value would be calculated by the model itself after it was fit with the training dataset and labels).

The `fit` method was defined, which took the inputs from the training dataset and the training labels and trained the model. The helper functions were defined for the computation of the between-class and in-class scatter matrices. Function calls to these methods were made in the fit function itself.

The operations on the dataset were performed using dictionaries based on the classes of the training labels in order to calculate the class means, covariance matrices, etc. corresponding to each class.

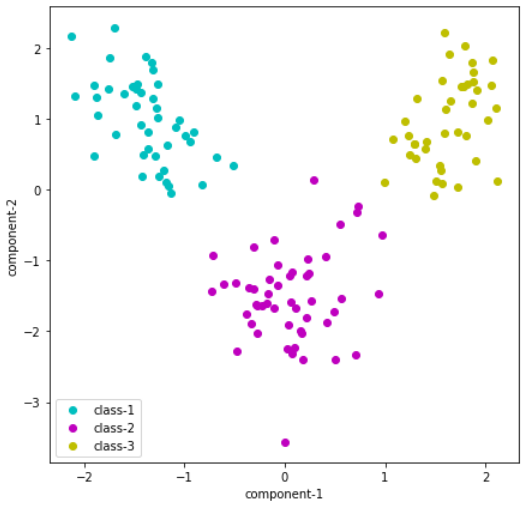
Another helper function was defined in order to automatically select the number of linear discriminants based upon the percentage of variance that needs to be conserved. A threshold was set in order to find the optimal value of the number of linear discriminants.

As was done before with PCA, a method was set up to find the ordered eigenvalues and eigenvectors that go with them. The eigenvector matrix was again sliced based on the number of linear discriminants.

The `transform` method was defined, which returned the dot product of the training, testing, or any other dataframe with the sliced eigenvector matrix in order to reduce its dimensions to the number of linear discriminants.

Visualization of the feature space of the linear discriminants with the highest impact on the classification tasks

Another helper method was defined in order to visualize the two linear discriminants with the highest impact on the classification tasks (by varying the variance). A scatter plot for the two discriminants was plotted.



Feature space visualization for features with high impact in classification tasks

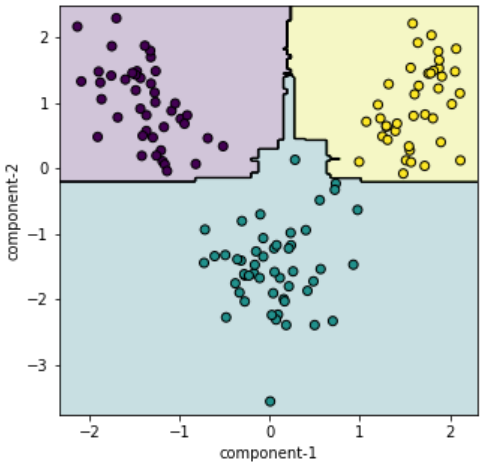
There were several other helper functions implemented in the LDA class. LDA as a classifier was based on the Bayes model. It used likelihoods and posteriori probabilities for classification. There were also functions implemented for returning prediction probabilities for each class.

Comparison between PCA and LDA
Classification techniques (RandomForestClassifier, BaggingClassifier and SVMClassifier in our case) were employed in order to compare the two techniques. The following accuracy scores were obtained during the testing:-

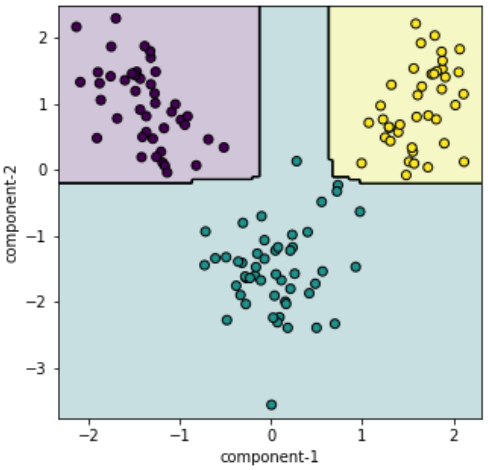
	PCA	LDA
Random Forest Classifier	0.0	0.9777777777777777
Bagging Classifier	0.06666666666666667	0.9777777777777777
SVM Classifier	0.0	1.0

From the accuracy scores, it is clearly visible that performing PCA on the dataset reduced the accuracy of the models horribly. In contrast, after performing LDA, the models achieved high accuracy scores (even 100% in the case of SVM).
Performing dimensionality reduction using PCA led to heavy information loss and poor results.

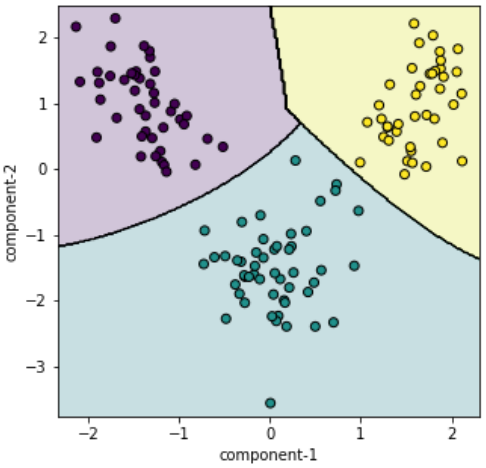
Scatter plot of the two best features and the Decision Boundary for LDA
Decision boundaries for the previously used classification techniques were plotted for the transformed dataset. The first two components had the best features based on their variance.
From the plots, it is evident that the Linear Discriminant Analysis model was able to classify the dataset very efficiently.



Decision boundary for RandomForest Classifier



Decision boundary for Bagging Classifier



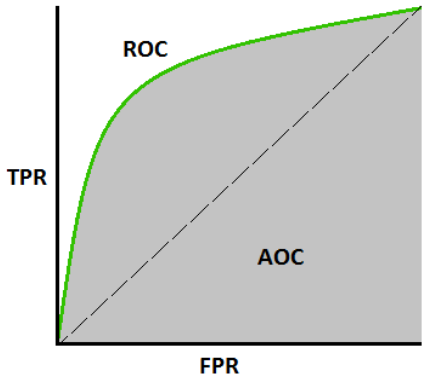
Decision boundary for SVM Classifier

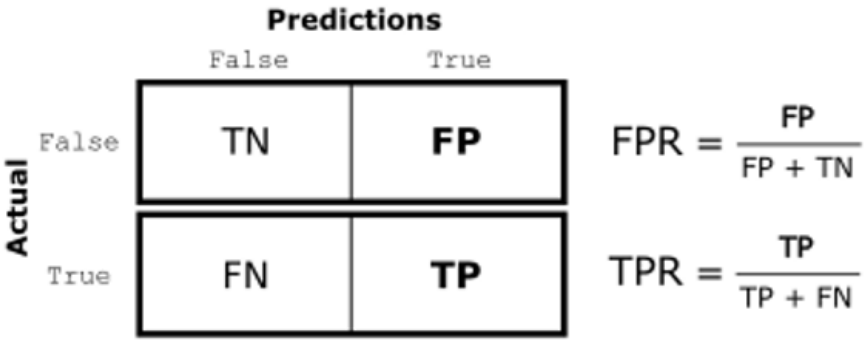
5-fold cross-validation, plotting ROC and AUC computation from scratch for LDA

Wine classification data contained more than two classes.

Since the ROC curve can only be applied to binary classification, we manipulated the data at each fold in order to make it binary. We considered one class at a time, and changed its class value to 1, and changed the class values for the rest of the classes to 0. We implemented K-fold cross-validation from scratch by performing permutations on the dataset and then splitting it into 5 parts.

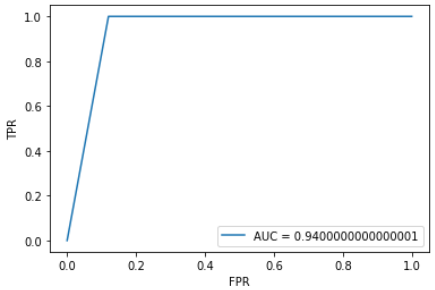
The ROC curve was implemented by considering an array of thresholds, comparing each of the thresholds with the predicted class probability for each data point, and appending the values of true-positives, true-negatives, false-positives, and false-negatives accordingly. These values were appended to arrays and used in plotting the curve.



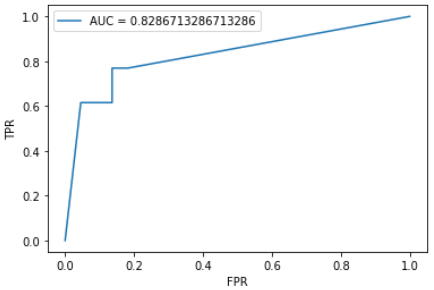


The following ROC curves were obtained for each fold corresponding to each class:-

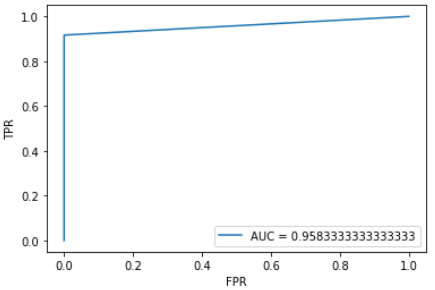
for fold-1



Class-1

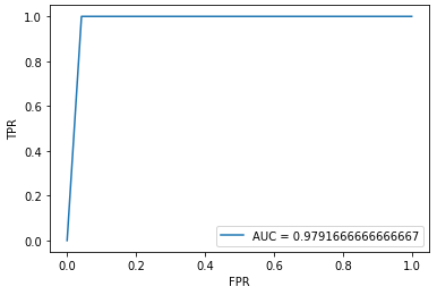


Class-2

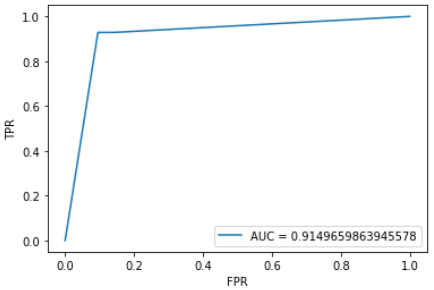


Class-3

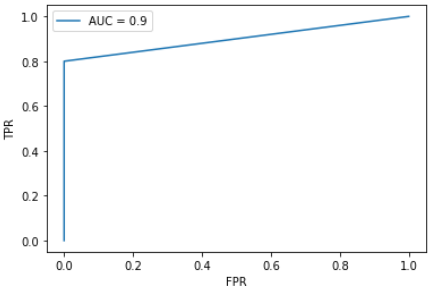
for fold-2



Class-1

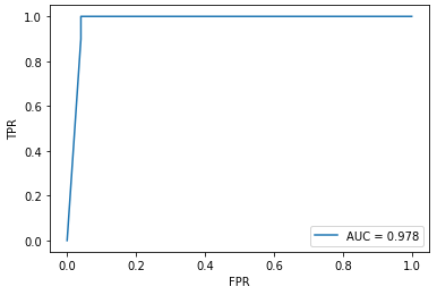


Class-2

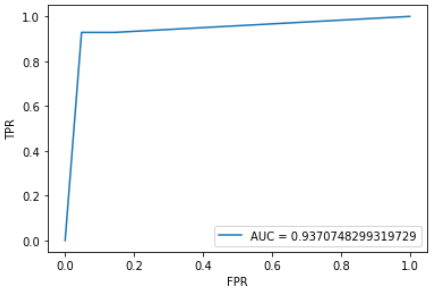


Class-3

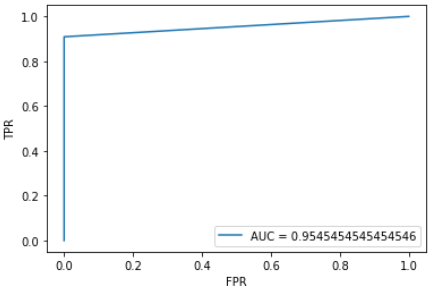
for fold-3



Class-1

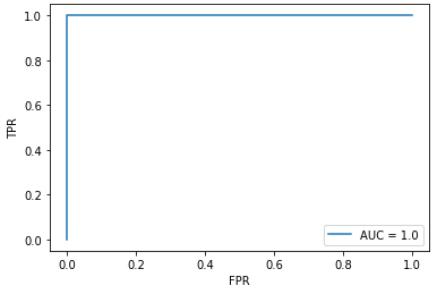


Class-2

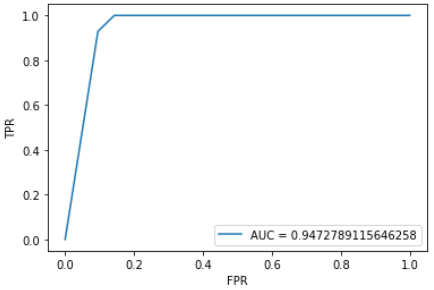


Class-3

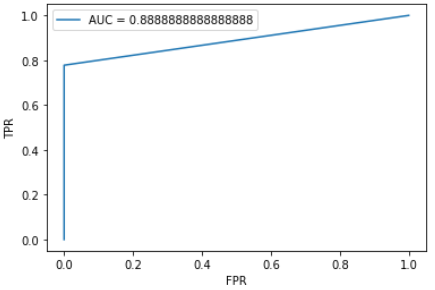
for fold-4



Class-1

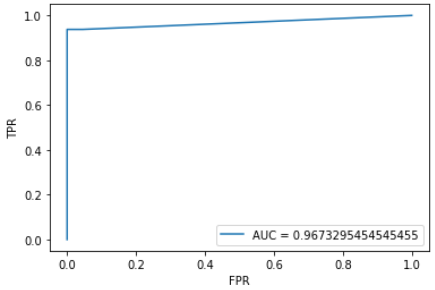


Class-2

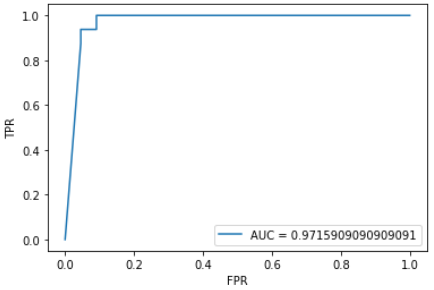


Class-3

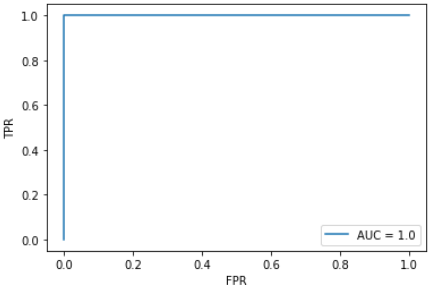
for fold-5



Class-1



Class-2



Class-3

The following cross-validation scores were obtained:-

```
individual cross-validation_scores: [0.8        0.91428571 0.94285714 0.88571429 0.94736842]
average cross-validation score: 0.8980451127819549
```