Table of Contents

I. Introduction **3**

II. Methodology and Results

II.I Unsupervised Learning **3-6**

Raisin Dataset

II.II Supervised Learning **6-9**

Red Wine Dataset **6-8**

Rice Dataset: Cammeo and Osmancik **8-9**

III. Discussion and Conclusion **9-11**

IV. References **12**

V. Appendix **13-39**

VI. Veriguide Report **40**

I. Introduction

A supervised learning algorithm uses input and output data with labels, whereas an unsupervised learning algorithm does not. This project applies unsupervised and supervised learning methods to analyze the dataset in real life and is divided into two parts. One dataset for unsupervised learning and two datasets for supervised learning. The ultimate goal is to achieve the best model for unsupervised and supervised learning by applying different models and selecting the one with the best clustering results based on indicators like correctly clustered instances, accuracy, and precision.

II. Methodology and Result

II.I Unsupervised Learning

Unsupervised Learning: Raisin Dataset

Dataset raisin is chosen in the study of unsupervised learning which contains different characteristics and classes of raisins. In this report, methods like the Elbow Curve, AIC, and hierarchical clustering to confirm the number of clusters. Followed by applying the K-means clustering, Gaussian mixture model and ensemble clustering methods to make the comparison and choose the best model to classify types of raisin by different characteristics. There are eight variables in the raisin dataset to study the class of raisin including area, perimeter, major axis length, minor axis length, eccentricity, convex area, and extent **[Appendix 2.1.1]**. There are a total of eight attributes (including class) with 900 rows. All attributes are numerical data.

Preliminary steps: data cleaning and processing

Data cleaning and processing were the first considerations. There is no missing value in the data **[Appendix 2.1.2]**. The class attribute is removed in unsupervised learning. Features with different measurement units have a similar range of variance after processing after implementation of data by ($z = (x-\mu) / \sigma$) **[Appendix 2.1.3]**.

Distribution Plot/ QQ-plot/ Box plot

Some characteristics can be observed from the data. On the graph of relationships between multiple variables, linear relationships, divergence and spherical relationships are observed from the scatter plot. Uniform, quasi exponential and irregular distribution of each variable is also found **[Appendix 2.1.4]**. By QQ plot, none of the 7 variables followed a normal distribution **[Appendix 2.1.5]**. In the box plot, all variables contained outliers, so clustering methods easily affected by outliers should not be considered **[Appendix 2.1.6]**.

Step 1: Eliminating the possibility of problems of the database balance

Since the variable ‘class’ will not appear throughout the rest of the process, it is necessary to first give a supplementary explanation of the issues of database balance. According to the **[Appendix 2.1.7]**, it can be discovered that the counts of the two classes (‘Kecimen’ and ‘Besni’) are of similar size, meaning that in this case, there will be no problems of uneven data distribution, uneven access frequency, and uneven node performance, not causing some nodes to be overloaded, thus not affecting the performance and availability of the system. Sometimes the imbalance of the counts of different classes will cause the inaccuracy of the result of the following clustering methods and the possibility of this impact has been ruled out.

Step 2: Reducing the dimensions of the data

Before everything started, it is necessary to first use the t-Distributed Stochastic Neighbor Embedding algorithm to change the original seven dimensions which are unable to be displayed into two dimensions being able to be displayed. The **[Appendix 2.1.8]** is a scatter plot displaying the raw data points through the reduction of the dimensions.

Actually, there are a lot of methods for dimensionality reduction, including PCA and t-SNE most commonly. The t-SNE algorithm is eventually chosen because it is a popular non-linear dimensionality reduction algorithm for mapping high-dimensional data into a low-dimensional space, preserving the local structure and similarity between data points, and for complex data sets, it can show different clusters and distributions. However, PCA is a linear dimensionality reduction technique that works only on linearly related data. For non-linearly related data, PCA may fail because it cannot find valid principal components, thereby affecting the accuracy of subsequent analysis.

Step 3: Confirming the number of clusters to be used in the clustering

The number of clusters to be used in the clustering is an essential parameter to be judged by the operators. Afterwards, the most suitable option in this situation can be confirmed. Based on the elbow curve method, according to the **[Appendix 2.1.9]**, the elbow point, also known as the turning point on the curve, is the value of k where the decrease in WSS starts to level off. This point represents the optimal number of clusters for the dataset and obviously in this case that is two. The elbow curve method only requires running the clustering algorithm multiple times and computing the WSS, which is a relatively fast and efficient process.

According to the Silhouette Analysis method with the curve **[Appendix 2.1.10]**, the optimal number of clusters is often the value of k that maximizes the average silhouette score, which measures how well each data point fits into its assigned cluster, based on the distance to other data points in the same cluster and the distance to data points in neighboring clusters and obviously, in this case that is also two. The silhouette analysis can evaluate and compare the quality of individual clusters and the overall clustering structure, and to identify potential issues such as overlapping or poorly separated clusters.

Hierarchical clustering can also be used to identify the optimal number of clusters in a dataset by examining the dendrogram. The objective function of hierarchical clustering is to group similar data points together into clusters based on their proximity or similarity. Here, all the three linkage methods are conducted, however the complete linkage and the average linkage **[Appendix 2.1.11 & 2.1.12]** are chosen eventually because the output of the simple linkage method is complicated and unclear. Then, we can examine the dendrogram to identify the level at which the clusters are most distinct and well-separated, and that level reflects the optimal number of clusters, in this case the answer should also be two in both linkage methods. This level can be identified by looking for a large vertical distance between the branches of the dendrogram, indicating a large difference in distance or dissimilarity between clusters. The dendrogram of the average link indeed contains a very small amount of the data on the rightmost, which can be neglected for its size is too small.

Finally, the AIC and BIC methods are utilized to judge the fittest number of clusters. AIC and BIC are both based on the principle of balancing model complexity and model fit, but they differ in their approach to penalizing more complex models. The objective function of AIC is to minimize the information loss between the true data-generating process and the estimated model, while balancing the trade-off between model complexity and model fit. The objective

function of BIC is to find the model that maximizes the posterior probability of the model given the data, based on Bayesian inference.

The smaller the AIC and BIC, the better the predictive ability of the model and the smaller the complexity, with the consideration of the influence of sample size. The number of clusters that minimizes the AIC or BIC is often chosen as the optimal number of clusters. However, according to the **[Appendix 2.1.13 & 2.1.14]**, with the increase of value k, the value of both AIC and BIC decrease, thus not being able to draw valid conclusions within these trends due to the thoughtlessness of characteristics and distributions of the dataset. Therefore, this method cannot be applied here, and according to the several other methods mentioned above, a conclusion can be made that the most proper number of clusters is two.

Step 4: Applying Ensemble clustering methods and K-means clustering methods
Since the resulting clusters of K-means can vary depending on the initial random assignment of cluster centroids, leading to instability and inconsistency in the clustering results, sensitive to the initial conditions, therefore the K-means clustering method is tried as many times as possible to avoid the limitation of the impact of the initial conditions to a large extent. Here, the number of estimators is set to be seven because for some reasons, if it is set larger than seven, for example ten, at least one of the plots of the K-means clustering is to some reasons not similar to others, which can be acted as an outlier.

Another thing is that although it is confirmed that the most appropriate number of clusters is equal to two, if the k value is small, the k-means algorithm may cause underfitting problems, assigning multiple different categories to the same cluster, resulting in inaccurate clustering results. Therefore, in integrated clustering, in order to improve the accuracy of clustering, it is necessary to increase the k value as much as possible, so that each cluster is more refined and can better distinguish different categories. Therefore, here the number of clusters is changed from two to five and that's the seven plots of the K-means clustering with 5 numbers of classes shown below. **[Appendix 2.1.15]**. It can also be discovered that some of the points in different plots show different results, especially some points in the edge region, like the bottom right points of the fifth plot and the seventh plot.

After conducting seven consecutive K-means clustering, now it's time to ensemble these methods together. Here, it is necessary to know the concept of the ensemble clustering method and the example will be used to demonstrate the concept. Ensemble clustering combines multiple clustering algorithms or multiple runs of the same clustering algorithm to improve the quality and robustness of clustering results by exploiting the diversity of different clustering algorithms to overcome the limitations of individual algorithms or runs. And here, we try to ensemble the run of the K-mean clustering algorithm seven times together to try to make up for the limitations of the K-means clustering algorithm mentioned before. To achieve this goal, the cluster ensemble approach is used, where multiple clustering results are combined by creating a consensus partition that maximizes the agreement between the clustering results. We use the Co-association matrix-based methods which represent the pairwise similarity or association between nodes, and then use this matrix as input to a clustering algorithm. The quality of the clustering results is evaluated by using external validation metrics.

One important thing to be mentioned is that since the most appropriate number of clusters proves to be two, but it may cause some underfitting problems because the class is small enough, in order to solve this problem, the cluster aggregation approach is applied. In this approach, the results of different clustering algorithms or runs are combined by assigning

each data point to the cluster that it belongs to in the majority of the clustering results and finally we still reach a two-cluster classification result. We use the voting-based methods, constructing a consensus matrix, which represents the pairwise agreement or disagreement between nodes in the different initial clustering or partitions. Each entry in the consensus matrix represents the number of times that two nodes are assigned to the same cluster in the different initial clustering or partitions. Finally, we can get the ensemble clustering plot **[Appendix 2.1.16]** with the table of the true cluster labels and predicted cluster labels **[Appendix 2.1.17].** The correct instances rate can be calculated, equaling to 70.11%, which is a relatively high accuracy rate.

The ensemble clustering plot can also be compared with the original plot. Obviously, it can be discovered that in the original plot, the data points of the two classes have a very high coincidence, basically not being able to see a clear distinction. However, through the ensemble clustering, we can observe that it can mainly separate the dataset into two classes, with just a few outliers and relatively high accuracy rate and the decision boundary is quite smooth. **[Appendix 2.1.18]**

Then, we also randomly select one K-means clustering. Based on the plot, it can also separate the dataset into two classes, the correct instances rate is surprisingly high, equaling 76.33%. **[Appendix 2.1.19 & 2.1.20 & 2.1.21]**

Step 5: Comparing the ensemble clustering methods with other clustering methods

Therefore, comparing K-means clustering with Ensemble clustering, the correct instance rate of Ensemble clustering is lower than K-means clustering, and we select K-means clustering method rather than ensemble clustering method. **[Appendix 2.1.22]**

Besides the K-means clustering and ensemble clustering, we also conduct hierarchical clustering, DBSCAN clustering, and probability clustering. According to the performance table, we can understand that the K-means clustering is always the best option with the highest correction rate equaling 76.33% compared with other clustering methods and the ensemble clustering methods ranking the second. **[Appendix 2.1.23]**

The reason why the K-means clustering method is superior to the ensemble clustering method probably is that the ensemble clustering requires selection of multiple clustering algorithms and their parameters, which requires more expertise and experience. The performance of Ensemble clustering may suffer when choosing inappropriate algorithms or parameters. Moreover, Ensemble clustering may not perform as well as K-means when dealing with high-dimensional data sets, and ensemble clustering may require more complex algorithm integration.

II.II Supervised Learning

II.II.I Supervised Learning: Red Wine Dataset

We are using the winequality-red dataset for our supervised learning methods. It contains 13 columns with no missing or duplicate values and 1599 rows.**[Appendix 3.1.1]** We have created a binary variable, quality_level, based on the quality column. If quality is less than or equal to 5, it is classified as low_quality; otherwise, it is classified as high_quality. The dataset includes columns such as fixed acidity, volatile acidity, critic acid, residual sugar, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, quality, and chlorides.

Lower dimensional plot (UMAP/Local Linear Embedding)
We used UMAP **[Appendix 3.1.2]** to visualize the dataset with N_nei=15, Euclidean distance, and min_distance=0.1. The resulting graph showed no clear boundary for separating the two classes, with many overlapping points in the middle. This suggests that the similarity between the two classes is not very high. The LLE graph**[Appendix 3.1.3]** showed a concentration of datapoints on the middle left side, with the High_Quality wine class having a wider distribution than the Low_Quality class. It may be necessary to remove irrelevant columns to simplify the model.

Distribution Plot/Box plot and Final_Model
We have selected variables with distinct distributions across classes for our final model **[Appendix 3.1.4-Appendix 3.2.4]**, including volatile acidity, fixed acidity, free sulfur dioxide, total sulfur dioxide, density, sulphates, and alcohol. We have excluded sugar, citric acid, pH, and chlorides, but will provide their graphs in the report. We will use 70% of the data for cross-validation and 30% for testing, after normalizing the data. In the next section, we will apply various supervised learning methods and analyze their performance using the testing set.

Random forest
Random forest involves extending the decision tree algorithm through the use of bagging, specifically Bootstrap Aggregating . This method involves constructing multiple decision trees using random subsets of features and data. To evaluate the performance of the model, we will use the out-of-bag (OOB) or unseen data to estimate the generalization error.

To construct a table of feature importance **[Appendix 3.2.5]**, we will utilize the mean decrease impurity method. This entails calculating the total decrease in impurity caused by a given feature across all trees in the ensemble. This information can be used to rank the relative importance of each feature in the model. The results of our Python analysis, including the importance table, confusion matrix, parameter settings, and other score functions such as f1-score, accuracy, and Cohen kappa, will be presented in the appendix section of our report**[Appendix 3.2.6-Appendix 3.2.7]**.

GBM (Gradient Boosting Method)
GBM involves using an ensemble model based on decision trees to improve the accuracy and robustness of our classification task. The idea behind this method is to iteratively add decision trees to the model, with each tree seeking to minimize the residual error of the previous trees. To begin, we will use a predicted value based on the log odds value to predict the class of each instance. We will then construct a decision tree based on the residual error, calculate the loss function, and repeat the process to construct additional trees. Each previous tree will be multiplied by a learning rate, and each prediction will be based on all the trees that have been constructed in the previous iterations. This process will be repeated until the model converges and the desired level of accuracy is achieved**[Appendix 3.2.8 - 3.3.0]**.

Logistic regression
Logistic regression is a supervised machine learning algorithm for binary classification. We will fit a logistic function to the training data using maximum likelihood estimation. We will then apply the softmax function to the logistic model to calculate the probability of each observation belonging to either the High_Quality or Low_Quality class. To optimize the model, we will use gridSearchCV to tune the regularization parameter, denoted as C. Based on our analysis, the best value for C was determined to be 0.1. By tuning this parameter, we aim to improve the accuracy and performance of our logistic regression model **[Appendix 3.3.1 - 3.3.2]**.

SVM (Support Vector Machines)
SVMs utilize a hyperplane to separate the different classes in the input space. The distance of the margins between the hyperplane and the closest data points in each class determines the classification accuracy. By using RBF or polynomial kernels to map the input space into high-dimensional feature spaces, it can handle nonlinear separation problems. By using GridSearch CV , we have determined that the best hyperparameters for our SVM model are a constant C of 0.1 and an RBF kernel by considering the accuracy rate.**[Appendix 3.3.3]**

Decision tree
Decision tree is a model that processes decision-making based on input features. Maximum likelihood estimation is used to fit the tree to the training data. However, due to the poor performance of this model, we can conclude that there is not a clear rule for separating the different data types. Additionally, the similarity among the different observations in the dataset is not particularly high.**[Appendix 3.3.4-3.3.5]**

II.II.II Supervised Learning: Rice Dataset: Cammeo and Osmancik

In this rice dataset, there are 8 variables, includes the variables of 'Area','Perimeter', 'Major_Axis_Length', 'Minor_Axis_Length', 'Eccentricity', 'Convex_Area',' Extent', 'Class'. The 'Class' variables illustrated two Classes in the dataset, with Cammeo and Osmancik class rice, detailed information in **appendix 4.1**. The class of two types of rice will be classified and will be discussed by the following methods.

Distribution Plot and Box plot
From **appendix 4.2 to appendix 4.8**, it summarized and visualized the distribution of the variables in the rice dataset, the plot shows the density of the data at each point along the x-axis. It can show the median, quartiles of each variable. All variables will be used as the features to classify two classes.

Lower dimensional plot (Local Linear Embedding)
The lower dimension plot of LLE is used, it is the non-linear dimensionality reduction methods. It can represent the clusters or the patterns among the data which may be hard to observe in the original high dimensional space.
Referring to the LLE plot **(appendix 4.9)**, the patterns of two classes are highly similar to the variable distribution of two classes. It shows that most of the points stay on the left, only a small number of points are far from the middle.The plot showed that two types of rice did not form distinct clusters, most of them overlapped with each other. Hence, it showed that the similarity of Cammeo and Osmancik classes are high.

Decision tree
Decision tree method is used to classify different classes based on the variables from the dataset. It considers the possible combinations when making decisions on how to split the data, and the relationship between variables. From **appendix 4.10**, we set the max_depth as 5; then it can be observed that the variables are first split by the 'major_axis_length' variable. The accuracy of the decision tree is 91.6% (**appendix 4.11**); and we found the order of importance of the top three variables are 'major_axis_length, 'Perimeter' and 'convex area'.

Random forest

Besides, random forest is the extension of decision tree method, it can handle high-dimensional data with many variables. It combines multiple decision trees to create a forest of trees, separates the features, and each tree is trained on a different subset of the features. In **appendix 4.12**, the accuracy of the random forest is 92.2%, it can be observed that the order of importance variable is the 'major_axis_length'.

SVM (Support Vector Machines)

SVM works by finding the hyperplane that separates the data points while maintaining a specified margin of error. The hyperplane is the line that maximizes the distance between the closest points. The SVM classifier is created as a linear Kernel, training the SVM model by finding the set of parameters that maximize the margin between the support vectors. it will finally test the SVM model on the testing set. The hyperparameter is grid search CV, from **appendix 4.13** result, the accuracy of SVM is 93.16%, which is similar to the FI-score.

K-Nearest Neighbour(KNN)

The concept of the KNN classification is classifying samples into the class that show similar character in the dataset. The procedure of the KNN classification is to compute the distance between the new sample and every observation in the dataset. Then, choose k observation(s) in the dataset that are the nearest to the new sample. The new sample is classified to the most frequent class among the k nearest neighbor. The hyperparameter 'k' is tuned by grid search, trying different values of k and finding the best k that generate the highest accuracy by the KNN classification. After the KNN classification, the 10-fold Cross Validation is used to evaluate the model. The accuracy of the model is 93.7% **[Appendix 4.14]**.

Decision Tree with SAS EM

The concept of a decision tree is finding the mutual characteristics of the data as the classifier. If the majority of the observations in a class have the same character, that character will be considered as a criterion to decide whether the new sample belongs to the class. The observations are partitioned recursively based on the selected attribute until there are no remaining attributes for further partitioning, all examples for a given node belong to the same class. The attributes are selected on the basis of an impurity function which means the attribute can represent the class. The decision tree was built with the use of SAS EM. 70% of the data is used for training models and 30% of the data are used for testing. Despite the train test ratio, the parameter of the decision tree algorithm using the SAS EM default setting. The confusion matrix and the accuracy of the decision tree model is calculated after exporting the result tables from the SAS EM **[Appendix 4.15& 4.16]**. The accuracy of the model is calculated by (TP+TN)/(P+N). The accuracy of the decision tree model after the 10-fold cross validation is 91.70% **[Appendix 4.16]**.

III.Discussion and Conclusion

Summary of Unsupervised Learning in the Raisin Dataset

In unsupervised learning, if we consider all the seven variables together, we should choose the K-means clustering method and at some time we can choose the ensemble clustering method due to its performance is much superior to other clustering methods, though weaker than the K-means clustering method.

Although k-mean clustering has been selected among others when considering all variables together, it is also worth studying how data affects the selection of clustering tools. We also

consider several variables separately in pairs to demonstrate the advantage and disadvantage between K-means clustering and K-medians clustering as well as three linkage methods in hierarchical clustering.Variables convex area and perimeter are randomly chosen. By k-means and k medians, the correct rate of k medians (9+324)/900 = 37% **[Appendix 5.1]** is higher than that of k-means (265+3)/900 = 29.8% **[Appendix 5.2]**, attributed to the presence of outliers. That means k-medians handle outlier problems better. Also, in hierarchical clustering, a single link has a bad result of (449 + 0)/900 = 49.89% **[Appendix 5.3]** of correctly clustered instances as the data is linear and may be affected by the chain effect. The complete link also has a bad result of (417+1)/900 = 46.44% **[Appendix 5.4]** as it required a spherical cluster shape, but the data has a linear relationship. Average link performed better with (175+448)/900 = 69.22% of correctly clustered instances **[Appendix 5.5]**, which is the highest in k-means clustering and hierarchical clustering. Therefore, it can be concluded that the selection of clustering tools is highly relevant to the shape and characteristics of the data. Thus, some methods could be eliminated in the preliminary step when they do not match the shape and characteristics of the data, so that the process of unsupervised learning could be smoother and more efficient.

Summary of Supervised Learning in the Red Wine Dataset

We have included an overall performance table of the different models in the appendix section of our report. Based on the table, we have observed that most of the models, with the exception of the decision tree, have performed fairly well. However, overall, we have found that random forest has the best performance across all metrics, indicating that it is the most effective model for predicting both "High_Quality" and "Low_Quality" classes.

One possible reason why Gradient Boosting Machine (GBM) performed poorly is due to the presence of outliers in the data. GBM constructs models based on the residual, which can be affected by outliers and lead to incorrect classifications. Similarly, logistic regression may have underperformed due to its limited ability to handle nonlinear relationships between variables, as well as the absence of kernels, polynomial terms, and interactive features in the model. As the use of SVM with kernel may lead to overfitting, the performance of this model is somewhat inferior to that of random forest.

In contrast, random forest has demonstrated robust performance across a variety of metrics, making it the most suitable model for our classification task. Overall, we believe that random forest is the optimal model for our analysis and look forward to further exploring its capabilities.**[Appendix 3.3.6]**

Summary of Supervised Learning in the Rice Dataset

To conclude, it can be observed that from the lower dimensional plot, the similarity of two classes are high. The decision tree and random forest, the importance of variables of ‘Major_axis_Length’, ‘Perimeter’ and ‘convex area’ are relatively high that may affect the classification on Cammeo and Osmancik.

The accuracy of each model is summarized in **Appendix 4.17** .The K-NN classification algorithm performs the best. It has the highest accuracy (93.7%). That may be due to K-NN having no assumption required about the data that can effectively capture complex interactions between variables in the rice dataset.

On the other hand, all of the selected supervised learning methods perform well in terms of accuracy. The accuracy of the models is greater than 90%, that may be due to the decision boundary of the rice dataset being highly irregular. It is shown that the two groups(Cammeo

and Osmancik) are overlapping **[Appendix 4.18]**. The distance between each group is very short. Both decision trees, K-NN and random forest can generate a highly irregular decision boundary that classifies data points from each group, while the SVM can generate a hyperplane to classify the data.

The difference in accuracy of the decision tree algorithm between using SAS EM and python may be caused by different ways of correction in numbers.The random forest performs slightly better than the decision tree since a random forest can reduce overfitting of the model. While the SVM performs better than the random forest, that may be due to it can maximize the distance between 2 classes.

IV. References

CINAR, I. and KOKLU, M., (2019). “Classification of Rice Varieties Using Artificial Intelligence Methods.” *International Journal of Intelligent Systems and Applications in Engineering, 7*(3), 188-194.

CINAR I., KOKLU M. and TASDEMIR S., (2020). Classification of Raisin Grains Using Machine Vision and Artificial Intelligence Methods. Gazi Journal of Engineering Sciences, vol. 6, no. 3, pp. 200-209, December, 2020.
https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009. https://archive.ics.uci.edu/ml/datasets/wine+quality

V. Appendix

Unsupervised Learning:

Variable	Interpretation	type
Area	Size of raisins	numeric
Perimeter	Shape of the raisins	numeric
Major Axis Length	Elongation of the raisins	numeric
Minor Axis Length	Roundness of the raisins.	numeric
Eccentricity	Symmetry of the raisins.	numeric
Convex Area	Compactness and Convexity of the raisins.	numeric
Extent	Spread and Coverage of the raisins.	numeric

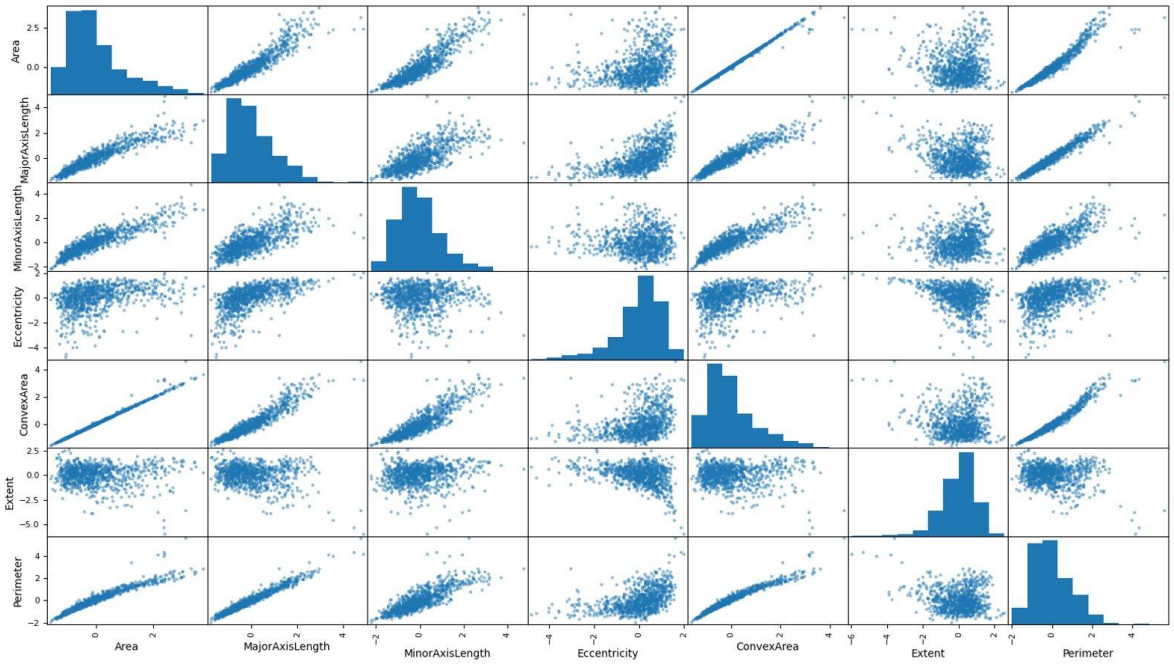
[Appendix 2.1.1]

```
     ...: data.isnull().sum()
Out[108]:
Area               0
MajorAxisLength    0
MinorAxisLength    0
Eccentricity       0
ConvexArea         0
Extent             0
Perimeter          0
Class              0
dtype: int64
```

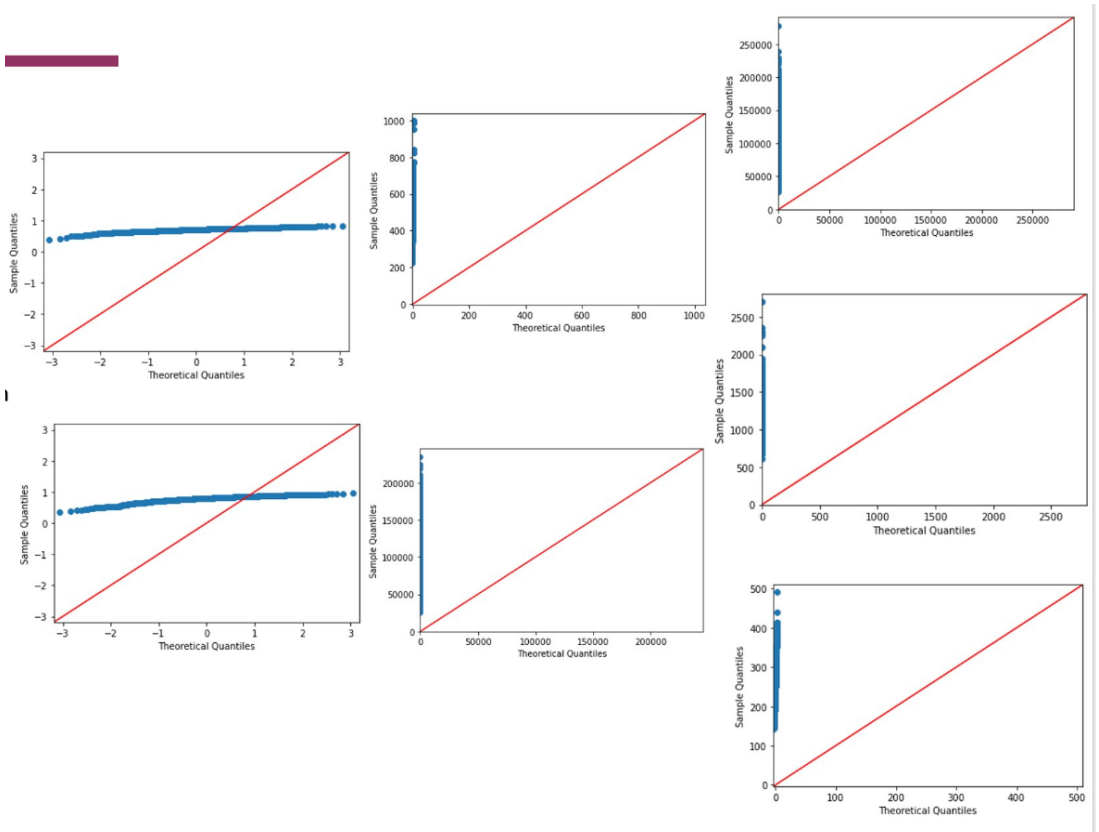
[Appendix 2.1.2]

```
        Area  MajorAxisLength  MinorAxisLength  ...    Extent
Perimeter    Class
0   -0.007182         0.097523        -0.023945  ...  1.106128
0.066237  Kecimen
1   -0.324037        -0.208896        -0.229165  ... -0.287617
-0.161163  Kecimen
2    0.078249         0.097704         0.236856  ... -1.157606
0.155858  Kecimen
3   -1.073689        -1.244359        -0.914765  ...  0.001711
-1.175261  Kecimen
4   -0.215274        -0.678581         0.726949  ...  1.744289
-0.338450  Kecimen
```

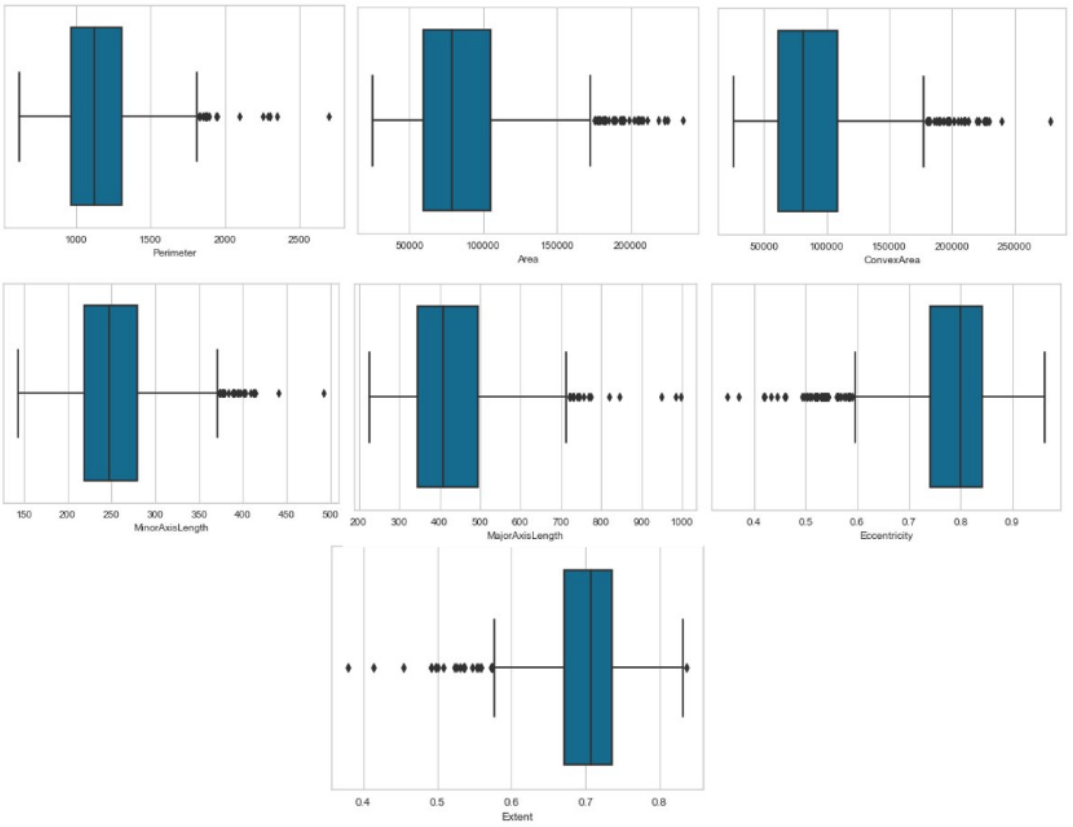
[Appendix 2.1.3]



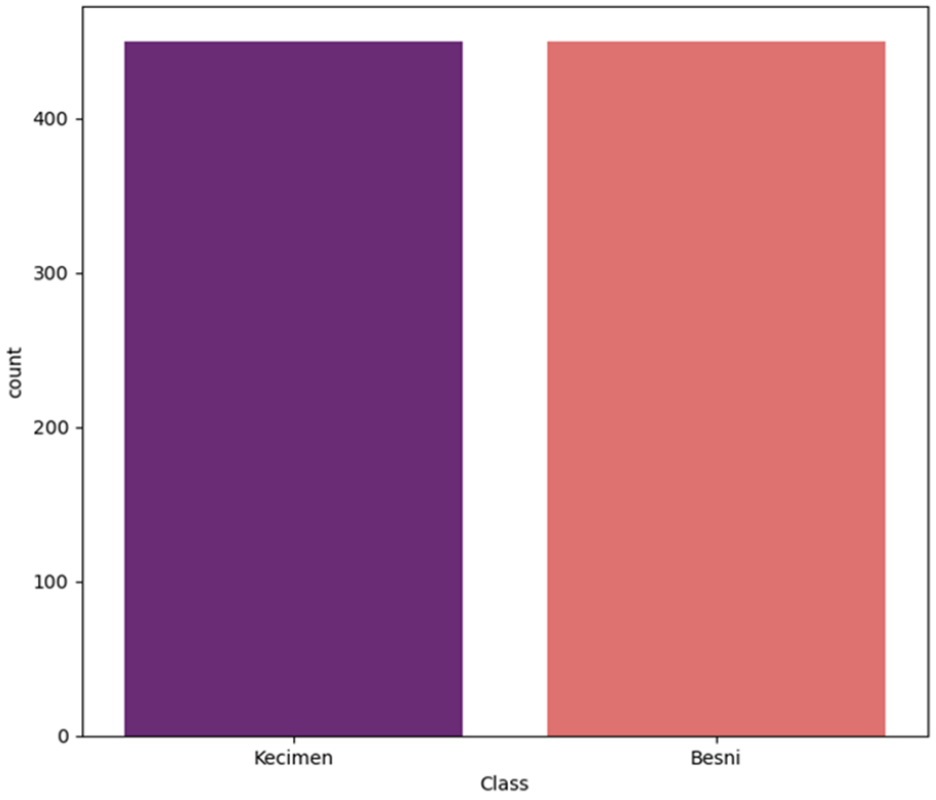
[Appendix 2.1.4]



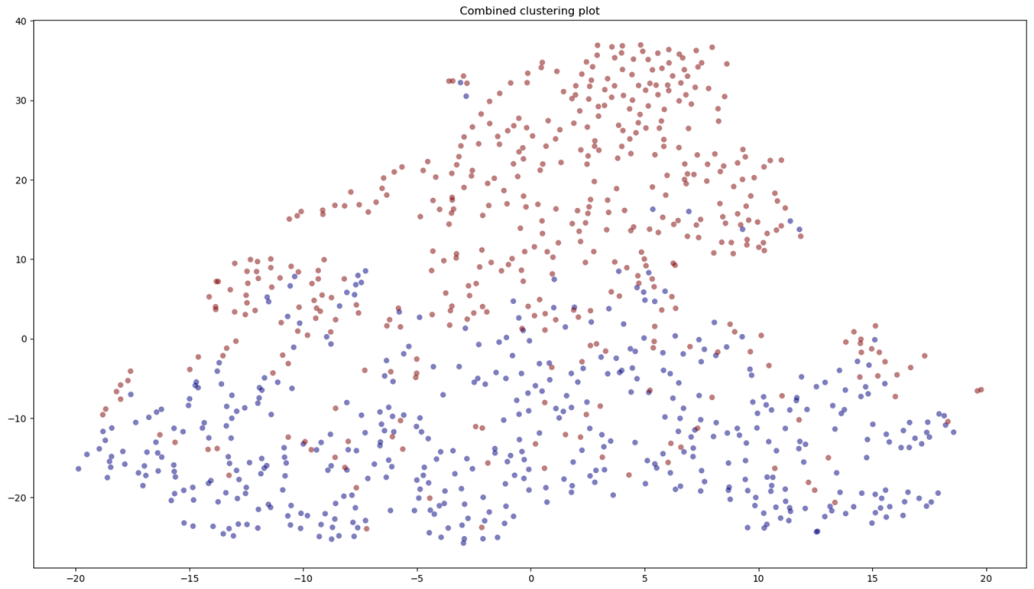
QQ-plot [Appendix 2.1.5]



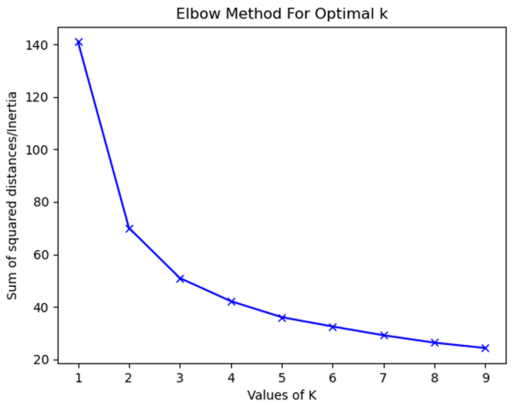
Box plot [Appendix 2.1.6]



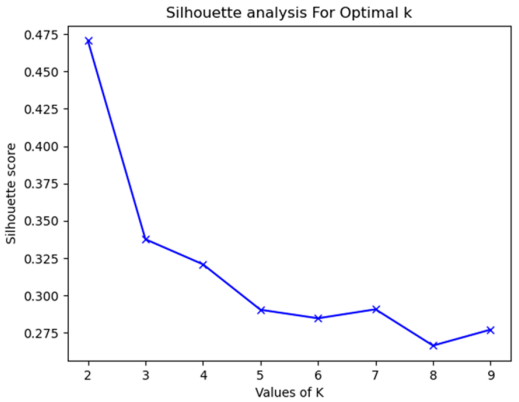
[Appendix 2.1.7]



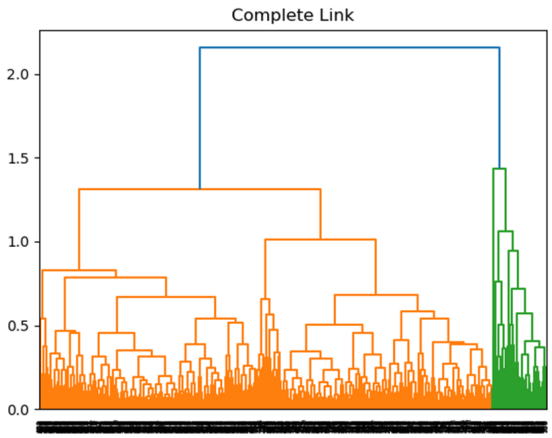
[Appendix 2.1.8]



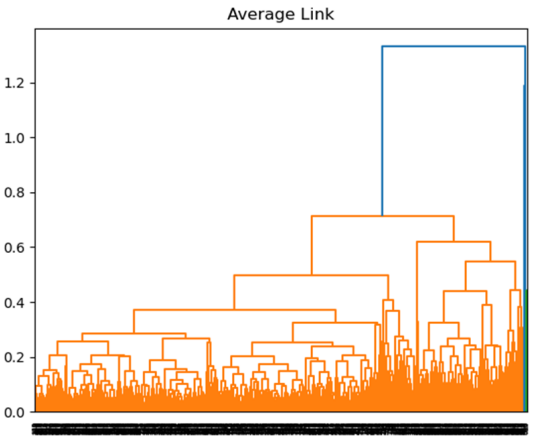
[Appendix 2.1.9]



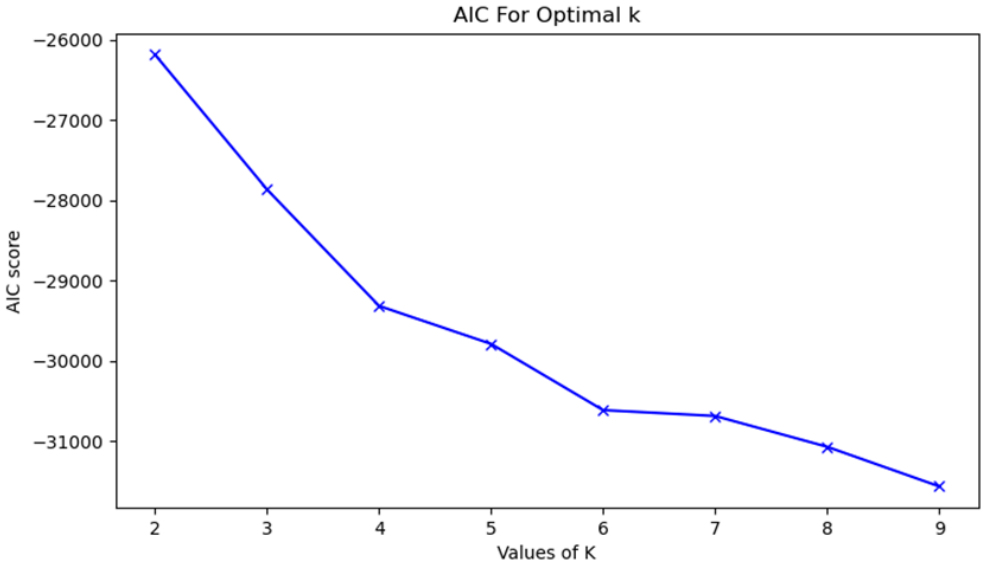
[Appendix 2.1.10]



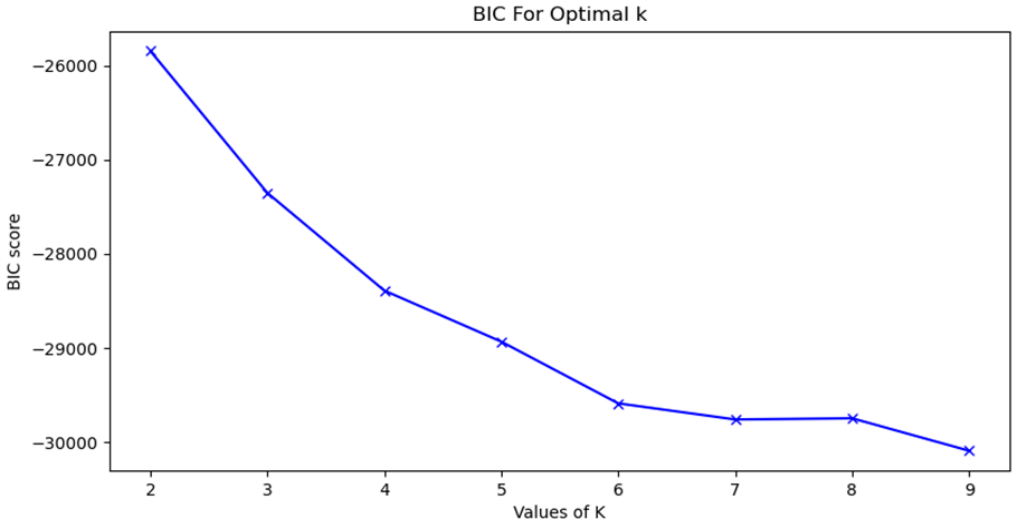
[Appendix 2.1.11]



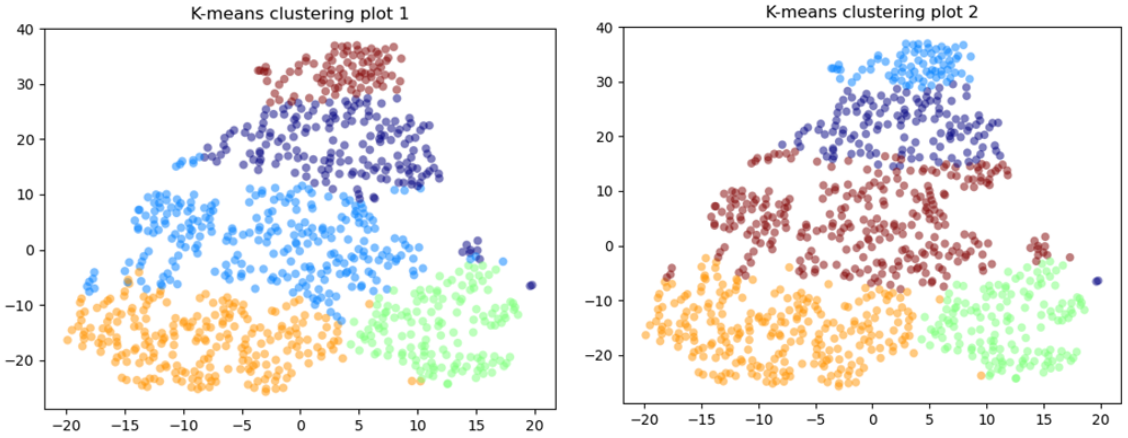
[Appendix 2.1.12]

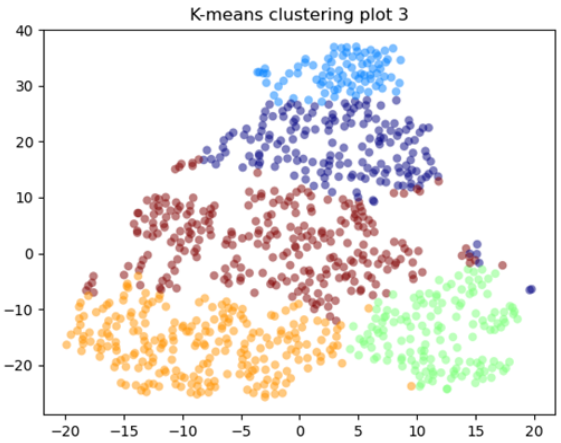


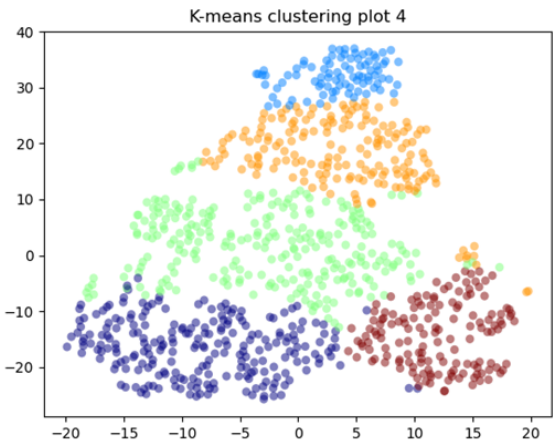
[Appendix 2.1.13]

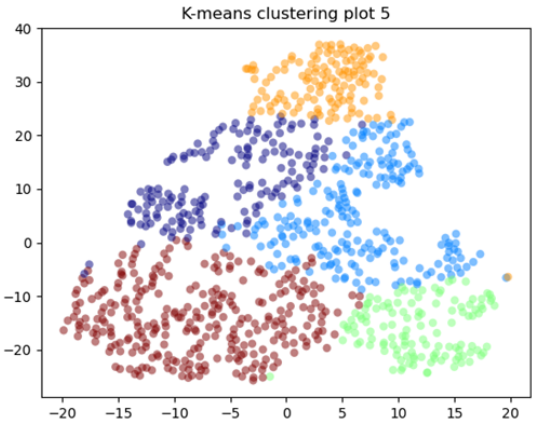


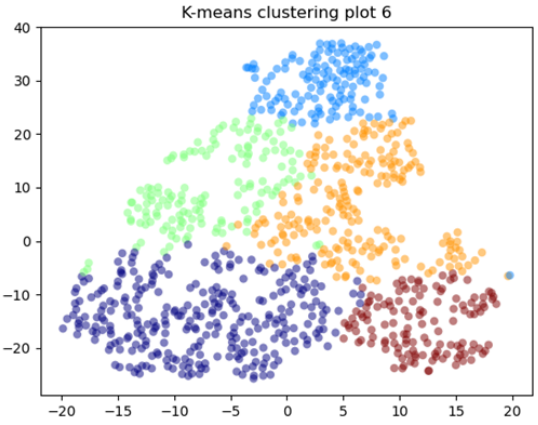
[Appendix 2.1.14]

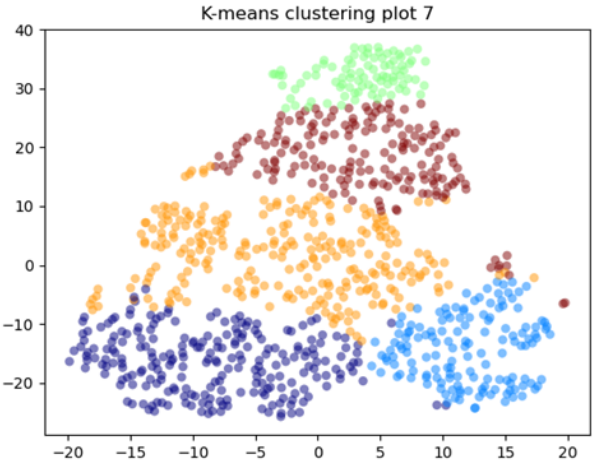




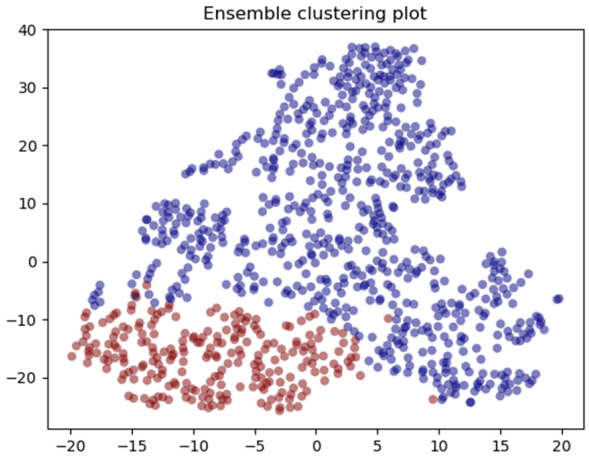








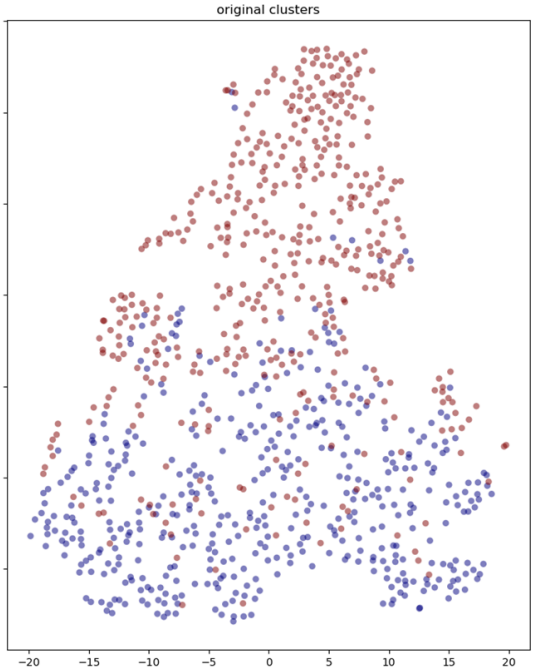
[Appendix 2.1.15]



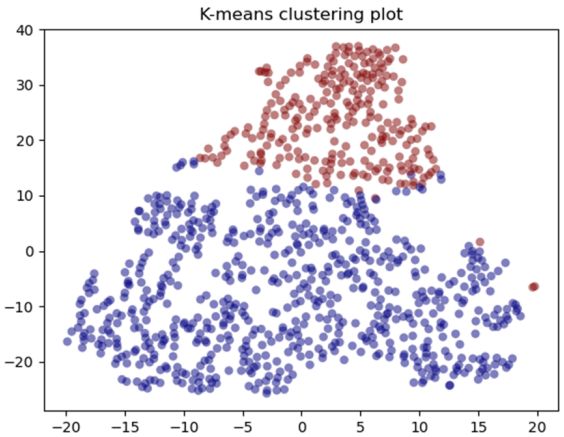
		Cluster Labels	
		K	B
True Cluster Lables	K	428	29
	B	240	203

[Appendix 2.1.16]

[Appendix 2.1.17]

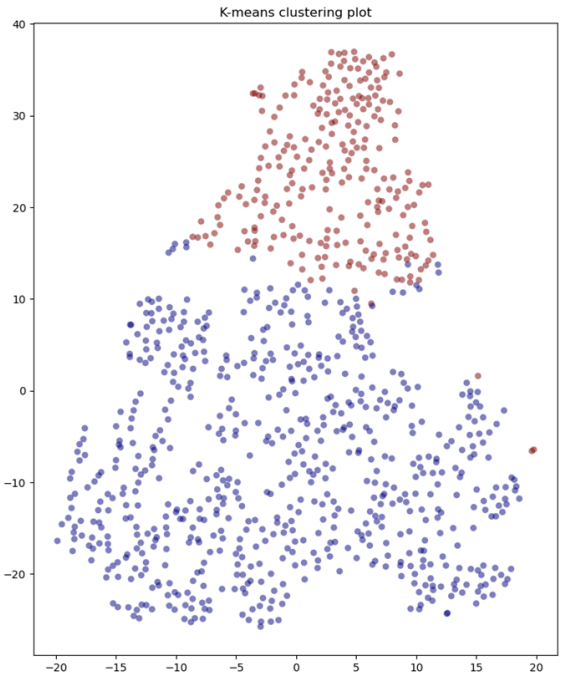


[Appendix 2.1.18]

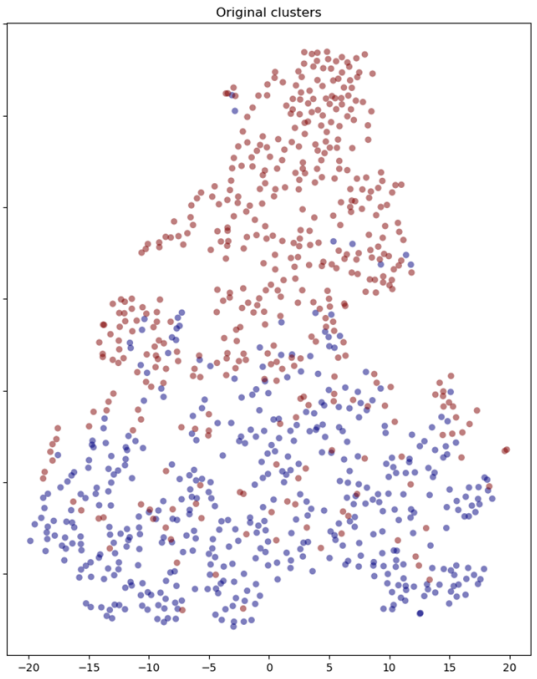


		Cluster Labels	
		K	B
True Cluster Lables	K	242	208
	B	5	445

[Appendix 2.1.19]



[Appendix 2.1.20]



[Appendix 2.1.21]

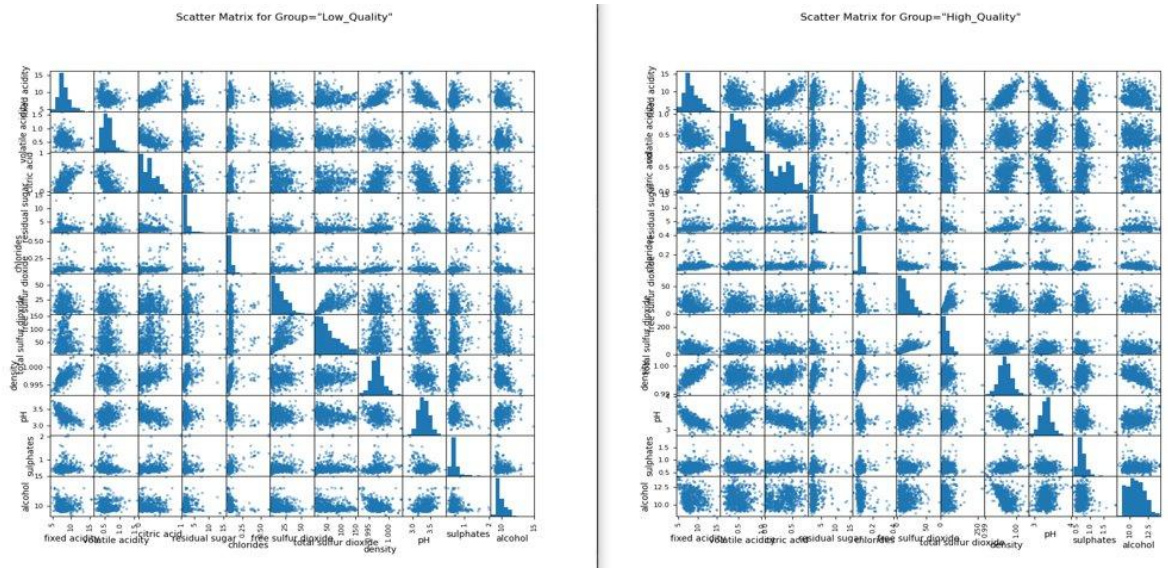
Summary	Correct rate	Incorrect rate
K-means clustering	76.33%	23.67%
Ensemble clustering	70.11%	29.89%

[Appendix 2.1.22]

Comparison	Correct rate	Incorrect rate
K-means clustering	76.33%	23.67%
Hierarchical clustering	60.22%	39.78%
DBSCAN clustering	50.44%	49.56%
Probability clustering	55.73%	44.27%
Ensemble clustering	70.11%	29.89%

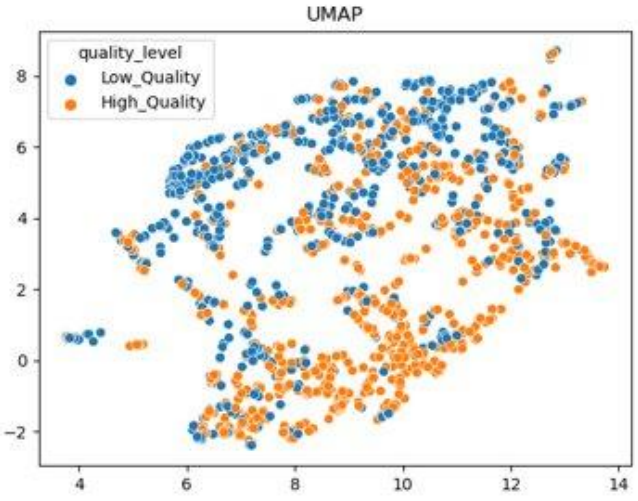
[Appendix 2.1.23]

Scatter plot Matrix for Group=Low_Quality and High Quality



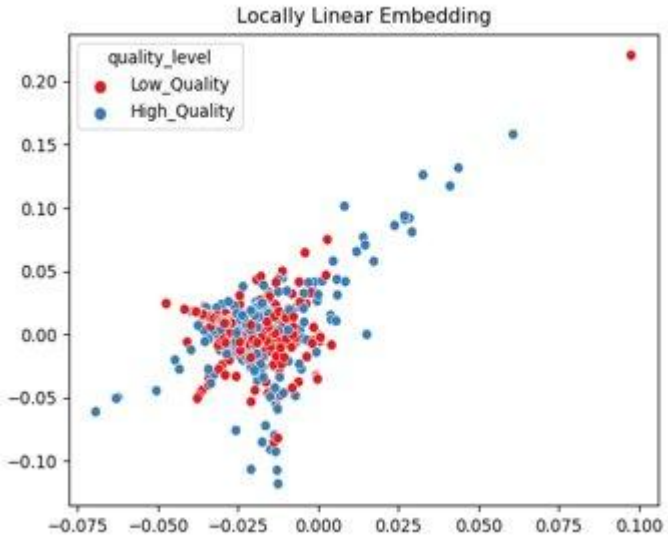
[Appendix 3.1.1]

UMAP plot after normalization



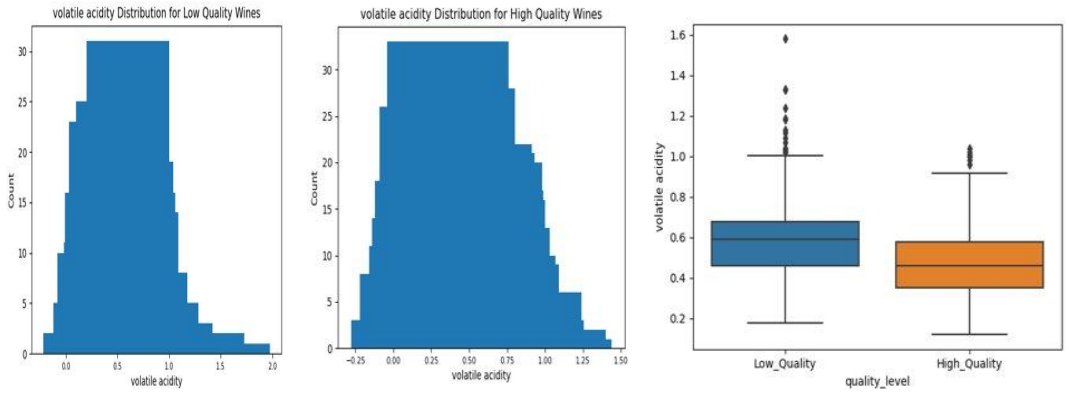
[Appendix 3.1.2]

LLE plot after normalization



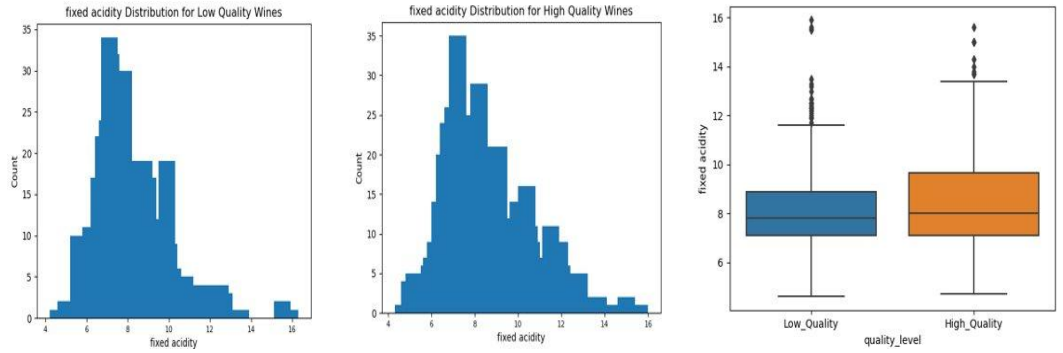
[Appendix 3.1.3]

Histogram/Box plot for variable volatile acidity among different class



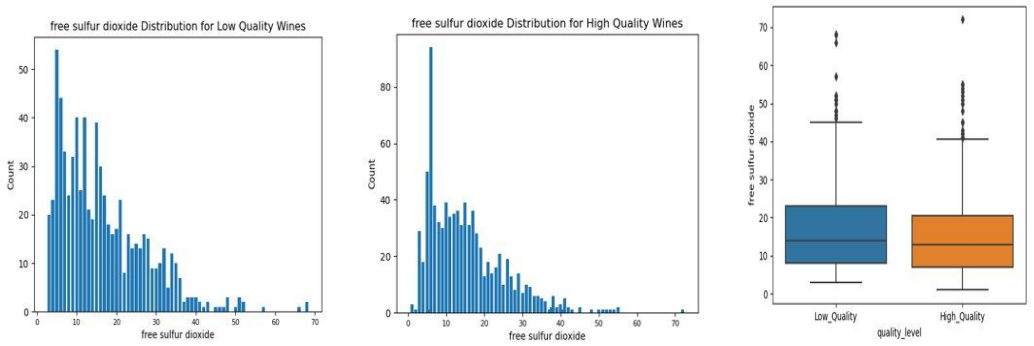
[Appendix 3.1.4]

Histogram/Box plot for variable Fixed acaidity among different class



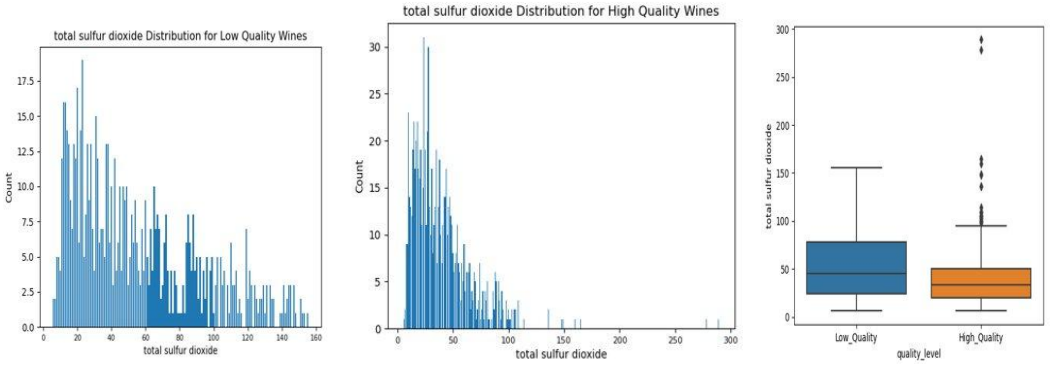
[Appendix 3.1.5]

Histogram/Box plot for variable Free sulfur dioxide among different class



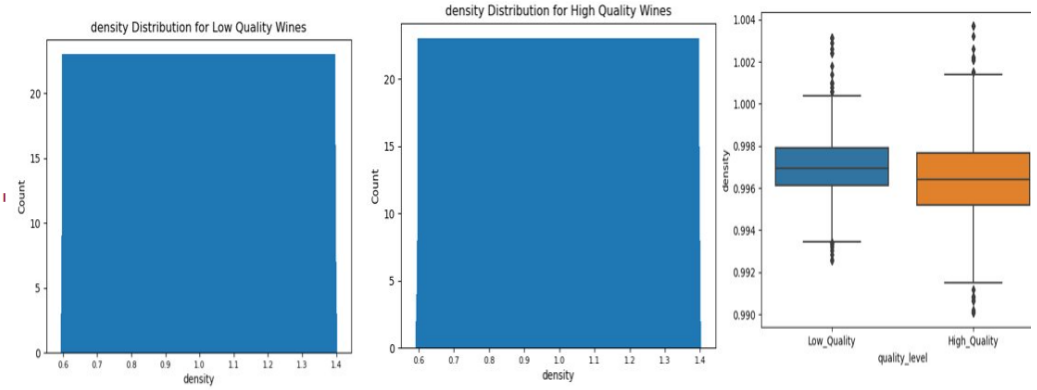
[Appendix 3.1.6]

Histogram/Box plot for variable Total sulfur dioxide among different class



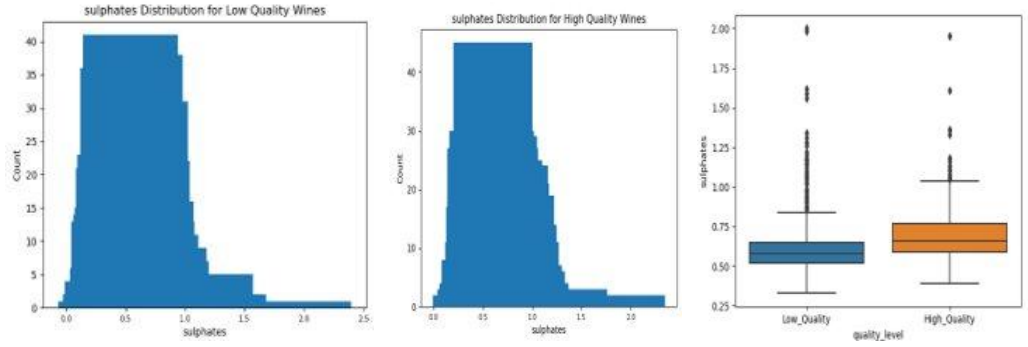
[Appendix 3.1.7]

Histogram/Box plot for variable Density among different class



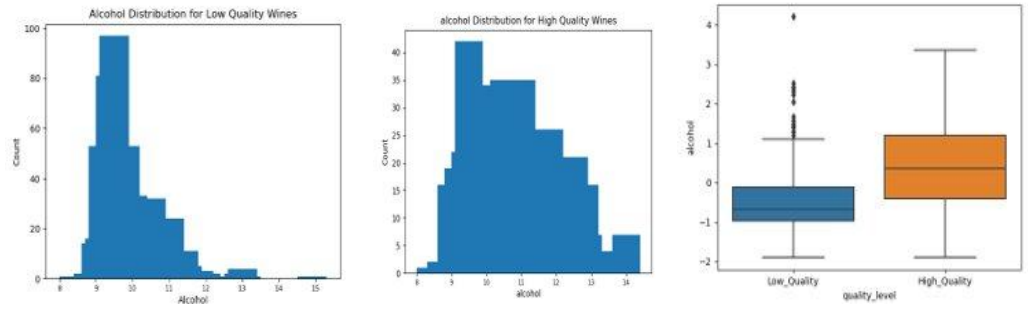
[Appendix 3.1.8]

Histogram/Box plot for variable Sulphates among different class



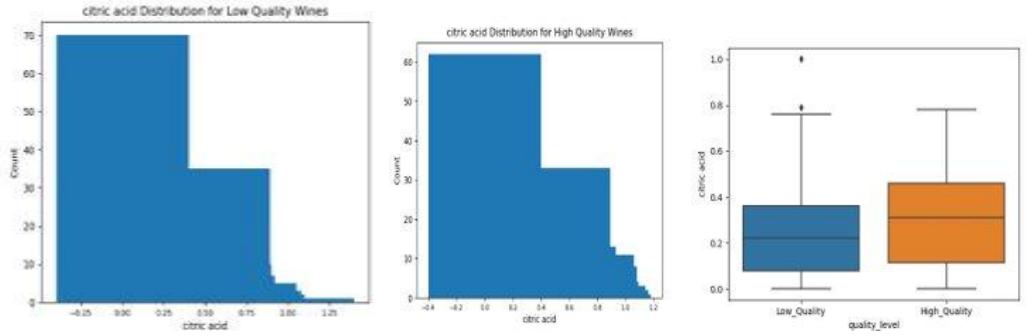
[Appendix 3.1.9]

Histogram/Box plot for variable Alcohol among different class



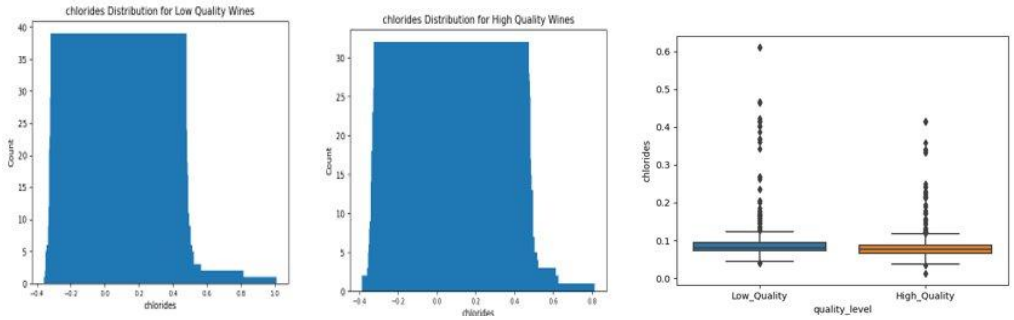
[Appendix 3.2.0]

Histogram/Box plot for variable Critic Acid among different class(Not selected variable)



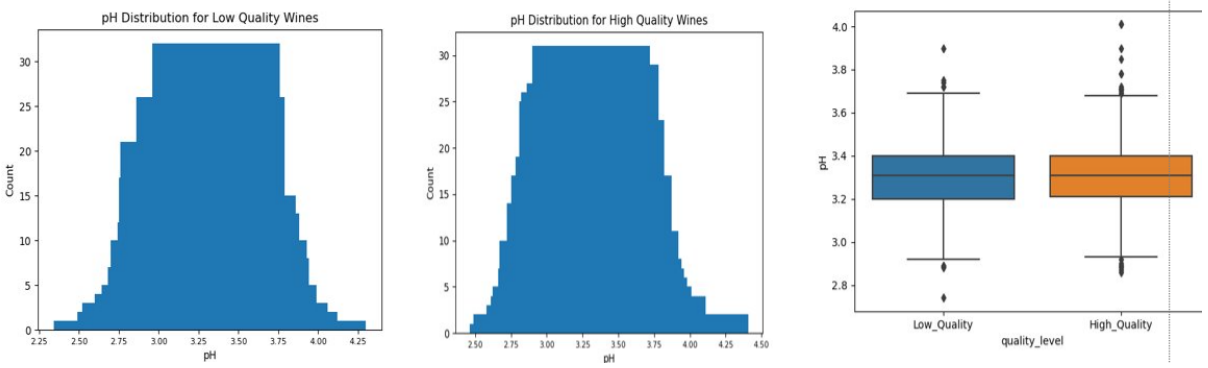
[Appendix 3.2.1]

Histogram/Box plot for variable Chlorides among different class(Not selected variable)



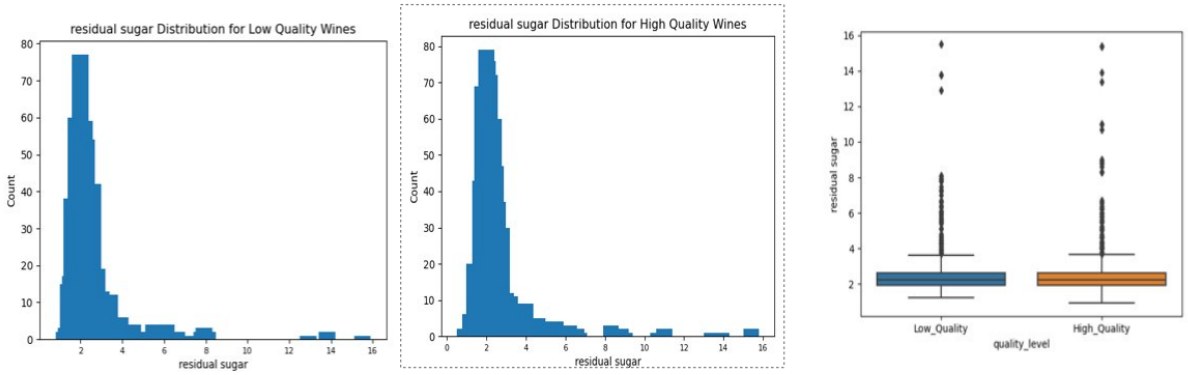
[Appendix 3.2.2]

Histogram/Box plot for variable pH among different class(Not selected variable)



[Appendix 3.2.3]

Histogram/Box plot for variable Residual sugar among different class(Not selected variable)



[Appendix 3.2.4]

Random forest importance table

```
      dtype='object')
Average accuracy score: 0.8255424528301887
                 Feature  Importance
6                alcohol    0.211649
5              sulphates    0.160263
1       volatile acidity    0.151443
3   total sulfur dioxide    0.142351
4                density    0.132427
0          fixed acidity    0.111781
2    free sulfur dioxide    0.090086
```

[Appendix 3.2.5]

Confusion matrix and other performance index

```
[[162  51]
 [ 51 216]]
Accuracy: 0.7875
Precision: 0.8089887640449438
Recall: 0.8089887640449438
F1 Score: 0.8089887640449437
Sensitivity: 0.8089887640449438
Cohen Kappa Score: 0.569552144326634
```

[Appendix 3.2.6]

Parameter setting for random forest

-Max-depth:None(Number of depth of tree that are built>>High may result over fitting, Low may result in bias)

-n_estimator:500(Number of tree that are built>>High may result over fitting, Low may result in bias)

-min_sample_leaf:1(Minimum numbers of lefts required to split in internal node>> High may result over fitting, Low may result in bias)

-min_sample_split:2(Minimum numbers of samples required to split in internal node>> High may result over fitting, Low may result in bias)

[Appendix 3.2.7]

Parameter setting for GBM

-Loss function: Deviance (Residual Deviance because of classification problem)

-Learning_rate: 0.1 (Concept from gradient descent>>too high will contribute the results from passing the local minimum, low will contribute the results from failing to attain local minimum)

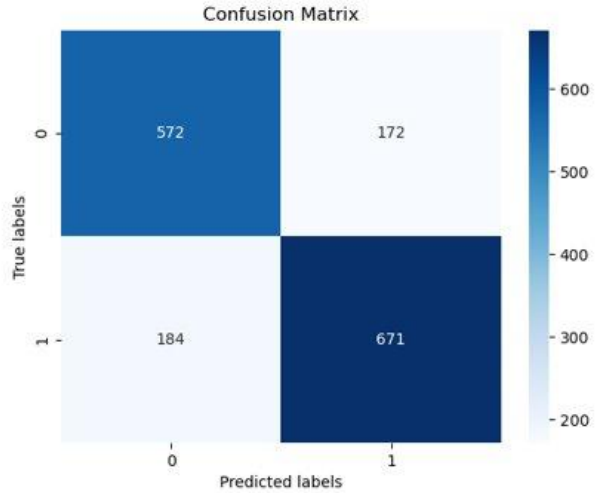
-n_estimators: 100(Number of tree that are built>>High may result over fitting, Low may result in bias)

-max_depth: 3(Number of depth of tree that are built>>High may result over fitting, Low may result in bias)

-Min_samples_split: 1(Minimum numbers of samples required to split in internal node>> High may result over fitting, Low may result in bias)

[Appendix 3.2.8]

Confusion Matrix for GBM



[Appendix 3.2.9]

Performance index For GBM:

-Precision: 0.795967

-Recall:0.7847953

-Accuracy:0.7773608

-Cohens' Kappa: 0.5530

-F1_Score=0.7903

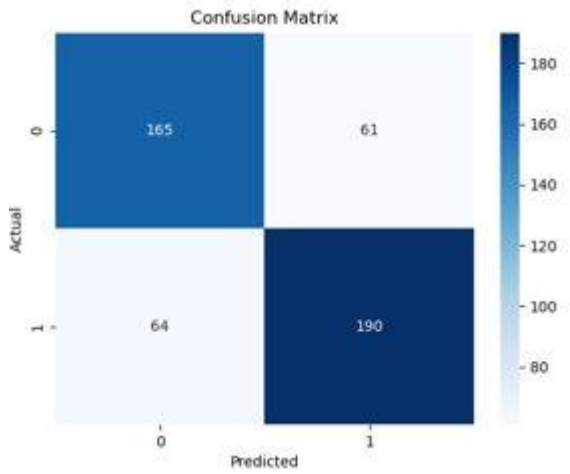
[Appendix 3.3.0]

Coefficient table for logistic regression model

	Variable	Coefficient
0	fixed acidity	0.125402
1	volatile acidity	-0.508960
2	free sulfur dioxide	0.262363
3	total sulfur dioxide	-0.559962
4	density	-0.067298
5	sulphates	0.347712
6	alcohol	0.948070

[Appendix 3.3.1]

Confusion matrix:



Performance index:

-Precision: 0.760

-Recall:0.7480

-Accuracy:0.7396

-Cohens' Kappa: 0.478

-F1_Score=0.7525

[Appendix 3.3.2]

SVM confusion matrix and other performance index

```
Best hyperparameters: {'C': 0.1, 'kernel': 'rbf'}
Best mean cross-validation score: 0.7408864221364222
Accuracy: 0.78125
Precision: 0.8008474576271186
Recall: 0.7651821862348178
F1 Score: 0.7826086956521738
Sensitivity: 0.7651821862348178
Cohen Kappa Score: 0.5627125702783369
Confusion matrix:
 [[186  47]
 [ 58 189]]
```

[Appendix 3.3.3]

Confusion matrix for decision tree

[[120,51],

[42,107]]

[Appendix 3.3.4]

Performance index for decision tree

-Accuracy=0.709375

-Cohen-kappa=0.41825

-Precision=0.74

-Recall=0.70

-f1-score=0.72

[Appendix 3.3.5]

Overall Performance for different model

Model	Random forest	GBM	Logistic regression	SVM	Decision tree
Accuracy	0.7875	0.7774	0.7396	0.78125	0.709375
Cohen Kappa	0.56955	0.553	0.478	0.562713	0.41825

Precision	0.8090	0.7960	0.760	0.80085	0.74
Recall	0.8090	0.7880	0.7480	0.76518	0.70
f1-score	0.8090	0.7903	0.7525	0.78231	0.72

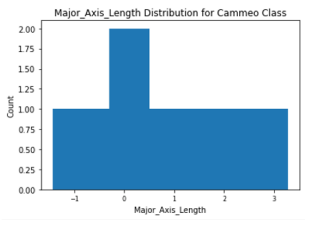
[Appendix 3.3.6]

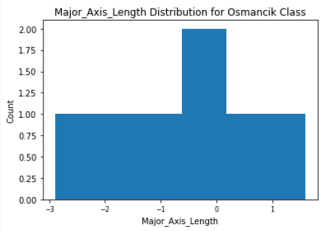
Supervised Learning dataset 2: Rice dataset

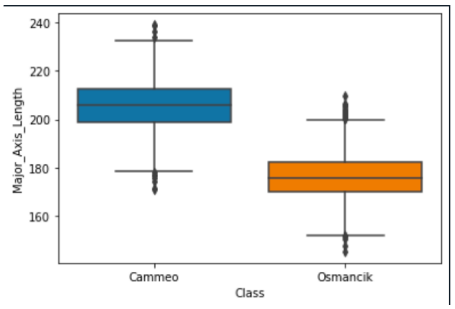
Variables Name	Description	Variable type
Area	number of pixels within the boundaries of the rice grain.	numeric
Perimeter	the circumference by calculating the distance between pixels around the boundaries of the rice grain.	numeric
Major_Axis_ Length	The longest line that can be drawn on the rice grain, i.e. the main axis distance, gives.	numeric
Minor_Axis_ Length	The shortest line that can be drawn on the rice grain, i.e. the small axis distance, gives.	numeric
Eccentricity	measures how round the ellipse, which has the same moments as the rice grain, is.	numeric
Convex_Area	Returns the pixel count of the smallest convex shell of the region formed by the rice grain.	numeric
Extent	Returns the ratio of the regionformed by the rice grain to the bounding box pixels.	numeric
Class	Cammeo or Osmancik rices	categorical

[Appendix 4.1]

Histogram/Box plot for variable Major_axis_Length of Cammeo and Osmancik Class

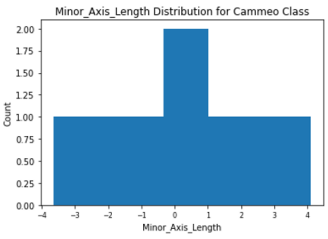


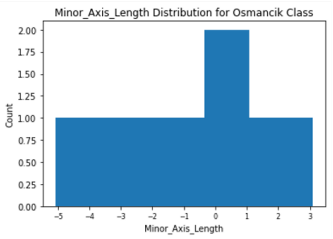


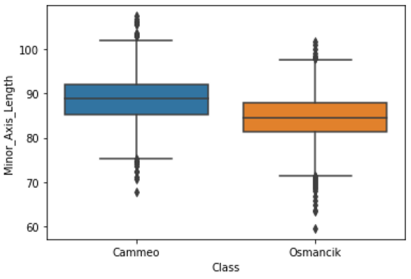


[Appendix 4.2]

Histogram/Box plot for variable Minor_axis_Length of Cammeo and Osmancik Class

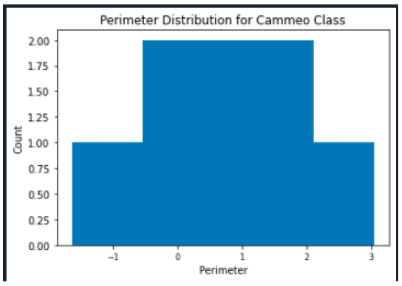


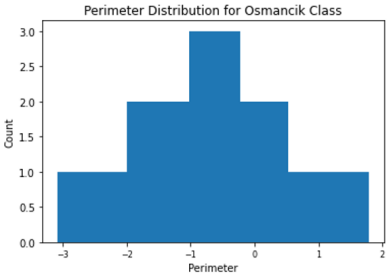


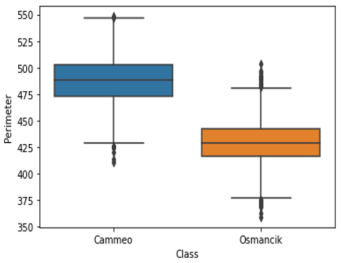


[Appendix 4.3]

Histogram/Box plot for variable Perimeter of Cammeo and Osmancik Class

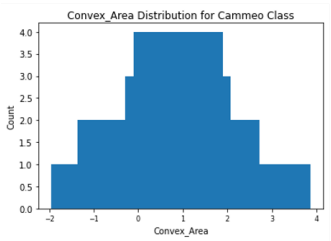


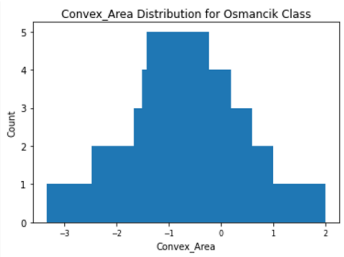


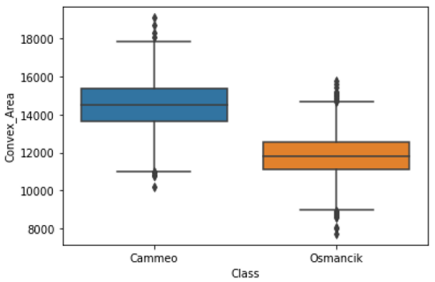


[Appendix 4.4]

Histogram/Box plot for variable Convex_Area of Cammeo and Osmancik Class

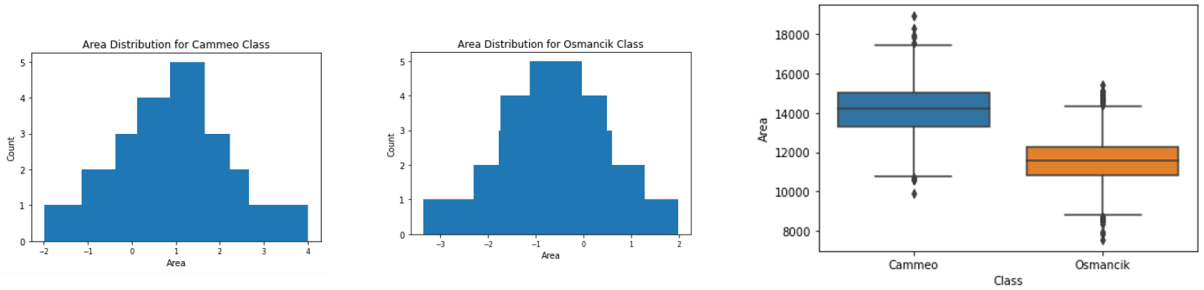






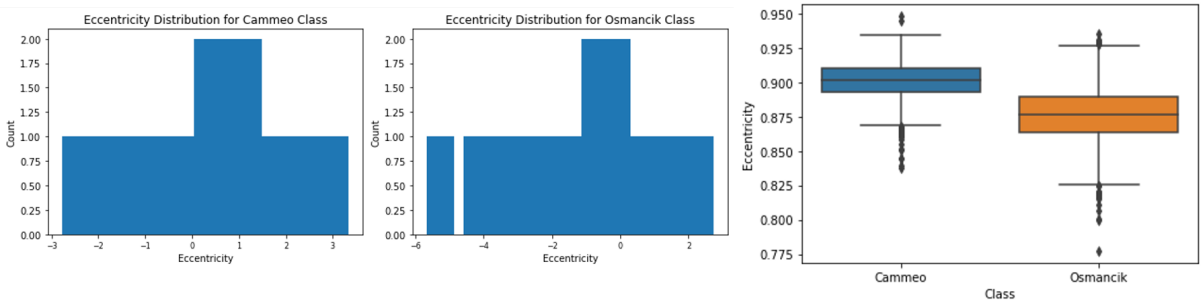
[Appendix 4.5]

Histogram/Box plot for variable Area of Cammeo and Osmancik Class



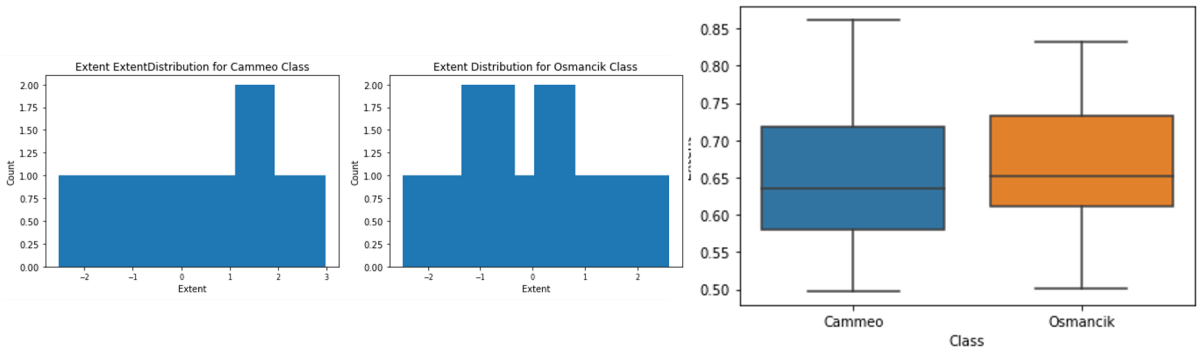
[Appendix 4.6]

Histogram/Box plot for variable Eccentricity of Cammeo and Osmancik Class



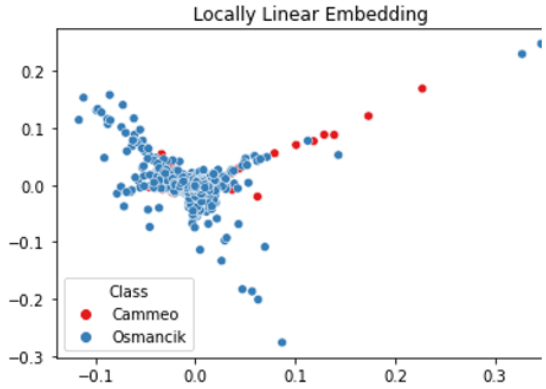
[Appendix 4.7]

Histogram/Box plot for variable Extent of Cammeo and Osmancik Class



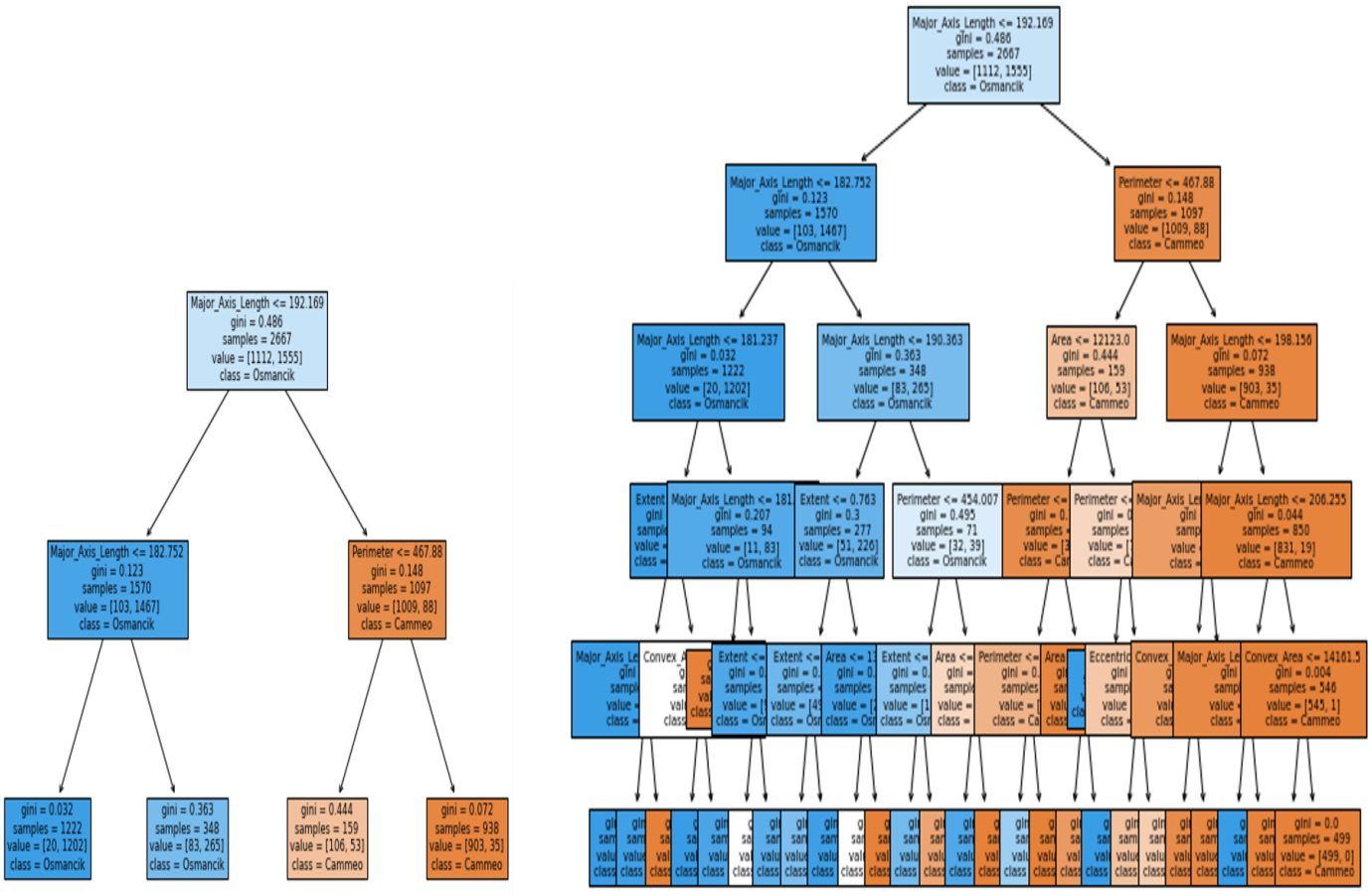
[Appendix 4.8]

Lower dimensional Plot LLE plot



[Appendix 4.9]

Decision Tree



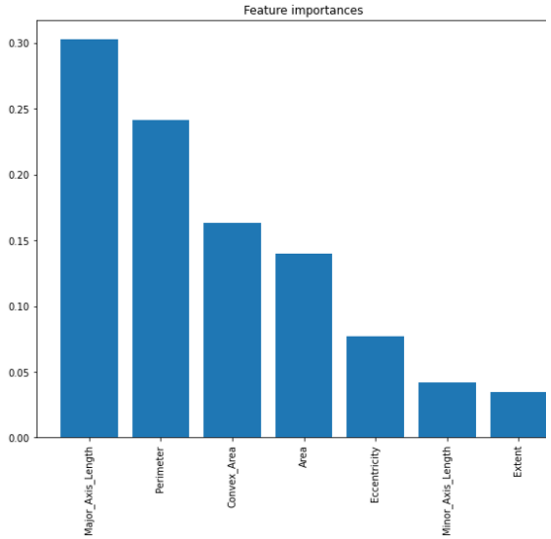
[Appendix 4.10]

Accuracy of Decision Tree

```
In [92]:
    ...:     accuracy_scores.append(accuracy_score(y_test, y_pred))
    ...: avg_accuracy = sum(accuracy_scores) / len(accuracy_scores)
    ...: print(f"Average accuracy score: {avg_accuracy}")
    ...: importance_table = pd.DataFrame({
    ...:     'Feature': X.columns,
    ...:     'Importance': rfc.feature_importances_
    ...: }).sort_values(by='Importance', ascending=False)
    ...: print(importance_table)
    ...: y_pred = rfc.predict(X_test)
Average accuracy score: 0.916010498687664
            Feature  Importance
2  Major_Axis_Length    0.294844
1          Perimeter    0.277092
5       Convex_Area    0.138832
0               Area    0.134462
4       Eccentricity    0.081114
3  Minor_Axis_Length    0.043366
6             Extent    0.030290
```

[Appendix 4.11]

Random Forest




```
    ...:     # Compute the accuracy score for this fold
    ...:     accuracy_scores.append(accuracy_score(y_test, y
    ...: avg_accuracy = sum(accuracy_scores) / len(accuracy
    ...: print(f"Average accuracy score: {avg_accuracy}")
    ...: importance_table = pd.DataFrame({
    ...:     'Feature': X.columns,
    ...:     'Importance': rfc.feature_importances_
    ...: }).sort_values(by='Importance', ascending=False)
    ...: print(importance_table)
    ...: y_pred = rfc.predict(X_test)
erage accuracy score: 0.9220472440944881
            Feature  Importance
Major_Axis_Length    0.322022
        Perimeter    0.287559
       Convex_Area    0.137251
             Area    0.103527
     Eccentricity    0.079068
Minor_Axis_Length    0.040962
           Extent    0.029610
```

[Appendix 4.12]

SVM

```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100],
                         'kernel': ['linear', 'poly', 'rbf']},
             scoring='accuracy')

In [21]: print('Best hyperparameters:', grid_search.best_params_)
    ...: print('Best accuracy:', grid_search.best_score_)
Best hyperparameters: {'C': 0.1, 'kernel': 'linear'}
Best accuracy: 0.9295065033623542
```

```
   ...: # Evaluate the performance of the SVM classifier on the test data
   ...: acc = accuracy_score(y_test, y_pred)
   ...: prec = precision_score(y_test, y_pred, average='weighted')
   ...: rec = recall_score(y_test, y_pred, average='weighted')
   ...: f1 = f1_score(y_test, y_pred, average='weighted')
   ...:
   ...: print('Accuracy:', acc)
   ...: print('Precision:', prec)
   ...: print('Recall:', rec)
   ...: print('F1 score:', f1)
Accuracy: 0.931758530183727
Precision: 0.9321594217634598
Recall: 0.931758530183727
F1 score: 0.9316083766016912
```

Model	SVM
Accuracy	0.93158
Precision	0.9322
Recall	0.93176
F1-Score	0.9316

[Appendix 4.13]

Accuracy of the KNN classification

```
In [4]: # Print the model score on the test data using GridSearchCV score method
   ...: print('Test accuracy: %.3f' % clf.score(X_test, y_test))
Test accuracy: 0.937
```

[Appendix 4.14]

Confusion Matrix of the decision tree model

	Actual Cammeo (CAMMEO)	Actual Osmancik (OSMANC)
Predicted Cammeo (CAMMEO)	456	62
Predicted Osmancik (OSMANC)	33	593
Total	489	655

[Appendix 4.15]

Accuracy of the decision tree model

	Accuracy
Decision Tree (with SAS EM)	91.70%

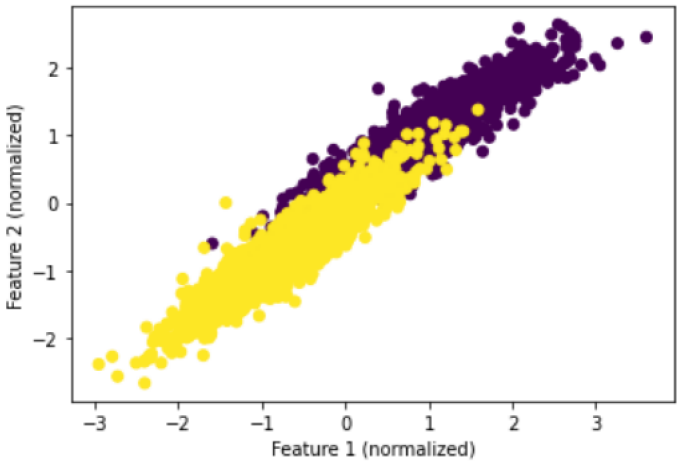
[Appendix 4.16]

Summary table of the accuracy for the rice dataset

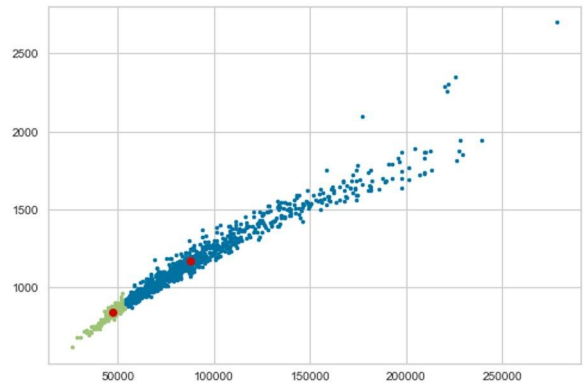
Method	Accuracy
Decision Tree	91.6%
Decision Tree (with SAS EM)	91.69%
Random Forest	92.2%
SVM	93.16%
K-NN	93.7%

[Appendix 4.17]

Distribution of the rice dataset in scatter plot

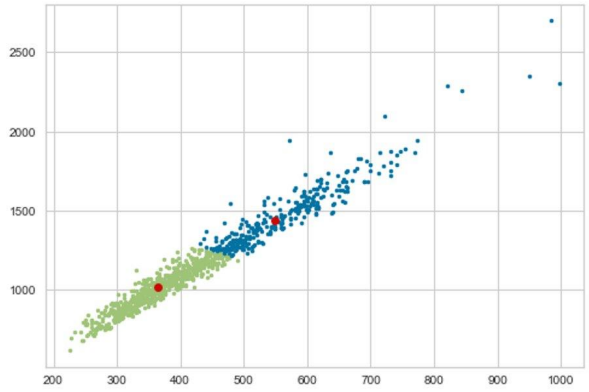


[Appendix 4.18]



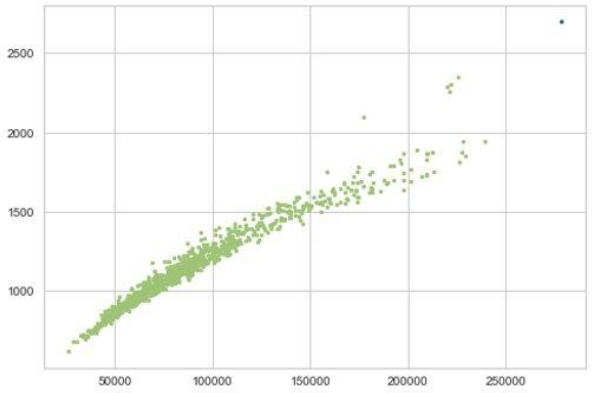
		Cluster Labels	
		K	B
True Cluster Lables	K	9	441
	B	126	324

k-medians [Appendix 5.1]



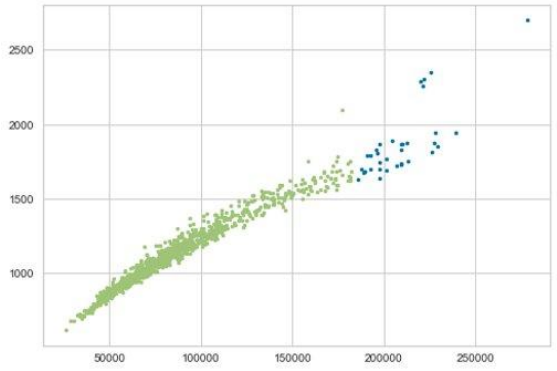
		Cluster Labels	
		K	B
True Cluster Lables	K	256	185
	B	447	3

k-means [Appendix 5.2]



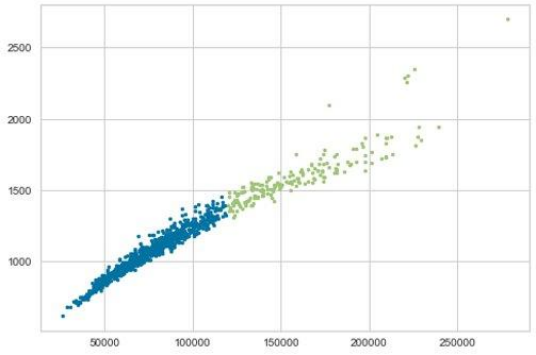
Single link		Cluster Labels	
		K	B
True Cluster Lables	K	449	1
	B	450	0

Hierarchical Clustering - single link [Appendix 5.3]



Complete link		Cluster Labels	
		K	B
True Cluster Lables	K	417	33
	B	449	1

Hierarchical Clustering - complete link [Appendix 5.4]



Average link		Cluster Labels	
		K	B
True Cluster Lables	K	175	275
	B	2	448

Hierarchical Clustering - average link [Appendix 5.5]

VI Veriguide Report