Data Modelling – Using Python (Jupyter Notebook)

Pranav Dev Kottimukkalur Ramasubramanian

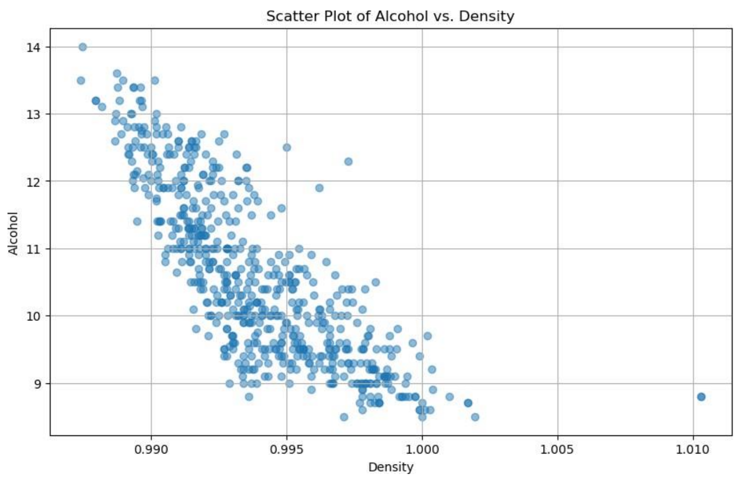
Task 1: Data Preparation and Analysis

Task 1.1

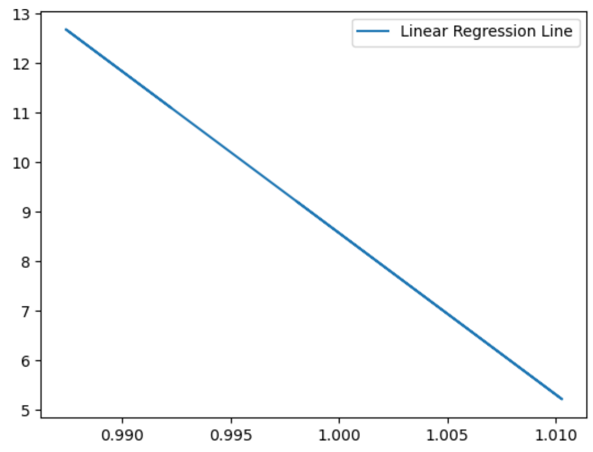
The given dataset 'A2data.csv' has been imported into Jupyter Notebook using the pd.read_csv(). There are 4781 rows and 12 columns. The column names have been displayed using the .column. The first 10 observations have been displayed using the .head(10) function for getting to know the overview of the data. For the next step of data exploration, the data is been checked for any missing values. If there are any, they are dropped using the dropna(). Firstly, the datatypes of the dataframe are found using .dtypes. There are some empty cells in 'residual sugar' and 'density' columns. They have been replaced using NaN values and dropped in the next step. Now, the datatypes for the two columns 'residual sugar', 'quality' and 'density' have been changed from object to float. After that, the number of rows is checked again and now found to be 4750. Finally, a random sample of 600 rows have been made and written to .csv file and named as 'A2RandomSample.csv'.

Task 1.2

The csv has been loaded as random sample. Now the scatter plot for Density vs Alcohol is taken and generated using matplotlib.pyplot. This has been visualised using scatter plot. While looking into the plot, we are able to observe a downward trend. This shows a negative correlation between alcohol content and density of wine. This means that as the alcohol content increases, the density of the wine decreases. Different wine qualities exhibit different alcohol content.

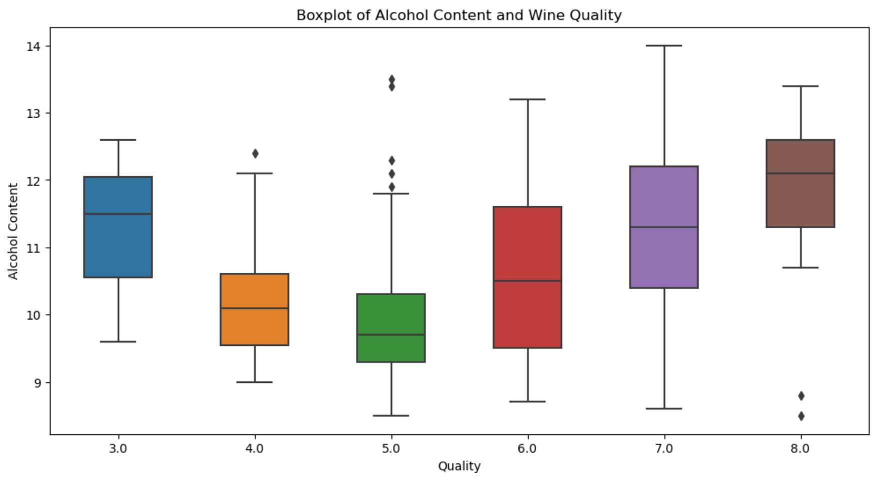


A simple linear regression model has been built using LinearRegression class from the sklearn (Scikit-Learn) library. Linear regression helps to determine the relationship between alcohol content and density in the wine dataset. It helps to provide insights about how alcohol content can be predicted from the wine density based on this linear model. Here density is set as independent variable and alcohol is set as dependent variable. Similarly, the linear regression line has also been visualised.



The co-efficient and intercept are found.
Intercept (334.7451): The intercept is the value of alcohol content when the density is set to zero. However, it is not practically possible, since density cannot be zero in wine.
Coefficient for Density (-326.1727): This coefficient shows that alcohol content is projected to fluctuate by about -326.1727 units for each unit change in density, demonstrating an inverse relationship between the two variables.

Task 1.3



The boxplot shows the quality starting from quality 3 (based on the random sample). It has alcohol content as 11.5. This value decreases in the next quality 4 and 5. Then an upward trend is observed in the alcohol content. Still the alcohol content is lesser in qualities 6 and 7 compared to quality 3. Quality 8 has alcohol content of 12. There is no specific trend observed in this graph. This is not practical. However, wine quality is influenced by multiple factors, not just alcohol content and we need more variables to prove it.

Task 2: Classification

Task 2.1

Decision Tree model has been selected for training and evaluating the data. Decision Tree classifier is a supervised machine learning model used for classification. Decision tree does not assume any specific data distribution unlike KNN (K- nearest neighbours) which can be sensitive to the distribution of data points. This makes this model more robust. This is also capable of capturing more complex, non-linear relationships in data (quality may also show non-linear relationships with some characteristics). This code effectively selects the model, trains and evaluates its performance using the metrics accuracy (ratio of correctly predicted instances to the total number of instances), precision (accuracy of positive predictions) recall (model's ability to identify all relevant instances) along with the F1-score (harmonic mean of precision and recall) as shown below:

Accuracy	0. 4888888888888889
Precision	0. 488967883967884
Recall	0. 4888888888888889
F1-score	0. 4873170413365272

In a general perspective, the model is capable of good predictions but also has room for improvement. It is able to predict nearly half correct. Here 30% of the data has been used as the test set. Decision tree has many parameters like maximum depth, minimum samples per leaf and also criterion for splitting nodes, while in KNN, optimal k is the preferred one. Due to wide availability of parameters, decision tree is chosen for this data and the tuning is shown in the next step.

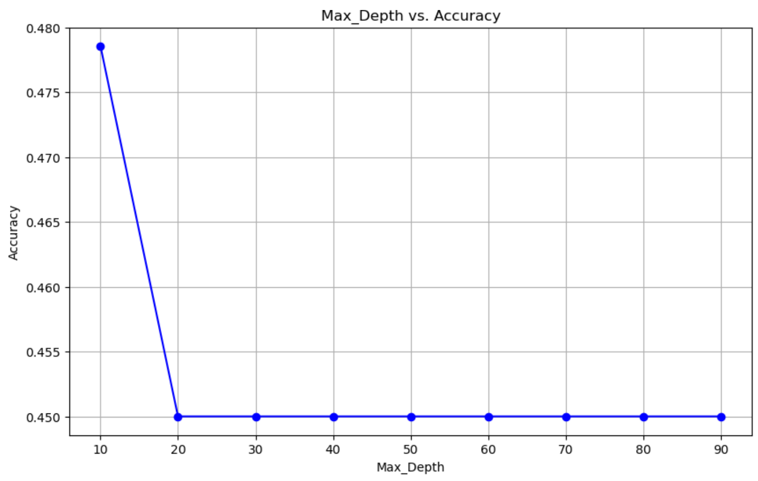
Task 2.2

We have seen results of the model. For the task, the parameter chosen for studying the impact is max_depth. This parameter depicts the maximum depth of the tree during model training. Selection of an appropriate max_depth values is important as it helps to improve interpretability with limited depth. This is also helpful in faster training, prediction, robustness and avoiding complexity of the model. The values have been provided below:

Maximum Depth	Accuracy
None	0.4500
10	0.4786
20	0.4500
30	0.4500
40	0.4500
50	0.4500
60	0.4500
70	0.4500
80	0.4500

90	0.4500

When the maximum depth is set to unlimited(none), the model achieves accuracy of 0.4500. It tends to overfit the data. When is it set to 10, the accuracy increases to 0.4786. Increasing it any further does not do any good and makes the accuracy 0.4500. This shows that a Decision Tree with maximum depth of 10 is recommended for this model to perform well and to classify approximately 47.86% of the test samples correctly.



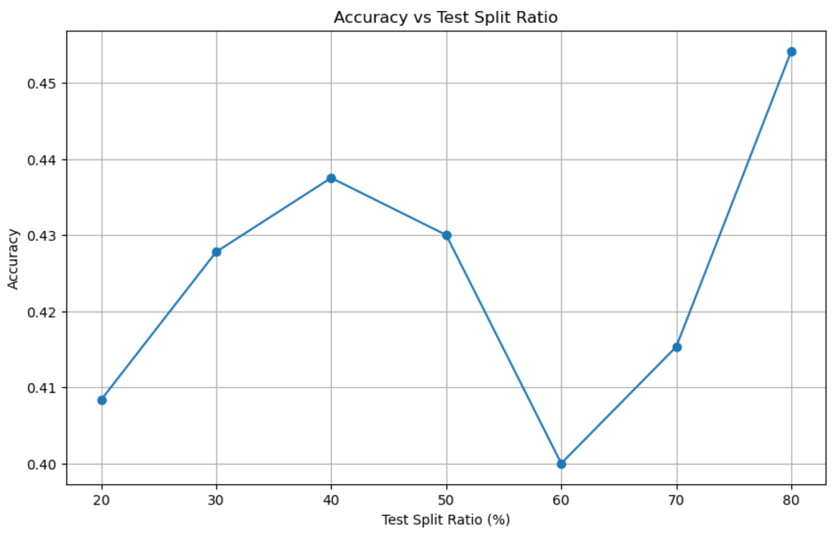
This line graph depicts the Max_depth vs the accuracies.

Task 2.3

The train-test split ratio also known as the data split ratio is a crucial parameter that determines how a dataset is divided into training and testing set. This ratio denotes the proportion of the data that shall be allocated to training and testing. Here it has been asked to evaluate these ratios: 20:80, 30:70, 40:60, 50:50, 60:40, 70:30, 80:20. The results after evaluating them has been shown below:

Split Ratio	Accuracy
20%	0.4083
30%	0.4278
40%	0.4375
50%	0.4300
60%	0.4000
70%	0.4153
80%	0.4542

Here, the choice for the best train/test split ratio is the 80:20 split ratio with an accuracy of 45.42%. Overall, the 80:20 train/test split ratio provides a reasonable compromise between model performance and efficient use of the available data. It ensures that the model is well trained and to provide the classification.



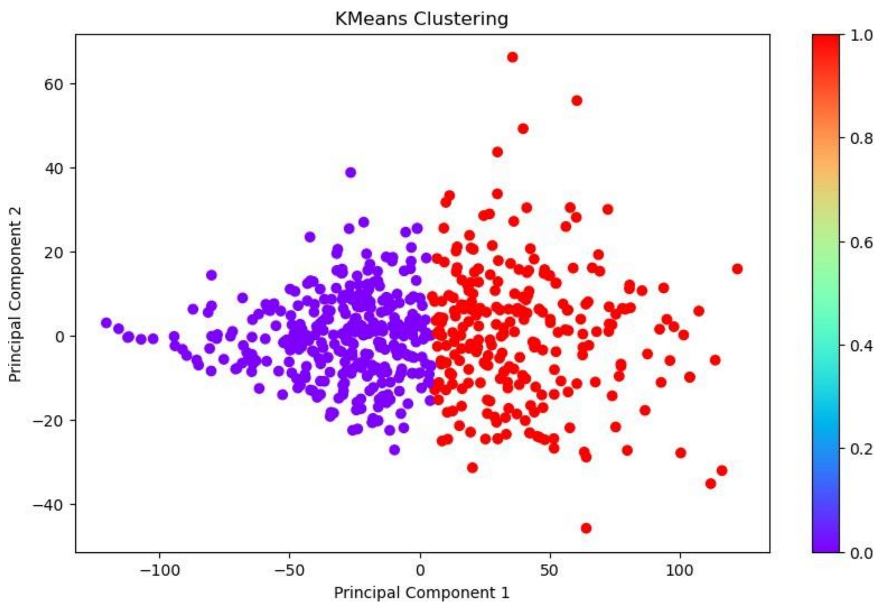
This graph depicts the split ratio vs accuracies.

Task 3: Clustering

K-means is a simple and easy clustering algorithm. It assigns data points to clusters by using the Euclidean distance. K-means is computationally efficient compared to DBSCAN and is also versatile.

Task 3.1

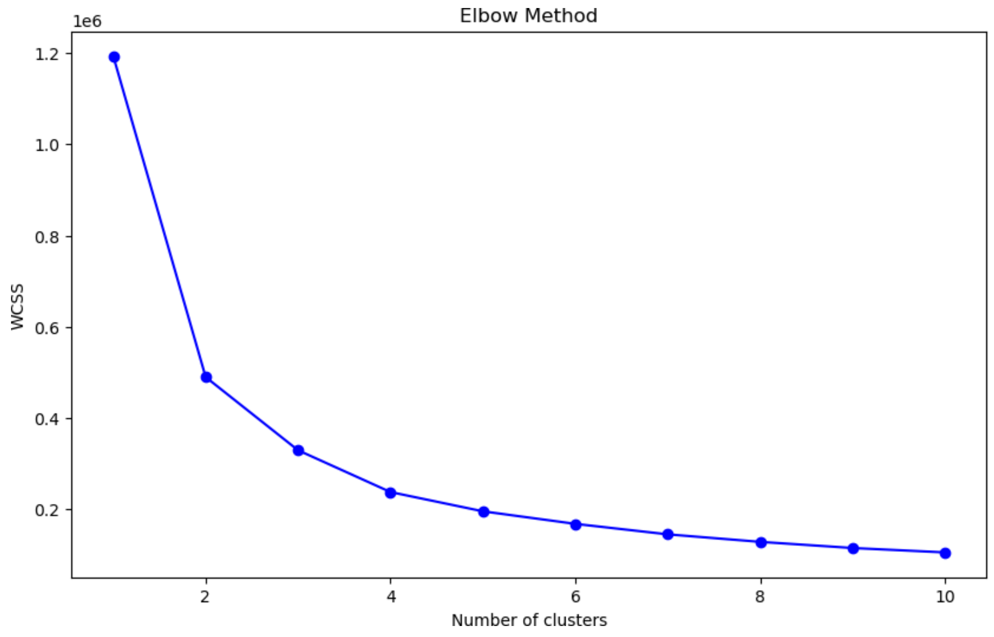
The code here applies Principal Component Analysis to reduce the dimensionality of the data by breaking down into two principal components. After reducing, K-means is applied to identify clusters in this reduced data. The optimal_k is set to 2.



The silhouette score for 2 clusters is calculated and found out to be **0.49256464693113655**. This is calculated to evaluate the quality of clustering results. It typically ranges from -1 to 1. This score indicates a reasonable level of clustering. The optimal number of clusters is achieved as 2 and is justified in the next steps.

Task 3.2

The code iterates through different values of k (number of clusters) from 1 to 10. For each 'k', it applies the K-means clustering and also calculates the WCSS (Within Cluster Sum of Squares) which is the sum of squared distances between data points and their respective cluster centers. The elbow point is determined as the optimal k which is the optimal number of clusters. It is the point where increasing the number of clusters would not result in a substantial reduction in the WCSS.



Here, the elbow point (optimal_k) is found to be 2. This determines that the optimal number of clusters is 2. The value of WCSS decreases with increase in the number of clusters. In the graph, after 2 there is no significant decrease in WCSS and decreases only gradually. This proves that 2 should the optimal k. This can also be proved by calculating the silhouette scores of different clusters as below. Score for 2 clusters is the highest.

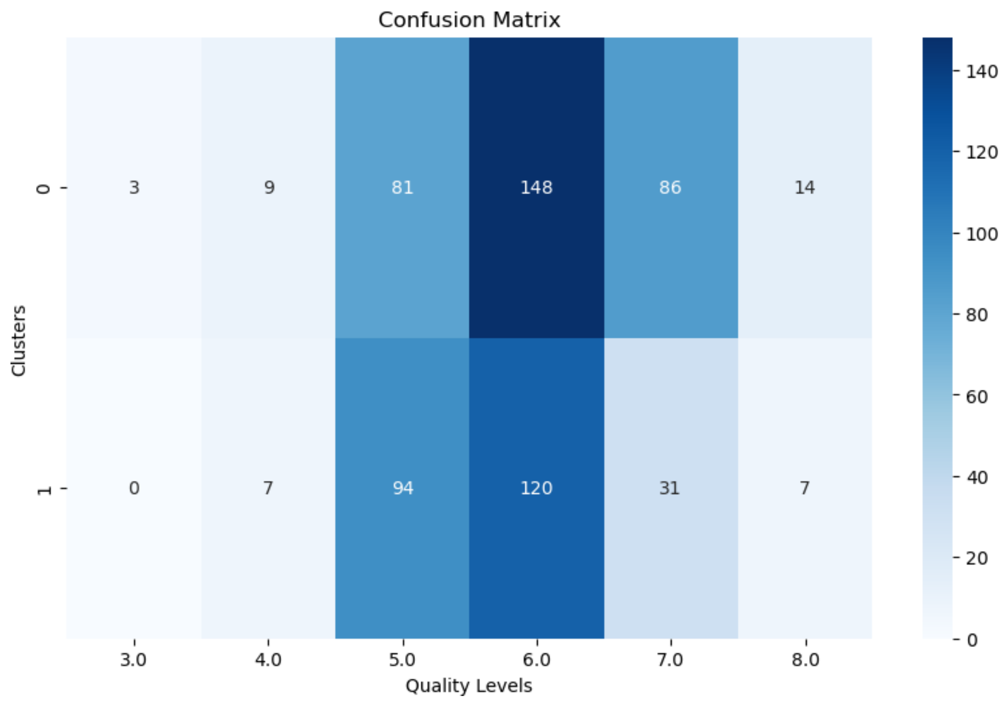
Clusters	Silhoutte Score
2	0.4926
3	0.4026
4	0.3836
5	0.3419
6	0.3373
7	0.3228
8	0.3237
9	0.3287
10	0.3158

Task 3.3

A new data frame is created here to consolidate the cluster assignments and true quality labels from the original data. A copy of the original data frame 'wine_without_quality' is created which contains all the features except quality. The first columns cluster contains the cluster assignments obtained from the K-means clustering. The second column true_quality contains the true quality labels from the original data.

Then the confusion matrix is constructed to assess how well the clustering aligns with the actual quality levels of the samples. The clusters and quality levels are created in a data frame and a cross-tabulation is created to capture the relationship between them.

Each cell in the matrix indicates the number of wine samples that fall into a particular combination of clusters and true quality level. These counts help to assess how accurately the clustering has grouped wine samples based on their actual quality.



Cluster 0 contains wine samples mostly in quality 6 (148 samples), 7 (86 samples) and 5 (81samples) and lesser samples from 8 (14 samples), 4 (9 samples) and 3 (3 samples).

Cluster 1 contains samples mostly in quality 6 (120 samples), 5 (94samples) and 7 (31 samples) and lesser samples in 4 (7 samples) and 8 (7 samples) too. Quality 3 has no samples.

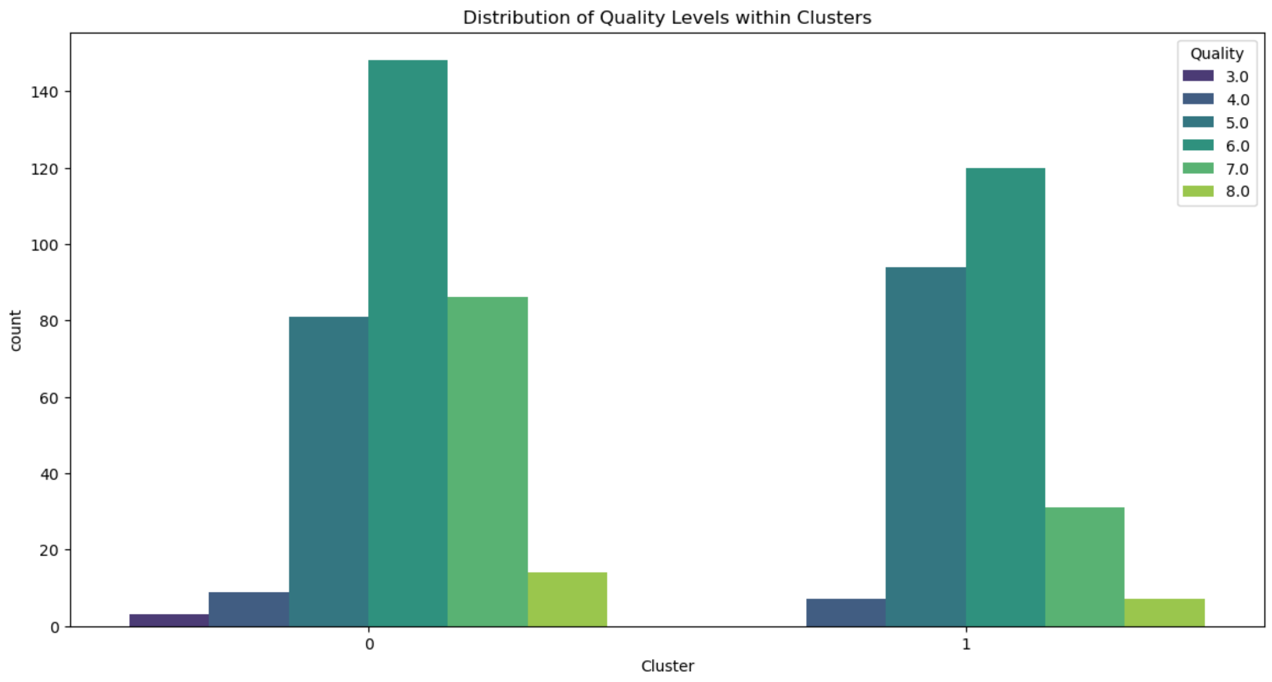
Observations:

In summary, Cluster 0 captures wine samples across a broader quality spectrum while Cluster 1 is more focused on the moderate to good-quality wines. And also, cluster 1 shows the maximum of good quality wines (quality 8) and low quality wines (quality 3).

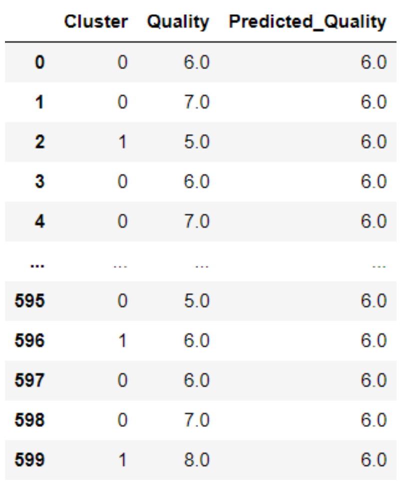
The most dominant wine quality in each cluster has been shown below:

Cluster	Dominant Quality
0	6
1	6

Below here is the graph which also proves that cluster 0 is more widespread.



A table is created to show the comparison between the predicted quality levels alongside the existing clusters and quality levels. This allows us to understand and compare the quality levels based on the clustering to the actual quality levels and analyse how well the clustering model aligns with the true quality labels.



	Cluster	Quality	Predicted_Quality
0	0	6.0	6.0
1	0	7.0	6.0
2	1	5.0	6.0
3	0	6.0	6.0
4	0	7.0	6.0
...	...	...	...
595	0	5.0	6.0
596	1	6.0	6.0
597	0	6.0	6.0
598	0	7.0	6.0
599	1	8.0	6.0

Furthermore, to explore the accuracy of the model, it is checked for matching by comparing the column Quality and Predicted Quality and the average of accuracy has been taken.

Inference:

The average accuracy is 44.66% which implies that the model's predictions match the actual quality levels with an accuracy of about 45%. The model's performance in predicting the wine quality is moderate but not exceptionally high. The model has predictive capabilities but needs improvement.

Conclusion:

In this comprehensive analysis of wine quality classification and clustering, we embarked on a journey to understand and predict the quality of wines based on various attributes and features. Our exploration involved data preprocessing, model selection, parameter tuning, and thorough evaluation. The findings from this analysis have practical implications for wine quality prediction and clustering. Further improvements in model accuracy and cluster interpretability can be explored.

In conclusion, this multifaceted analysis of wine data showcased the power of data preprocessing, modelling, and evaluation techniques. It underscores the importance of selecting the right models and parameters, and it highlights the potential for valuable insights and applications in the realm of wine quality assessment and also room for improvement.

Libraries used:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import random
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
classification_report
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.model_selection import cross_val_score, StratifiedKFold
```

References:

RMIT – COSC2670 Modules 1 to 8

Accessed: Oct. 08, 2023. [Online]. Available: https://e2eml.school/how_modeling_works_4#:~:text=To%20honestly%20split%20the%20data,us%20a%20more%20realistic%20assessment

J.-C. Chouinard, 'Train Test Split in Python (Scikit-learn Examples)', JC Chouinard. Accessed: Oct. 08, 2023. [Online]. Available: https://www.jcchouinard.com/train-test-split/

J. Verma, 'Split Training and Testing Data Sets in Python - AskPython'. Accessed: Oct. 08, 2023. [Online]. Available: https://www.askpython.com/python/examples/split-data-training-and-testing-set

'pandas - Python Data Analysis Library'. Accessed: Oct. 08, 2023. [Online]. Available: https://pandas.pydata.org/

'scikit-learn: machine learning in Python — scikit-learn 1.3.1 documentation'. Accessed: Oct. 08, 2023. [Online]. Available: https://scikit-learn.org/stable/index.html

‘1.10. Decision Trees’, scikit-learn. Accessed: Oct. 08, 2023. [Online]. Available: https://scikit-learn/stable/modules/tree.html

‘sklearn.model_selection.GridSearchCV’, scikit-learn. Accessed: Oct. 08, 2023. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

‘2.3. Clustering’, scikit-learn. Accessed: Oct. 08, 2023. [Online]. Available: https://scikit-learn/stable/modules/clustering.html

‘sklearn.metrics.silhouette_score’, scikit-learn. Accessed: Oct. 08, 2023. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.metrics.silhouette_score.html

‘Matplotlib — Visualization with Python’. Accessed: Oct. 08, 2023. [Online]. Available: https://matplotlib.org/

‘sklearn.decomposition.PCA’, scikit-learn. Accessed: Oct. 08, 2023. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html

K. Arvai, ‘kneed: Knee-point detection in Python’. Accessed: Oct. 08, 2023. [Online]. Available: https://github.com/arvkevi/kneed

----------------------------------- *END OF REPORT* ---------------------------------------