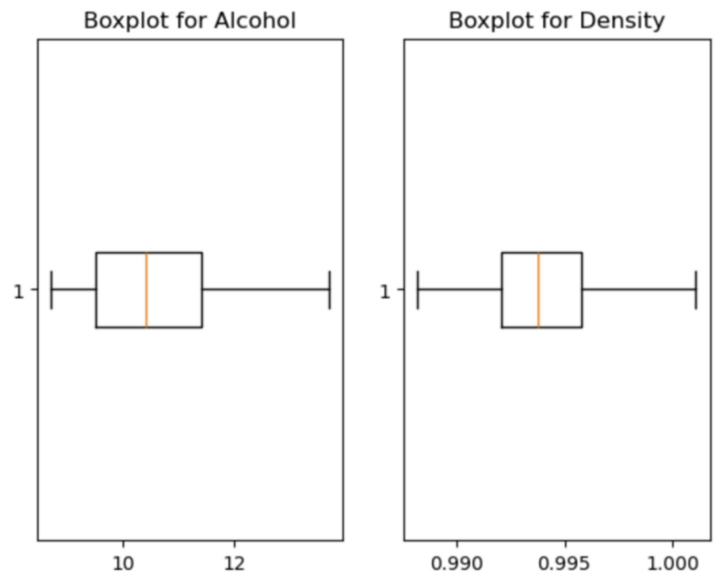
Assignment 2: Data Modelling

Deepthi Suresh (S3991481)

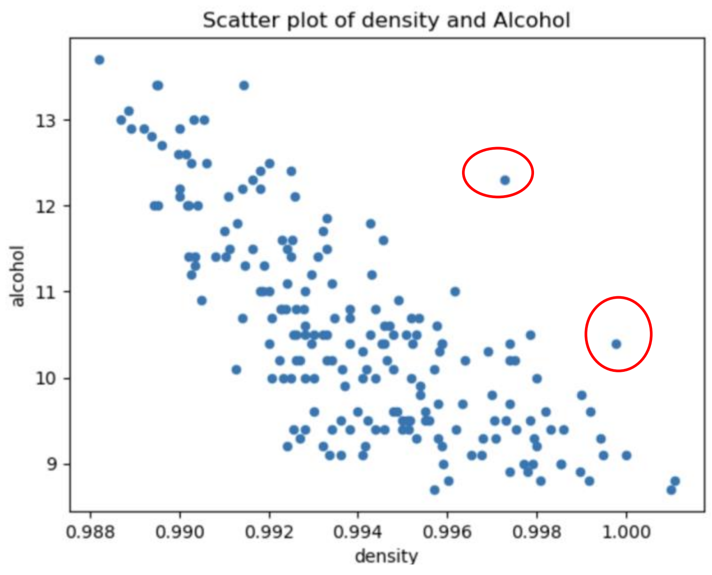
Task 1: Regression

Data Preparation

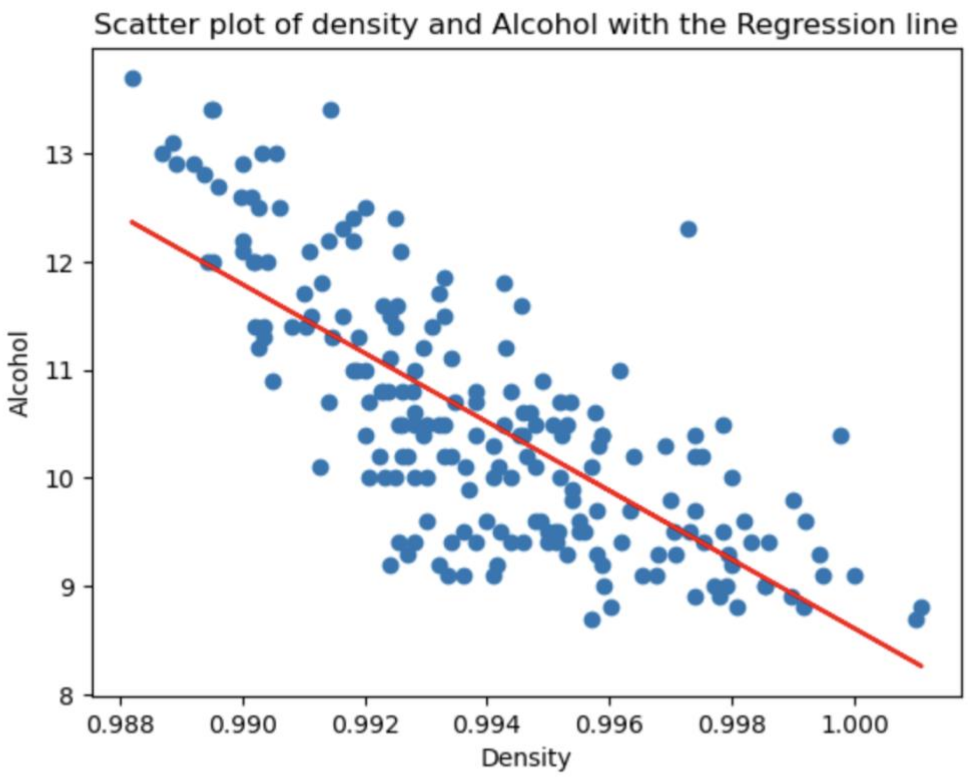
Once the data has been loaded into Jupyter Notebooks, it was noticed that a few of the columns were loaded as object. These have been converted into numeric for easier processing. There were also missing values in the dataset in all columns except fixed acidity, alcohol and quality. The rows with missing values have been dropped as it would affect the any modelling that we perform, besides since the dataset has a huge number of observations, the rows shouldn't impact the overall proportions of the data. Using the cleaned data, a random sample of 200 observations are considered to create the Linear Model. For this modelling, alcohol and density variables are considered. A box plot is used to check for any outliers. The boxplot below shows that there are none in density and alcohol.



Simple Linear Model



A scatter plot is used to describe the relationship between alcohol and density. It can be noted that there is a strong negative correlation between alcohol and density, which means that as density increases the alcohol content of the wine also decreases. It can also be noted that there could be a potential outlier (marked in red) in the relationship between density and alcohol, though none were found in the individual variables.



A linear regression model was developed to study the relationship between the density and alcohol and predict the density values. In this model, alcohol is the dependent variable, and density is the independent variable. The X values have been standardised as alcohol and density are in two different units of measurement, and this standardised value is used for further training and predictions. To evaluate the model, the data has been split into training and testing data set with 25% of the data considered as a testing dataset. The model returned a R square value of 0.68 and Mean Squared Error value of 0.52. The R-squared value , explains the proportion of the variance in the dependent variable (alcohol percentage) that can be explained by the independent variable (density). In this case, R-squared value of 0.68 means that approximately 68% of the variability in alcohol percentage can be explained by the changes in density. The Mean Squared Error is a measure of the average squared difference between the predicted values and the actual values of the dependent variable i.e. how big the average error of the prediction is. An MSE of 0.52 indicates that, on an average, the squared difference between the predicted alcohol percentages and the actual alcohol percentages is 0.52.

The model returned an intercept of 10.52 and a coefficient of -0.87, giving us a linear model of:

$$Y = aX + B$$

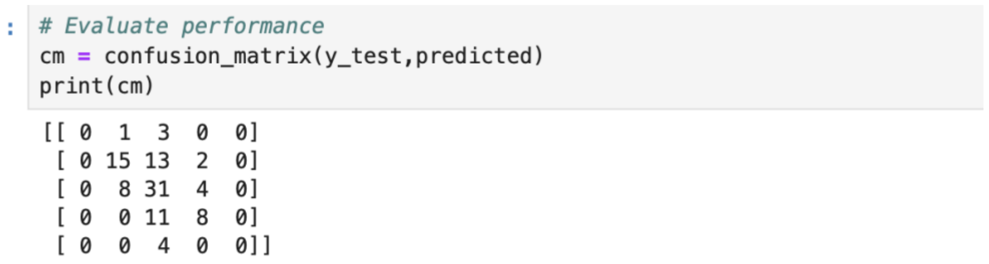
$$\text{i.e Alcohol} = 10.52 + -0.87 * \text{Density}$$

The intercept of 10.52, shows the alcohol content in the wine when the density is 0 i.e. if the density of wine were zero, the model predicts an alcohol percentage of approximately 10.52%. The coefficient in a linear model indicates how much the dependent variable changes with the change in the independent variable. In this case the alcohol content in predicted to decrease by -0.87% for every unit increase in density. This further confirms the relationship that we found earlier in the scatter plot as well indicating that wines with higher density tend to have lower alcohol content..

Task 2: Classification

kNN

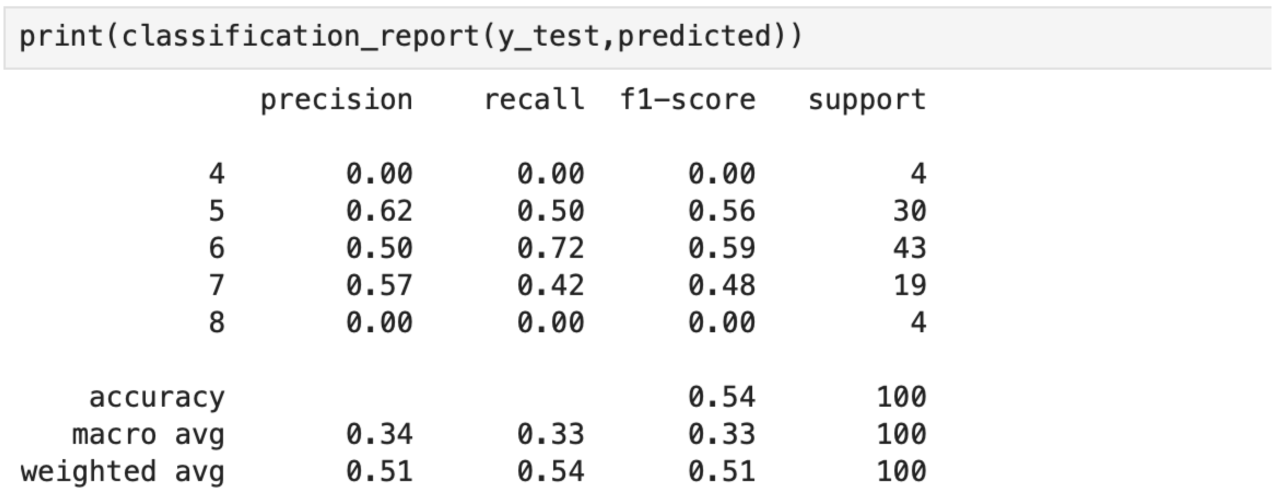
A sample of 500 observations have been selected from the dataset. In this task the objective is to classify the wine **quality** based on the other variables, which indicates that Quality is the target variable, and the rest of the columns are the features. To execute the KNN classifier, the data is split into training and testing data. Since there is an imbalance in the classes, stratified sampling is used to maintain the class distribution between training and testing dataset. Since all the features have a different unit of measure, these features have also been standardised using the StandardScaler() function. As a rule of thumb, the initial k value is calculated as square root of the No of training data, which was 20. However, it is best to take the closest odd no, which resulted in the KNN to be run with a initial k of 21. The model was then used to predict on the test dataset. To evaluate the KNN model, Confusion matrix and classification report was extracted.



```
# Evaluate performance
cm = confusion_matrix(y_test,predicted)
print(cm)
```

```
[[ 0  1  3  0  0]
 [ 0 15 13  2  0]
 [ 0  8 31  4  0]
 [ 0  0 11  8  0]
 [ 0  0  4  0  0]]
```

The confusion matrix shows that there has been no correct predictions for wine quality 4 and 8, which could indicate that the model struggles to identify these classes which could be because of the class imbalances. Out of all the classes, class 6 seems to be most accurately predicted, followed by class 5.



```
print(classification_report(y_test,predicted))
```

```
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         4
           5       0.62      0.50      0.56        30
           6       0.50      0.72      0.59        43
           7       0.57      0.42      0.48        19
           8       0.00      0.00      0.00         4

    accuracy                           0.54       100
   macro avg       0.34      0.33      0.33       100
weighted avg       0.51      0.54      0.51       100
```

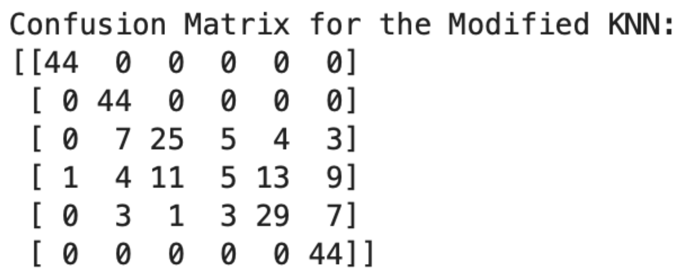
The classification report shows that the accuracy for this model is 54%, however since we know that the classes are imbalanced, the accuracy may not be the correct metric to evaluate the model. In terms of precision, classes 5,6 and 7 have a precision over 50% and in terms of recall this class have over 40% of recall. Since the dataset is imbalanced, F1 score might be a better metric as it is the harmonic mean of precision and recall, providing a balance between the two metrics. Out of all the classes, class 6 has the best performance with 0.59, followed by class 5 and class 7. Overall, the model has only 0.33 as its F1-score which is not a really good performance. Classes 4 and 8 still show 0 precision, recall, and F1-scores, indicating issues with these classes.

Modified kNN

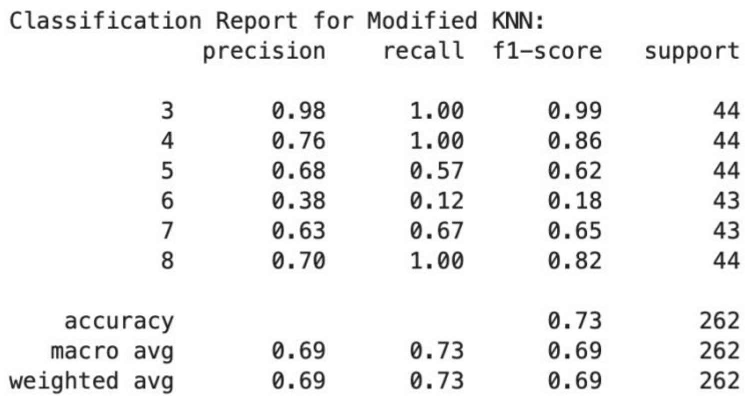
To improve the KNN classifier, it is important to identify the key parametrs of the model. In order to do this, GridSearchCV() function is used. This function conducts an exhaustive search over specified parameter values for an estimator to identify the most optimal value that provides the best score. The parameters given for the search are the number of neighbours (given as a range between 1 to 100 to consider the large dataset), weights ("uniform" and "distance"), "p"(range (1, 10)) and "metric" ("minkowski", "euclidean", "manhattan"). The Gridsearch returned the optimal parameters as k =23 (which is very close to our earlier k), weight = distance (distance ensures that closer Neighbours of a query point will have a greater influence than Neighbours which are further away. This can help with an imbalanced dataset such as the one used here.), and metric ='minkowski' and p=2. For a high dimensional data like this, a smaller p value normally results in better results.

Another method that has been used to improve the balance of the dataset is the SMOTE function. SMOTE() generates synthetic samples for the minority class to balance the class distribution. From the initial KNN model, it was quite clear that class 4 and class 8 had 0 precisions, leading to a poor model performance. SMOTE helps in improving the model performance by giving KNN a larger sample to learn from, and therefore reducing any bias towards the majority classes. Another issue that we faced in the KNN model was that all the features were of different units. To correct these issues, the features have been standardised using StandardScaler () function. StandardScaler standardizes the dataset by removing the mean and scaling it to unit variance. This is particularly useful as KNN is a distance-based ML algorithm, and therefore having a similar unit ensures that all features contribute equally to the distance calculations.

Based on these parameters the KNN model was trained and used for predictions on the testing dataset. The confusion matrix and the classification report for the Modified KNN model is given below:



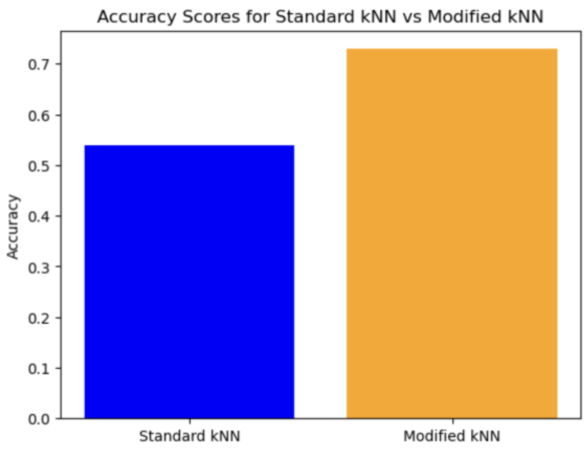
```
Confusion Matrix for the Modified KNN:
[[44  0  0  0  0  0]
 [ 0 44  0  0  0  0]
 [ 0  7 25  5  4  3]
 [ 1  4 11  5 13  9]
 [ 0  3  1  3 29  7]
 [ 0  0  0  0  0 44]]
```

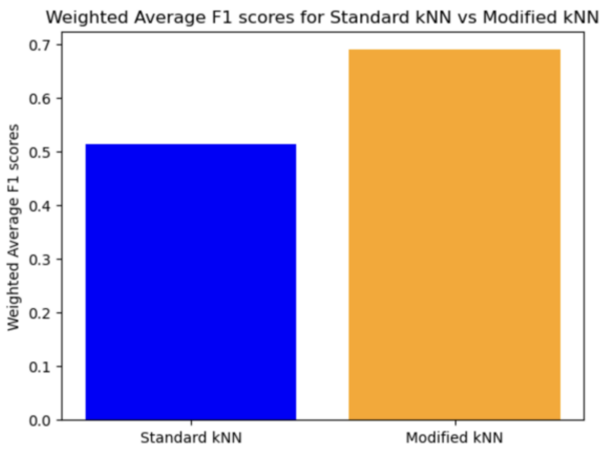


```
Classification Report for Modified KNN:
              precision    recall  f1-score   support

           3       0.98      1.00      0.99        44
           4       0.76      1.00      0.86        44
           5       0.68      0.57      0.62        44
           6       0.38      0.12      0.18        43
           7       0.63      0.67      0.65        43
           8       0.70      1.00      0.82        44

    accuracy                           0.73       262
   macro avg       0.69      0.73      0.69       262
weighted avg       0.69      0.73      0.69       262
```

It is easily noticeable from the confusion matrix that the data is more balanced, and the predictions are more accurately made for the classes compared to the Standard KNN model. The classification report shows that the model has an accuracy of 73 %. The precision of the model in correctly predicting class 3 is at a 98% followed by class 4 and class 8. Class 3,4 and 8 has a perfect recall, however the model struggled to positively identify the values in class 6 with a lowest recall of 0.12. F1 score shows the harmonic mean of precision and recall. Class 3,4 and 8 seems to have a strong performance while class 5 and 7 has a moderate performance. The model struggled with class 6. Overall, the model performs very well for classes 3, 4, and 8, with high precision, recall, and F1-scores. Class 6 is the most challenging, with very low precision, recall, and F1-score.

Comparison of the standard Knn and Modified Knn



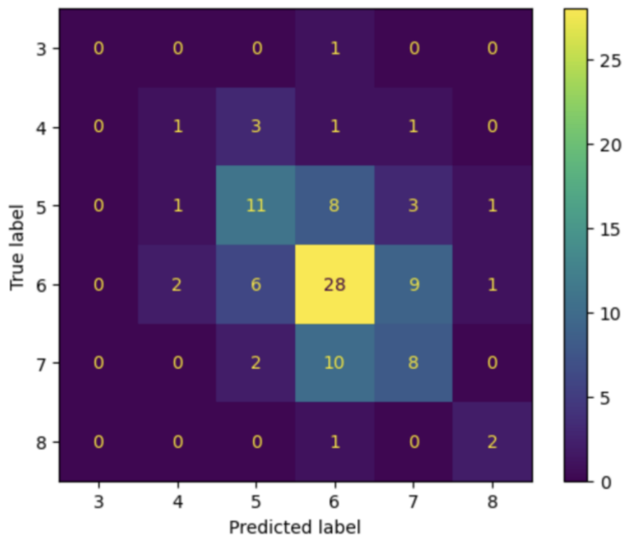


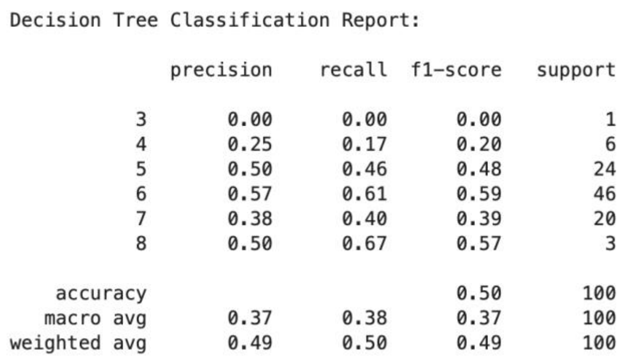
From the above classification report and confusion matrix of the standard kNN and the modified Knn, it is clear that the modified kNN has improved the performance of class 3,4 and 8 with improved precision, recall and support. The F1 scores have also significantly improved for all the classes except class 6 in modified kNN. The overall accuracy has increased to 0.73, a significant improvement over the 0.54 from the standard KNN model. As shown in the graph above, the weighted average F1 scores for the overall model has also improved from 0.33 to 0.69

In conclusion, it can be noted that the standard kNN model was improved by implementing a combination of techniques like SMOTE, StandardScaler, and hyper tuning the parameters of kNN. SMOTE helped in dealing with the imbalanced dataset by generating synthetic samples for the minority classes. By balancing the dataset, SMOTE improved the recall for minority classes like 3, 4 and 8, ensuring the model is more sensitive to detecting them. Standard Scaler helped in dealing with the bias that would arise from the larger range in the features by standardizing all the values. This ensures all features contribute equally to distance calculations in KNN. By normalizing the data, the model improved in accuracy from 54% to 73%, as the distances are now meaningful across all features. By using GridSearch for Hyperparameter Tuning, we found an optimised k value that provide a higher model score. The modified KNN also uses the Minkowski distance with distance-based weighting parameters to prioritise closer neighbours. Fine-tuning these hyperparameters improves accuracy and F1-scores, as seen in the confusion matrix and classification reports for the modified KNN.

Decision Tree (and Comparison)

Implementing an initial Decision tree (without tuning the parameters provided the confusion matrix and classification table below. The initial tree model best classifies class 6 and moderately classifies class 5 and 7 as seen in the precision, recall and f1-scores. However, the model performs very poorly for class 3 and 8, probably because of the imbalance in dataset. The overall accuracy of the dataset is 50% meaning that it only correctly classifies 5% of the observations and the weighted average F1 score is 0.49.





```
Decision Tree Classification Report:

              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.25      0.17      0.20         6
           5       0.50      0.46      0.48        24
           6       0.57      0.61      0.59        46
           7       0.38      0.40      0.39        20
           8       0.50      0.67      0.57         3

    accuracy                           0.50       100
   macro avg       0.37      0.38      0.37       100
weighted avg       0.49      0.50      0.49       100
```

In order to take into account, the imbalance in the class distribution in the dataset, ensemble method of decision tree was Implemented. According to Sckilearn “A random forest is a meta estimator that fits a few decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.” The Random Forest classifier is an ensemble method that uses multiple decision trees to improve prediction accuracy and control overfitting.

The RandomForestClassifier model was run with initial values and then the parameters were tuned to get a better fit using the Gridsearch function. This function returned the following as the best parameters based on the accuracy scores of the model. The accuracy score was used for easier comparison with the KNN. Best parameters: 'max_depth': 20 – which determines the maximum no levels in a tree

'max_features': 'sqrt' - This parameter determines the number of features to consider when looking for the best split. This means that the model will randomly select a number of features equal to the square root of the total number of features.

'min_samples_leaf': 1, 'min_samples_split': 8 – this means that the model allows leaf nodes to contain just one sample and a node must have at least 8 samples in order to be split further. This helps to reduces the chance of overfitting.

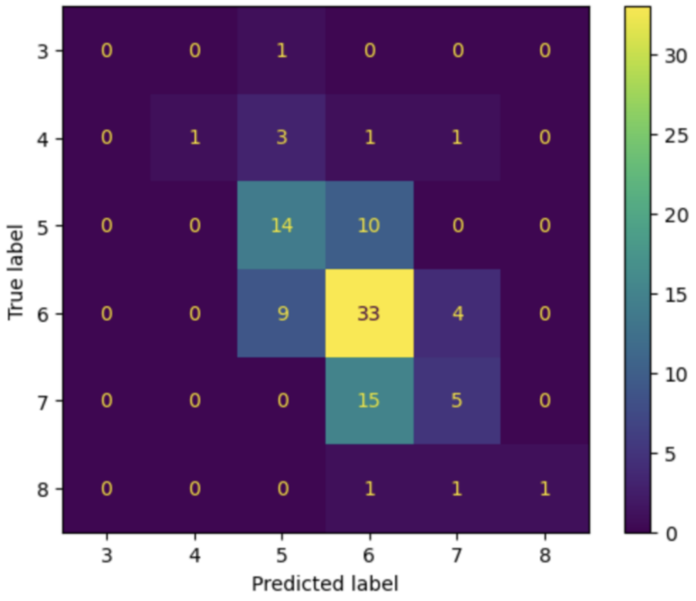
'n_estimators': 150 - This parameter indicates the number of trees in the forest.

By using class_weight='balanced', the Random Forest model can give more importance to minority classes, which can improve performance on those classes.

Evaluation of the Random Forest decision Tree

The random forest decision tree gives the classification report and confusion matrix given below.

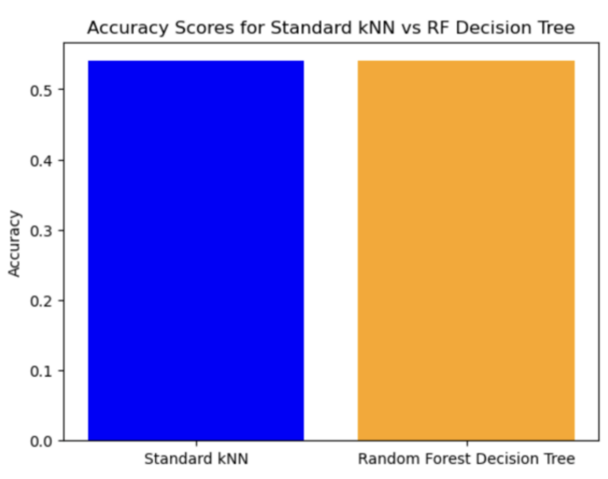
```
Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       1.00      0.17      0.29         6
           5       0.52      0.58      0.55        24
           6       0.55      0.72      0.62        46
           7       0.45      0.25      0.32        20
           8       1.00      0.33      0.50         3

    accuracy                           0.54       100
   macro avg       0.59      0.34      0.38       100
weighted avg       0.56      0.54      0.51       100
```

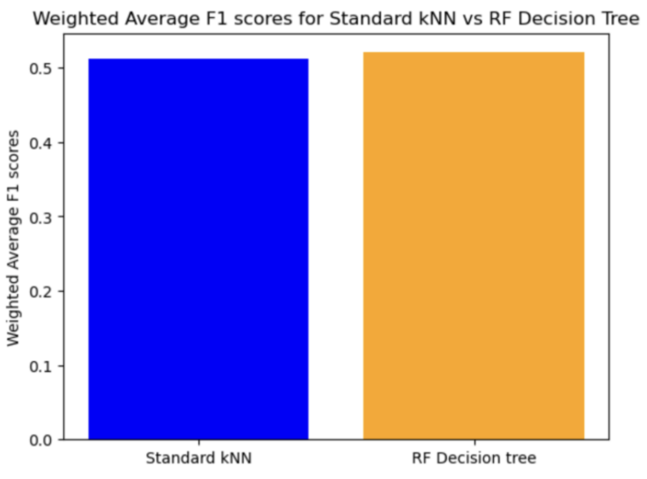


True \ Predicted	3	4	5	6	7	8
3	0	0	1	0	0	0
4	0	1	3	1	1	0
5	0	0	14	10	0	0
6	0	0	9	33	4	0
7	0	0	0	15	5	0
8	0	0	0	1	1	1

The confusion matrix shows that the model is not able to correctly predict class 3, this could be because there is only 1 observation. Class 6 has the best precision,recall and F1 score, inficating that the model is able to predict the observations in this class. The model made one correct prediction out of six instances, resulting in a high precision (1.00) but low recall (0.17). This suggests that while the model correctly identified one instance as class 4, it missed most of them. In class 7 the precision is 0.45 and recall is low at 0.25, indicating a lack of reliability for this class. Overall, the model achieves an accuracy of 0.54, meaning it correctly classified about 54% of the total instances and has a weighted average F1 of 0.51. The Random Forest classifier shows a mixed performance across different classes; however, it gives a better performance than the initial Decision tree model that was implemented indicating that this is a better model.

Comparison with kNN





The above graph shows the accuracy score and the weighted F1 score comparison between kNN and RF Decision Tree. It is interesting to note that in terms of accuracy, both the models offer very close accuracy whereas in terms of the F1 score (which would be prioritised in this case because of the imbalanced dataset) RF decision trees seem to be performing slightly better than the kNN model. KNN struggles significantly with classes 4 and 8, achieving 0.00 precision and recall for both, whereas Random forest struggles with class 3. The F1-scores for both classifiers reveal the same trends. KNN's highest F1-score is 0.59 for class 6, while Random Forest's is also 0.62 for the same class. Overall, Random Forest may be slightly better than KNN for this dataset due to its ability to reduce overfitting and handle class imbalance better, although both classifiers struggle significantly with certain classes.

Strengths and Weaknesses

KNN is a lazy learner and therefore was easy to implement. Identifying k using the square root formula was quite easy and was very close to the best parameter k. However, KNN model struggles with imbalanced datasets, often biasing predictions toward the majority class.

Modified KNN (Using SMOTE and Hyperparameter Tuning), helped mitigate the issue of imbalanced dataset to an extent, however it increases more complexity than kNN. Running the grid search did take some computing power, however once the best parameters were found it was easy to implement.

Decision trees are quite easy to visualise, however it was affected by the imbalanced dataset. Random Forest classifier was able to handle the imbalanced dataset better and provided a better model, however with the accuracy and F1 score that it returned, kNN might be a better choice as it was less complex and time consuming. Another strength of decision trees is that it can handle missing values effectively and do not require extensive data preprocessing. However, identifying the parameters even with Grid search takes a lot more time and are very prone to overfitting, which means the parameters need to be continuously tuned until a satisfactory result is achieved. Random forests mitigate the overfitting problem commonly associated with single decision trees by averaging the predictions of multiple trees. By combining multiple trees, the model tends to generalize better to unseen data. On the downside, training multiple trees can be computationally expensive, especially on large datasets.

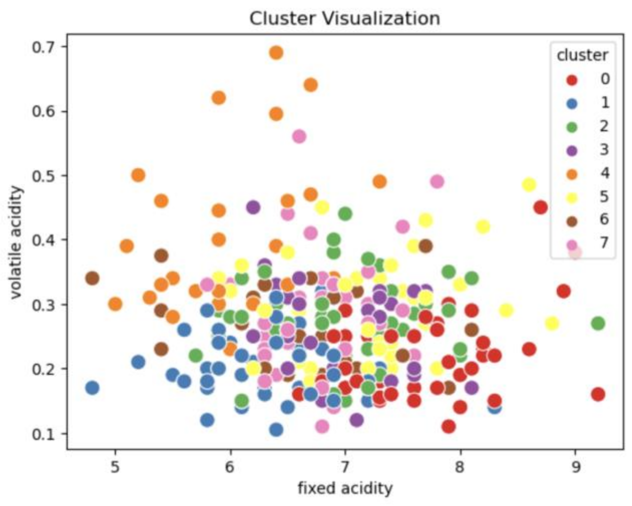
Task 3: Clustering

kMeans

Initially run of kMeans have been conducted on 8 clusters as there are 8 different quality rating. we also perform feature scaling to ensure that all features are on the same scale before applying the K-Means clustering algorithmSince the model is an unsupervised one, it has been evaluated based on Inertia (Within-Cluster Sum of Squares), Silhouette Score, Davies-Bouldin Index and Calinski-Harabasz Index. Inertia represents the "tightness" of the clusters. The lower the inertia, the better the clustering, as the data points are closer to their assigned cluster centroids. The silhouette score measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation). Higher silhouette scores indicate better-defined clusters. The Davies-Bouldin index measures the similarity of clusters. A lower Davies-Bouldin index means the clusters are compact and well-separated. Calinski-Harabasz Index evaluates the ratio of the sum of between-cluster dispersion to within-cluster dispersion. A higher score means the clusters are dense and well-separated.

The score for the initial run is given below:

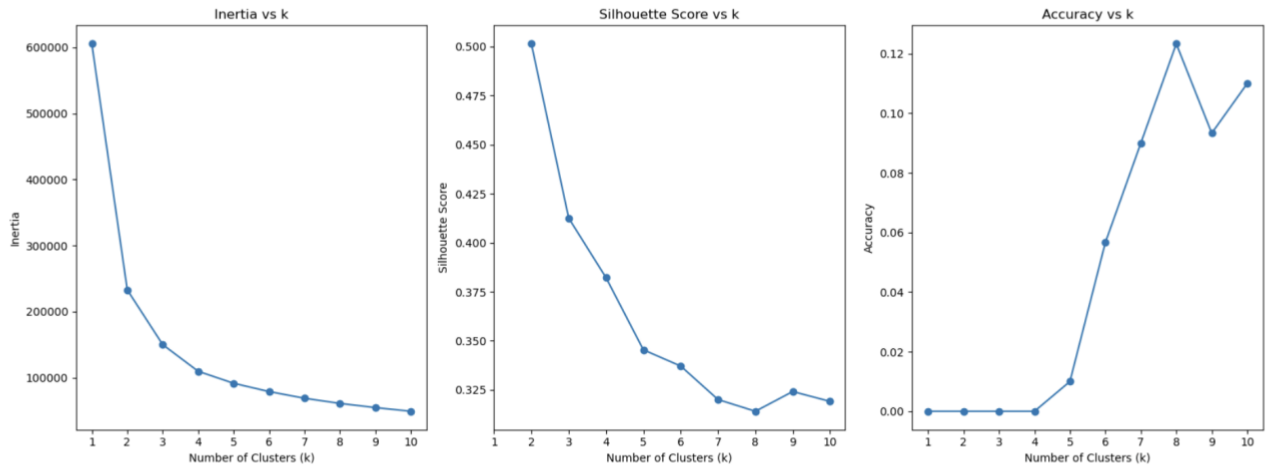
```
Inertia (WCSS): 1834.0990406111605
Silhouette Score: 0.15220539941141845
Davies-Bouldin Index: 1.7878260157378782
Calinski-Harabasz Index: 40.16320576588447
```



A silhouette score of 0.152 is relatively low, indicating that the clusters formed are not very distinct and might be overlapping. A Davies-Bouldin index of 1.788 suggests that the clusters are not well separated. The visualisation of fixed acidity and volatile acidity also shows that the clusters are not well defined.

In order to improve k, we have to consider the impact of the number of clusters (*k*) on the performance of the k-Means algorithm. Given below is the results of the evaluation. It can be noted that inertia decreases as the number of clusters increase. Ideally, lower inertia values suggest better-defined clusters. The inertia plot shows that there is a sudden drop in the inertia values when k is increased from 1 to 2. The decrease continues, but the rate at which the decrease occurs slows down significantly. In terms of the silhouette score, higher value indicates better-defined clusters, and the highest score is assigned to k =2, suggesting that the clusters are reasonably well-separated. As a rule of thumb, silhouette score above 0.5 is considered as a good indicator. Accuracy in terms of kmeans measures the proportion of observations that are correctly labelled. However, it is important to keep in mind that the kMeans clustering is a unsupervised learning model and the cluster labels do not directly correspond to the actual classes. The highest accuracy observed is 0.123 (or 12.3%) at k =8, indicating that only about 12.3% of the predicted clusters align with the true wine quality categories.

```
    k        Inertia  Silhouette Score  Accuracy
0   1  605549.007890               NaN  0.000000
1   2  233018.240325          0.501396  0.000000
2   3  150162.261482          0.412544  0.000000
3   4  109808.399462          0.382176  0.000000
4   5   91690.640388          0.345415  0.010000
5   6   79107.442488          0.337139  0.056667
6   7   68992.010654          0.320066  0.090000
7   8   61182.879073          0.314095  0.123333
8   9   54863.239196          0.324153  0.093333
9  10   49290.388115          0.319126  0.110000
```

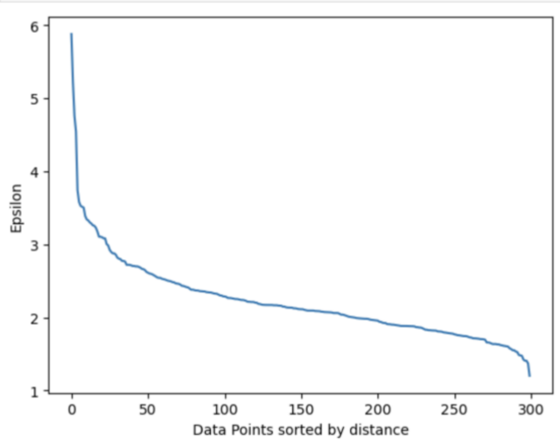


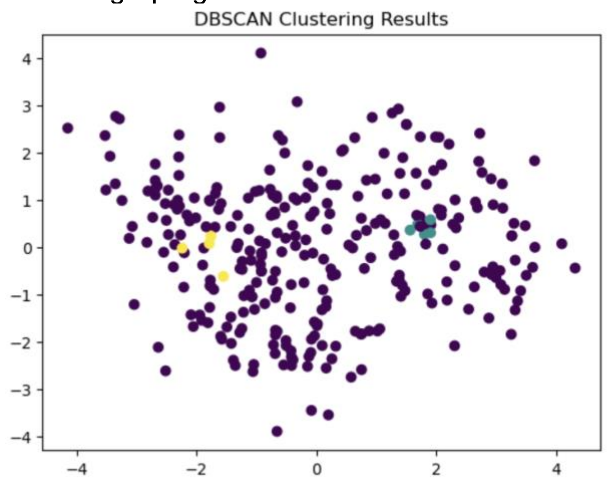
Based on the above scores, the best value seems to be at k=2 as it yields the highest silhouette score while maintaining a good inertia score. However, according to accuracy k should be 8. This might indicate that K-Means may not be the best clustering method for this dataset.

One of the major drawbacks for kMeans is that it assumes the data is spherical, which may not be true for many real-world datasets. Another challenge is the setting of the k values beforehand. If kk is too small, clusters may be merged; if too large, clusters may be split unnecessarily. Elbow method can used to identify the best k value. Another challenge pertaining to this dataset was that the cluster labels assigned by the model is not the same as the class label which can bring down the accuracy scores of the model. To avoid this more weightage has been given to other evaluation metrics while selecting the k value.

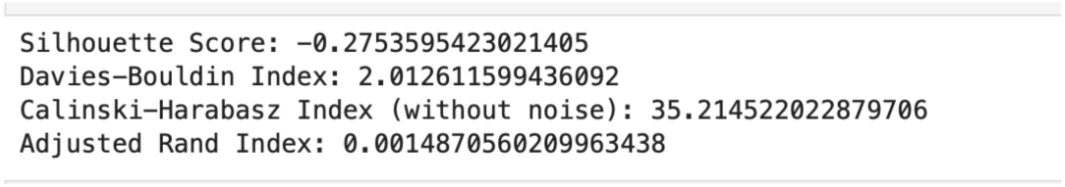
DBSCAN

DBSCAN has been implemented on the scaled features of the sample dataset. Scaling is essential in this dataset because of the different units of measure, therefore preventing features with larger ranges from dominating the clustering process. The DBSCAN was initialised with eps of 0.3 and minpoints of 3. K-distance graph was used to identify the optimal eps value. With multiple trial and errors, the n_neighbours were set as 4, resulting in the k-graph given below:





According to this graph the optimal eps is at the elbow which is at 3.5. Since the dataset has high dimensions the Manhattan distance has been used because it emphasizes differences along each dimension, making it less sensitive to outliers than Euclidean distance.

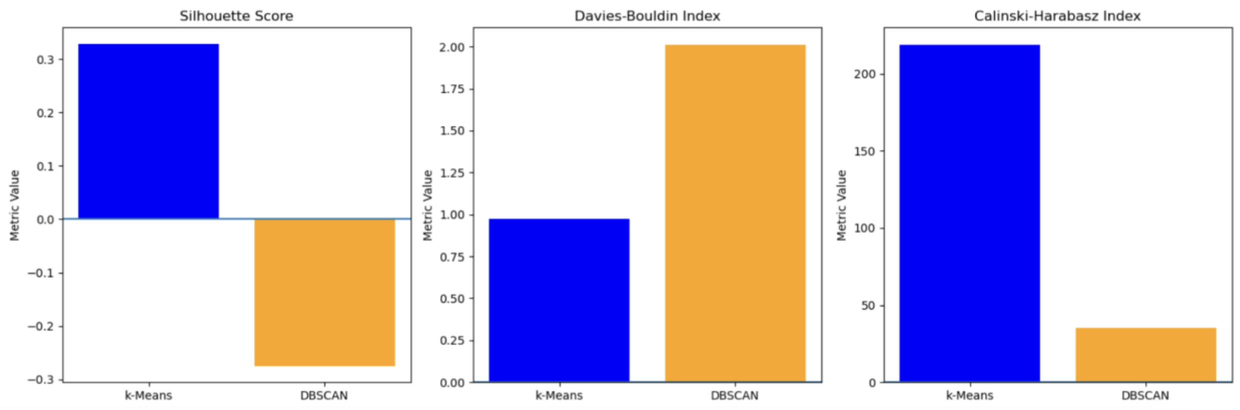


```
Silhouette Score: -0.2753595423021405
Davies-Bouldin Index: 2.012611599436092
Calinski-Harabasz Index (without noise): 35.214522022879706
Adjusted Rand Index: 0.0014870560209963438
```

The updated DBSCAN model has been evaluated using Silhouette Score, Davies-Bouldin Index, Calinski-Harabasz Index and Adjusted Rand Index. The Silhouette Score measures how similar an object is to its own cluster compared to other clusters. In this case, a Silhouette Score of -0.275 suggests that the clustering is poor and that many points are likely misclassified. a Davies-Bouldin Index of 2.012 indicates a relatively poor clustering solution. This suggests that the clusters are not well-separated from each other and that the internal cohesion of the clusters is low. A Calinski-Harabasz Index of 35.21 suggests that the clusters may be reasonably compact and somewhat well-separated. An ARI of -0.0019 indicates that the there is low agreement between the predicted clusters and the actual quality labels

Comparison with KMeans

A Silhouette Score of 0.32885 in K-means indicates a moderate level of clustering meaning that k-Means model has successfully separated the observations into distinct clusters whereas the clustering in DBSCAN is poor and there are several misclassifications. The Davies-Bouldin Index of 0.973053 in K-Means indicates good clustering, as lower values suggest better separation between clusters, however DBSCAN performs poorly here as well. Compare to the Calinski-Harabasz Index of 35.214522 in DBSCAN, K-Means index of 218.819909 suggests that the clusters formed are compact and well-separated, further reinforcing the better performance of the k-Means clustering model. The k-Means algorithm performed better than DBSCAN on all metrics. This might be due to the unbalanced class distribution in the dataset. Another key point is that in DBSCAN the label does not really match with the class label which could have affected the score.



References

DecisionTreeClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Gadre, V. (2022, January 4). Simple linear regression using Python - geek culture - medium. *Medium*. https://medium.com/geekculture/simple-linear-regression-bd4348e1ee62

How to disable warnings in Jupyter Notebook | Saturn Cloud Blog. (2023, October 20). https://saturncloud.io/blog/how-to-disable-warnings-in-jupyter-notebook/

How to make two plots side-by-side. (n.d.). Stack Overflow. https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side

Managing missing data with pandas. (n.d.). Python for Data Science. https://www.python4data.science/en/24.1.0/clean-prep/nulls.html

Sankalana, N. (2024, September 20). K-Means Clustering: choosing optimal K, process, and evaluation methods. *Medium*. https://medium.com/@nirmalsankalana/k-means-clustering-choosing-optimal-k-process-and-evaluation-methods-2c69377a7ee4

Viswa. (2023, July 26). Linear Regression model practical - viswa - medium. *Medium*. https://medium.com/@vk.viswa/linear-regression-model-practical-610f36d464d4

Yadav, A. (2024, September 14). Grid search for Decision Tree - Biased-Algorithms - Medium. *Medium*. https://medium.com/biased-algorithms/grid-search-for-decision-tree-ababbfb89833#:~:text=Grid%20Search%20is%20a%20methodical,hyperparameters%20for%20your%20decision%20tree.