Assignment 2: Data Modelling

Campbell Timms (s3720784)

Assignment 2 is cantered around the data modelling process. The data set for this assignment included several variables related to physiochemical tests of different wines and also how they scored out of 10 for quality which was our output variable we were aiming for.

Task 1: Regression

Task 1 involves applying a simple linear regression algorithm to find the trend between two of the variables in the data. The first is the alcohol content by volume of the wine and the other is its density (g/cm^3).

Simple Linear Model

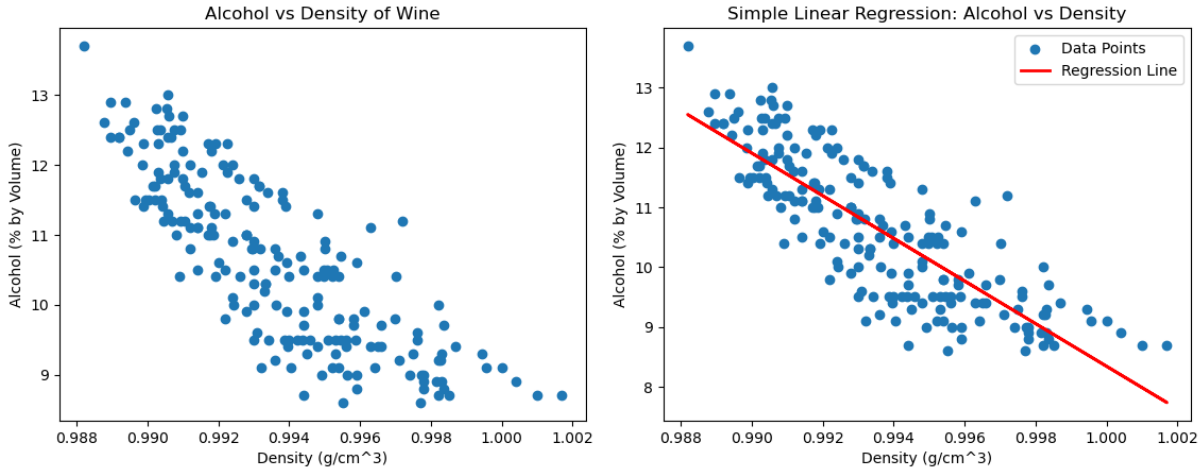


Figure 1: A scatter chart showing the alcohol and density data of our sample (Left), The same scatter chart with a regression line from out simple linear model (Right).

As we can see from the initial graph of the data from the left plot in figure 1, all of the data seems to follow the same trend as it uniformly shows a negative relation between density and alcohol content of the wine. That is to say that the larger the share of alcohol in the wine the less dense the wine is in total.

In the second graph to the right, you can see the output of the linear regression model of the relationship between the two variables, otherwise known as a trendline. More accurately the model found a coefficient of -356.5 between the two variables with a mean squared error of 0.46. The mean squared error is effectively the average of difference between the data points (actual value) and the regression line (predicted value) and reflect the distance they are from each other.

Task 2: Classification

Task 2 involves classification models. That is models that can take an input and places that into one of the predefined categories (classes).

kNN

kNN or k Nearest Neighbours is a classification algorithm that classifies an object based on the class of the nearest k neighbours. To build this model we need to split our data into train and test data. A portion of data that we can train the model on so it knows the classes of an object compared with the objects around it then we can use this model on data it hasn't seen to test it and make a prediction of its class. Choosing an appropriate k value is important to the accuracy of the model. Too few and

it the decision is based on a small sample size but too many and the points are no longer nearby the original point.

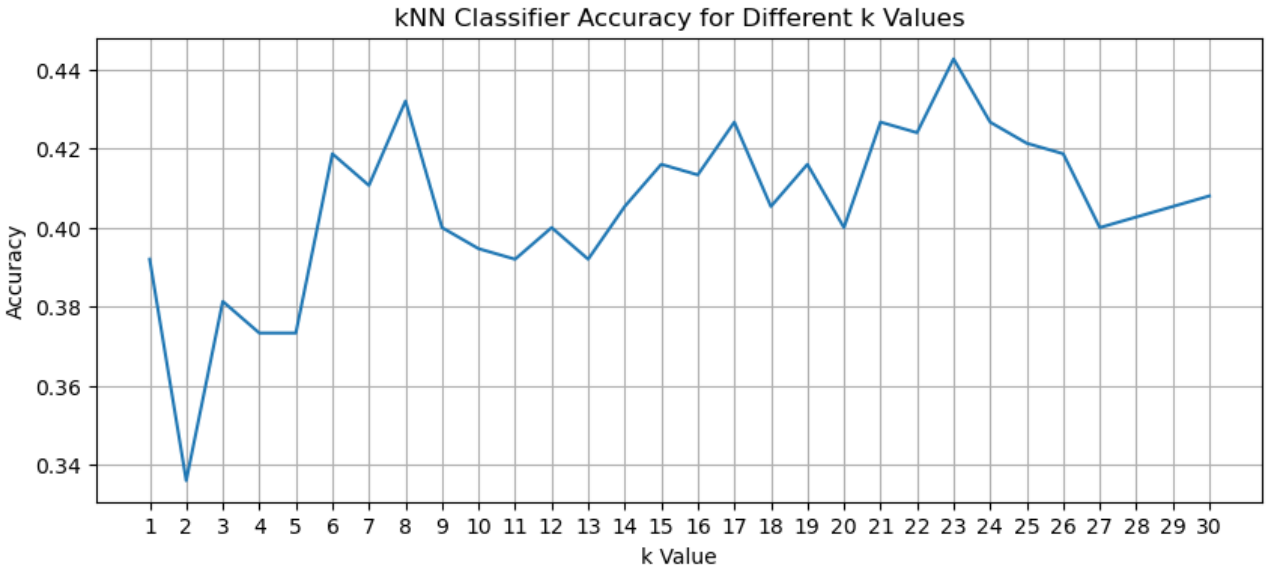


Figure 2: A chart showing the accuracy of a kNN model based on a sample of k values.

To determine the k value for the kNN model, I used a technique called stratified k fold cross validation (StratifiedKFold, 2024). Although a daunting title the process can be explained quite simply. Essentially all the data is broken up into a number of smaller sets called folds, then what makes it stratified is that the class representation in each fold is preserved to the same proportion as the original data set. Then you can run your model of each of these folds one at a time as a test set and using the rest of the folds as a training set allowing you to evaluate the model in different conditions on the same set of data. This was done for each value of k and the accuracy was plotted as a result. As we can see here the k with the highest accuracy was 23 which is also an odd number which helps for the rare case where all the classes are equal.

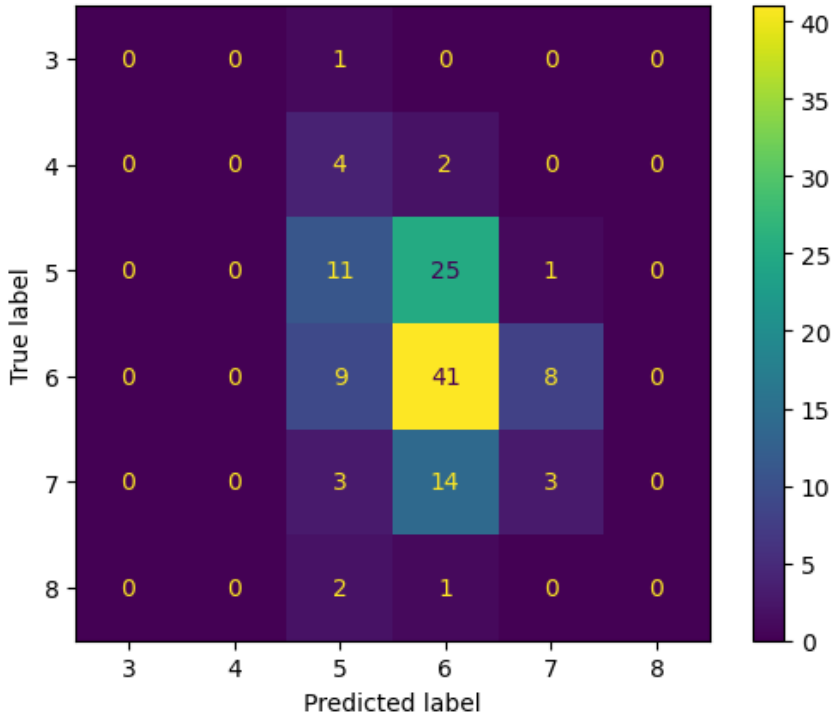


Figure 3: A confusion matrix for the kNN algorithm with the true labels for the quality of the wine on the y axis and the predicted quality from the kNN algorithm.

Figure 3 shows the confusion matrix for the kNN algorithm for k = 23. A confusion matrix shows the true class of each observation and what it was predicted as. If the true class = the predicted class then the model made a correct prediction, on the confusion matrix all the correct predictions are along the diagonal of the matrix. Looking at this confusion matrix we can see that a significant portion of the predictions were a 6/10 for quality, which probably is accurate for most of the ratings as 6 seems to be the median value.

In terms of the performance of the model there are several metrics we can use to evaluate it, they are:

Accuracy: How many were predicted correctly out of all predictions.
Precision: For those predicted to be true, how many were actually true.
Recall: For all the true labels, how many was predicted to be true.
F1-score: F-score helps to measure Recall and Precision at the same time with the following equation.

$$F\text{-}measure = \frac{2*Recall*Precision}{Recall + Precision}$$

These definitions were informed by towardsdatascience.com. (Science, 2024)

For the kNN classifier these metrics are summarised in the following table.

	ACCURACY	PRECISION	RECALL	F1-SCORE
KNN	0.440	0.378	0.440	0.397

We'll use these metrics to compare to the improvements made at the end of the following section.

modified kNN

For the next portion of this task, it's required to improve on the kNN algorithm. Not by tuning it parameters like selecting an optimal k value but by changing process. One of the first things you can do to improve the kNN algorithm is to add a weighting factor to each of the nearest n neighbours so closer neighbours have a larger importance on the classification of the object. This was easy enough to do with the scikit learn package by using an argument to add this feature, so on top of this I also added feature scaling (Learn S. , 2024).

Feature scaling is especially useful when we have many dimensions of vastly different magnitudes. The idea is if we're calculating the distance when using the weighted feature, variables with a much larger magnitude will have a greater effect on the distance calculation and therefore the classification. An example of how this works is given below.

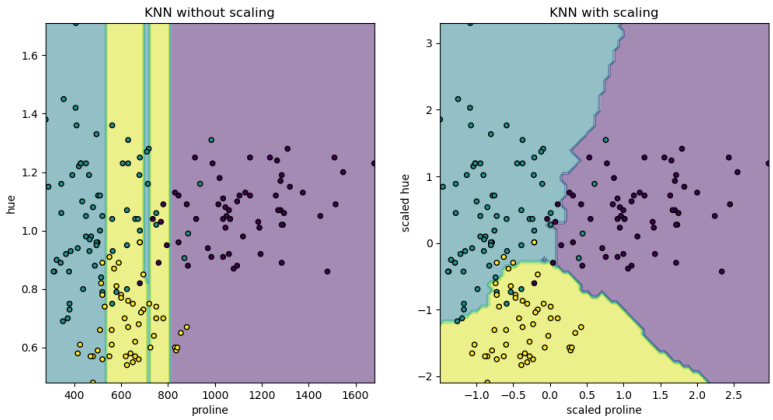


Figure 4: A demonstration of the effect feature scaling can have in terms of classification.

For the most part in this data set all the variables are within a magnitude or two of each other although the largest difference was between the chlorides (g / dm^3) variable and the total sulfur dioxide (mg / dm^3) variable with a magnitude difference of 10^5. Scaling these variables should allow better predictions for classification.

The metrics for the improved kNN classifier with distance weighting and feature scaling is given in the following table.

	ACCURACY	PRECISION	RECALL	F1-SCORE
KNN	0.440	0.378	0.440	0.397
MODIFIED KNN	0.488	0.447	0.448	0.467

Here we can see the modifications made to improve the kNN algorithm were effective and did improve all the metrics even though only slightly. Like I mentioned earlier these improvements would have a much larger effect if more of the variables were vastly different magnitudes as normalisation would be more useful.

Decision Tree

Another classification method we can use is that of the decision tree. A decision tree classifier is a supervised classification algorithm which classifies objects into groups by splitting them into nodes based on a conditional feature. Decision trees are often simple to understand and can be easily visualised although if not configured correctly they can make over-complex trees which overfit the data.

To determine the correct parameters for the decision tree classifier, I used a function called GridSearchCV (GridSearchCV, 2024) which goes through a list of predefined parameters and runs the model with each combination of these parameters comparing their performance via cross validation to output the best set of parameters which gave the best performance.

Running the model with the max depth as 5, minimum samples to split as 10 and minimum samples in a node as 20 gave me a tree that looks like figure 5.

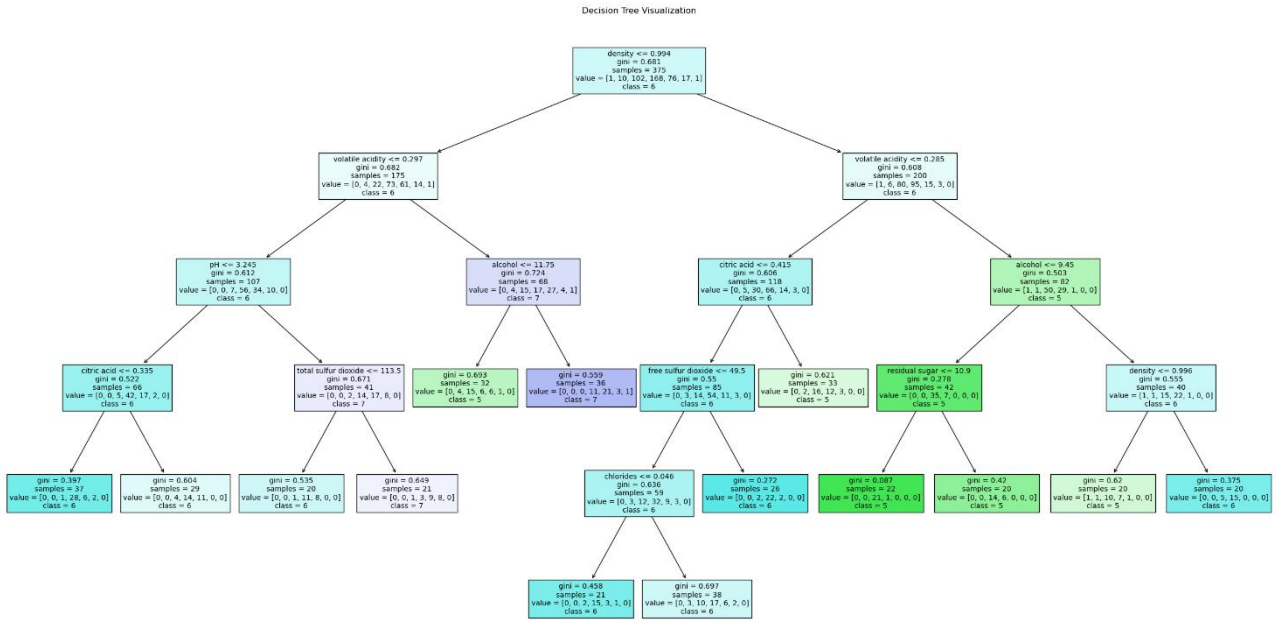


Figure 5: Decision tree classifier visualisation classifying the quality of wine off physiochemical tests.

Figure 5 may look a bit complicated but as viewed full screen on my monitor it is in fact quite legible. I hope the pdf retains the definition of the visualisation as I cannot display this in another manner. For a full-sized image please see the appendix.

The decision tree classifier uses the Gini index to determine if each node that is split is homogeneous (all the same class) or disordered (made up of different classes). Ideally for a decision tree the split that best separates the sample into classes with the same class is best.

Comparison between kNN and Decision Tree

Below in figure 5 we can see the performance difference between the two classifiers. The performance metrics for the decision tree does indicate that this model is better at predicting the quality of the wine from the variables given.

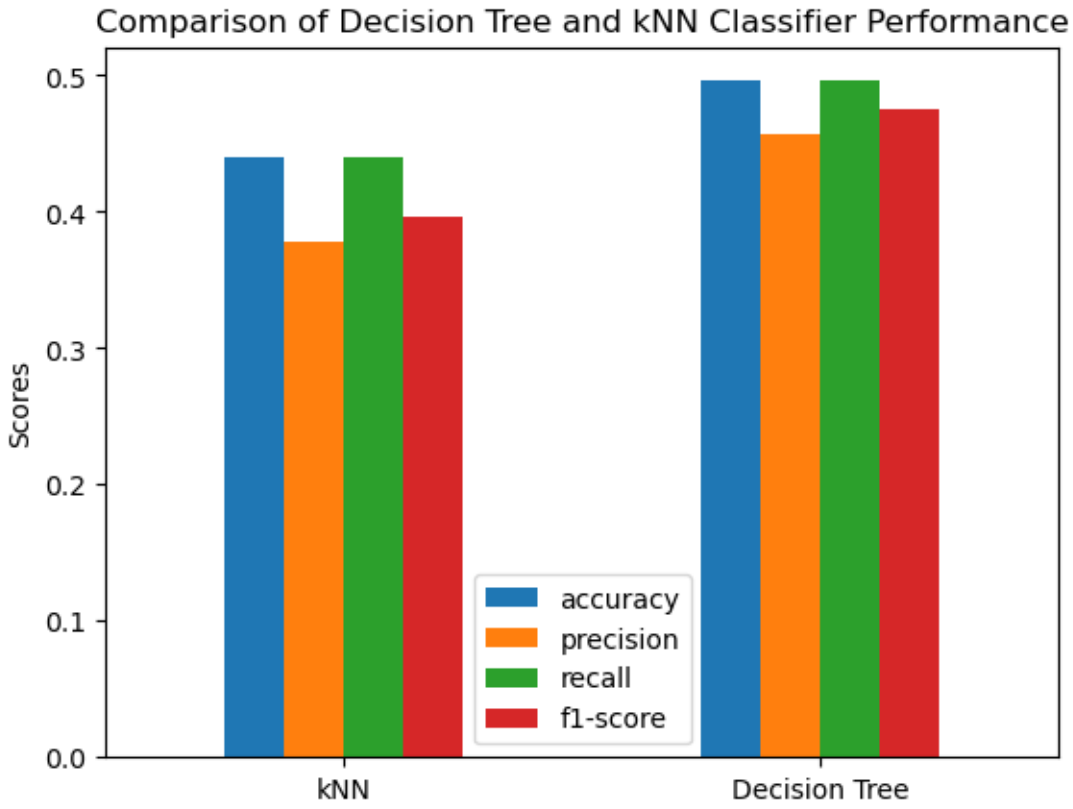


Figure 6: Comparison between a kNN classifier and Decision Tree classifier based on performance metrics.

The vales of the performance evaluation from figure 6 can also be found on the table below.

	ACCURACY	PRECISION	RECALL	F1-SCORE
KNN	0.440	0.378	0.440	0.397
DECISION TREE	0.496	0.457	0.496	0.475

Despite the decision tree having a better accuracy we can see from figure 5 that is rather complicated despite my best efforts. It's likely this performance does suffer from potential overfitting and will not have the same performance on new data. It's especially true as when optimising the hyperparameters for performance using the grid search function we did select the variables that would give the highest accuracy despite making a potentially more complicated tree.

Task 3: Clustering

The next algorithms we're going to try are clustering algorithms which unlike classifiers don't predict the label of an object but merely group them into clusters based on their similarity.

k-Means

The first of these algorithms is k-Means. K-Means starts with a number, k, which indicates the number of divisions into groups it looks for then divides all the data into said groups then finds the mean distance between all the points and the initial centroid of that group then moves the centroid to that position then divides all the data again. This process repeats until the centroids no longer move.

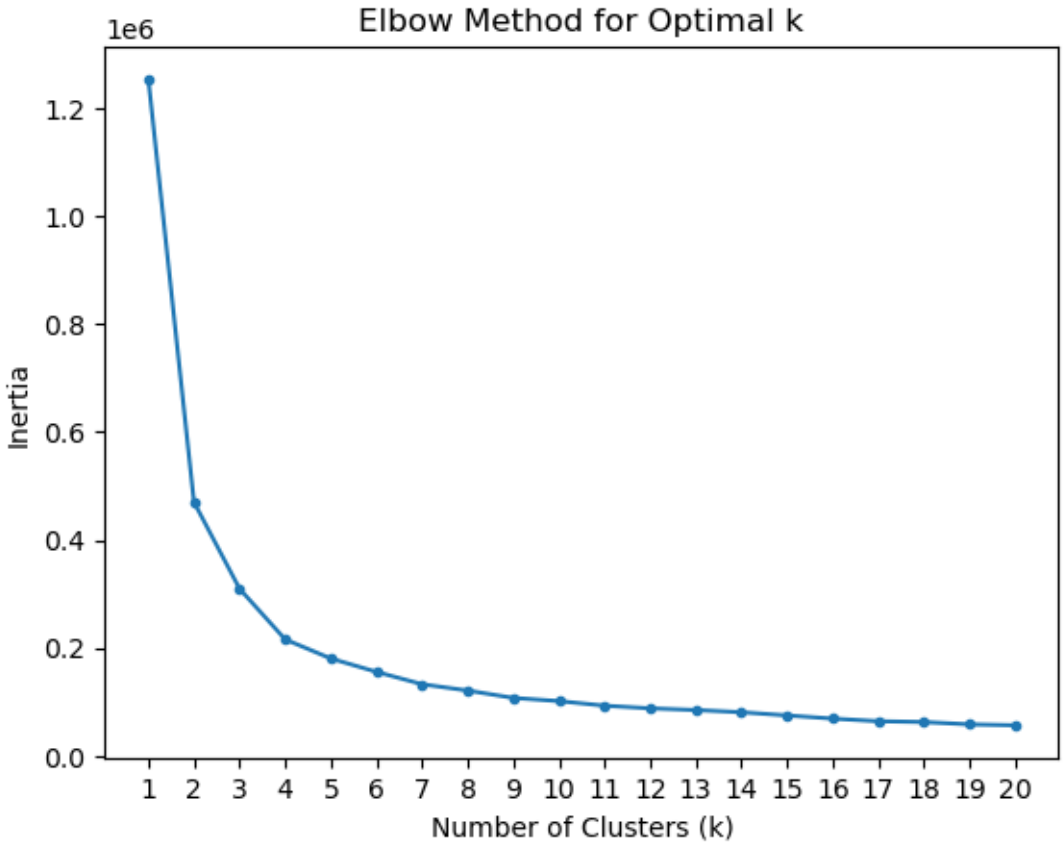


Figure 7: Elbow method to find optimal number k clusters to use.

Although, how do you determine the initially number of clusters without knowing anything about the data set yet. One method which is featured in figure 7 above is the elbow method which calculates how far each data point is away from each cluster centroid which is its inertia and is repeated for each number for k. You'll see that after adding a few clusters the distance between the centroid and all the data points quickly decreases but slows down as k increases. The best value for k is at the 'elbow' of this graph. For this assignment, I know there are only 6 class labels so that is the value for k I chose but by figure 7 any number of k between 3-6 seems to be acceptable.

If we were to use the number of clusters as a prediction for the of the quality like we did in the classification section we could produce the following confusion matrix again, as seen in figure 8 below. Since k means isn't searching to identify these clusters into classes it is only clustering data into their most similar other values it does a poor job. It does not predict any cluster as anything but 5 or 6's as they are over represented.

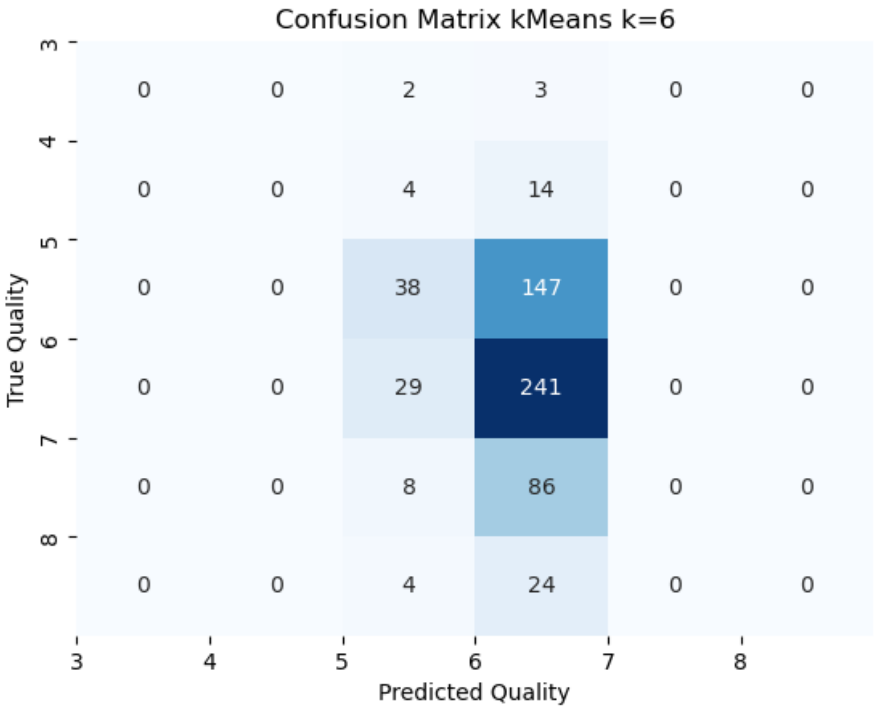


Figure 8: Confusion matrix for the k Means clustering algorithm for the quality of wine.

A better set of metrics to use to evaluate the clustering algorithm is shown in the following table.

	RAND INDEX	HOMOGENEITY	COMPLETENESS	V-MEASURE
K MEANS	0.418	0.00899	0.0286	0.0136

These metrics were recommended in the weekly tutorials via the documentation page (Clustering performance evaluation, 2024) for clustering performance evaluation. A quick summary of these metrics:

Rand Index: Measures the similarity between the true labels and the predictions
Homogeneity: Each cluster contains only members of a single class.
Completeness: All members of a given class are assigned to the same cluster.
V-Measure: A mix between Homogeneity and Completeness calculated as follows:

$$v = \frac{2 \times \text{homogeneity} \times \text{completeness}}{(\text{homogeneity} + \text{completeness})}$$

Aside from the Rand index the k means doesn't do a good job for classifying after it's clustering. It would seem the clusters would more likely be made up of wines that are physiochemically similar and doesn't relate to their quality a subjective value. It seems that when given this task it almost always predicts the quality as a 6 as that is the dominant class as I said before as 6 is likely the mean. We can see this with such low homogeneity and completeness values that very few clusters have all it's data from the same class.

DBSCAN

DBSCAN is another clustering algorithm although instead of clustering the data based on a predetermined value k, DBSCAN clusters data based on density and also identify points as noise if they do not below to any cluster.

The only parameters needed for DBSCAN are an epsilon (eps) value, which is the maximum distance from a data point to search for other data points and a minimum samples parameter, which determines the minimum points are in that neighbourhood which make a cluster. Yet again we can use a similar method to the elbow method to determine the appropriate epsilon value.

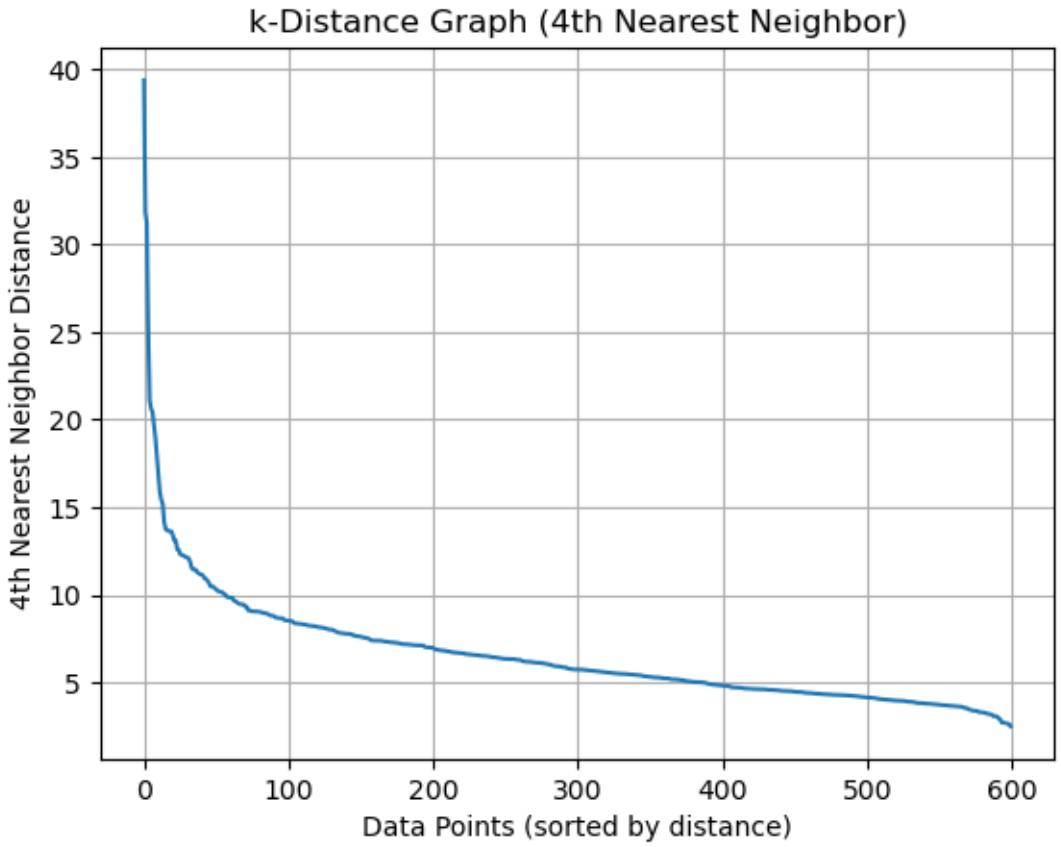


Figure 9: The distance between a data point and its 4th nearest neighbour sorted by distance.

Here we can see the distance on the y axis which will determine out epsilon value. Similar to the elbow method we want to choose a value which captures as many values as possible near each data point and not those far away. Using this graph, I determined an epsilon value of 8 was appropriate.

Running the same metrics that we did with the k means algorithm, the performance of the DBSCAN algorithm is shown in the table below.

	RAND INDEX	HOMOGENEITY	COMPLETENESS	V-MEASURE
K MEANS	0.418	0.00899	0.0286	0.0136
DBSCAN	0.337	0.00877	0.0817	0.0158

DBSCAN is better at clustering data which are in non-uniform shapes when compared to k means although with the high dimensionality of the data it does not seem to make much of a difference when attempting to evaluate if a cluster relates to a quality value to a wine. The one large difference can be seen in the completeness metric where it seems more data points of the same class were clustered together using DBSCAN although the cluster still overall was just as homogenous.

These values can also be seen in the following figure below.

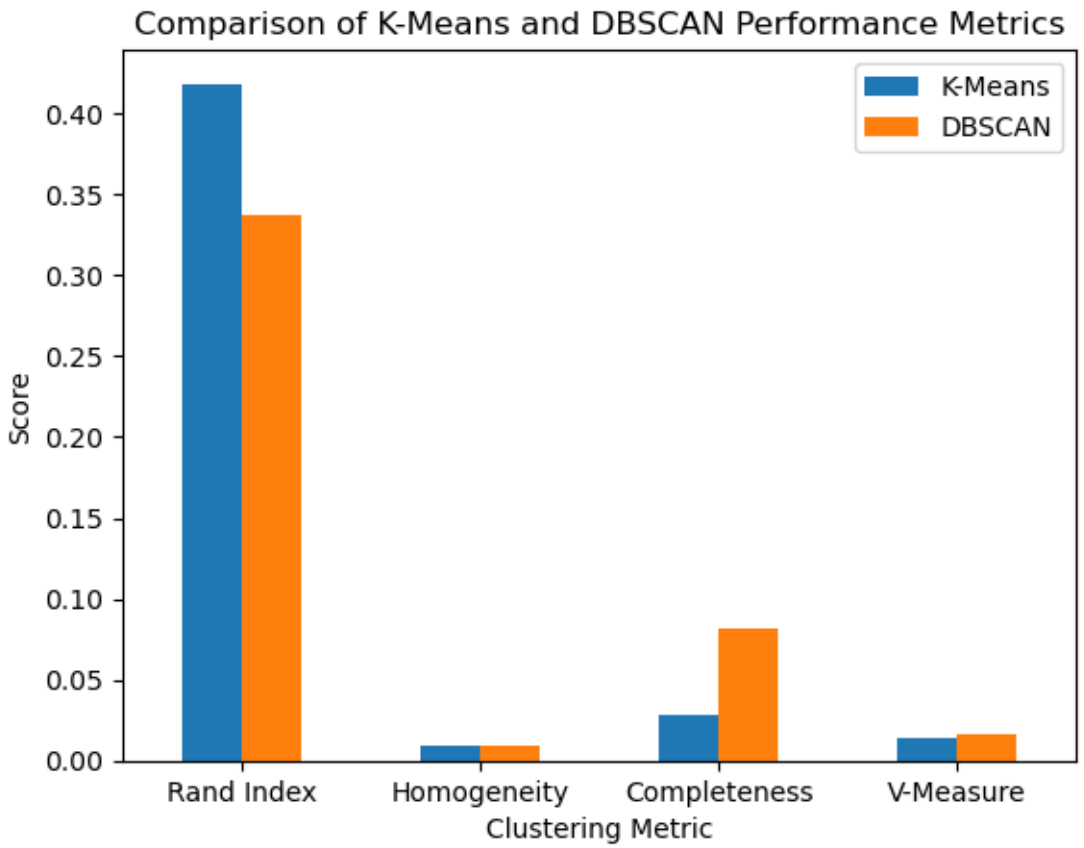


Figure 10: A comparison of performance between the k means and DBSCAN algorithms.

Use of AI Tools:

AI tools were used in the completion of this assignment. The use of AI was also often complimented by inspecting the documentation for the all the packages it recommended and gaining an understanding of the underlying method and procedures of any code it produced.

The main AI used was ChatGPT (ChatGPT-4o, 2024) which was used to both help understand the content and methods as well as any coding errors and functions I have not used before.

References:

ChatGPT-4o. (2024, 10). Retrieved from https://chatgpt.com/

Clustering performance evaluation. (2024, 10). Retrieved from Scikit Learn: https://scikit-learn.org/dev/modules/clustering.html#clustering-performance-evaluation

GridSearchCV. (2024, 10). Retrieved from Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

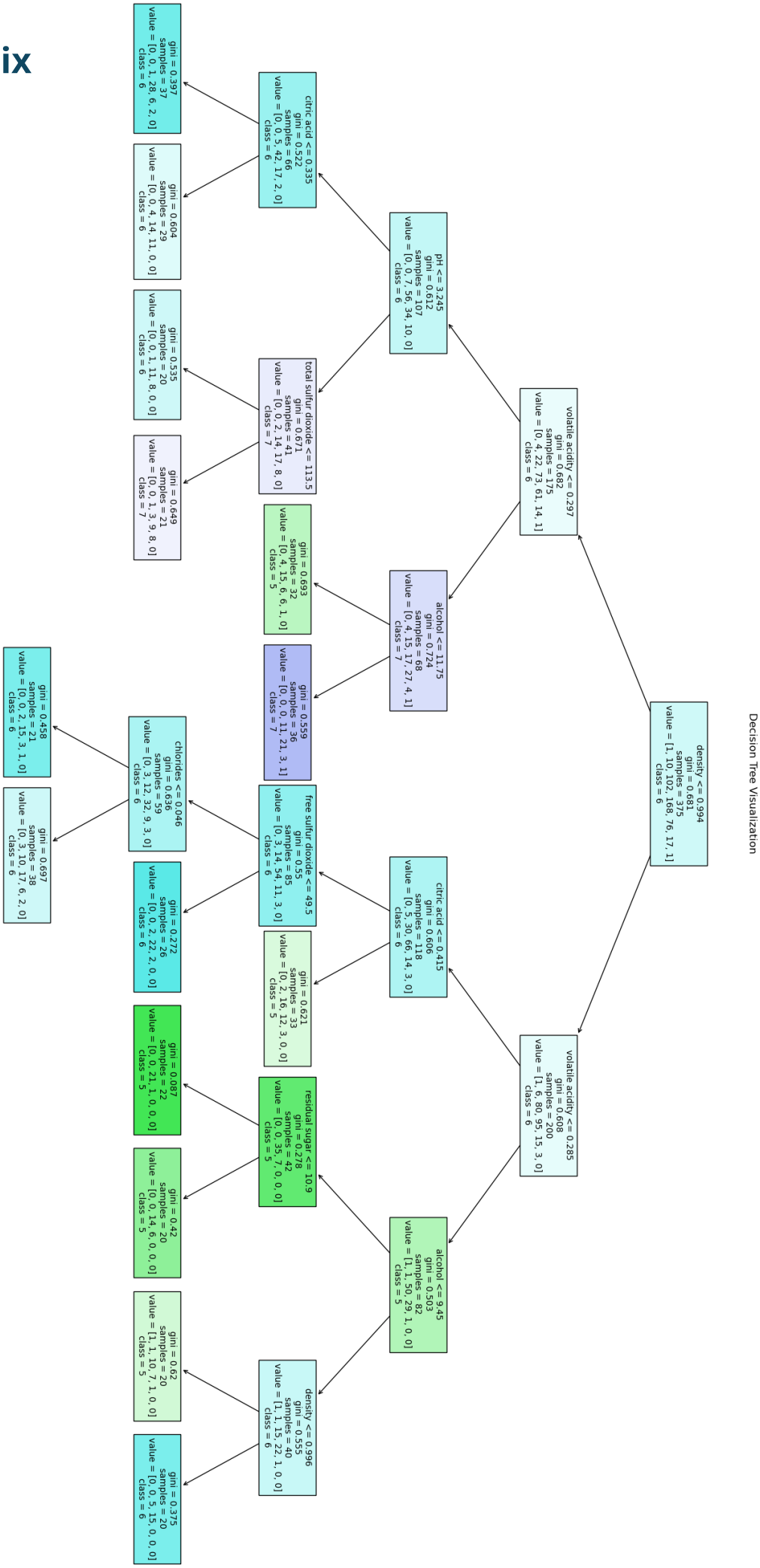
Learn, S. (2024, 10). Retrieved from Scikit Learn: https://scikit-learn.org

Learn, S. (2024, 10). *Importance of Feature Scaling*. Retrieved from SciKit Learn: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html#

Science, T. D. (2024, 10). *Confusion Matrix*. Retrieved from Towards Data Science: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

StratifiedKFold. (2024, 10). Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Appendix


Decision Tree Visualization
density <= 0.994
gini = 0.681
samples = 375
value = [1, 10, 102, 168, 76, 17, 1]
class = 6
volatile acidity <= 0.297
gini = 0.682
samples = 175
value = [0, 4, 22, 73, 61, 14, 1]
class = 6
volatile acidity <= 0.285
gini = 0.608
samples = 200
value = [1, 6, 80, 95, 15, 3, 0]
class = 6
pH <= 3.245
gini = 0.612
samples = 107
value = [0, 0, 7, 56, 34, 10, 0]
class = 6
alcohol <= 11.75
gini = 0.724
samples = 68
value = [0, 4, 15, 17, 27, 4, 1]
class = 7
citric acid <= 0.415
gini = 0.606
samples = 118
value = [0, 5, 30, 66, 14, 3, 0]
class = 6
alcohol <= 9.45
gini = 0.503
samples = 82
value = [1, 1, 50, 29, 1, 0, 0]
class = 5
citric acid <= 0.335
gini = 0.522
samples = 66
value = [0, 0, 5, 42, 17, 2, 0]
class = 6
total sulfur dioxide <= 113.5
gini = 0.671
samples = 41
value = [0, 0, 2, 14, 17, 8, 0]
class = 7
gini = 0.693
samples = 32
value = [0, 4, 15, 6, 6, 1, 0]
class = 5
gini = 0.559
samples = 36
value = [0, 0, 0, 11, 21, 3, 1]
class = 7
free sulfur dioxide <= 49.5
gini = 0.55
samples = 85
value = [0, 3, 14, 54, 11, 3, 0]
class = 6
gini = 0.621
samples = 33
value = [0, 2, 16, 12, 3, 0, 0]
class = 5
residual sugar <= 10.9
gini = 0.278
samples = 42
value = [0, 0, 35, 7, 0, 0, 0]
class = 5
density <= 0.996
gini = 0.555
samples = 40
value = [1, 1, 15, 22, 1, 0, 0]
class = 6
gini = 0.397
samples = 37
value = [0, 0, 1, 28, 6, 2, 0]
class = 6
gini = 0.604
samples = 29
value = [0, 0, 4, 14, 11, 0, 0]
class = 6
gini = 0.535
samples = 20
value = [0, 0, 1, 11, 8, 0, 0]
class = 6
gini = 0.649
samples = 21
value = [0, 0, 1, 3, 9, 8, 0]
class = 7
chlorides <= 0.046
gini = 0.636
samples = 59
value = [0, 3, 12, 32, 9, 3, 0]
class = 6
gini = 0.458
samples = 21
value = [0, 0, 2, 15, 3, 1, 0]
class = 6
gini = 0.697
samples = 38
value = [0, 3, 10, 17, 6, 2, 0]
class = 6
gini = 0.272
samples = 26
value = [0, 0, 2, 22, 2, 0, 0]
class = 6
gini = 0.087
samples = 22
value = [0, 0, 21, 1, 0, 0, 0]
class = 5
gini = 0.42
samples = 20
value = [0, 0, 14, 6, 0, 0, 0]
class = 5
gini = 0.62
samples = 20
value = [1, 1, 10, 7, 1, 0, 0]
class = 5
gini = 0.375
samples = 20
value = [0, 0, 5, 15, 0, 0, 0]
class = 6