# CONTENTS

# 1. Introduction

## 1.1. Problem Statement

The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope - a boring and time-consuming task. We want to predict the age using different physical measurements which is easier to measure. In this project we will use Machine Learning to predict the age of abalone from physical measurements.

*Figure 1. The Abalone*

## 1.2. History of Data Set / Access

The dataset comes from a 1994 study "The Population Biology of Abalone (Haliotis species) in Tasmania. I. Blacklip Abalone (H. rubra) from the North Coast and Islands of Bass Strait". It was donated to the UCI Machine Learning Repository in 1995. The dataset is available from the University of California Irvine Machine Learning data repository (http://archive.ics.uci.edu/ml/datasets/Abalone). We downloaded data set in .csv format from this site.

# 2. Dataset Description

## 2.1. Summary of Data Set

The UCI Machine Learning Repository provides one dataset abalone.data, it contains 4177 observations, 8 descriptive features and 1 target feature. The target feature is the rings of abalone. It is an integer to describe the age of abalone, number of rings add 1.5 gives the age in years of them.

| Data Set Characteristics | Number of Instances | Number of Attributes | Attribute Characteristics |
|---|---|---|---|
| Multivariate | 4177 | 9 | Categorical, Integer, Real |

## 2.2. Attribute of Data Set

The data consist of measurements of the type (male, female and infant), the longest shell measurement, the diameter, height and several weights (whole, shucked, viscera and shell). The outcome is the number of rings. The age of the abalone is the number of rings plus 1.5. Given is the attribute name, attribute type, the measurement unit and a brief description. The number of rings is the value to predict: either as a continuous value or as a classification problem.

| Attribute | Data Type | Units | Description |
|---|---|---|---|
| **Sex** | nominal | N/A | M, F, and I (infant) |
| **Length** | continuous | mm | Longest shell measurement |
| **Diameter** | continuous | mm | perpendicular to length |
| **Height** | continuous | mm | with meat in shell |
| **Whole weight** | continuous | grams | whole abalone |
| **Shucked weight** | continuous | grams | weight of meat |
| **Viscera weight** | continuous | grams | gut weight (after bleeding) |
| **Shell weight** | continuous | grams | after being dried |
| **Rings** | integer | N/A | +1.5 gives the age in years |

*Table 1. Attributes of Abalone*

## 2.3. Data Set Statistics

NA values were checked for each attribute. There is no NA value in our data set. In addition, the type values for each attribute are shown in the table.

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
```

*Table 2. df.info() Table*

Statistical values were analyzed for each attribute. Labelencoder () transformation was done for “Sex” values. After this transformation, the value representations are as follows: 0: Female, 1: Infant, and 2: Male

| | Sex | Length | Diameter | Height | Whole weight | Shucked weight | Viscera weight | Shell weight | Rings |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 |
| **mean** | 1.052909 | 0.523992 | 0.407881 | 0.139516 | 0.828742 | 0.359367 | 0.180594 | 0.238831 | 9.933684 |
| **std** | 0.822240 | 0.120093 | 0.099240 | 0.041827 | 0.490389 | 0.221963 | 0.109614 | 0.139203 | 3.224169 |
| **min** | 0.000000 | 0.075000 | 0.055000 | 0.000000 | 0.002000 | 0.001000 | 0.000500 | 0.001500 | 1.000000 |
| **25%** | 0.000000 | 0.450000 | 0.350000 | 0.115000 | 0.441500 | 0.186000 | 0.093500 | 0.130000 | 8.000000 |
| **50%** | 1.000000 | 0.545000 | 0.425000 | 0.140000 | 0.799500 | 0.336000 | 0.171000 | 0.234000 | 9.000000 |
| **75%** | 2.000000 | 0.615000 | 0.480000 | 0.165000 | 1.153000 | 0.502000 | 0.253000 | 0.329000 | 11.000000 |
| **max** | 2.000000 | 0.815000 | 0.650000 | 1.130000 | 2.825500 | 1.488000 | 0.760000 | 1.005000 | 29.000000 |

*Table 3. Statistical values for each attribute.*

In addition, histogram drawings for each attribute are as follows.

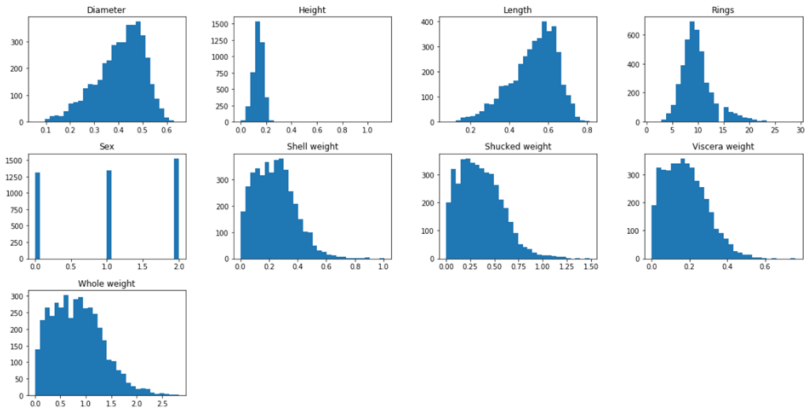


*Figure 2. Histogram for each attributes*

## 3. Analysis

### 3.1. Choosing Polynomial Function in Regression

The code that we write first about choosing degree of polynomial regression and we used validation which helps models tuning the parameters of the model (choosing k in KNN, choosing the order of polynomial function in regression) or selecting the set of features to be used in the model. As always firstly we imported our data separated our target variable and features from dataset and created our dummy variables and we separated our dataset randomly as train and validation set and we wrote function which will implement our nth polynomial regression degree to our dataset so in the next step we can put this function in for loop to show till how many degrees we want to see our dataset on this example we decided to try till 9'th degree and at the end of the for loop we got a list of "Train Error" "Validation Error" for every degree and we sorted this numpy list by Validation error so we can decide which polynomial regression will fit the best for us and we saw the result as this:

| | Training Error | Validation Error | Degree |
|---|---|---|---|
| 0 | 4.762501e+00 | 5.001233e+00 | 1 |
| 1 | 4.195638e+00 | 7.167765e+00 | 2 |
| 2 | 3.681223e+00 | 1.020727e+02 | 3 |
| 3 | 2.613059e+00 | 4.312449e+07 | 4 |
| 4 | 7.909899e-01 | 1.306140e+12 | 5 |
| 8 | 3.839613e-06 | 3.510621e+13 | 9 |
| 7 | 5.890947e-06 | 6.461072e+13 | 8 |
| 6 | 3.556150e-07 | 3.074502e+14 | 7 |
| 5 | 7.224245e-06 | 9.032195e+14 | 6 |

*Table 4. Choosing Polynomial Function in Regression*

As result we understand that the best model for us is linear regression (because 1st degree giving the least validation error) that we will use for other steps

### 3.2. Linear Regression

Linear regression and its values that I created using x_train and x_test in the next example.

```
Mean Squared Error : 5.388776577744681
Train MSE:  4.565852132204131

Length           -0.260554
Diameter          1.654378
Height            0.489724
Whole weight      5.583438
Shucked weight   -5.855818
Viscera weight   -1.494195
Shell weight      1.938175
Sex_F            -0.171340
Sex_I            -0.897241
dtype: float64
```

*Figure 3. Coefficients and MSE for Linear Regression*

### 3.3. Best Subset Selection and Regularization (Forward/Backward)

In this section, we have worked with linear regression, which is the best polynomial example we found in the previous section. First of all, the purpose of doing this section is to think that some of the features we have are independent of the model, and we decided to solve this problem by using the feature selection, which are forward, backward selection, and regularization, which ridge and lasso and based on the models we will compare MSE and find the best model. How we are prepared my train and test data set is first of all, we need to convert the sex column we have as dummy variable. 3 new columns we have obtained here are 3 new columns SEX_M SEX_F and SEX_I. After that, data is divided into two as train and test set, scaled train and test set with rebustscaler so that if the columns with very different scales are scaled close to each other, this will make the data better.
We did the first part, the forward selection (ExhaustiveFeatureSelector), and here we listed the validation values that we found by comparing negative mean squared errors using cv, and in this order, cluster is selected with the lowest validation value, so we made a model selection. We found the set of columns that is found in the forward selection as ordered columns (1, 3, 4, 5, 6, 8) as you can see below. Using these columns which is selected by model selection, we gave the column order number and created my new X_train_selected and X_test_selected consisting of only (1, 3, 4, 5, 6, 8) columns. For the new dataframe that we rebuild, we created Linear Regression for X_train selected and tested it on X_test_selected and found MSE (mean squared error) for X_test_selected.

| | feature_idx | cv_scores | avg_score | feature_names | ci_bound | std_dev | std_err |
|---|---|---|---|---|---|---|---|
| 453 | (1, 3, 4, 5, 6, 8) | [-4.860503574559526, -4.528849381460856, -4.76... | -4.662365 | (Diameter, Whole weight, Shucked weight, Visce... | 0.342775 | 0.266690 | 0.133345 |
| 481 | (0, 1, 3, 4, 5, 6, 8) | [-4.85984126357364, -4.533344630802083, -4.765... | -4.662844 | (Length, Diameter, Whole weight, Shucked weigh... | 0.341380 | 0.265606 | 0.132803 |
| 499 | (1, 3, 4, 5, 6, 7, 8) | [-4.848967687013617, -4.524956218864693, -4.75... | -4.665105 | (Diameter, Whole weight, Shucked weight, Visce... | 0.358265 | 0.278742 | 0.139371 |

*Figure 4. Forward Selection (ExhaustiveFeatureSelector)*

In the second stage, we used the other feature selection SequentialFeatureSelector and found the same result as the 1st by sorting the validation values (1, 3, 4, 5, 6, 8). Here we did not create new X_train_selected and X_test_selected because our result is the same as previous feature selection and we would get the same result.

| | feature_idx | cv_scores | avg_score | feature_names | ci_bound | std_dev | std_err |
|---|---|---|---|---|---|---|---|
| 6 | (1, 3, 4, 5, 6, 8) | [-4.860503574559526, -4.528849381460856, -4.76... | -4.662365 | (Diameter, Whole weight, Shucked weight, Visce... | 0.342775 | 0.266690 | 0.133345 |
| 7 | (0, 1, 3, 4, 5, 6, 8) | [-4.85984126357364, -4.533344630802083, -4.765... | -4.662844 | (Length, Diameter, Whole weight, Shucked weigh... | 0.341380 | 0.265606 | 0.132803 |
| 8 | (0, 1, 3, 4, 5, 6, 7, 8) | [-4.848253419318976, -4.529578750088185, -4.75... | -4.665578 | (Length, Diameter, Whole weight, Shucked weigh... | 0.356876 | 0.277661 | 0.138831 |

*Figure 5. Forward Selection (SequentialFeatureSelector)*

In the 3rd stage, we used SequentialFeatureSelector as a backward selection, and how backward is done in this code, we simply convert forward = False forward to false in our code, and we have reduced the value of k_features by 1, only 1 feature at a time. Wused the same method when choosing the model for the 3rd stage, calculating negative mean squared error, we took the feature set with the largest negative mean squared error value(avg_score) and the columns I found for the backward selection were again (1, 3, 4, 5, 6, 8). Here we did

not create new X_train_selected and X_test_selected again because our result is the same as in example 1 so we would get the same MSE value.

### 3.4. Ridge Regression

We performed Ridge Regression, which is the 4th stage, while choosing a model in Ridge Regression, the important part is to try certain alpha values by making cv and get the optimum cross validation score, that is, the alpha value that has the largest negative mean squared error value. logspace became (-4, -0.5,60) we tried a total of 60 alpha values in this value range and found the optimum alpha = 0.316228 and created the ridge regression with this alpha value. In fact; we could give alphas = np.logspace (-4, -0.5,60) and do the ridgecv_alpha function, which is a simpler method, alpha = alphas. Both have very very close the coefficients.
MSE's we obtained for Ridge Regression are as follows

| | Lambda | Validation Error |
|---|---|---|
| 59 | 0.316228 | 4.820995 |
| 58 | 0.275853 | 4.821013 |
| 57 | 0.240633 | 4.821035 |
| 56 | 0.209910 | 4.821058 |
| 55 | 0.183110 | 4.821082 |
| 54 | 0.159731 | 4.821106 |
| 53 | 0.139337 | 4.821129 |
| 52 | 0.121547 | 4.821151 |

*Table 5. Chossing Best Alpha*

```
Length            -0.250842
Diameter           1.642411
Height             0.490801
Whole weight       5.396456
Shucked weight    -5.768758
Viscera weight    -1.450057
Shell weight       1.997923
Sex_F             -0.170323
Sex_I             -0.897887
dtype: float64
5.393122628466965
```

*Figure 6. Coefficient of Ridge Regression and MSE at the bottom*

### 3.5. Lasso Regression

While applying The Lasso, we did the same things with Ridge Regression, and by searching for the alpha value in the same intervals, we got the biggest Negative mean squared error for cross validation and found the best alpha value with that we found the coefficients and MSE for lasso as follows.

```
Length            -0.000000
Diameter           1.385563
Height             0.488295
Whole weight       4.816910
Shucked weight    -5.519448
Viscera weight    -1.267617
Shell weight       2.168515
Sex_F             -0.146551
Sex_I             -0.882915
dtype: float64
```

*Figure 7. Coefficient of Lasso Regression*

When we Compare the performance of different techniques we used MSE values of each technique to interpreate them so lower MSE means better performance here list down here is the names of techniques and their MSE's.

| Name of technique | Mean Squared Error (MSE) |
|---|---|
| Forward selection (ExhaustiveFeatureSelector) | 5.4304 |
| Forward selection (SequentialFeatureSelector) | 5.4304 |
| Backward selection | 5.4304 |
| Ridge regression | 5.3887 |
| The Lasso | 5.4058 |

*Table 6. Performance Comparison between Technique*

When we compare their MSE lower MSE means better performance, so the best performance among these 5 techniques is ridge regression.

### 3.6. Tree Based Methods (Without penalty, Bagging, and Random Forest)

Firstly, we imported the required libraries and read out data. Converted categorical variable (sex) into dummy variables with get_dummies. We have splitted the data set into a training set and a test set. Then fitted a regression tree to the training set. Plotted the tree, the maximum vertical depth of the tree is 29 and our regressor made predictions on test data. MSE is equal to 8.84 It does not seem to be successful enough, we tried to get lower error rates. We have used cross-validation in order to determine the optimal level of tree complexity. Instead of manually testing alpha values, we defined a specific alpha range / values. Errors are calculated using the cross validation method using those alphas. We drew the errors, to see the differences between training and validation error for different alphas. Alpha values are on the x-axis.

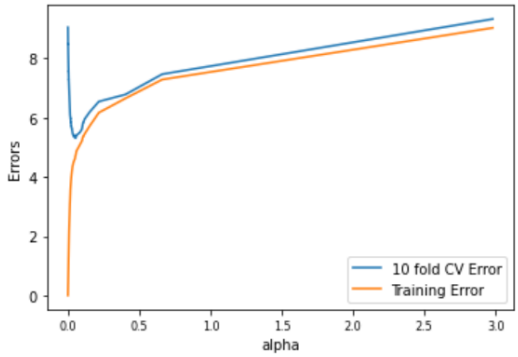


*Figure 8. CV - Alpha Error*

With the lowest validation error, we have fitted our regression tree with alpha=0.051. Pruning the tree improved the test MSE. New test MSE is equal to 5.85 The maximum vertical depth of the tree is 8 now. Then we tried to apply bagging method to see whether we can obtain lower MSE. The new MSE is equal to 4.93.

Then we have used random forests to analyze this data. Compared different number of trees (among [50, 100, 150, 200, 250, 300, 350, 400, 450, 500]) with their MSE's. We have selected 500 estimators which gives the lowest error. New MSE is equal to 4.63. Then we tried to find the best alternative among estimators of "50, 100, 200, 300, 400, 500" and max

features “1, 2, 3, 4, 5, 6, 7, 8, 9, 10”. Each dual had different MSE’s and they are really changing the test data error. The best alternative was max 2 features and 200 estimators. Then the new MSE is equal to 4.56 This is a successful progress from 8.84 to 4.56.

By comparing MSE’s with all these tree-based methods, we plotted them to visualize.

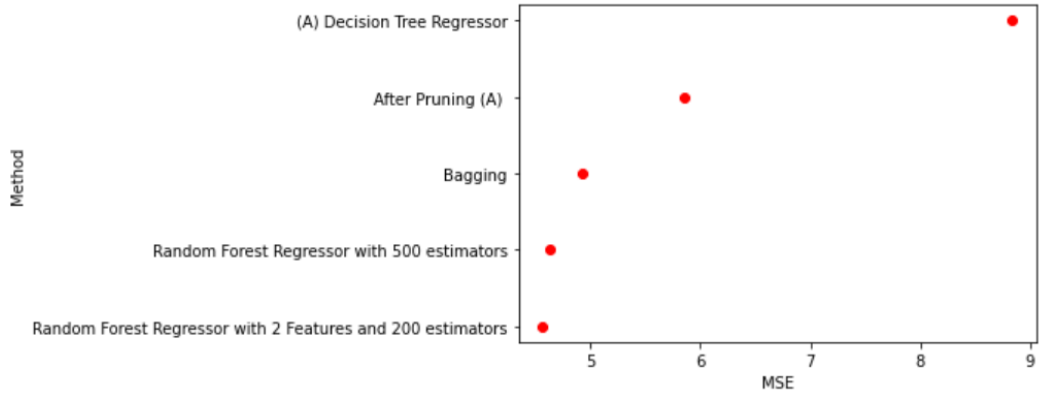


*Figure 9. Comparison of Methods*

Then we found the importance of each variable according to the method with lowest MSE, Random Forest Regressor. It can easily be said that Shell Weight is the most important estimator of rings according to Gini. Whole weight and shucked weight follow it. Plus, sex is not as important as the other features.

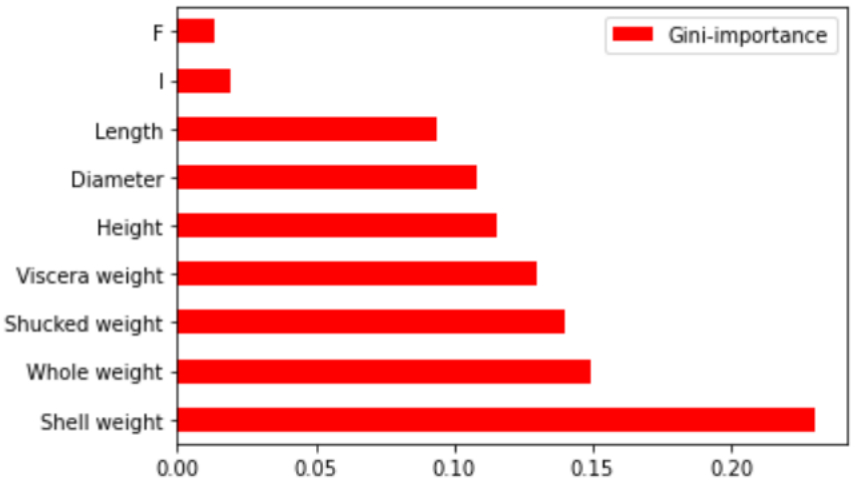


*Figure 10. Gini Importances of Features*

### 3.7. K-Means Clustering

First, we convert our "Sex" values from F-I-M to 0-1-2 values using LabelEncoder (). Later, when we look at our attribute values, some are 0-0.8, some 0-2, and our Rings values vary between 1 and 29. Scaling transformation is required at this point. Therefore, the attributes were scaled using MinMaxScaler ().

Then I created a list of k values that contains values from 1 to 11. In addition, we have created two separate lists for recording distortion values and silhouette scores. For KMeans calculation, I changed the default value of the n_int parameter from 10 to 100. As a result,

KMeans algorithm will get closer to finding the best option by trying 100 different starting points.
Also, we need to choose our starting points as further away as possible. In sclit learn, there is k-means++ which works before k-means algorithm and for tries to place these randomly selected points as further away as possible local minimum convergence. That's why I used k-means++ as the init value.

*Figure 11. Elbow Method*

When we look at the Inertia / Distortion graph, we see that the Elbow point is at k = 3 (candidate). Sometimes elbow method might not always give clear result, so we used silhouette score as well.

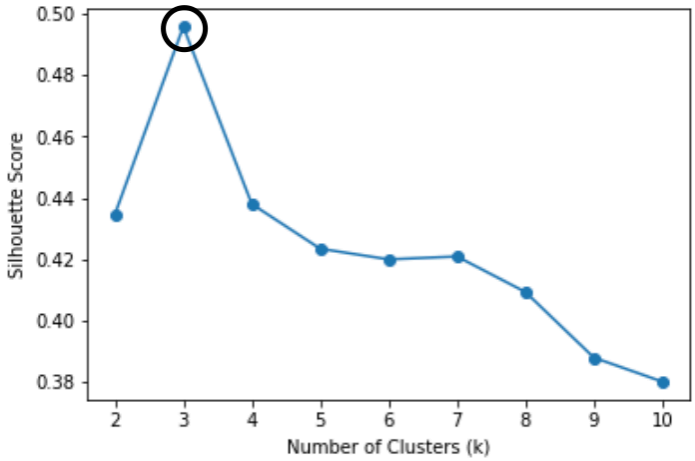

*Figure 12. Silhouette Scores*

As seen on the Silhouette score graph, it gives the highest score at k = 3. Likewise, the elbow method gave a value of 3 for k. That's why we set our optimal clusters k value to 3.

After, our data is re-fit with the value of k = 3 for KMeans method. Our center values for 3 clusters are as in the table below.

| | Sex | Length | Diameter | Height | Whole weight | Shucked weight | Viscera weight | Shell weight | Rings |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.549245 | 0.450689 | 0.430635 | 0.091413 | 0.130041 | 0.108905 | 0.103235 | 0.108849 | 0.240737 |
| 1 | 1.000000 | 0.697518 | 0.687153 | 0.141546 | 0.389847 | 0.323937 | 0.314860 | 0.309717 | 0.360191 |
| 2 | 0.028947 | 0.695087 | 0.685443 | 0.142172 | 0.382593 | 0.309702 | 0.313049 | 0.309797 | 0.367803 |

*Table 7. Interpretation of Final Clusters*

**Interpretation of Final Clusters:**

As can be seen from the table, our "Sex" value emerges as a very distinctive feature for each cluster. While Infant sex is dominant in cluster number 0, all members of cluster number 1 are male. In cluster number 2, women dominate.

The values of Length, Diameter, Height, Whole weight, Shucked weight, Viscera weight, Shell weight and Rings are almost the same for clusters 1 and 2. In cluster number 0, all these attribute values differ from Cluster 1 and 2.

At this point, we can say that the only difference between cluster 1 and 2 is Sex. However, Cluster 0 differs from these two clusters in all attribute values.

### 3.8. Agglomerative Clustering

As in K-Means Clustering, we first converted the "Sex" attribute values to numeric values with LabelEncoder (). Then we have Scaling Transformation to our data with MinMaxScaler ().

After, I will use the dendogram to determine the k value. While doing this, I will use the Ward method as a method. Because Ward method tries to combine clusters with small volumes. In addition, Ward is the most obvious method and seems most comfortable in reading dendrogram.

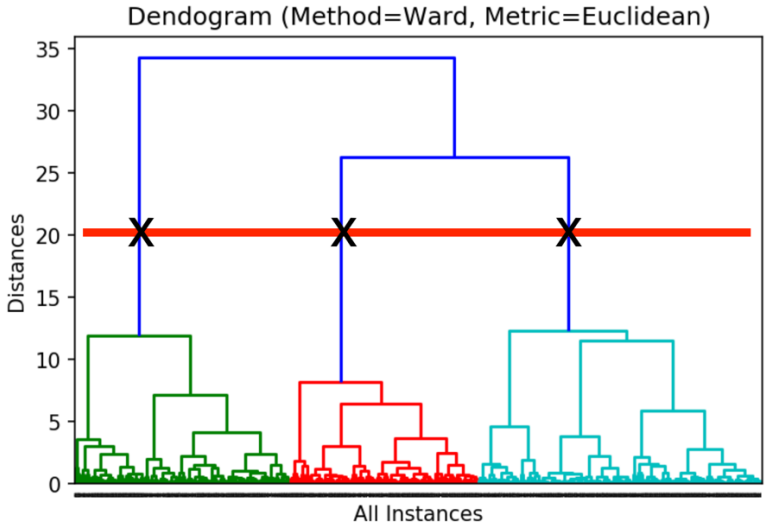


*Figure 13. Dendrogram for Ward Method*

As seen in the dendrogram in the figure, when we draw a horizontal line from the point where the distance is greatest, it cuts our dendrogram at 3 points. This shows us that the best value of the number of clusters according to the dendrogram is 3. In the code file, dendrograms are also drawn for the Single, Complete and Average methods, but we did not include them in the

report because we used the Ward method. For more details, check the code files. Euclidean is used as the metric for all these methods.
K values were determined in the dendrogram graphs of each of the 4 methods. According to these k values, Silhouette Score values were calculated for all 4 methods. The display of these scores in the form of bar charts is as follows. Ward's Method gives the best Silhouette Score value with k = 3.

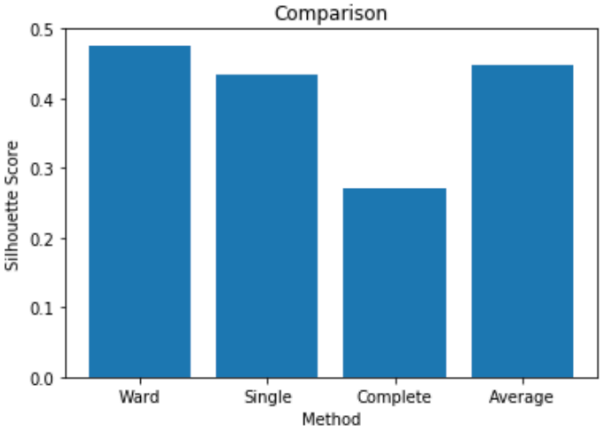


*Figure 14. Silhouette Score for each method*

## 4. Conclusion

All techniques used in the report were compared within their own groups. The individual results of each are detailed below their output. All coding parts were added separately in the report file as .ipyb. As a result, all techniques were used in detail and compared with each other. The calculations and comparisons made were supported by graphics. In addition, test-train separation, validation and performance analysis were also performed at all necessary points.