Preprocessing and Clustering Techniques

Mario García Berenguer
Eder Tarifa Fernández

Contents

1 Data Presentation **1**

2 Preprocessing **3**
2.1 Imputation of Missing Values 3
2.2 Encoding the `Cleft` Variable 4
2.3 Scaling the Variables 6

3 Hierarchical Clustering **6**
3.1 Data Used 7
3.2 Linkage Methods 7
3.3 Dissimilarity Measures 8

4 KMeans Clustering **9**

5 Evaluation of Clusterings **10**
5.1 Hierarchical Clustering Evaluation 10
5.2 KMeans Clustering Evaluation 11

6 Conclusions **12**

1 Data Presentation

In this project, we will work with a CSV file containing data from different fruit instances, though we do not know the type of each fruit. The following variables are known:

- **Weight**: Represents the weight of the fruit in kilograms.
- **Length**: Represents the length of the fruit in centimeters.
- **Width**: Represents the width of the fruit in centimeters.
- **Regularity**: Represents the degree of symmetry of the fruit.
- **Cleft**: Represents the size of the cleft in the fruit.

	weight	length	width	regularity	cleft
0	1.205	4.603915	2.847	5.691634	Small
1	1.726	5.978000	3.594	4.539000	Large
2	1.126	4.516534	2.710	5.965993	Average
3	1.755	5.791000	3.690	5.366000	Large
4	1.238	4.666888	2.989	6.153947	Small
...	...	...	...	...	...
175	1.345	NaN	3.065	3.531000	Very small
176	1.237	4.508595	2.960	4.655452	Small
177	1.437	5.569000	3.153	1.464000	Average
178	1.273	5.412000	2.882	3.533000	Very small
179	1.480	5.656000	3.288	3.112000	Average

Figure 1: Dataframe.

All variables are numerical and continuous except for the `cleft` variable. `Cleft` is a categorical variable, which can take five distinct values: "Very small", "Small", "Average", "Large", and "Very large". It can be considered an ordinal variable, as these values follow a specific order based on the size of the cleft.

By performing a boxplot analysis of the numerical variables, we observe one extreme outlier in the `Weight` variable and a less extreme one in the `Regularity` variable.

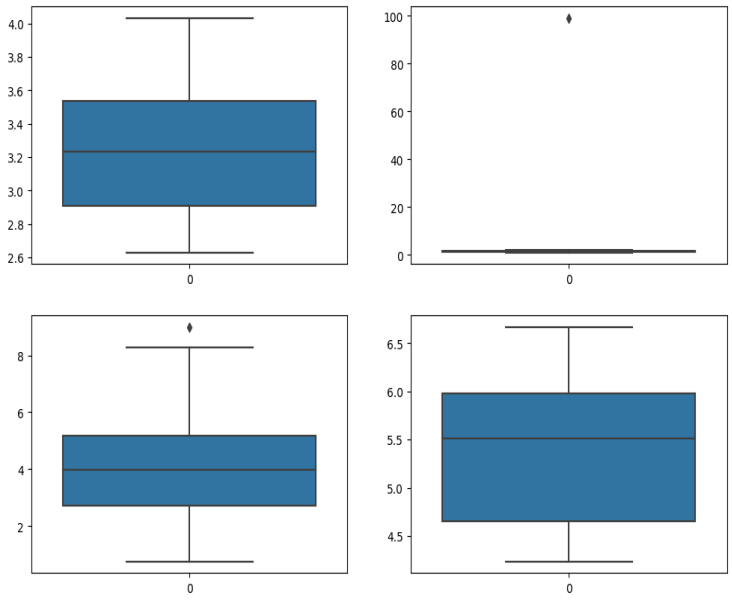


Figure 2: Boxplot of our data

Additionally, all variables except for `Cleft` contain some missing values

(NaN).

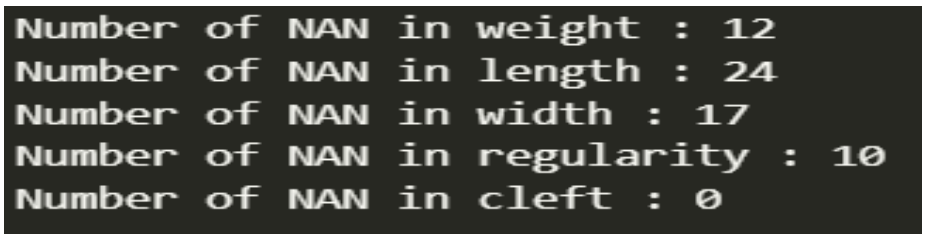


Figure 3: NaN values.

Based on these observations, we plan the following preprocessing steps:

- Impute missing values for instances with NaN.
- Encode the Cleft variable to assign numerical values.
- Remove extreme outliers.
- Scale all variables to standardize the range and avoid issues with variable distances.

Once the preprocessing steps are outlined, we proceed to the next stage.

2 Preprocessing

We begin by imputing the NaN values. As the outlier in the Weight variable is clearly an error and not useful for our analysis, we decide to treat it as NaN as well.

2.1 Imputation of Missing Values

For this task, we have two main options: the SimpleImputer or the KNNImputer from the sklearn.impute library. We performed tests with both methods, considering different values for K in the KNNImputer, and found that the KNNImputer works better since it considers multiple variables when imputing the values. After testing, we found that the best value for K is 5, as it generates intermediate values between the minimum and maximum values from other K values.

Figure 4: Simple Imputer

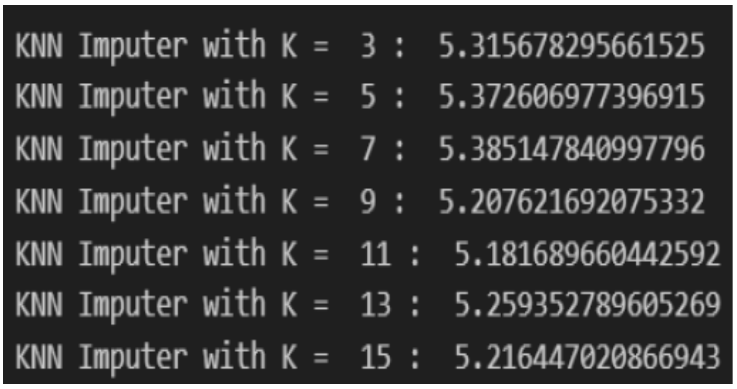


Figure 5: K-NN Imputer

2.2 Encoding the `Cleft` Variable

Next, we need to encode the `Cleft` variable. We have two main encoding options: `OrdinalEncoder` or `OneHotEncoder`, both from `sklearn.preprocessing`. Let's compare the pros and cons of each method:

Pros of Ordinal Encoder	**Cons of Ordinal Encoder**	**Pros of OneHotEncoder**	**Cons of OneHotEncoder**
Takes into account the order of data	Cannot apply arithmetic operations (addition, subtraction, etc.)	Simple encoding	Doesn't account for the order of data
Single variable is retained	-	-	Creates at least 4 new columns

Table 1: Comparison of Ordinal Encoder and OneHotEncoder

Conclusion

In conclusion, we have:

1. We have 5 possible values for the variable, which would translate into at least 4 additional columns using OneHotEncoder. However, using OrdinalEncoder would maintain just one variable.

2. The values appear to follow an order, from smallest to largest, so using OrdinalEncoder makes more sense.

Once the encoding technique is decided, we move on to the actual encoding process. We will assign values from 0 to 4, where 0 represents "Very small" and

4 represents "Very large". At this point, we can see that the `cleft` variable has been encoded. Now, let's examine all the variables together to observe how they behave.

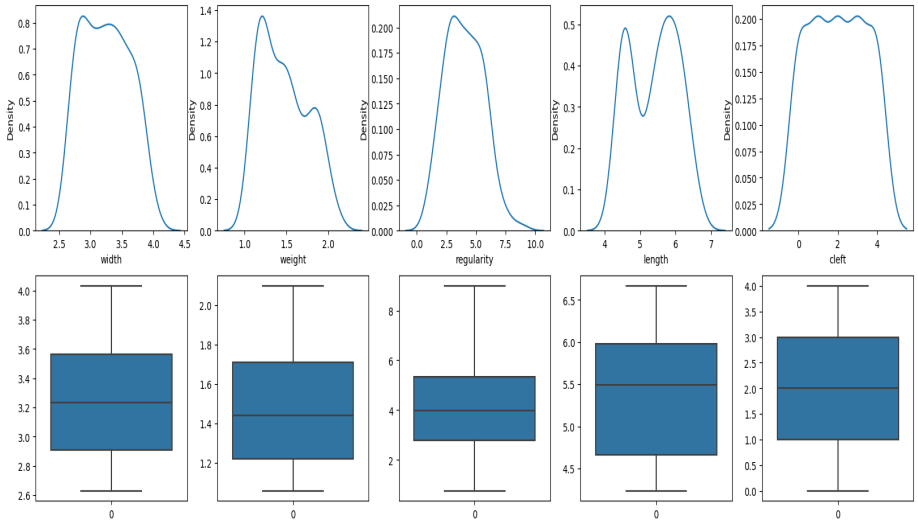


Figure 6: Plots after inputing.

We can appreciate that all outliers have disappeared from the boxplots, including the one from the `regularity` variable. It seems that imputing this value has made it no longer an outlier. We also observe the typical values each variable takes, though the maximum and minimum values are clearer here, along with the difference between them.

```
'weight': (1.059, 2.097, 1.038),
'length': (4.236811412670006, 6.666, 2.4291885873299943),
'width': (2.63, 4.032, 1.4020000000000001),
'regularity': (0.7651, 8.986146203041347, 8.221046203041347),
'cleft': (0.0, 4.0, 4.0)}
```

Figure 7: Range of variables.

Although the ranges of the variables are not significantly different, it is likely that scaling will improve the performance of clustering methods.

2.3 Scaling the Variables

To ensure proper scaling for clustering, we apply different scaling techniques and compare them: `StandardScaler`, `MinMaxScaler`, and `RobustScaler`.

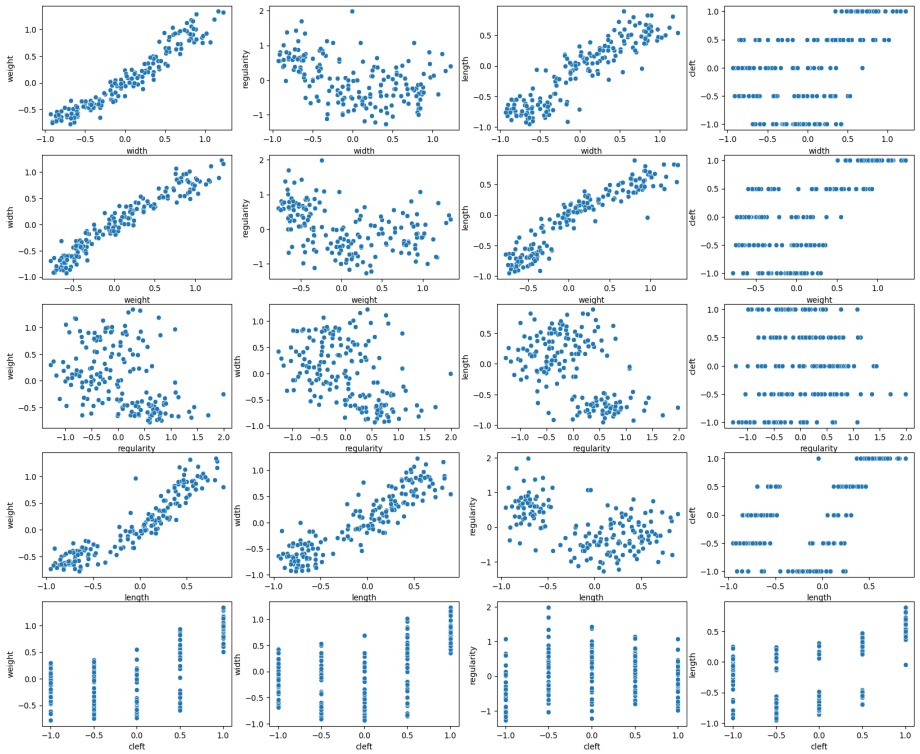


Figure 8: Data distribution after applying Robust Scaler.

We create scatter plots for each scaled dataset and observe that relationships between many variables, such as `Weight` and `Width`, remain linear. The `Cleft` variable continues to take values from 0 to 4, as expected. This suggests that we have correctly scaled and prepared the data.

3 Hierarchical Clustering

We first examine different linkage methods (single, average, ward, and complete) and dissimilarity measures (Euclidean, Manhattan, Chebyshev, and correlation) to determine which provides the most reliable results. We test with between 2 and 7 clusters to find the best model.

3.1 Data Used

In this section, we highlight that models based on scaled data perform better than models using unscaled data. For example, when using correlation as a dissimilarity measure and complete linkage, the unscaled data generates only 2 clusters, while the scaled data generates between 4 and 5 clusters.

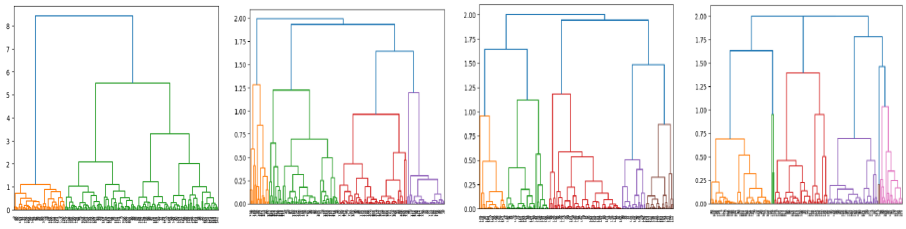


Figure 9: Scaled vs uncaled data

3.2 Linkage Methods

We explore the results of different linkage methods:

- `Single`: Generates highly unbalanced and extreme dendrograms, often with either too many or too few clusters. The clusters are irregular in size.

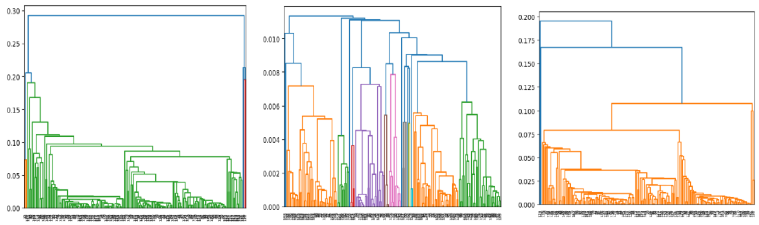


Figure 10: Single linkage

- `Average`: Sensitive to scaling. When applied to unscaled data, it produces noticeably different results than when applied to scaled data.

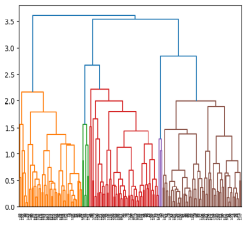


Figure 11: Unscaled average linkage

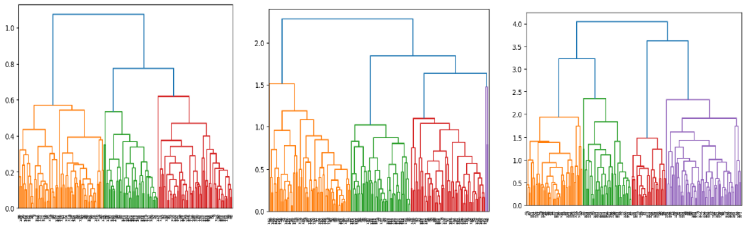


Figure 12: Scaled average linkage

- `Ward` and `Complete`: These methods provide the best clustering results, generating more balanced and stable clusters.

3.3 Dissimilarity Measures

The best results are obtained using the Euclidean and Manhattan distances. These measures generate more regular models, unlike Chebyshev and correlation, which result in inconsistent and poorly structured clusters.

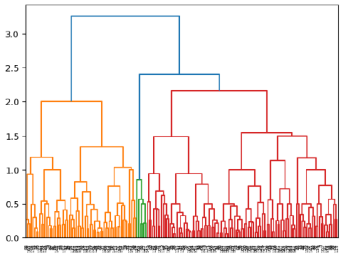


Figure 13: Chebyshev

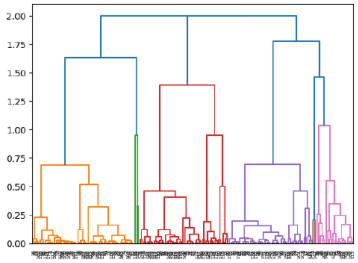


Figure 14: Correlation

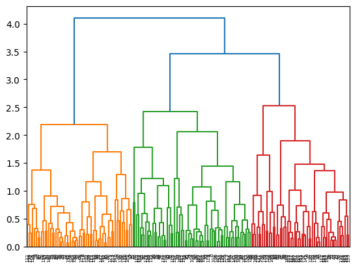


Figure 15: Euclidean

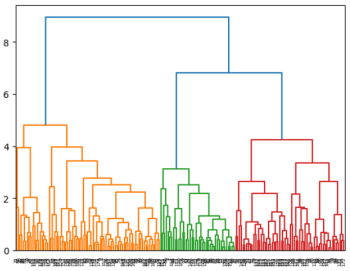


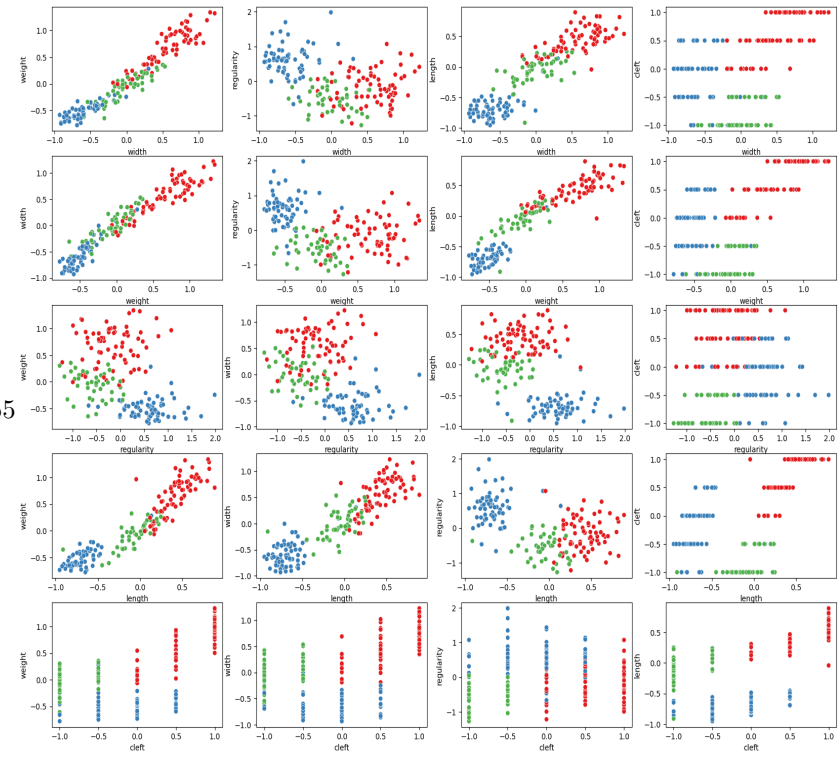
Figure 16: Manhattan

4 KMeans Clustering

We will now use the `KMeans` function from sklearn.cluster to determine the optimal number of clusters for our dataset. To do this, we have calculated the Silhouette coefficient and inertia for different values of `K` in `KMeans`. By analyzing the graphs, we focus on the maximum Silhouette coefficient and the "elbow" shape in the inertia plot to identify the optimal `K`.

Additionally, we have performed this process for each type of data scaling we applied. This indirectly allows us to verify which scaling method yields the best results, as we did in the previous section.

As expected, the results match those of hierarchical clustering. The highest Silhouette coefficient we obtain is slightly below 0.6 at `K`=3. Furthermore, we observe that the "elbow" shape also appears at this same K for the `RobustScaler`.



55

Figure 17: Clustered variables.

5 Evaluation of Clusterings

5.1 Hierarchical Clustering Evaluation

To evaluate the hierarchical clustering models, we use the silhouette coefficient. After testing various combinations, we find that the best results come from the Ward linkage, with either Euclidean or Manhattan distances, and with the `RobustScaler` applied to the data.

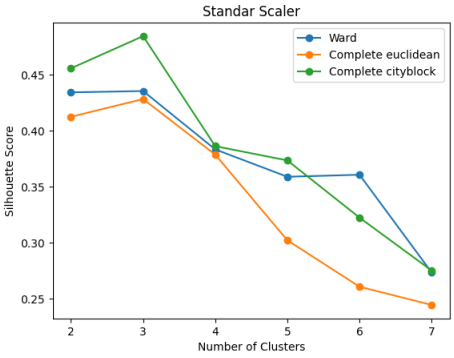


Figure 18: Standar Scaler

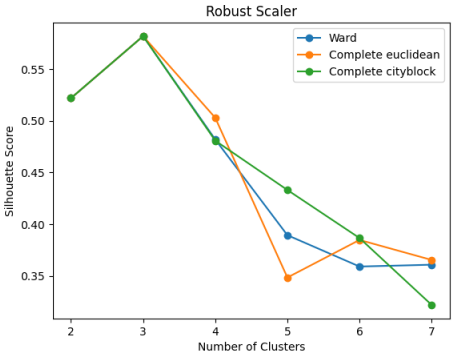


Figure 19: Robust Scaler

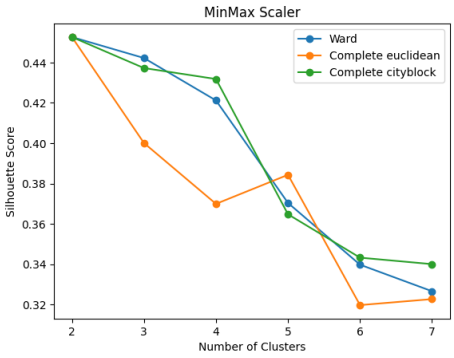


Figure 20: MinMax Scaler

5.2 KMeans Clustering Evaluation

For KMeans, the silhouette coefficient and inertia graphs confirm that `K=3` is optimal. The `RobustScaler` provides the highest silhouette coefficient, confirming it as the best scaling method.

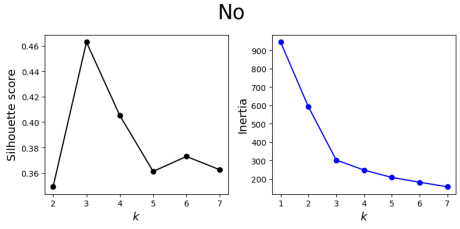


Figure 21: Unscaled

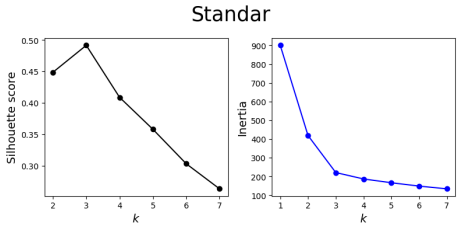


Figure 22: Standar Scaler

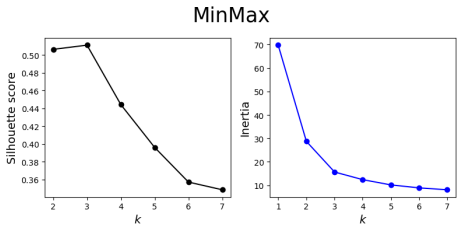


Figure 23: MinMax Scaler

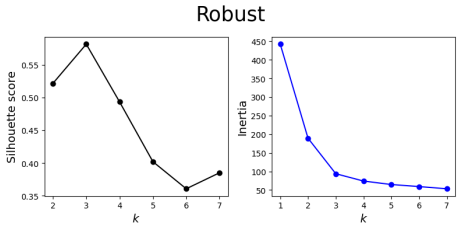


Figure 24: Robust Scaler

6 Conclusions

After performing both hierarchical and partitioning clustering using KMeans, we selected the model that yielded the highest Silhouette coefficient, which is the KMeans model with robust scaling, achieving a coefficient close to 0.6. However, the agglomerative hierarchical clustering also provided a good Silhouette coefficient, slightly lower than that of KMeans, indicating that it is also a suitable model for our data.

Next, we examine the model created with KMeans, separating the variables into pairs for visualization. We have three distinct clusters, but we can observe that the blue cluster tends to have the highest values across all variables, except

for regularity. In contrast, the red cluster has low values for most variables, except for regularity, which has relatively high values, and cleft, which shows both high and low values. Finally, the green cluster represents intermediate values across all variables, except for cleft, where it only takes low values. This chart helps us understand how our model has interpreted the most influential variables for each cluster and the range of values they may take.

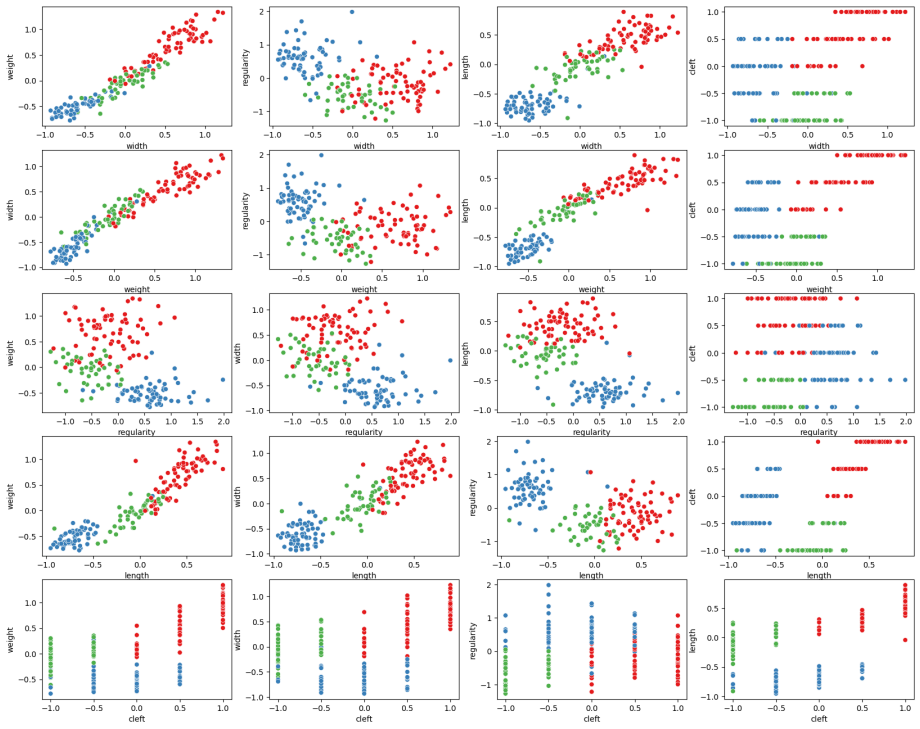


Figure 25: Best model