Introduction to Intelligent Systems
Lab Session 7

Group 30
Sergio Calogero Catalano (s3540294) & Emmanouil Gionanidis (s3542068)

October 30, 2017

Assignment 1

Remarks

The legend has not been included in most of the plots, as they show only a dendrogram, whose elements are lines, having all the same role. The only plots which have a legend are the ones which have other elements, such as lines.

Introduction

For the hierarchial clustering methods, the dendrogram can be used as a graphical tool to evaluate different cluster soutions: it shows the objects which are clustered along the x-axis, and the distance at which the cluster was formed along the y-axis. The Dendrogram allows you to trace backward or forward to any individual case or cluster at any level. The bigger the distances before two clusters are joined, the bigger the differences in these clusters.
The dendrograms shown in this assignment have been created with the code in **Listing 1** , which uses 'pdist' to calculate the pairwise distances and 'linkage' to actually link the observations in clusters. The function in **Listing 1** allows to choose different distance measures, linkage methods and number of clusters, so that different dendograms can be shown, and returns the cophenetic coefficient of the dendrogram, which, as explained in the next section, describes the "accuracy" of a dendrogram.

Listing 1: *Function that shows a dendrogram and a possible clustering, returning the cophenetic coefficient*

```
function cophenetic_coefficient = plot_dendrogram(data,
    distance_measure, linkage_method, num_clusters, fig, labels, xlab,
    ylab, plot_title)
     Y = pdist(data, distance_measure);
     Z = linkage(Y, linkage_method);
     cophenetic_coefficient = cophenet(Z,Y);
     figure(fig(1));
     subplot(fig(2), fig(3), fig(4));
     if num_clusters <= 1
         [H, ~, ord] = dendrogram(Z);
     else
         color = Z(end+2-num_clusters,3)-0.005;
         [H, ~, ord] = dendrogram(Z, 'colorthreshold', color);
         l = line([0 20],[color color], 'Color', 'g');
         legend(l,'Division line');
     end
     set(gca, 'XTickLabel',labels(ord));
     xlabel(xlab);
     ylabel(ylab);
     title(plot_title);
end
```

The only remarks for the shown code are that, if the number of clusters is 1, we don't want the division line to highlight the clusters and thus the line (and its legend) is not shown, and that the labels on the x-axis are reordered to show the correct stock, just by reordering the labels based on the variable 'ord'.

First dendrogram

Figure 1 shows the dendrogram obtained from the given data, using the Euclidean distance and a complete linkage, as requested.
Looking at the dendogram, there is clearly a group of close elements on the left, while most of the elements on the right side are relatively far from each other. For this data set, it looks like we should not split it in more than 3-4 groups, otherwise the elements on the right side will all form single-sized clusters.

Dendrogram generated with the input data

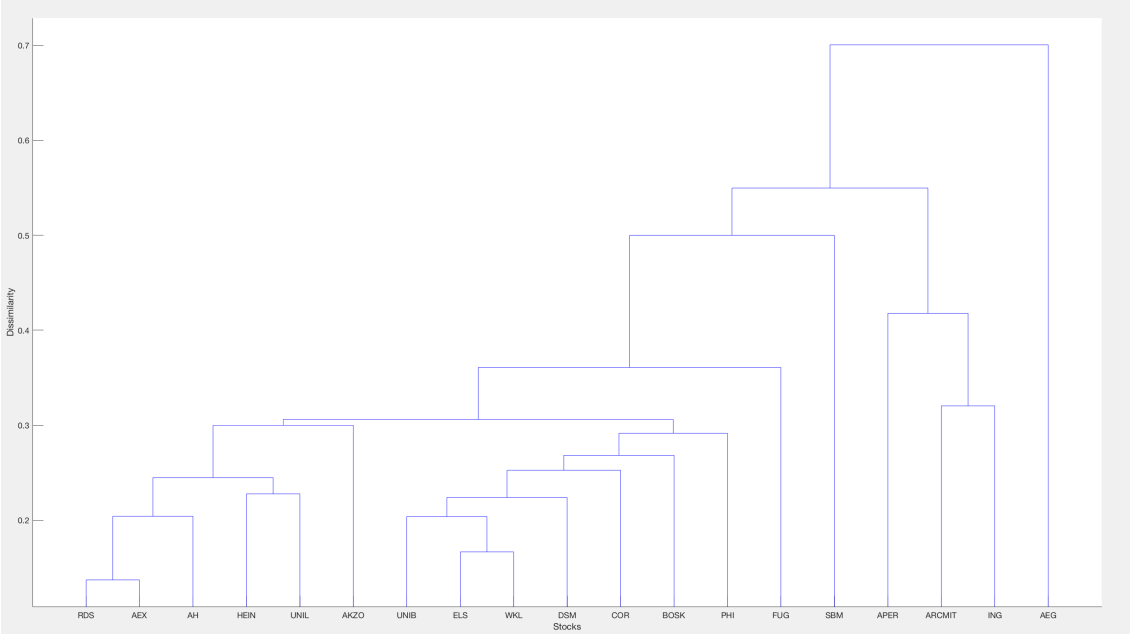

Figure 1: *shows dendrogram obtained analyzing the data given as input, produced with euclidean distance and complete linkage*

The dendrogram shows different clustering options available to cluster the stocks, each with different properties. If we choose any height on the y-axis, and move horizontally across the dendogram, counting the number of lines that we cross, each line represents a possible cluster (represented by the branches of the dendogram that spread out below that line). For example, if we look at a height of 0.545, and move across the x-axis at that height, we'll cross 3 lines. That defines a 3-cluster solution: by following each line down through all its branches, we can see the names of the stocks that are included in each cluster. **Figure 2** shows, in the second plot, the 3 obtained clusters in different colors, with a green horizontal line, drawn to highlight where we cut the dendrogram.
There is not a way to determine which is the best clustering, as it significantly depends on what kind of information we want to obtain from the data, but, as a general rule, we can consider the following: since the y-axis represents how close together observations were when they were merged, clusters whose branches are very close together (in terms of the heights at which they were merged) probably aren't very reliable, but if there's a big difference along the y-axis between the last merged cluster and the currently merged one, we are probably considering a natural division of the data set. That is why we would consider as good clusterings the ones which create 2 or 4 clusters, but not 3 (**Figure 2** shows that the line to create 3 clusters is really near to another merging). In particular, for this data set, we would choose 4 clusters, as it is not as trivial as the 2 clusters division (all observations in one cluster except one, which is in the second cluster).

Different possible clusterings

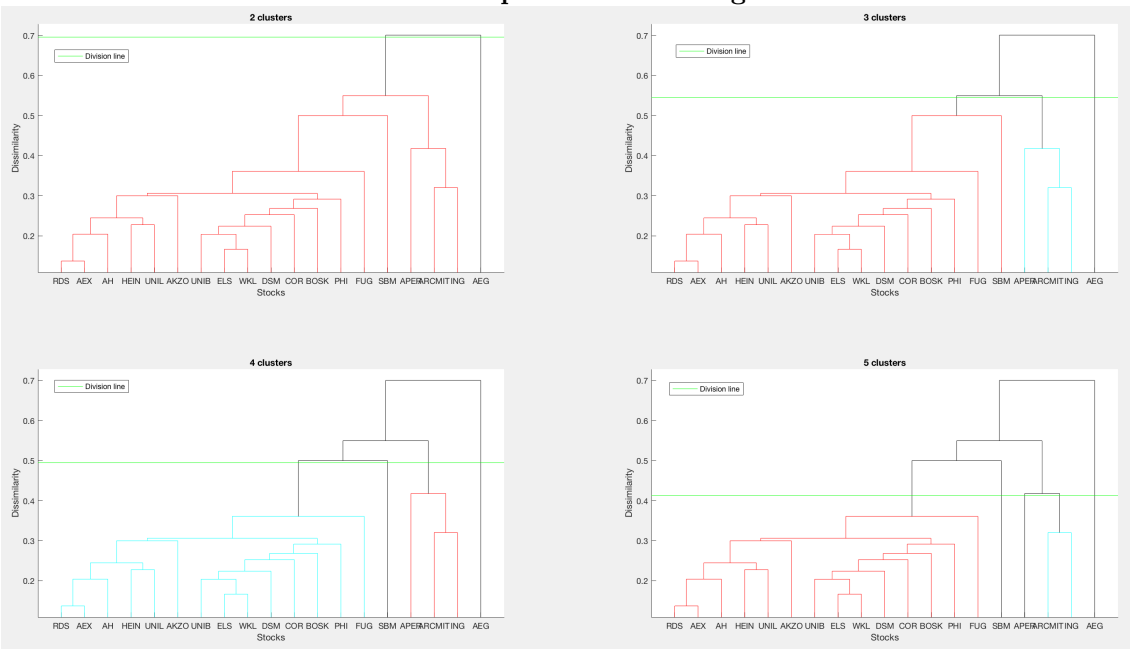

Figure 2: *shows some possible clusterings of the input data, produced with euclidean distance and complete linkage, with 2,3,4 and 5 clusters*

By observing **Figure 2** , we can notice a problem common to all the clusterings of this data set: there are clusters which contain only one element. Usually having clusters with a very small number of elements is not good, because our aim is to divide in clusters, not single elements; but with this data, using complete linkage and Euclidean distance, it is unavoidable (unless we consider the trivial case of one cluster). The problem of single-sized clusters becomes more relevant if we increase the number of clusters, as we can see in the other plots of **Figure 2** , which show the same dendrogram with more clusters. A possible solution to this problem is changing the linkage method or the distance measure, as discussed in the next section.
We can also evaluate how accurately our dendrogram reflects our data by computing the cophenetic coefficient. In a hierarchical cluster tree, two objects in the original data set are eventually linked together at some level. The height of the link represents the distance between the two clusters that contain those two objects. This height is known as the cophenetic distance between the two objects. If the clustering is valid, the cophenetic distances should have a strong correlation with the pairwise distances computed with pdist. The matlab 'cophenet' function compares two sets of values and computes their correlation, returning the cophenetic correlation coefficient: the closer the value of the cophenetic correlation coefficient is to 1, the more accurately the clustering solution reflects our data. The cophenetic coefficient of the previously shown dendrogram is 0.9172, meaning that the dendrogram representation is pretty accurate.

Different linkage methods

To see how good our dendrogram is, we can try to use a different linkage method and compare the results. As the single linkage algorithm is based on minimum distances, it tends to form one large cluster with the other clusters containing only one or few objects each. We can make use of this method to detect outliers, as these will be merged with the other clusters usually at very large distances. **Figure 3** shows the dendrogram obtained with single linkage method. As expected the elements are merged to one growing cluster one by one. Observing **Figure 3** we can determine that the stocks 'AEG', 'SBM' and 'APER' are relatively far from all the others, and thus can be

classified as outliers. We could also consider 'ING' as an outlier, but its distance from the main cluster is not much larger than the previously merged clusters, thus we decided not to include it.

Dendrogram generated with the simple linkage

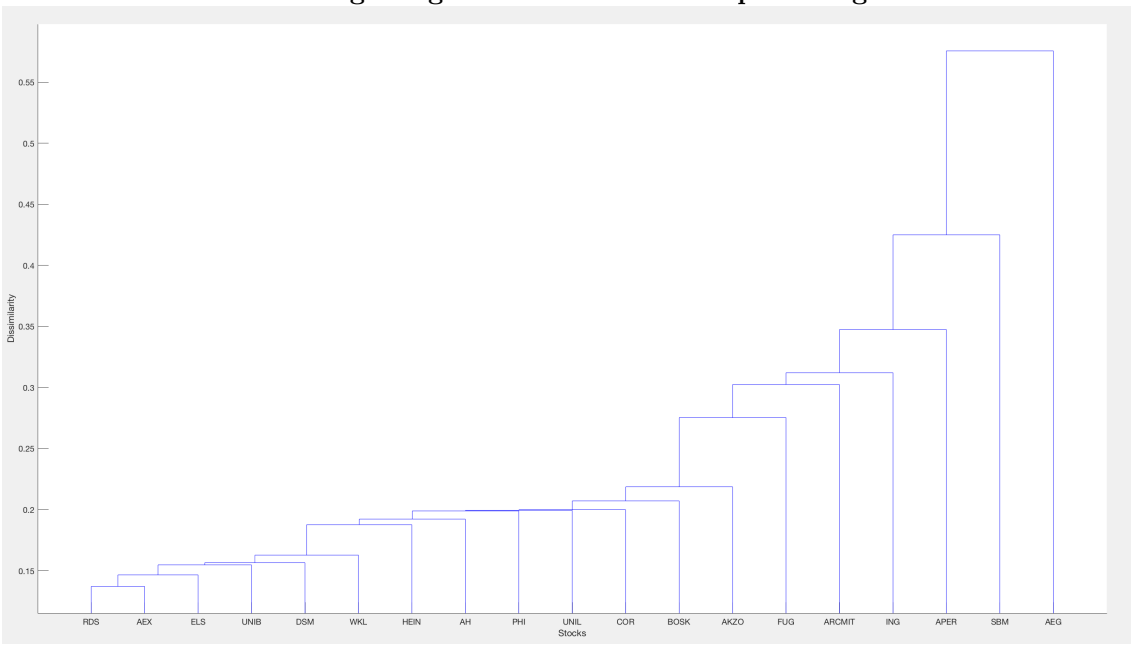

Figure 3: *shows the dendrogram produced with euclidean distance and simple linkage*

There are relatively many outliers in our dendrogram, therefore even a ward linkage method, which combines those objects whose merging increases the overall within-cluster variance to the smallest possible degree, might not solve the single-sized clusters problem. We tried it anyway, along with the average linkage method, as shown in **Figure 4** , but the results have not really improved. The dendrogram obtained by ward linkage method is very similar to the complete linkage one, while the average linkage method yuelds a very similar dendrogram to the single linkage one. This is caused by the particular structure of the data, which, with its outliers, leads to a bad element distribution between the clusters.

Dendrogram generated with average and ward linkage

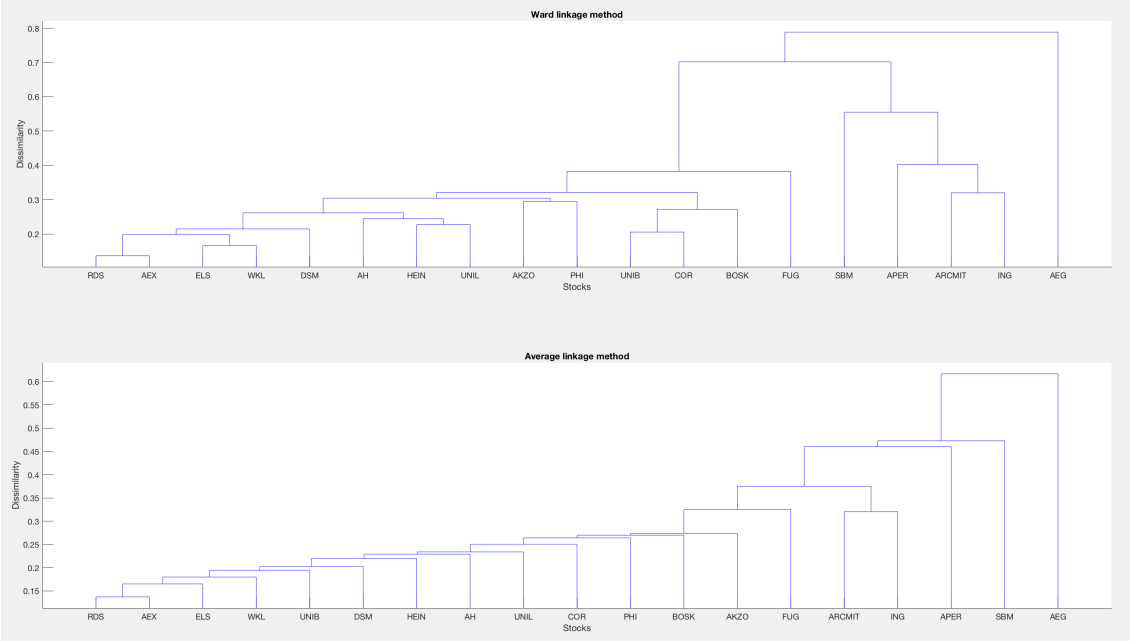

Figure 4: *shows the dendrogram produced with euclidean distance and average and ward linkages*

To decide which of the available linkage methods is more accurate we can use, as previously done, the cophenetic coefficient: **Table 1** shows the cophenetic coefficients for different linking methods, which would lead to the choice of the 'Average' method, as it has the largest coefficient.

Table 1: *Cophenetic coefficients for different linkage methods*

Average	Centroid	Complete	Median	Single	Ward	Weighted
0.9741	0.9659	0.9172	0.9492	0.9592	0.8817	0.9548

But more accuracy does not necessarily mean better clustering: as we can see in **Figure 4** , the average dendrogram produces a dendrogram similar to the single linkage one, which has the problem of single-sized clusters. Indeed if we cut this dendrogram at almost any height, we obtain a cluster with many elements and then only single-sized clusters. Thus, in this case, a complete linkage method would be a good compromise between accuracy and clusters' elements distribution.

Assignment 2

Introduction

A binary decision tree can be used to classify new incoming data based on some learning data or on some given information, as in this assignment.

Assumptions

We assumed that, if nothing is specified about the tusk, the whale does not have a tusk or it has a small one (so that the answer to the query "Has a long tusk" is No for all the whales but the Narwhal). The same assumption has been made for the frequency of the water blows. We also avoid using the length of the whale as a feature, if it is not strictly necessary, as we have no information about the length of the Beluga Whale (thus when we are considering a set of possible classes where the Beluga Whale is not present, we also consider the length as a feature to use). We do not have some training data, thus we will assume that the probability of being a certain whale species is distributed equally among all the species (therefore we can compute the impurities as if we had as learning data only 5 observations, one for each class, with exactly the values described in **Table 2**)

Tree Design Choices

Table 2 shows the features of the different whale species.

Table 2: *Features described by the biologist to distinguish the whales*

	Length	Fluke visible when dives	Has a fin	Has a long tusk	Blows water often
Beluga Whale	?	No	No	No	No
Blue Whale	Very long	Yes	Yes	No	No
Bowhead Whale	Long	Yes	No	No	No
Killer Whale	Small	No	Yes	No	Yes
Narwhal Whale	Very small	Yes	No	Yes	No

We are required to build a tree with simple queries and minimum height. As for the height, considering that we have to classify 5 species of whales and that the tree is binary, the minimum height that it can have is 3, because, if it had a smaller height, such as 2, it would have maximum 4 leaves (not enought for the 5 classes). Thus we aim to design a tree with a height of 3. Now we can analyze the features in the table. We have to choose a root feature, and to do that we analyze the impurity drop for each feature. The impurity in a node is:

$$i(N) = 1 - max_j P(\omega_j)$$

while the impurity drop is

$$\Delta i(N) = i(N) - P_L i(N_L) - P_R i(N_R)$$

Considering the assumptions made above about the data distribution, the impurity in a node is always equal to $\frac{n-1}{n}$, where n is the number of classes which have reached that node (because $P(\omega_j)$ is always $\frac{1}{n}$). Thus the only criterion which we can use to determine the impurity drop is the number of elements in each of the descendent nodes: the only possible combinations (the order does not really matter) are 1 and 4, or 2 and 3(we ignore the useless case of 0 and 5, as this would not reduce the number of possible classes in the descendent node), for the root node. In the former, the impurity drop will be:

$$\Delta i(N) = i(N) - P_L * i(N_L) - P_R * i(N_R) = \frac{4}{5} - \frac{1}{5} * 0 - \frac{4}{5} * \frac{3}{4} = \frac{4}{5} - \frac{3}{5} = \frac{1}{5}$$

while in the latter, the impurity drop will be:

$$\Delta i(N) = i(N) - P_L * i(N_L) - P_R * i(N_R) = \frac{4}{5} - \frac{2}{5} * \frac{1}{2} - \frac{3}{5} * \frac{2}{3} = \frac{4}{5} - \frac{1}{5} - \frac{2}{5} = \frac{1}{5}$$

which is the same value as the former. This means that we can choose the root feature just to minimize the height of the tree, without paying attention to the impurity drop. Thus we choose a feature which divides the set of possible classes in two sets as similarly sized as possible(2 and 3): the only two choices are "Fluke visible when dives" and "Has a fin", as we can see from **Table 2** . We choose the former, leading to a left descendent node with 3 possible classes, Blue Whale, Bowhead Whale and Narwhal Whale, and a right descendent node with 2 possible classes, Beluga Whale and Killer Whale (we chose left and right arbitrarily, but to have more order we will always create a left descendent node with all the classes which respond positively to the query, and a right descendent node with all the others), as shown in **Figure 5** .

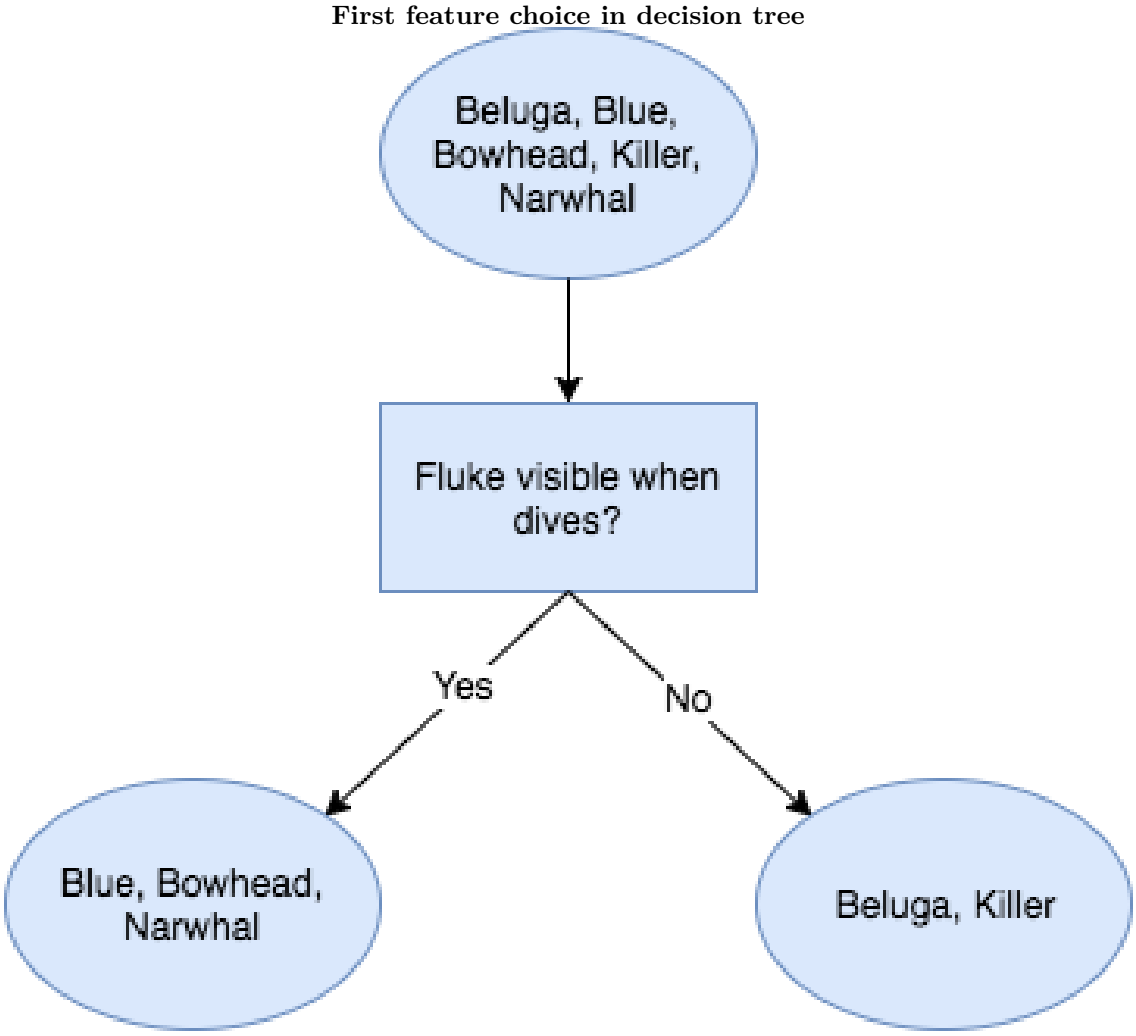

Figure 5: *shows the beginning of the decision tree and the classes division after using the first feature*

At this point, to complete the right sub-tree, we have only one choice, which is to split the possible classes in two leaves: since the two classes to split are Beluga and Killer Whale, we can use the features "Length", "Has a fin" or "Blows water often", for which the two classes have different values. We choose the second one, leading thus to two leaves and completing this part of the tree.

On the other part, we still have a node with 3 possible classes: also in this case, there is only one possible splitting, which divides the 3 classes in a group of 2 and a leaf. Looking at **Table 2** , we can see that some acceptable features for this are "Length", "Has a fin", "Has a long tusk": we choose the second one. The choice leads to a left leaf descendent node, which represents the Blue Whale, and a right descendent node, which has still two possible classes, namely Bowhead and Narwhal Whale. At this point, as previously, the only choice we have is to split it in two descendent leaves, and the features which can accomplish this are "Length" and "Has a long tusk": we choose the second one. So we obtain the last two leaves and we have a complete tree, which minimizes its height and maximizes the impurity drop at each node. As for the simplicity of the queries, we decided to avoid using the length, as it would have made the query more complicated (we would have had to specify a length and check if the whale was longer than that or not) and it is also more difficult to measure only by seeing a whale. We also avoided the "Blows often water" feature, as it is not an objective quality (often is not well defined) and it requires time to decide if a whale has this feature. Considering this, we could build a smaller table, **Table 3** , containing only the features we actually need to classify new whales:

Table 3: *Features selected to classify the whales*

	Fluke visible when dives	Has a fin	Has a long tusk
Beluga Whale	No	No	No
Blue Whale	Yes	Yes	No
Bowhead Whale	Yes	No	No
Killer Whale	No	Yes	No
Narwhal Whale	Yes	No	Yes

Following all the previous considerations, we have built the tree in **Figure 6** .

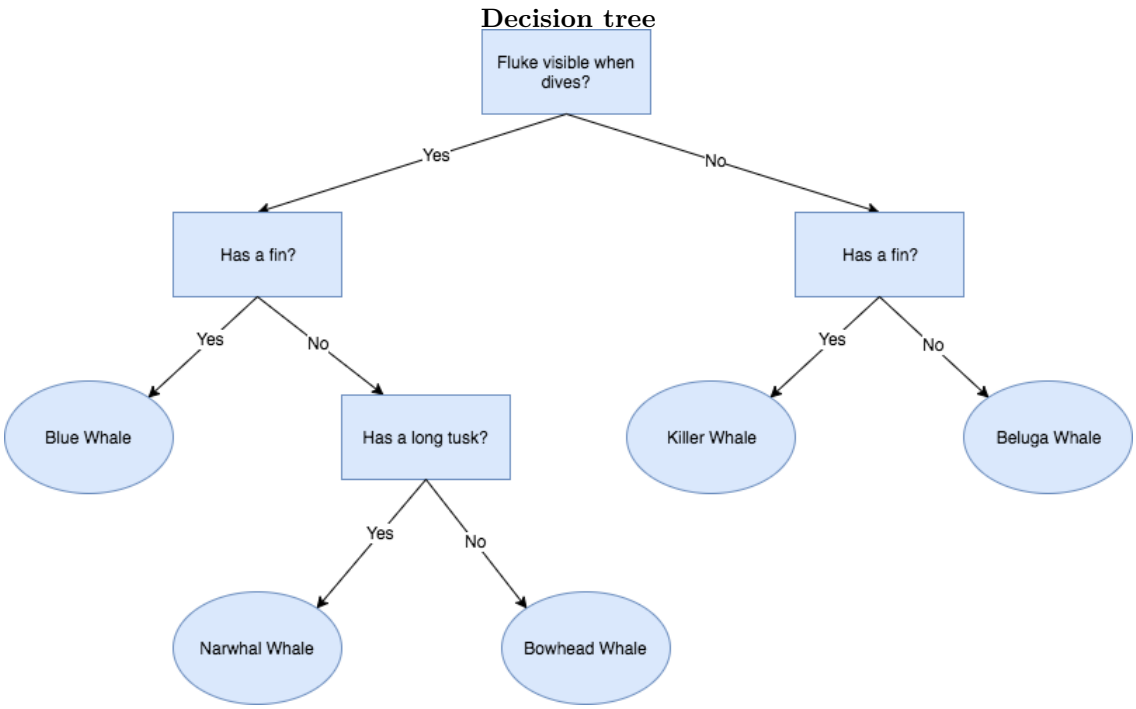

Figure 6: *shows the best possible decision tree in terms of height and query simplicity*