LAB 3 MACHINE LEARNING

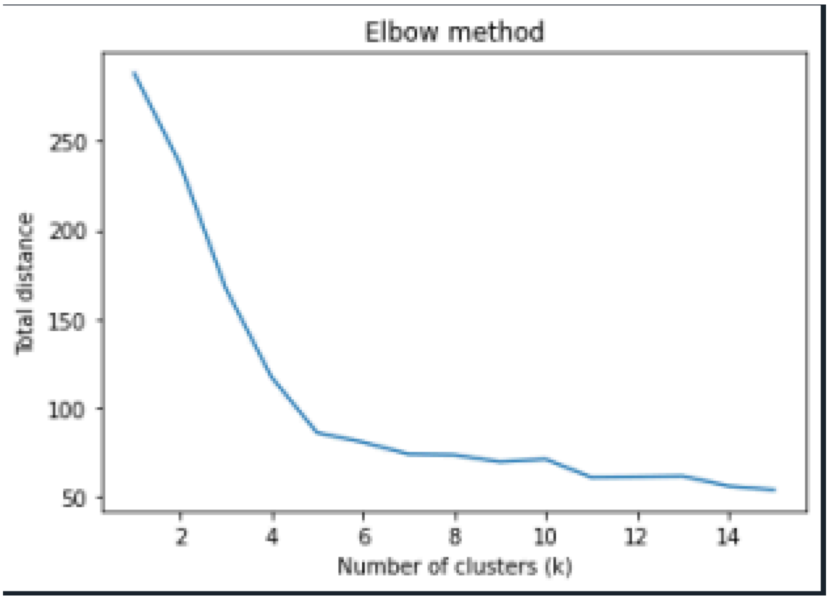
Ted Ljungsten, Viktor Abraham

K-MEANS ALGORITHM

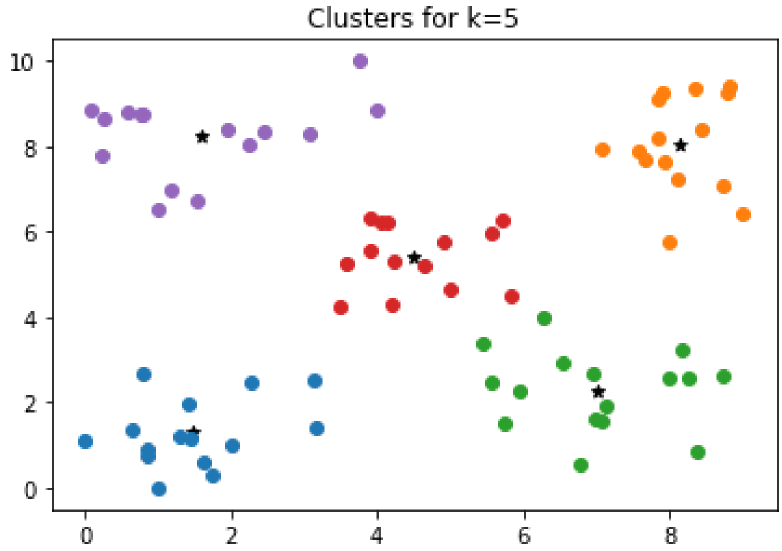
1. **Kmeans algorithm**
   You define how many clusters of data you want to create. Then, you randomly initialize k many centroids. Then, you look at all the data in the dataset and map them to the nearest centroid by calculating the distance between one centroid and the data point in the dataset. The nearest centroid to the data point gets allocated to each other. All points closest to one centroid create a cluster or a group of data points. Each cluster receives a different color in our code to indicate which cluster it belongs to.

   The centroids could be better placed when placed randomly, so you iterate this code for a set amount of iterations or until your centroids are optimally positioned. This optimization is done after each iteration, taking the mean value of all data points and setting this point as a new centroid. This is done for all clusters, and once done, we compare the old centroid to the new centroid by looking at the mean distance from centroids to the data points. If the distance is shorter now than before, we set this as the new centroid and redo the process all over again. We are Allocating data points to centroids and calculating new centroids. We do this for a maximum amount of iterations or until the improvement is negligible.

2. Here is the elbow plot produced by our clustering algorithm. 5 clusters are optimal for this dataset since adding more clusters increases the total average distance for all the points to a centroid.



3. Here is the optimal clustering plot with 5 clusters represented by blue, orange, red, green and purple.



4. We can get different results based on where the centroids are placed at the beginning since we look at the closest point to a centroid and take a mean. This results in inconsistency if the centroids are randomized since they get placed closer to other data points than in the first run. This is why, in our code, we set a seed for the randomization, making it so that our centroids are always placed in the same spot. This means this is the optimal number of clusters based on our randomized seed of centroids. Changing this random seed will result in different optimal amounts of clusters.

classification

1. The ID3 algorithm is an algorithm that takes a dataset and divides it into smaller subsets of data. This is done by calculating the entropy of all attributes in the dataset. We select the attribute with the smallest entropy and highest information gain, we then select this attribute with the lowest entropy and split the dataset to create a subset of the original dataset with this node as the root and this procedure is repeated for all subsets. The recursion stops once we cannot split anymore or until a satisfied depth is reached.

2. The first thing that happens is that the code reads the dataset from the provided golf_dataset file containing the input features and target Play Golf. Then, the provided decision tree code constructs a tree with a set height of 3 using our implemented Split function.
   Once the tree is generated to height 3, the code outputs the decision tree in text, and it looks like this:

```
Node<1>
  Non leaf node - Parent: None
  Split variable: Outlook
    Child_node: 2, split_value: Rainy
    Child_node: 5, split_value: Overcast
    Child_node: 6, split_value: Sunny
Node<2>
  Non leaf node - Parent: 1
  Split variable: Humidity
    Child_node: 3, split_value: High
    Child_node: 4, split_value: Normal
Node<5>
  Leaf node - Parent: 1, Decision: ('Yes',)
Node<6>
  Non leaf node - Parent: 1
  Split variable: Windy
    Child_node: 7, split_value: False
    Child_node: 8, split_value: True
Node<3>
  Leaf node - Parent: 2, Decision: ('No',)
Node<4>
  Leaf node - Parent: 2, Decision: ('Yes',)
Node<7>
```

```
  Leaf node - Parent: 6, Decision: ('Yes',)
Node<8>
  Leaf node - Parent: 6, Decision: ('No',)
```

We can look at “parent: “ to determine our depth. If the parent is none, then we are at depth 1. If the parent is 1, then the depth is 2. If the parent is two, then the depth is 3.

3. We gain more information about the relationships between the target and the different attributes for each level of the tree. At maximum height one we get the least complex version of the decision tree and also the least amount of information from the tree. We miss out on a lot of the critical relationships between the different attributes. Increasing the depth increases computation time but also gives us more accurate results. With a height of 2, we see relationships that were missed at the height of 1 at the cost of a more complex tree. With a depth of 3, we reach the optimal solution since we now go deep enough into the decision tree to see the relationships between all attributes in our dataset. We see both the relationship between the humidity and the wind factor but also pick up on the relation between wind and humidity in combination with a sunny day.

4. Confusion matrix

	Prediction Yes	Prediction No
Actual Yes	5	1
Actual No	2	6

Underfitting and overfitting:

Underfitting occurs when a model is too simple to capture some patterns in the data. This is what we see if we reduce the height of the decision tree, as mentioned above in the text. This results in some patterns not being detected, and as a result, the model's performance becomes very poor. However, this is not the case for our model since the confusion matrix only predicted wrong on one occasion.

Overfitting is, instead what happens when the model is too complex and recognizes false patterns in data. Increasing the height of the model could result in the model instead of recognizing patterns instead of simply memorizing the training data. This results in excellent training data performance since the model “remembers” the outcome of specific patterns instead of finding the general patterns that make up the solution. This means the model has excellent performance on the data it recognizes. However, it does not perform well in new situations. This can be seen in a confusion matrix with more false positives and negatives due to the model no longer recognizing the data making mistakes.

4.2 How to run?

To run the k-means code, you just need to run the class named kmeans.py and the code is written in python 3.8.10.
To run the ID3 code you do the same as k-mean but you run ID3.py and is also written in python 3.8.10.