UNIT – 2

K-Nearest Neighbor classifier

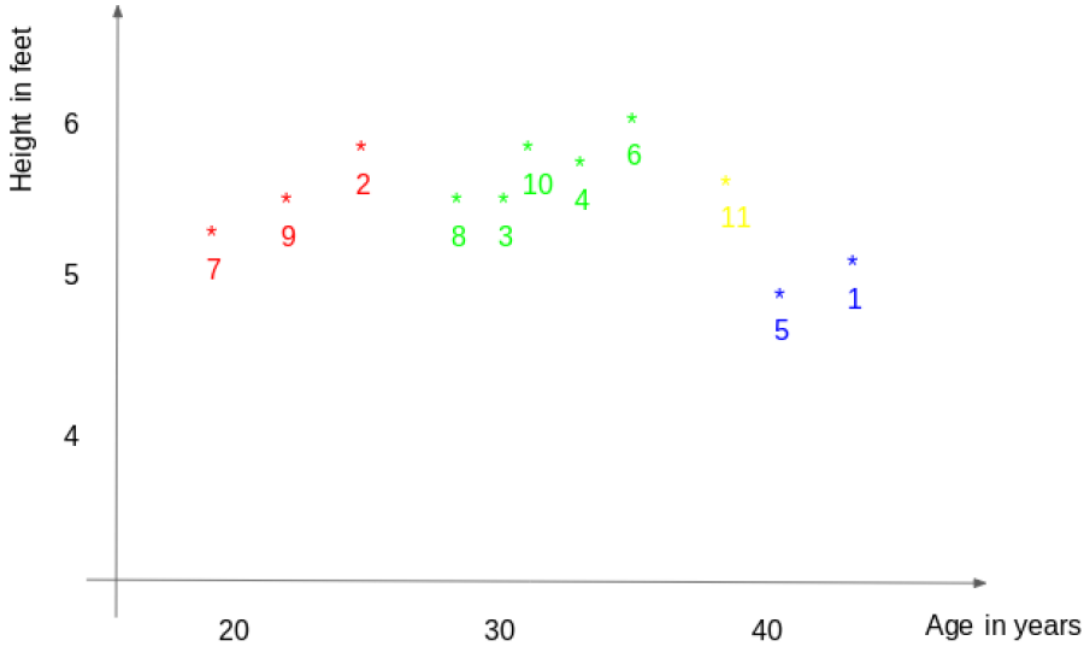
KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses _feature similarity' to predict the values of any new data points

Let us start with a simple example. Consider the following table – it consists of the height, age and weight (target) value for 10 people. As you can see, the weight value of ID11 is missing. We need to predict the weight of this person based on their height and age.

Note: The data in this table does not represent actual values. It is merely used as an example to explain this concept.

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

For a clearer understanding of this, below is the plot of height versus age from the above table:



In the above graph, the y-axis represents the height of a person (in feet) and the x-axis represents the age (in years). The points are numbered according to the ID values. The yellow point (ID 11) is our test point.

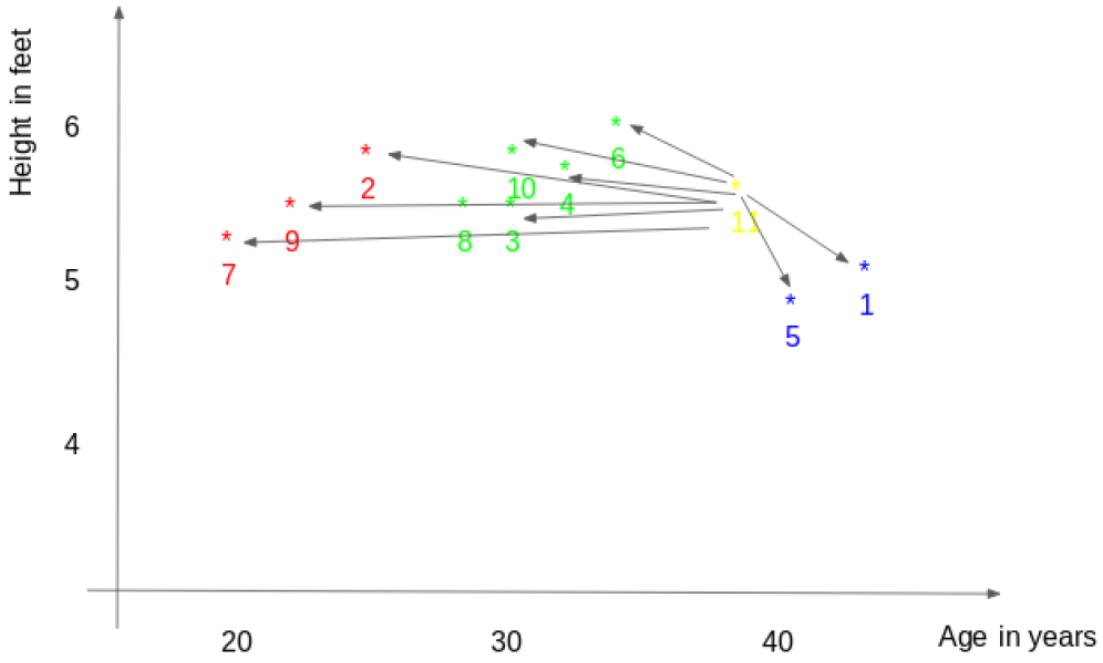
If I ask you to identify the weight of ID11 based on the plot, what would be your answer? You would likely say that since ID11 is **closer** to points 5 and 1, so it must have a weight similar to these IDs, probably between 72-77 kgs (weights of ID1 and ID5 from the table). That actually makes sense, but how do you think the algorithm predicts the values? We will find that out in this article.

As we saw above, KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses **_feature similarity**‘ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. From our example, we know that ID11 has height and age similar to ID1 and ID5, so the weight would also approximately be the same.

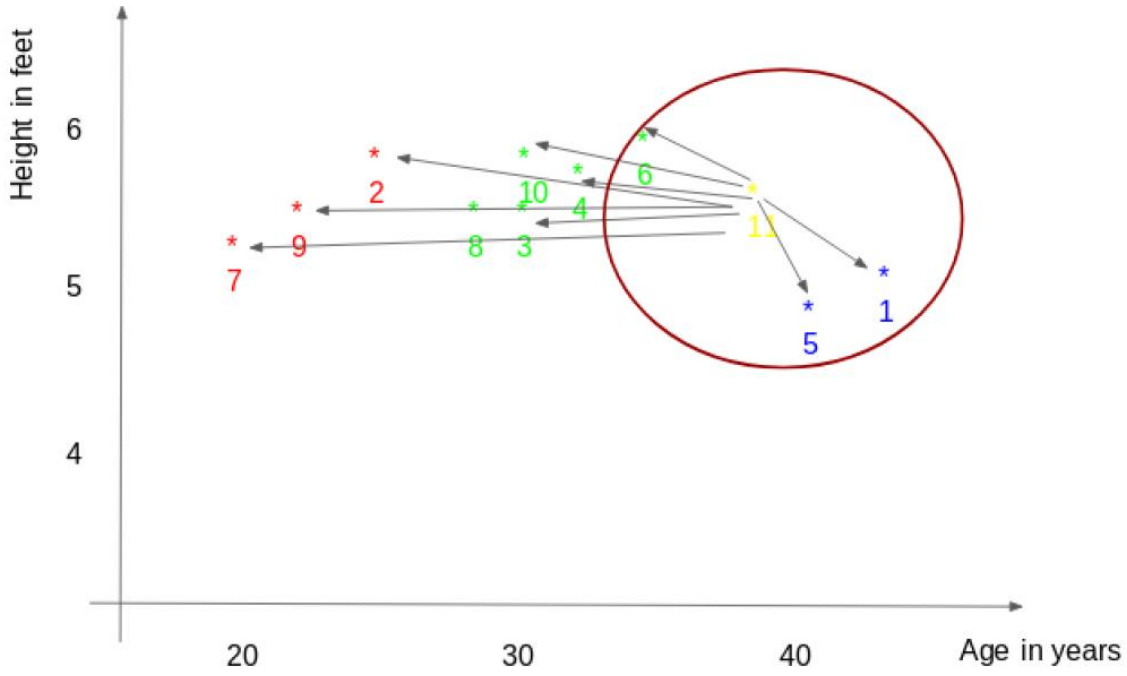
Had it been a classification problem, we would have taken the mode as the final prediction. In this case, we have two values of weight – 72 and 77. Any guesses on how the final value will be calculated? The average of the values is taken to be the final prediction.

Below is a stepwise explanation of the algorithm:

1. First, the distance between the new point and each training point is calculated.



2. The closest k data points are selected (based on the distance). In this example, points 1, 5, 6 will be selected if the value of k is 3. We will further explore the method to select the right value of k later in this article.



3. The average of these data points is the final prediction for the new point. Here, we have weight of ID11 = (77+72+60)/3 = 69.66 kg.
In the next few sections, we will discuss each of these three steps in detail.
3. Methods of the calculating distance between points

The **first step** is to calculate the distance between the new point and each training point. There are various methods for calculating this distance, of which the most commonly known methods are – Euclidian, Manhattan (for continuous) and Hamming distance (for categorical).
1. **Euclidean Distance:** Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (y).

2. **Manhattan Distance**: This is the distance between real vectors using the sum of their

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

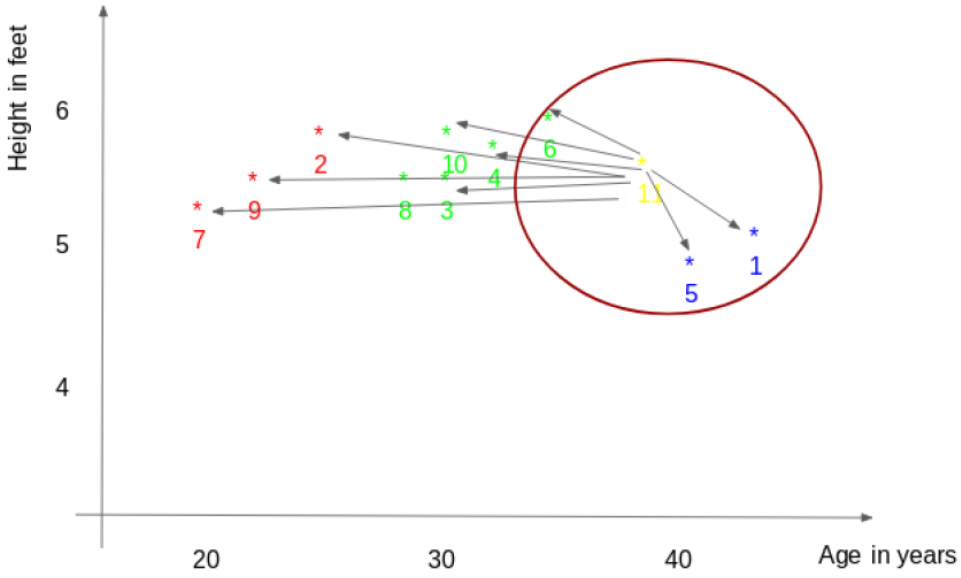
absolute difference.
3. **Hamming Distance**: It is used for categorical variables. If the value (x) and the value (y) are the same, the distance D will be equal to 0 . Otherwise D=1.

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Once the distance of a new observation from the points in our training set has been measured, the next step is to pick the closest points. The number of points to be considered is defined by the value of k.
4. How to choose the k factor?
The **second step** is to select the k value. This determines the number of neighbors we look at when we assign a value to any new observation.
In our example, for a value k = 3, the closest points are ID1, ID5 and ID6.



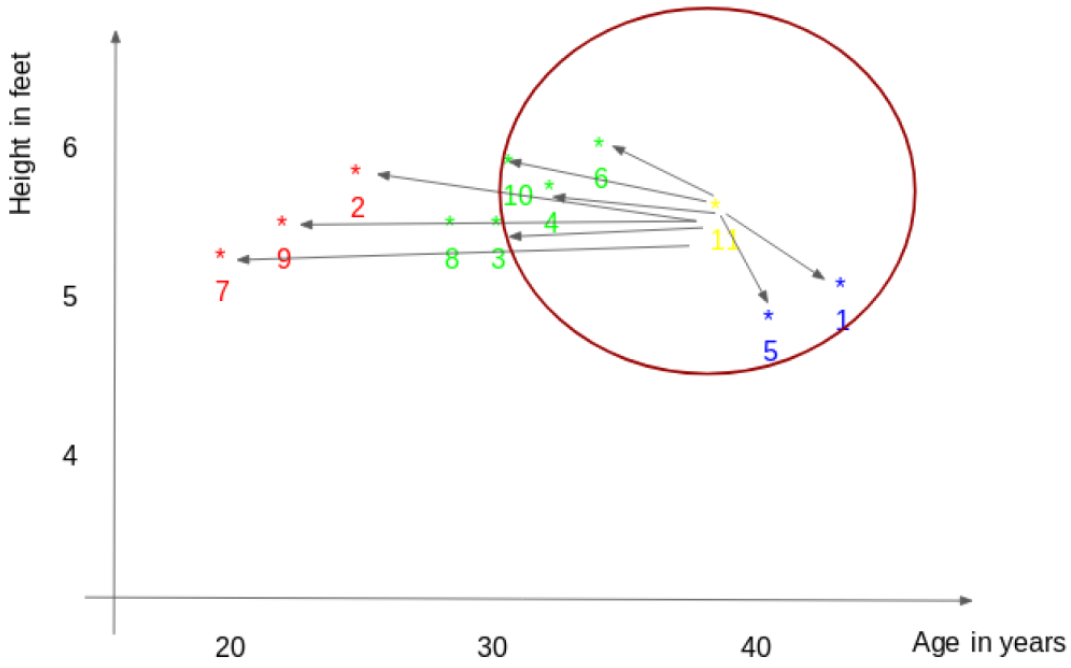
ID	Height	Age	Weight
1	5	45	77
5	4.8	40	72
6	5.8	36	60

The prediction of weight for ID11 will be:

ID11 = (77+72+60)/3

ID11 = 69.66 kg

For the value of k=5, the closest point will be ID1, ID4, ID5, ID6, ID10.



ID	Height	Age	Weight
1	5	45	77
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
10	5.6	32	58

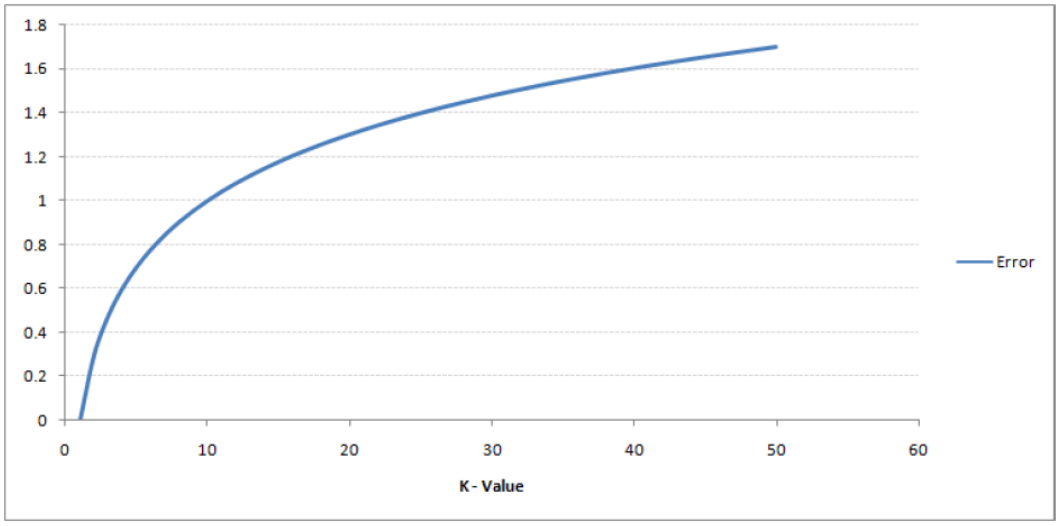
The prediction for ID11 will be :

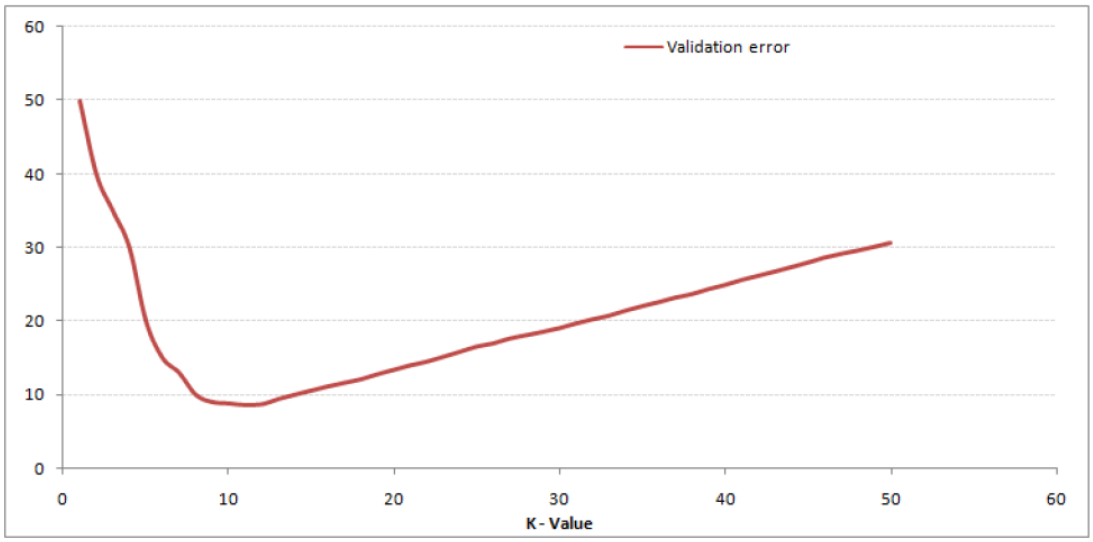
ID 11 = (77+59+72+60+58)/5

ID 11 = 65.2 kg

We notice that based on the k value, the final result tends to change. Then how can we figure out the optimum value of k? Let us decide it based on the error calculation for our train and validation set (after all, minimizing the error is our final goal!).

Have a look at the below graphs for training error and validation error for different values of k.





For a very low value of k (suppose k=1), the model overfits on the training data, which leads to a high error rate on the validation set. On the other hand, for a high value of k, the model performs poorly on both train and validation set. If you observe closely, the validation error curve reaches a minima at a value of k = 9. This value of k is the optimum value of the model (it will vary for different datasets). This curve is known as an '**elbow curve**' (because it has a shape like an elbow) and is usually used to determine the k value.

You can also use the grid search technique to find the best k value.

Reference:

1. https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/understanding-multivariate-classification.htm#:~:text=Each%20grouping%20of%20features%20is,stored%20in%20a%20signature%20file.