HOMEWORK #3

- Alexander Shender: 328626114
- Samuel Panzieri: 336239462

Question 1.

The claim is incorrect.

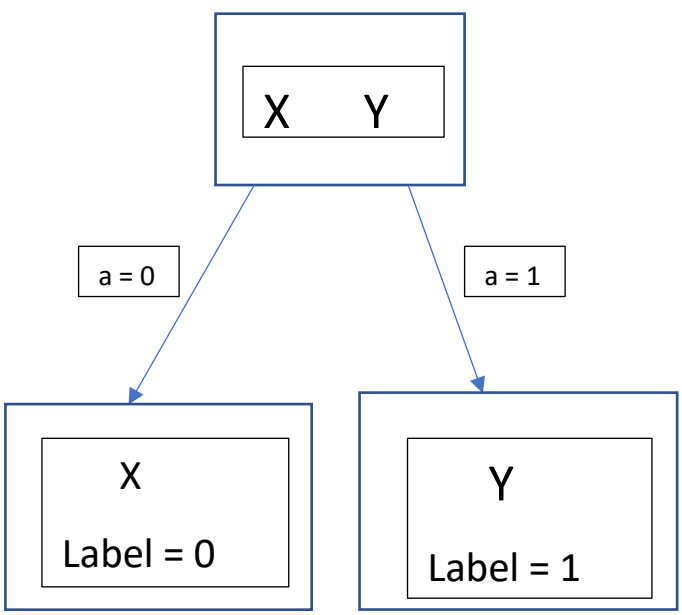
Example:

Assume:

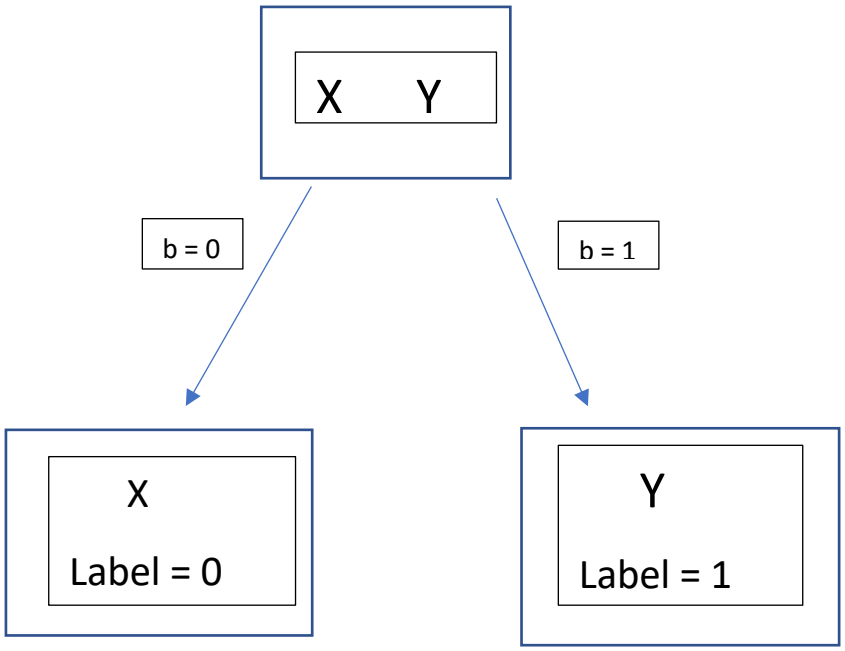
$$D = \{x, y\}$$
$$a \in \{0,1\}, b \in \{0,1\}$$
$$x.a = 0, x.b = 0$$
$$y.a = 1, y.b = 1$$
$$x.label = 0, y.label = 1$$
$$\forall x \in D \;\; f(x) = x$$

Meaning the function is the identity function.

Then if we build a possible decision tree T:



The tree T_a :



Now let's look at w, a test example for which holds:

$w.a = 0, w.b = 1$

We can see how the first tree will produce a label 0 whereas the second one will produce a label 1.

b)

The claim in incorrect.

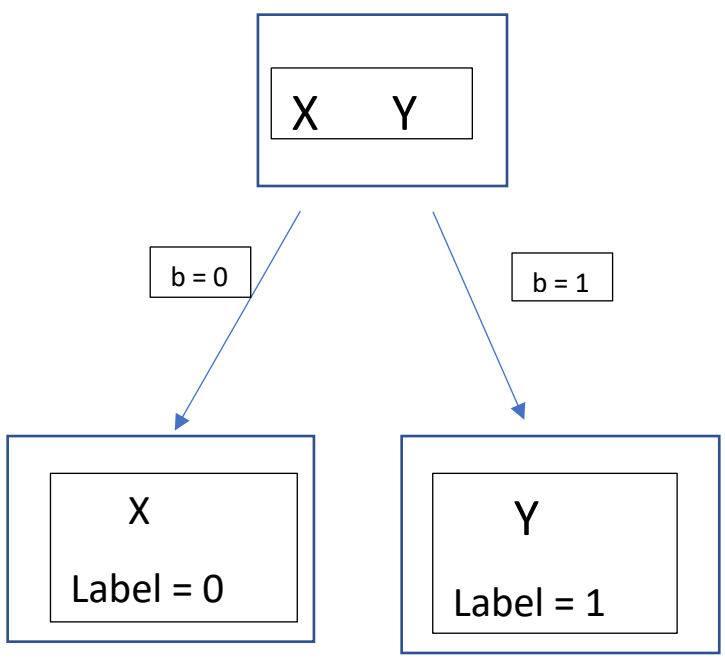
Example:
assume:

$$D = \{x, y\}$$
$$a \in \{0\}, b \in \{0,1\}, c \in \{0,1\}$$
$$x.a = 0, x.b = 0, x.c = 0$$
$$y.a = 0, y.b = 1, y.c = 1$$

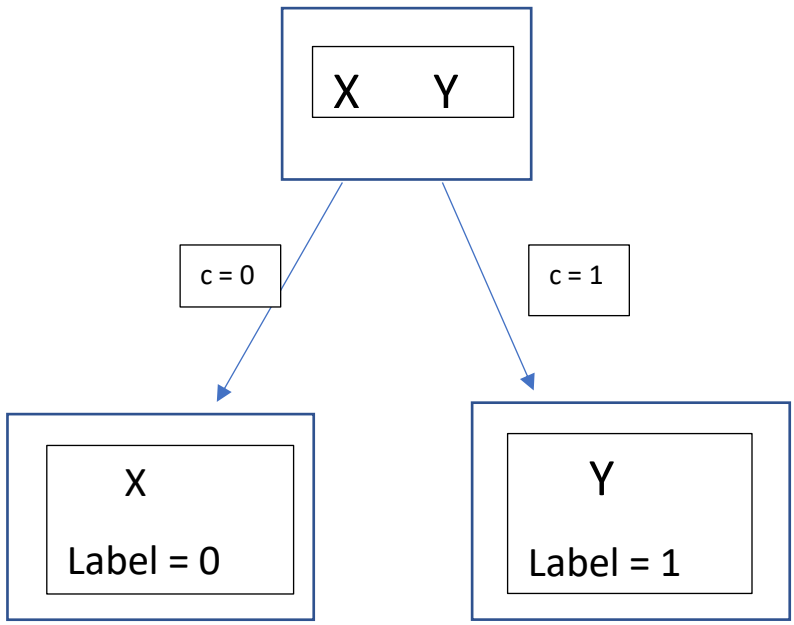
$$x.label = 0, y.label = 1$$

$$\forall x \in D \;\; f(x) = 0$$

The decision tree T:



And the tree T_b:



Now for a test example w which holds:

$w.a = 0, w.b = 1, w.c = 0$

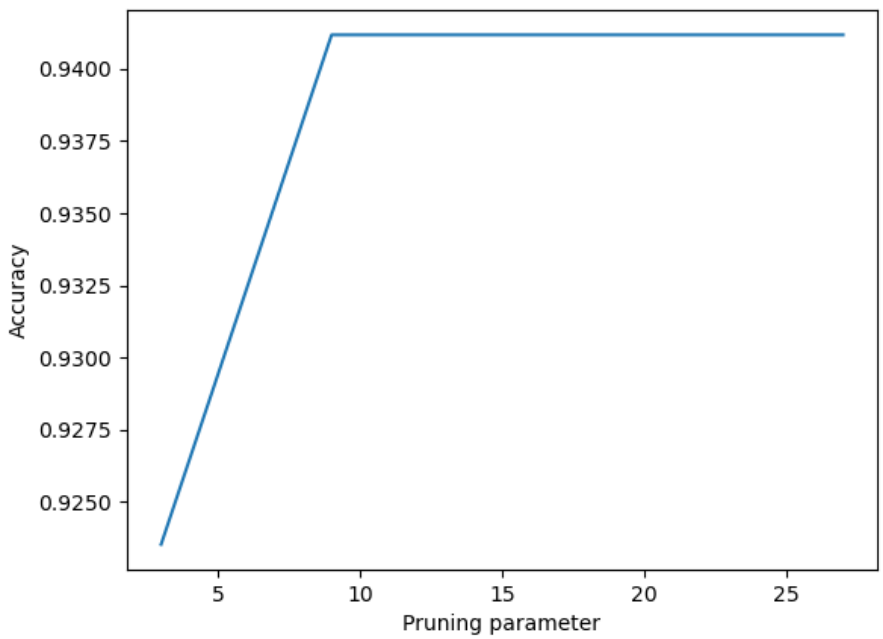
We can see that the first tree will output the 1 label and the second one the 0 label

Question 2.

The accuracy that was received is 0.9235

Question 3.

The pruning was implemented, and the following graph was obtained for difference x parameters:



Conclusions:

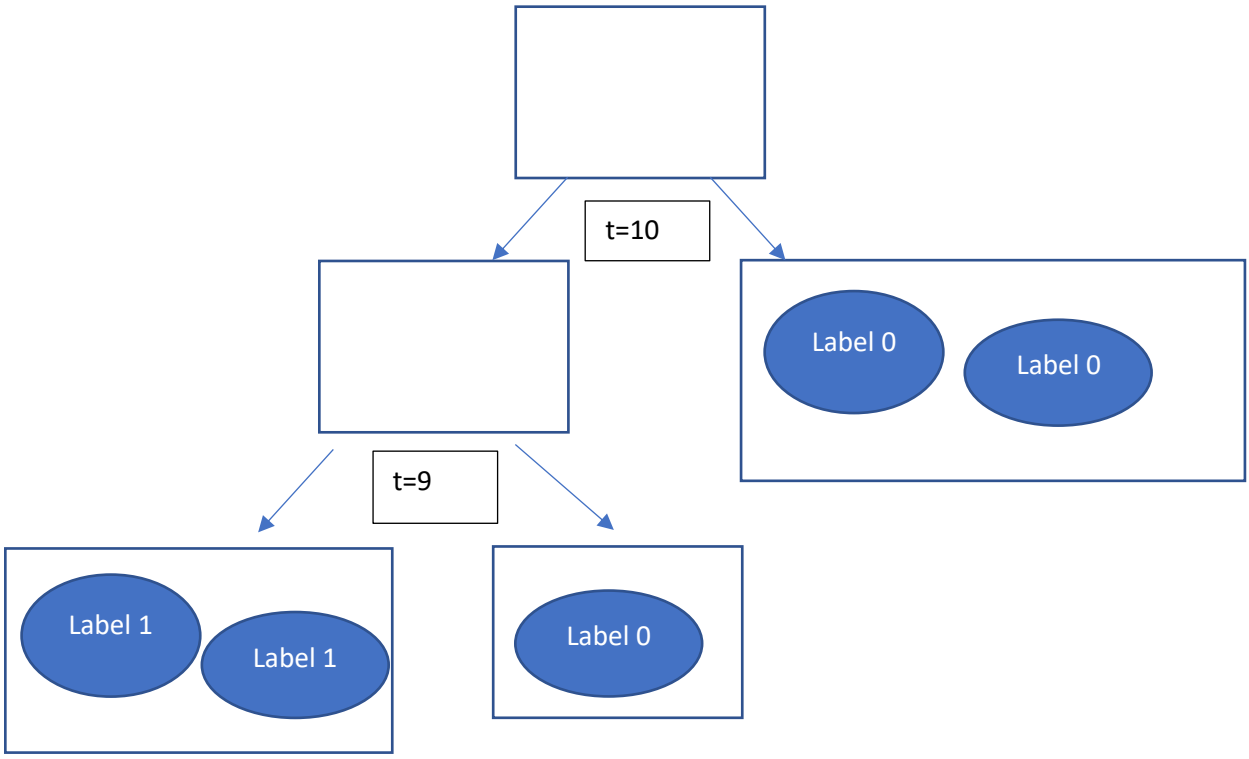
1. We can see that certain pruning does improve the accuracy of the decision tree, and reduces the over-fitting
2. Although it wasn't required to display, if we increase the pruning parameter even further, the accuracy will eventually start decreasing

Question 4.

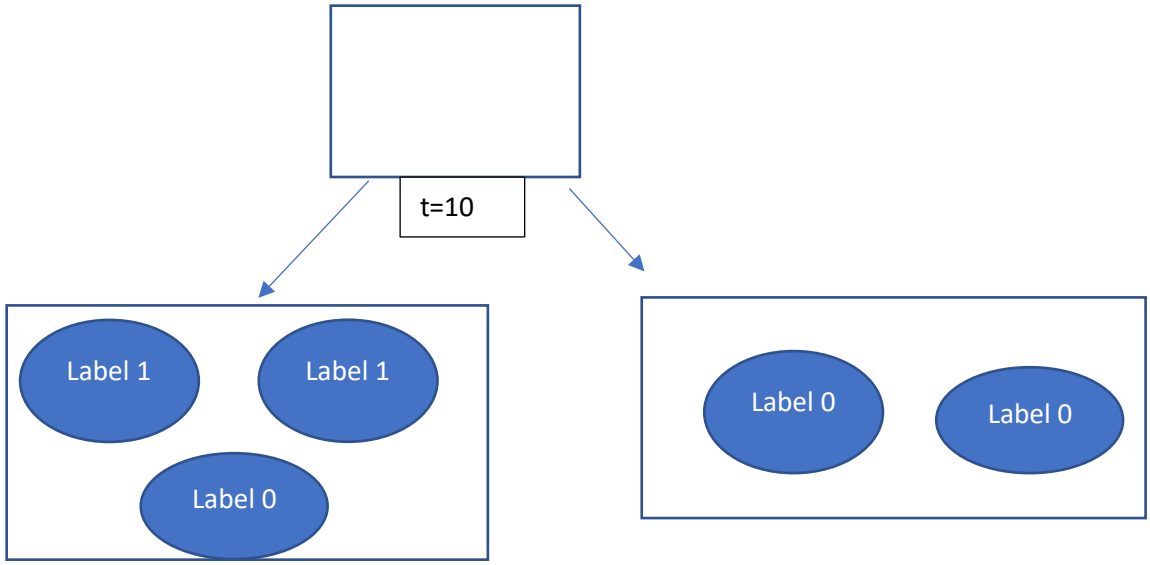
The claim is incorrect.

Example:

The tree T:



The tree T':



With an example with f = 9.5 T' will output 1 as a label.

With epsilon < 0.5, at the first node 9,5 will not be in the epsilon neighborhood of 1 and thus will go left. On the second node again for the same reason we will go right since 9.5 is not in the epsilon neighborhood of 9 and thus will go right and T will output 0.

For epsilon >= 0.5 9.5 is in the epsilon neighborhood for both nodes and thus the decision is based on the majority of the leaves, meaning 0.

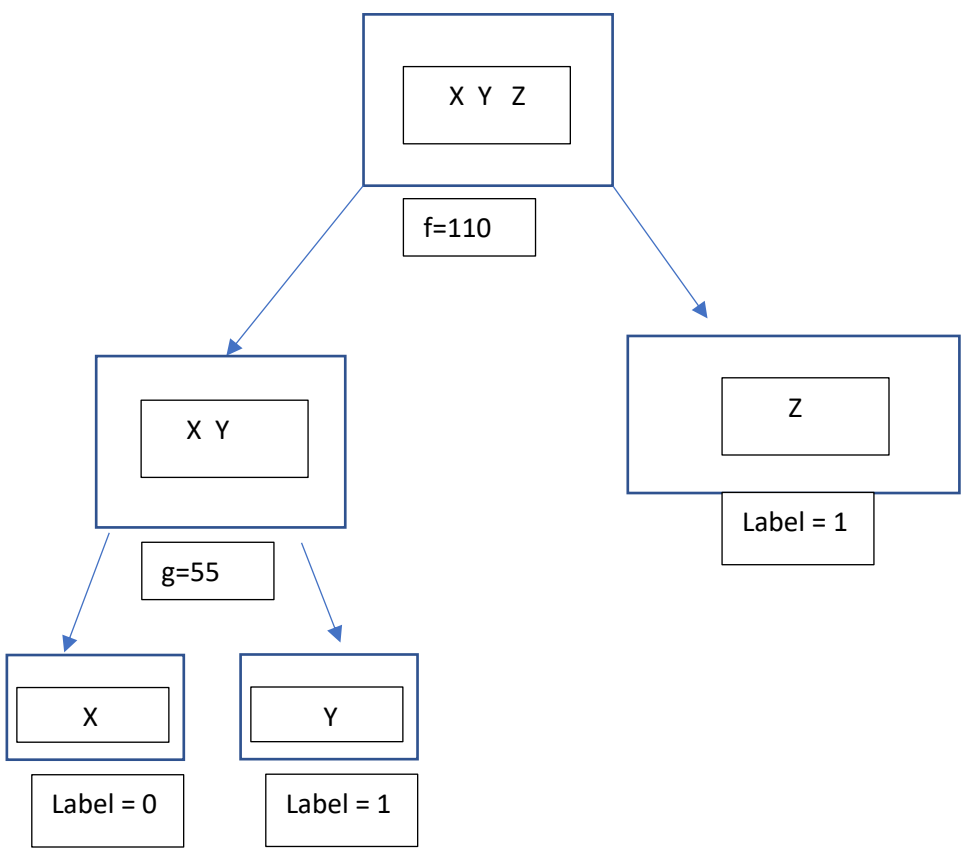
Thus for every epsilon the tree T will output 0 and T' will output 1.

Question 5.

The claim in incorrect.

Example:
assume:

$$D = \{x, y, z\}$$
$$x.f = 20, x.g = 10, x.label = 0$$
$$y.f = 20, y.g = 100, y.label = 1$$
$$z.f = 200, z.g = 10, z.label = 1$$



With epsilon = 0.01 the trees T_l and T_l' will be equal since there are no test example that are inside the epsilon neighborhood of the limit value.

Moreover with a test example that has: f = 109.9, g = 54.9:

Following the regular choosing rule with the above tree we will get 0 .

With the Epsilon rule we will get 1.

Thus the tree T_l with the regular rule and the tree T_l' with the epsilon rule will output differently.

Question 6.

The code was implemented, and the accuracy received for Epsilon-Decision-Tree with depth limit of 9 is 0.9. It is worth noting that if the default value for the case where the classification list (list of classification of leaf nodes that were reached) is 0 instead of 1, the accuracy will increase to 0.929, because most of the samples from the test set have classification False.

More conclusions:

- This algorithm doesn't account for the number of examples in each leaf node, giving each node similar weight when the final classification is calculated.
- We can see that the performance for this tree is worse / equal to the one using the simple tree (even without pruning). There were 0 cases where the simple tree (no epsilon) gave incorrect result, and the Epsilon-classification gave correct result. But there were cases when simple classification (no epsilon) gave correct result, and epsilon tree gave incorrect result, because it had reached 2 leaf nodes with classification 0 and 1, and chose 1 (as preference).

Question 7.

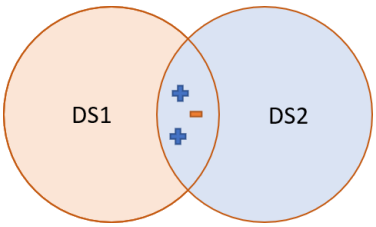
This statement is **FALSE**.

The proof:

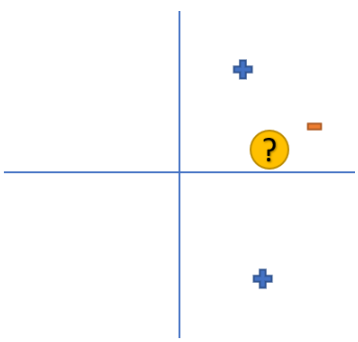
Let's suppose we have 3 examples in our sets:

- Examples 1, 2, 3 appear in BOTH $DS1$ and $DS2$

Thus, they appear in both of the sets: $DS1 \cup DS2$ and $DS1 \cap DS2$ (once in each of those sets).

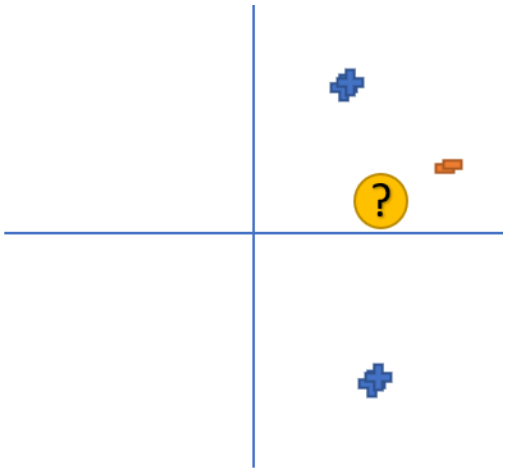


Let's say for simplicity it's 2D. The new test example is marked with "?" sign and we use the 3NN to classify it:



Using the 3NN method, it will be classified as TRUE for both : $DS1 \cup DS2$ and $DS1 \cap DS2$

But when we classify it with the $DS1 + DS2$ group, we receive the following:



(In this example the points are in the SAME location, but are slightly moved to show that they appear twice).

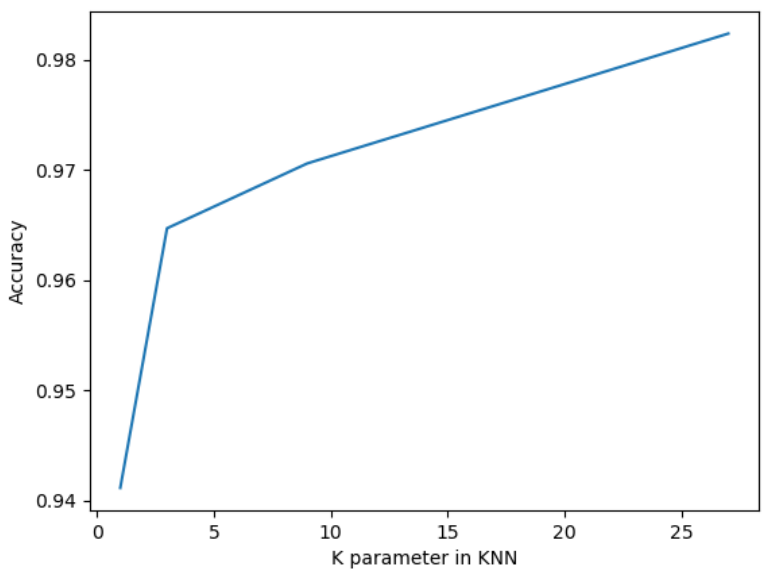
Now, the 3 closest points include two FALSE classifications, which means, the 3NN classification will be FALSE. Thus, this proves the statement is wrong.

Question 8.

The code was implemented, and the result for k=9 is 0.9706

Question 9.

The result:



Conclusions:

1. We can see that increasing the K parameter the accuracy increases as well. Taking more neighbors into account helps for the classification (until certain level)
2. We can see again that we haven't reached the 'peak' of the accuracy. Increasing the K values even more will eventually decrease the accuracy, since more distant samples will be taken into account

Question 10.

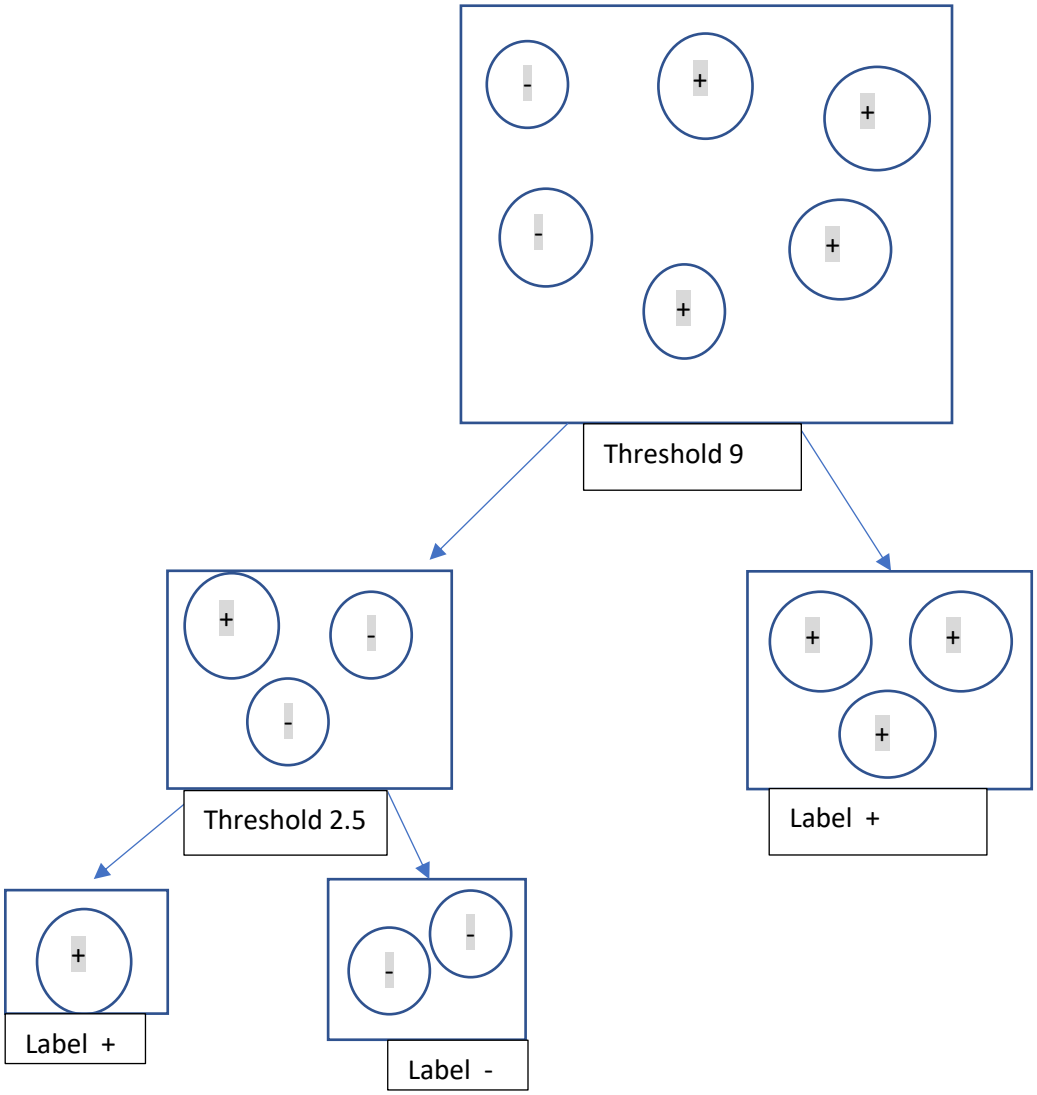
We will show that for epsilon = 1, the two classifiers will classify the same way.

The 3NN classifier will have a threshold value of 7.5 so each example with f < 7.5 will be classified as -, and the others as +

Thus the examples with f = 8,10,12,13 will be classified as +

The examples with f = 2,3 will be classified as –

Now the decision tree for the given training set:



The Maximum IG value is obtained with 9 as a threshold for the first node and for the second node with the value of 2.5

- f = 2 is not inside the epsilon neighborhood for the first node, but it will be for the second so the classifier will output –
- f = 3 is not inside the epsilon neighborhood for the first node, but it will be for the second so the classifier will output –
- f = 8 is inside the epsilon neighborhood for the first node, but it will not be for the second so the classifier will output +
- f = 10 is inside the epsilon neighborhood for the first node, but it will not be for the second so the classifier will output +
- f = 12 is not inside the epsilon neighborhood for the first node, so the classifier will output +
- f = 13 is not inside the epsilon neighborhood for the first node, so the classifier will output +

Question 11.

The code is implemented. The accuracy that was reached is 0.917.

Conclusions:

- We can see improvement over the simple Epsilon-Decision-Tree. Indeed, now the size of the samples at each node are taken into account. Indeed, if we look closer, we see that most samples that reached 2 leaves by using by simple Epsilon-Decision-Tree, which had classification results of [0, 1] (we choose '1' in case of 'tie' between 0 and 1) and are classified incorrectly using simple Epsilon-Decision-Tree are classified correctly using the addition of KNN, because KNN algorithm is used on the actual samples at the leaf nodes, and not on the classification of the leaf node.
- The regular KNN classifier with appropriating K parameter still gives better result. From short insight, most of the "wrong classification" samples got all wrong classification samples at the leaf nodes that were reached, so there was no chance to get the right classification.

Notes:

- The required code files are submitted, but also the file ID_total.py. This file includes the classifiers for questions 2,3,6,11.
- To save time, the trained tree classifiers were saved as ".pkl" files and are loaded upon need to classify new data. Using the parameter "FORCE_NEW" as 1 will allow the algorithm to load the classifiers that were generated before.
- In files DT.py, DT_epsilon.py, KNN_epsilon.py : all the classes and functions are GENERIC and defined in DT_total.py