Decision tree

My implementation of the ID3 algorithm is based on the ratio gain, it works recursively by splitting the dataset along the feature that maximize this value until there either are no features left to decide or no diversity among the classes of the target variable. The ratio gain is the information gain divided by the information split, which is the entropy of the feature. As such, feature with high entropy will divide more the information gain and thus will be less prioritized than features with lesser entropy. This allows to correct the bias of maximizing with the information gain, which tend to prefer features with lots of classes.

It is suited for problems where features are discrete, though they are ways to make it work for continuous values. Also, it is really good when visualization and confirmation of the results is needed because the predict algorithm and the decision tree work very much like human thinking, and it is very easy to understand the results of the computation by outputting the tree (see fig. 2,3). It also needs the training set to be fairly complete as there can be many cases not covered and that the algorithm won't infer (see fig. 1). Moreover, it assumes that all features are equally important to decide the target variable, while there can be some that don't add any actual information to the problem, like the birth month of the second data set.

To get around this issue, I first tried to replace how the splitting decision was made using ratio gain instead of information gain, as explained above, but it didn't work. Therefore, after a discussion with another student, I did some feature engineering on the data set, decided that the 'Birth Month' feature added nothing and removed it by adding a hyper parameter to my class, called ignored parameter.

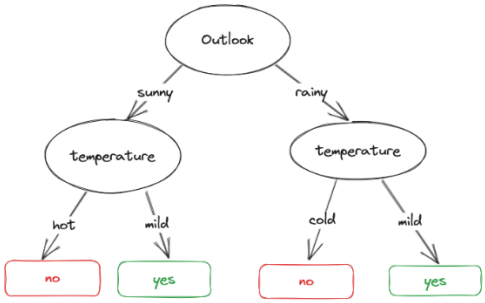

Figure 1: the algorithm can't handle the case Outlook : Sunny, Temperature : Cold.

Below are the results I obtained on the first and second dataset:

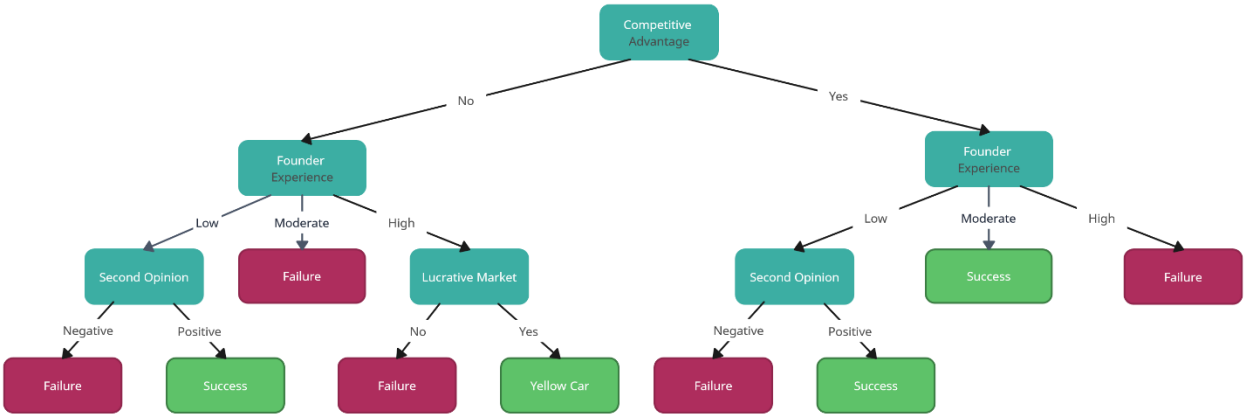

Figure 2: decision tree for the second dataset - Birth Month. Accuracy : 92%

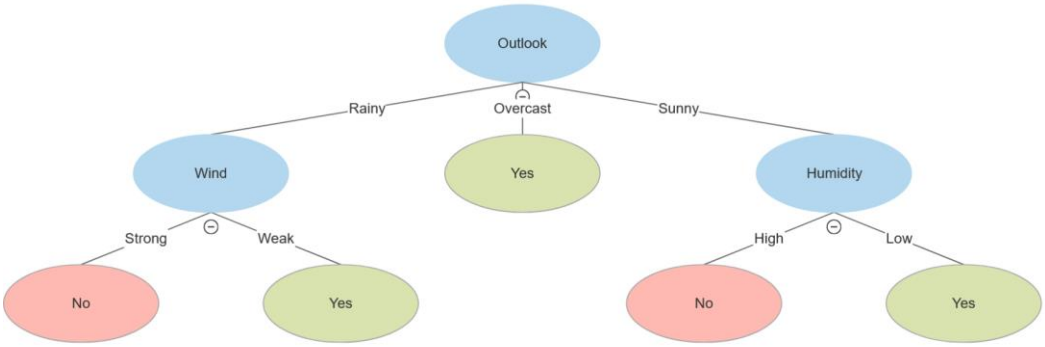

Figure 3: first dataset decision tree. Accuracy 100%

K-Means

The algorithm is very simple : pick K centroids, where k is an hyperparameter, for each point compute the closest centroid, compute the means of all points of a centroid for each centroid, do it again until the sum of the errors (distances) between each point and their centroids becomes smaller than a degree of tolerance, hyperparameter.

It is suited to find clusters of points in any dimension, given that we have a way to find how many clusters there should be, without having to specify any rules to group the points. It needs the features to be continuous.

Its inductive bias is the number of clusters and their shape; it won't find the clusters if they form a ring, like in the bonus dataset.

At first, the second dataset difficulty seemed to be the higher number of clusters but a closer look after seeing that the clusters where strangely grouped as vertical stripes saw that the scales were not at all the same, which was an issue because Euclidean distance needs an orthonormal basis. To solve this issue, I scaled the samples using z-score scaling, which was very easy with the tool panda give us, which gave me this nice cluster separation.