Yann RENARD
Marharyta TOMINA

Machine Learning project

on the topic

Machine Learning for classification

Professor

Mathilde MOUGEOT

February 26, 2023

Contents

1 Data preprocessing

1.1 Data analysis

In this section we are going to explore our data to get some insights about it and prepare it for our classifiers. Our feature data has 2556 instances and 11 attributes in the data set. We don't have any missing data.
There is a mixture of categorical and numerical variables in our dataset. More precisely, we have one categorical variable, *'time'* having a data type **object** and 10 other numerical variables with data type **int64**.

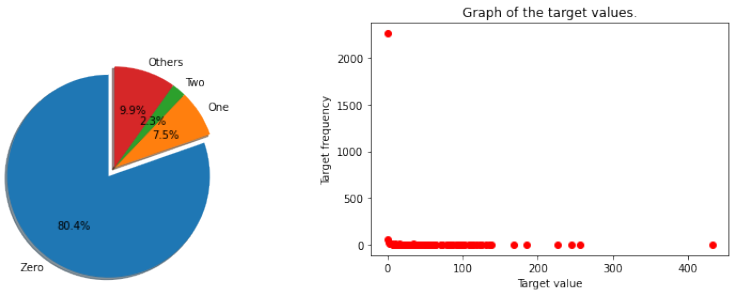
1.1.1 New feature creation

First of all, we will explore the categorical variable. The values in the *'time'* table have the form *'year-month-date'*. We can consider this data to be cyclical continuous features and we will create three new features: two of them (*'time_cos'*, *'time_sin'*) by deriving a sine transform and cosine transform for months and days, and the third one(*'year'*) by extracting the year.
After the feature creation we obtain feature data with 2556 instances and 13 attributes.

	time	t2m	u10	v10	SST	SIC	r1_MAR	r2_MAR	r3_MAR	r4_MAR	r5_MAR	time cos	time sin
0	2013-01-01	-21.926931	-0.973994	3.149094	-1.690511	90.745710	0.034537	0.033345	0.0	0.0	0.0	0.860961	0.508671
1	2013-01-02	-23.696195	-6.502908	2.494894	-1.690511	88.502980	0.034527	0.033326	0.0	0.0	0.0	0.852078	0.523416
2	2013-01-03	-25.644027	-3.557411	1.025486	-1.689860	88.734091	0.034523	0.033321	0.0	0.0	0.0	0.842942	0.538005
3	2013-01-04	-23.566887	-1.888075	-3.486122	-1.690511	89.149576	0.034509	0.033306	0.0	0.0	0.0	0.833556	0.552435
4	2013-01-05	-22.897768	-2.748844	-3.491206	-1.689860	91.613955	0.034492	0.033290	0.0	0.0	0.0	0.823923	0.566702

1.2 Target variables transform

Our initial target variables consist of 78 different numbers in range (0, 624), having different frequency. The most frequent target value is 0, more then 80%, and 47 values are met just once.



(a) (b)

Figure 1: Frequences of different target values

Therefore, for the better classification we are going to transform our variables into the binary ones. Primarily, we are going to consider all zero target values to be **0** and all other values to be **1**, as in the Figure1(b) we can distinctly see separation. We can choose, for example, the threshold 0.5 to perform such partition.
After the transformation, we obtain a binary targets, where about 80% of our targets are **0**, and 20% are **1**. We can clearly see, that we have an imbalanced dataset. This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely. This is a problem as it is typically the minority class on which predictions are most important. We are going to discuss approaches we used to cope with it a little bit later.

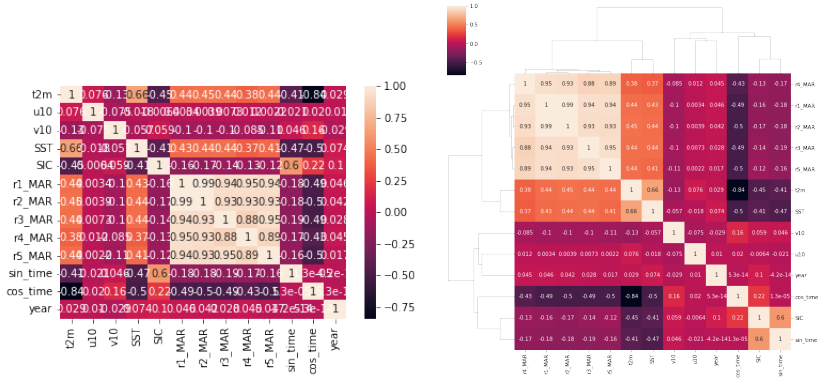
1.3 Delete all correlated tables

As a some of models, that we use (Bayes Naive, LDA, QDA, etc.)have primary assumptions, that the features are independent, it is really important for us to get rid of the tables with high correlations.

As we can see, the features 'r2_MAR', 'r3_MAR', 'r4_MAR', 'r5_MAR' are highly correlated, so we are going to drop them.

2 Commonly used techniques for the classifiers

Further, in this work we will oftentimes use and mix specific approaches to improve the accuracy of our models for the different metrics.



(a) (b)

Figure 2: Heatmap and Clustermap, representing the correlations

2.1 K-fold cross validation

As we have really imbalanced dataset, it might be extremely hard to choose the training and testing data, so that it would represent properly the abilities of our classifier and evaluate the quality of our model. To prevent this and the overfitting, in this work we will use k-fold validation with each classifier.

The only parameter in k-fold cross validation is k, which indicates the number of groups into which the given data sample should be divided. We chose the classic value $k = 10$, a value that has been experimentally found to generally guarantee low bias and low variance.

2.2 Random Resampling (Over- , Under-)

This is a simple and effective strategy for imbalanced classification problems.
Resampling involves creating a new transformed version of the training dataset in which the selected examples have a different class distribution. There are two main approaches to random resampling for imbalanced classification:

- Random Oversampling: Randomly duplicate examples in the minority class.
- Random Undersampling: Randomly delete examples in the majority class.

We are going to use both of those techniques in our experiments with classifiers to obtain the best result possible.

3 Choosing a measure of success

In our work we have an imbalanced dataset, which leads to some difficulties with the success measurement of the classifiers. In order to prevent this we are not going to use *Accuracy* score measurements, as this metric is useless in the tasks with not equal classes. Instead of it, we are going to use *Precision, Recall* and *f1-score*, as they don't depend on the ratio between classes.

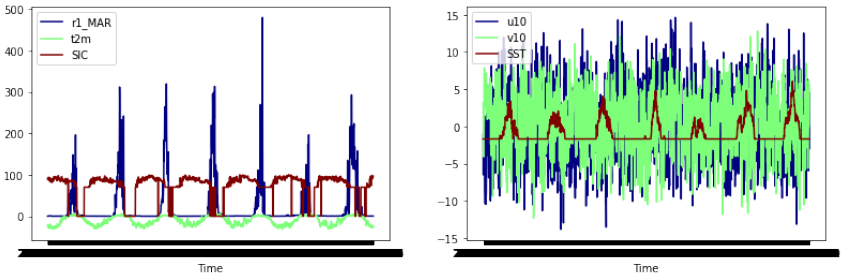
4 Binary classifiers

4.1 Bayes Naive

Here we use Gaussian Bayes Naive classificator. First thing, that we have to mention are model assumptions:

- independent features;
- continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution.

If with the first assumption we have no problems, because of the procedure we explained in 1.3, the second assumption can cause some problems. Let's see, how our data is distributed.

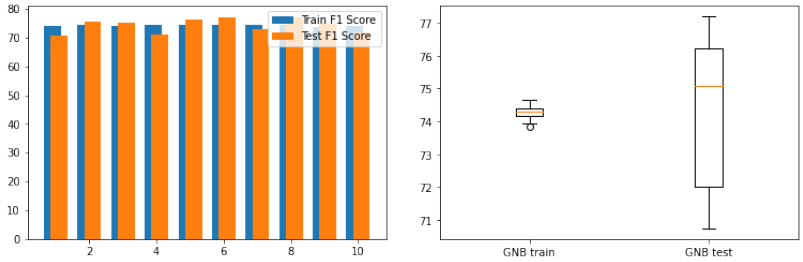


(a) (b)

Figure 3: Data distribution

It is easy to notice, that the distribution of our data does not have a Gaussian distribution. Nevertheless, Bayes Naive classifier can be used with such data, as well, and sometimes even can show some good results.

Let's apply Gaussian Bayes Naive classificator with 10-fold validation to our data and see, how it performs. It is good to notice, that for this model Resampling is not only unnecessary, but harmful. The thing is that NB uses ratio of the classes in the data to make its predictions, and Resampling makes equal ratio between classes.



(a) (b)

Figure 4: Confusion matrix, F1-score, Box plot for GNB

On the testing data GNB seems to performs really good in detecting zeros, but quite bad with detecting ones. Looking on the weighted test f1-score measurements with the k-fold validation we can obtain the information that it is varying between 70 and 77. For such simple model that is not a bad result, considering the fact, that one of the primary assumptions is neglected. The box plot shows, that, although, the mean on the training and testing data are close, the Interquartile ranges have a big difference. This represents the uncertainty of our classifier about the testing data.

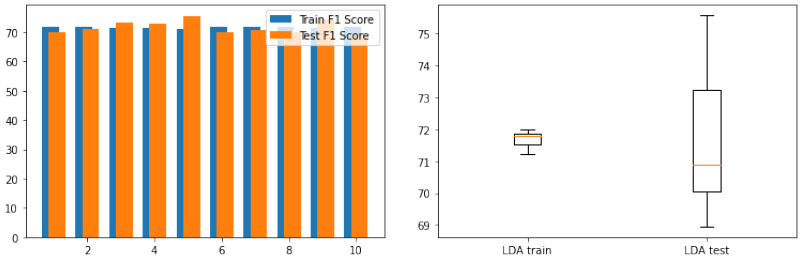
4.2 LDA

The assumptions for LDA are:

- independent features;

- Gaussian distributed features, having the same density.

Once again, the second assumption does not apply to our data. We use 10-fold validation and skip Resampling.



(a) (b)

Figure 5: Confusion matrix, F1-score, Box plot for LDA

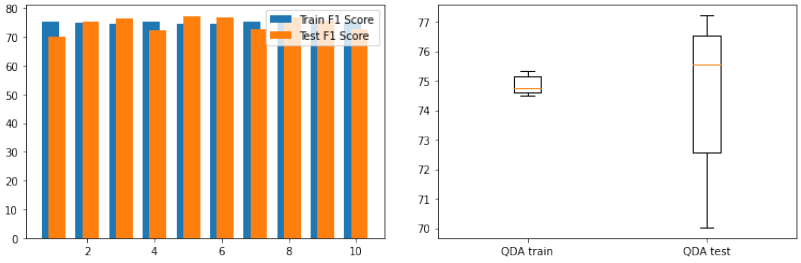
For the LDA it was extremely hard to work with our data, because nearly in every k-fold data batch it could not predict any values of the class 1. Considering f1-score and the Box plot graphs we see nearly the same thing as for the GNB.

4.3 QDA

The assumptions for QDA are:

- independent features;
- Gaussian distributed features, having different densities.

And here, the second assumption does not apply to our data. We use 10-fold validation and skip Resampling.



(a) (b)

Figure 6: Confusion matrix, F1-score, Box plot for QDA

QDA did better, than two models discussed above in distinguishing zero-class data. It is still hard for the model to track the data of the class one. The f1-score and the Box plot look similar to the GNB and LDA graphs.

4.4 Logistic Regression

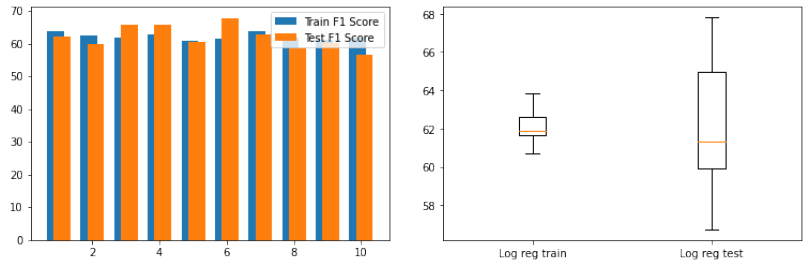
The assumptions of the logistic regression are :

- Binary target

- Independent observations
- No multicolinearity
- Sample size is sufficently large

These assumptions are verified in our model. Indeed, we have a binary target variable, the observations are said to be independent, we retrieve the highly correlated features, and our sample size is large (and we can conduct an oversampling if we feel it's not enough).

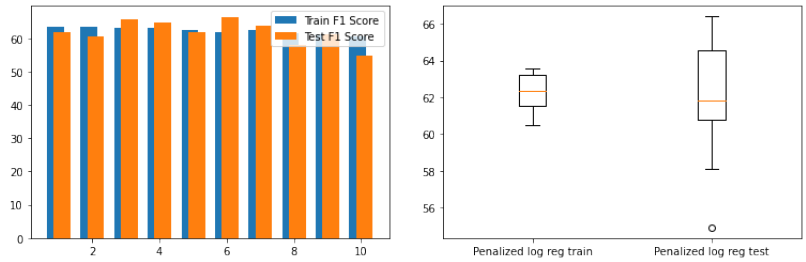
We first train a regular logistic regression with no penalization using the 10-fold validation and a resampling (oversampling). Here are the results.



(a) (b)

Figure 7: F1-score, Box plot for logistic regression

We can see that we get a worse result than the previous ones. We then tried to apply a penalization (l2) but it wasn't doing much better since the regression didn't seem to overfit.



(a) (b)

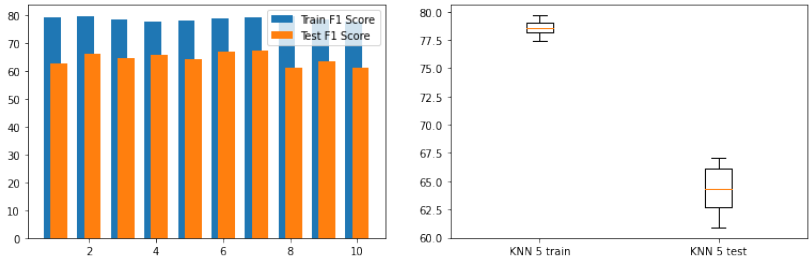
Figure 8: F1-score, Box plot for penalized logistic regression

The two trained logistic regression didn't seem to perform better than the previous models.

4.5 KNN

The main assumption of the K nearest neighbors model is that the data points close to each other are highly similar. We will train a KNN model and evaluate the best value of K to use. First we try with a KNN model with $K = 5$.

The F1 scores obtained seem to be better than the previous logistic regression. We need to see how the F1 score of the KNN model changes depending on the hyper parameter K. To do that, we made a chart representing the average F1 score of each KNN model on the 10-fold in function of the hyper parameter K.



(a) (b)

Figure 9: F1-score, Box plot for KNN with K=5

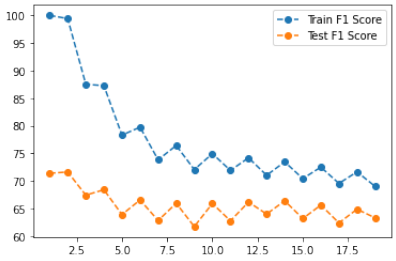
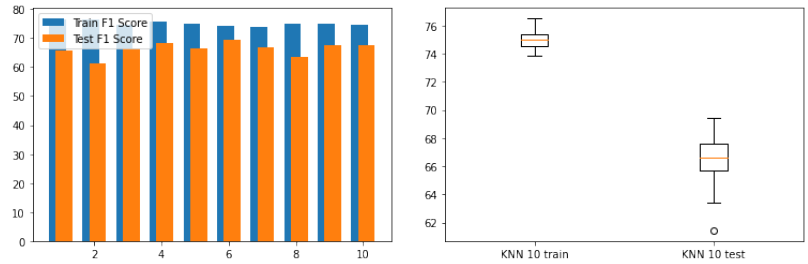


Figure 10: F1-score depending on K

We can see that the test F1 score doesn't increase a lot when the training F1 score decreases. We therefore took the value of K to be 10. We get the following results.



(a) (b)

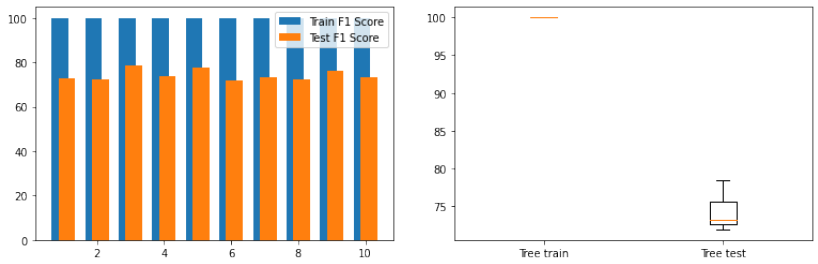
Figure 11: F1-score, Box plot for KNN with K=10

4.6 Decision Tree

We decided to train a decision tree on the dataset. Without changing the default parameters, here are the results we got.

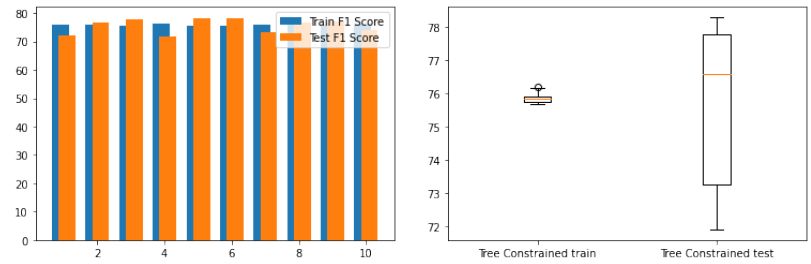
We can see that the model overfit on the training dataset case it gets a F1-score of 100%. We then tried to modify some parameters of the decision tree to prevent this from happening, especially the maximum leaf nodes that we set to 2.

The model now doesn't overfit on the training data and performs similarly as before on the testing data (or slighlty better).



(a) (b)

Figure 12: F1-score, Box plot for a decision tree



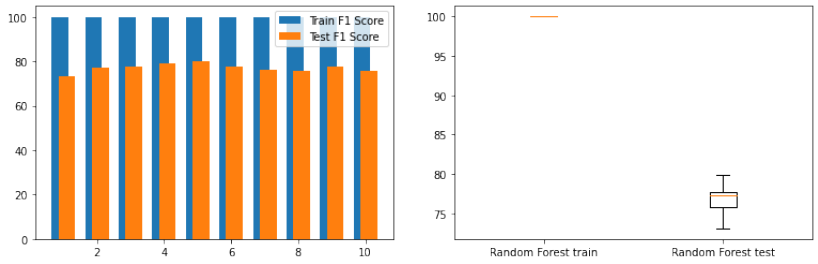
(a) (b)

Figure 13: F1-score, Box plot for a decision tree with max leaf nodes = 2

4.7 Ensemble Methods

4.7.1 Random Forest

The first ensemble method that we tried is the random forest. This is an extension of the previous decision tree model. We get the following results.

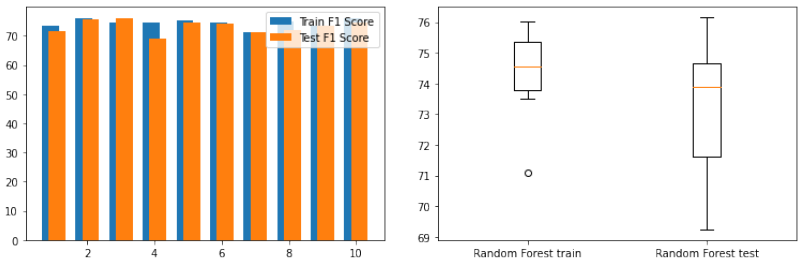


(a) (b)

Figure 14: F1-score, Box plot for a random forest

Just like with the decision tree, the model seem to overfit on the training data with the default parameters. We then tried to set maximum leaf nodes to 2 just like before.

Just like before, the model doesn't seem to overfit on the training data anymore, and perform the same or slightly betetr on the testing data.

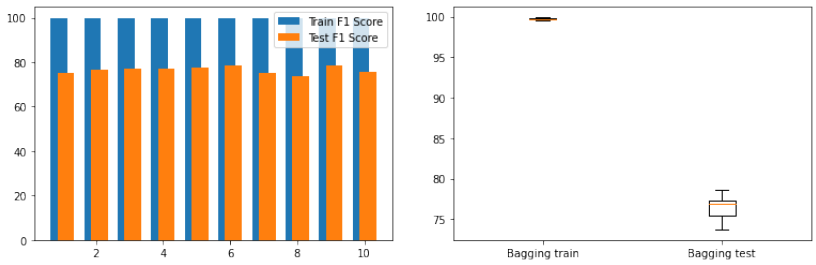


(a) (b)

Figure 15: F1-score, Box plot for a random forest with max leaf nodes = 2

4.7.2 Bagging

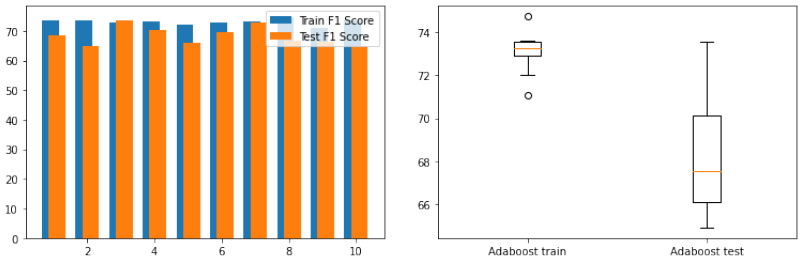
Another ensemble method that we tried is the bagging method. Bagging can be used with a lot of classifier (making groups of them), we decided to go with a decision tree classifier (as the default parameter). We didn't modify the default parameters of the maximum leaf nodes. Here are the results we got.



(a) (b)

Figure 16: F1-score, Box plot for a bagging classifier

We then tried to modify the base estimator (a decision tree) and set its maximum leaf nodes to 2 to prevent the overfitting on the training data, as well as modifying the number of estimators to 100. Here are the new results we got.



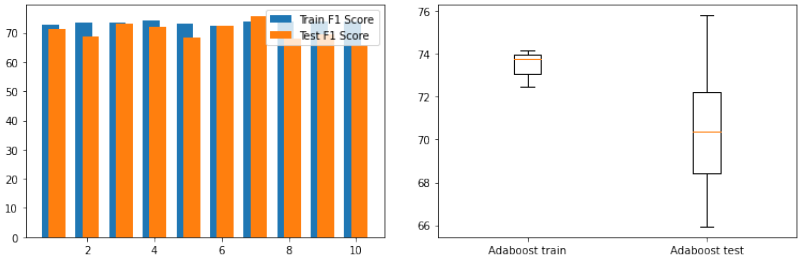
(a) (b)

Figure 17: F1-score, Box plot for a bagging classifier with max leaf nodes = 2 and 100 estimators

As with the decision tree, we can see that the model doesn't overfit on the training data and keeps its F1-score on the testing data.

4.7.3 Adaboost

Finally we tried the Adaboost ensemble method with a decision tree as a base classifier. Here are the results.



(a) (b)

Figure 18: F1-score, Box plot for an adaboost classifier

The model doesn't have the overfitting issue directly, but seem to perform a little worse than the two previous ensemble methods.

5 Conclusion

In conclusion, we made a graph of all the training and testing F1-score of all the models tested in the form of boxplots of the 10-fold.

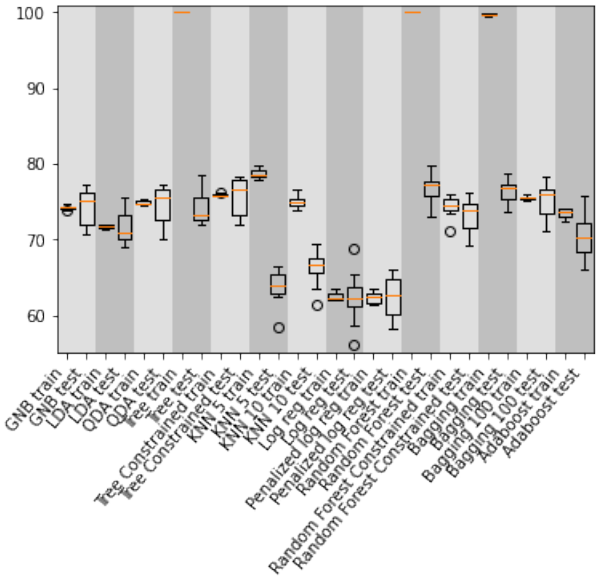
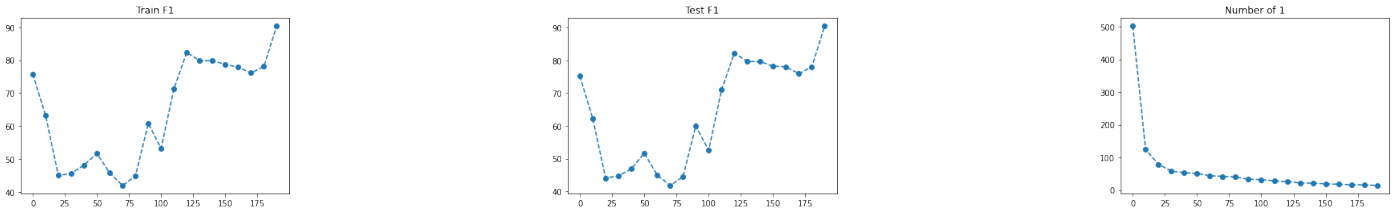


Figure 19: Training and testing F1-score of all the models

We can see that among the non ensemble methods, the decision tree and the gaussian models performed better than the others, and thus we chose the decision tree in our ensemble methods which performs similarly as the regular models.
We then tried to evaluate how the choice of the threshold impacted our models. Therefore we made a graph of

the average F1-score of a said model (a decision tree with a maximum leaf node of 2) depending on the threshold chosen for the classes 0 and 1.



(a) (b) (c)

Figure 20: training average F1-score, testing average F1-score, number of elements in class 1

It seems like our choice of 0.5 for the class 1 (thus ≥ 1) was a good enough choice, and that we could increase our F1-score by applying a much higher threshold to the data, but by doing so, we get a lot less data in the class 1, which makes the scores obtained less relevant.