Institut de Recherche
en Informatique de Toulouse

Internship Report

Gilet Cassandra

Département Sciences du Numérique - deuxième année
2020-2021

Contents

1 Introduction **3**

2 Neural Networks **4**
2.1 Custom Loss 4
2.2 First Network 4
2.3 Second Network 5
2.4 Third Network 6
2.5 Fourth Network 6
2.6 Fifth Network 7

3 AdaBoost **8**
3.1 Old Results 8
3.2 New Results 8

4 AdaCost **10**

5 Conclusion **11**
5.1 Neural Networks 11
5.2 AdaBoost 11
5.3 Continuation 11

1 Introduction

In this report, we will come back to the different methods used to predict WiFi cuts.
To compare the different results obtained, we will base ourselves on the data of 6 phones, differentiated by their IMEI, the important information of which you will find in the table below.

IMEI	% of 1 (1h slots)	% of 1 (15mn slots)	nb of measures (1h slot)	nb of measures (15mn slots)
203a	12.4	5.0	2399	8605
f1d9	9.4	2.7	12663	50660
936f	8.0	2.6	4938	19745
4517	1.1	0.1	2144	16087
677a	14.3	4.1	1076	4298
63cd	11.6	3.8	2461	9835

2 Neural Networks

The first method used to try to predict WiFi connection losses are neural networks.
As the results obtained by this method are not conclusive for any IMEI, we will only present the results of a single IMEI for the comparisons, the trends being very similar between the different IMEIs. We choose the IMEI with the highest percentage of cuts : 677a.

2.1 Custom Loss

For the first three networks, we created a personalized loss in order to be able to weight the penalty of not detecting a cut compared to the false detection of a cut.
To play on this parameter, all you have to do is change the "taux" value in the customloss class.
If we give a high value to this variable (for example 100), then the network tends to predict only cuts to minimize the loss. On the contrary, if we give a low value to the variable then this time no cut is detected. If we try to find an intermediate value, then the predictions are very random and the accuracy is low.

2.2 First Network

To start with, we built a network that took different features as input such as:

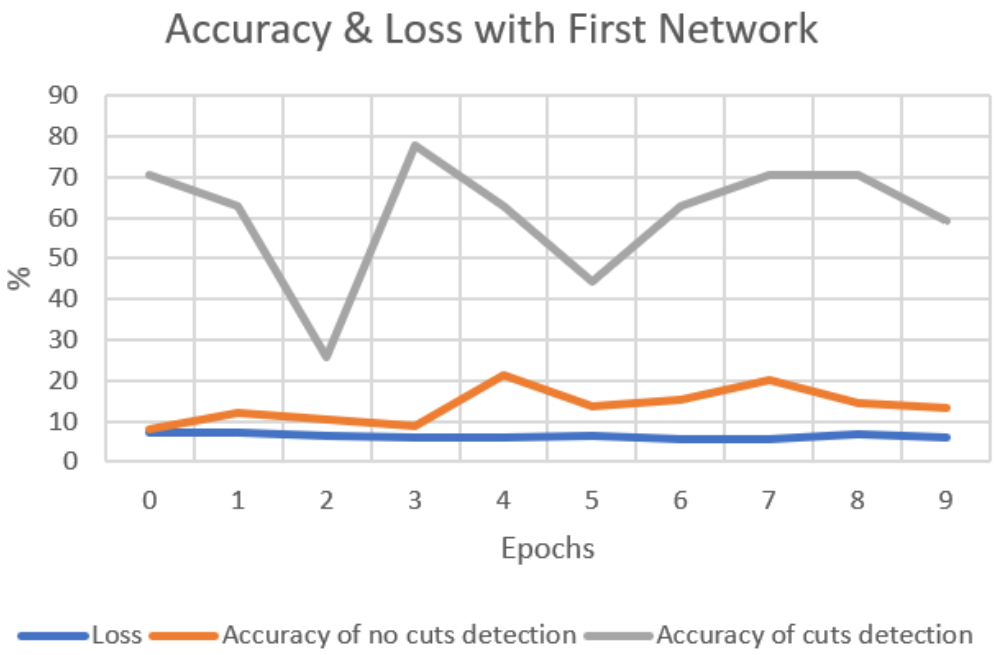
- Cut during this slot? (0/1)
- Type of connexion (0=3G/1=Wifi)
- Connected to the HomeWifi? (0/1)
- Connected to the WorkWifi? (0/1)
- Day (0-6)
- Number of Apslots
- Top 1 Wifi in list of Apslots? (0/1)
- Top 2 Wifi in list of Apslots? (0/1)
- Top 3 Wifi in list of Apslots? (0/1)
- Number of the Slot

The architecture of our network was as follows:

- Linear layer of 512 neurons
- Linear layer of 512 neurons
- Linear layer of 256 neurons
- Linear layer of 1 neuron
- Each layer is separated from the next by a ReLu function

For the training of our network we use the following parameters:

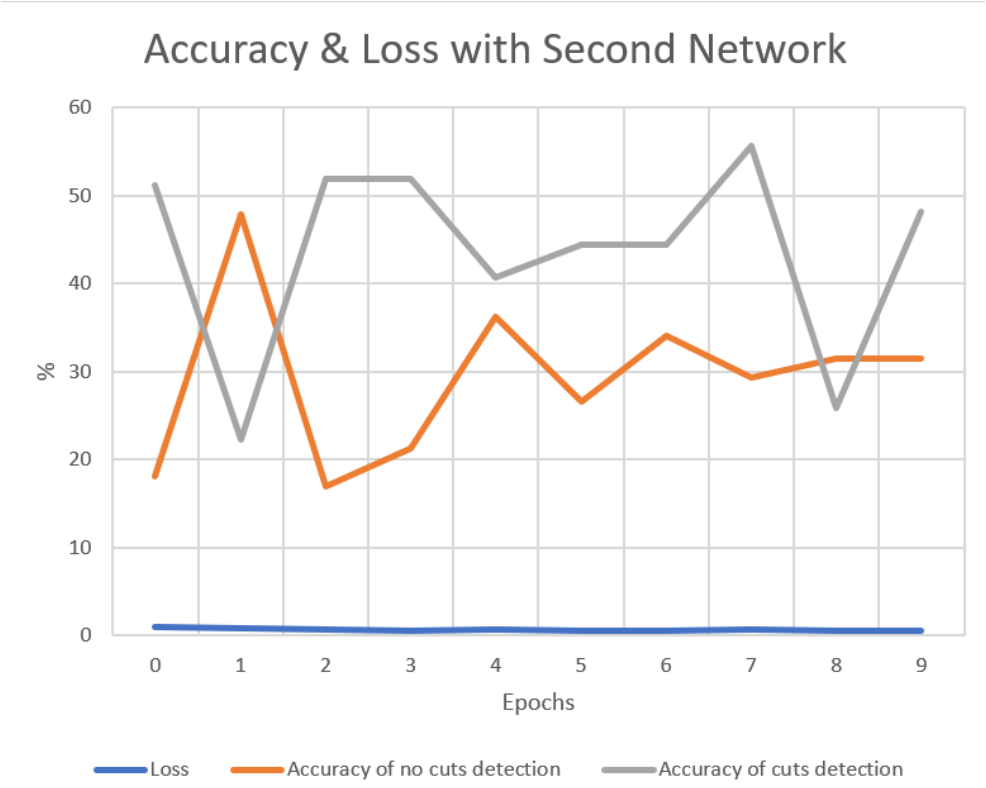
- optimizer: Adam
- learning rate: 1e-4
- number of epochs: 10
- batch size: 1



If the accuracy of the detection of cuts is high, then the accuracy of the detection of non-cuts is very low and vice versa. Even by changing the values or the custom loss we are unable to find a balance.

2.3 Second Network

In this network, we try, keeping the same inputs and parameters, to use the data from the two previous slots to predict a cut in the third slot.



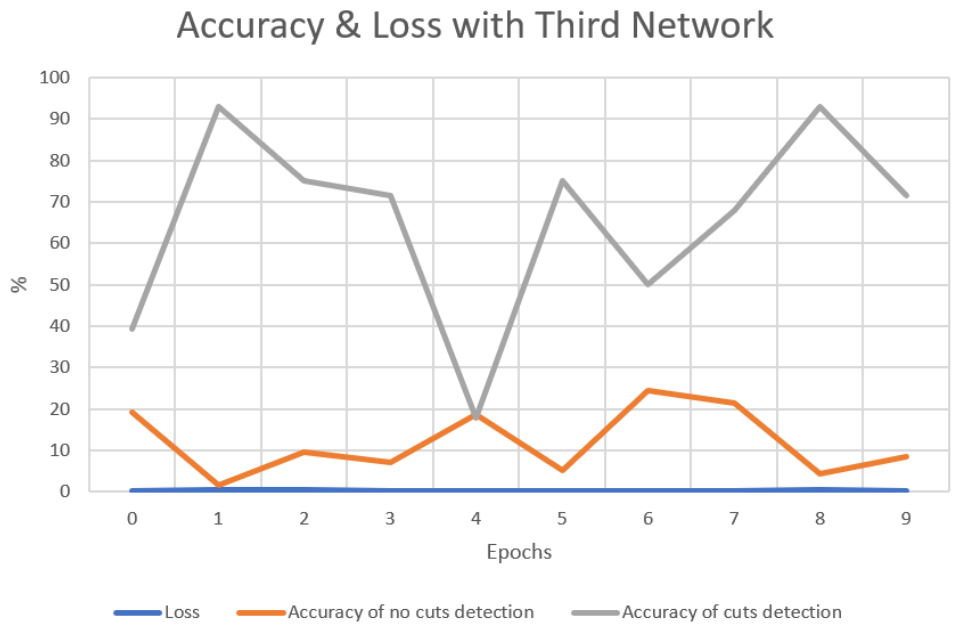
We obtain the same trends as with the first network, but even more marked. We also notice that although the loss is very low (0.6), this does not allow to obtain a high accuracy even by adjusting the parameters.

2.4 Third Network

For this network, we only give as input the data of a single slot to predict a cut in the next slot since we noticed that increasing the number of slots does not allow to obtain better results.

We decide to keep the same parameters but to change the input features. This time, we only enter a small amount of data into the network that seems most useful for predicting a cut:

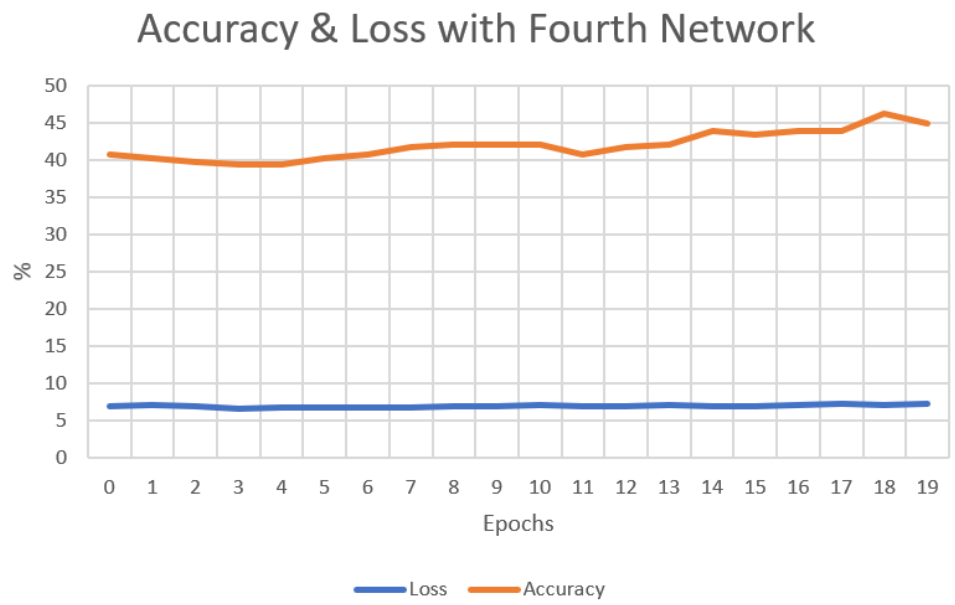
- Cut during this slot? (0/1)
- Type of connexion (0=3G/1=Wifi)
- Day (0-6)
- Number of the Slot



The trends remain the same as with other networks and the results are still not satisfactory and exploitable.

2.5 Fourth Network

To use this network, we decide to reformulate our prediction problem. Rather than predicting from a measurement if there will be a cut in the next slot, we instead try to determine from a measurement in how many slots the next cut will take place.
We therefore keep the same network, with the same parameters, but we abandon the customized loss and instead choose an L1 loss.

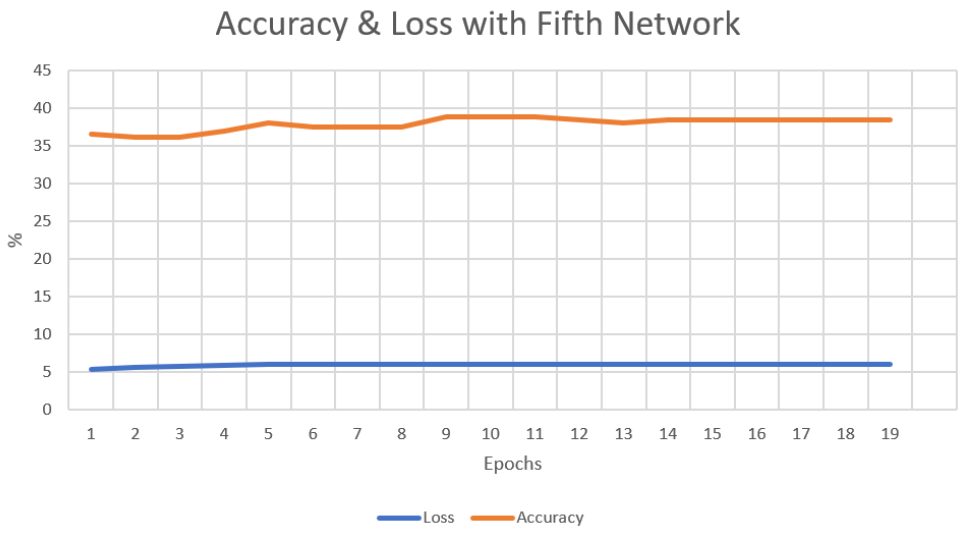


We can see that despite the epochs, our network doesn't seem to be learning. Loss and accuracy remain constant throughout the learning process.

2.6 Fifth Network

Since our network is very large, it requires a very large amount of data to learn properly. We do not have enough data to allow proper learning. We are therefore going to resume our last network (prediction of the number of slots before the next cut) and greatly reduce its number of neurons and layers to see if we can see better learning.
The architecture of our network was as follows:

- Linear layer of 5 neurons
- Linear layer of 1 neuron
- Each layer is separated from the next by a ReLu function



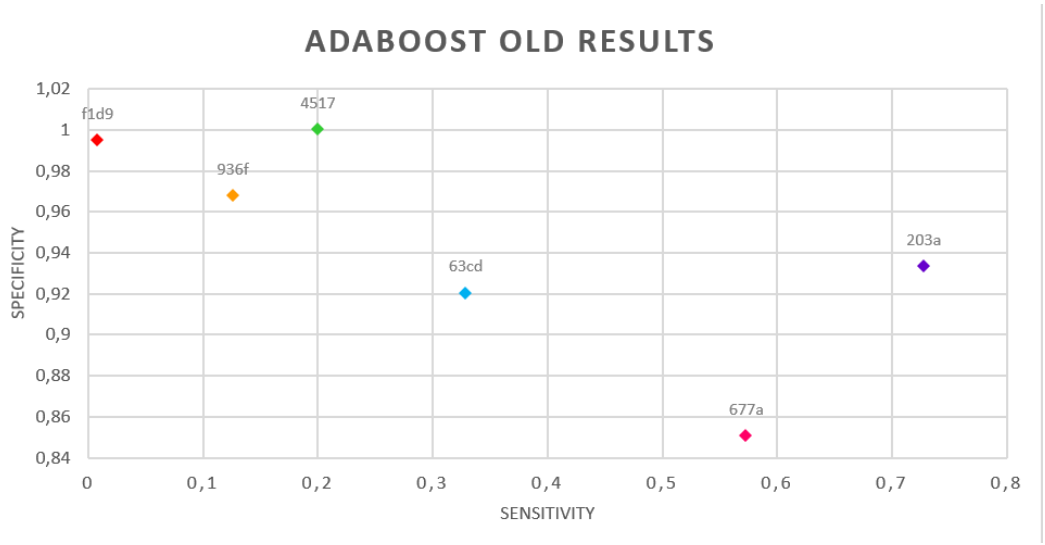
The results look very similar to what we were getting before. So we decided to try using AdaBoost rather than continuing to work on neural networks for the future.

3 AdaBoost

AdaBoost is an algorithm whose objective is to improve the performance of other learning algorithms. The outputs of the other algorithms (called weak classifiers) are combined into a weighted sum that represents the final output of the boosted classifier.

3.1 Old Results

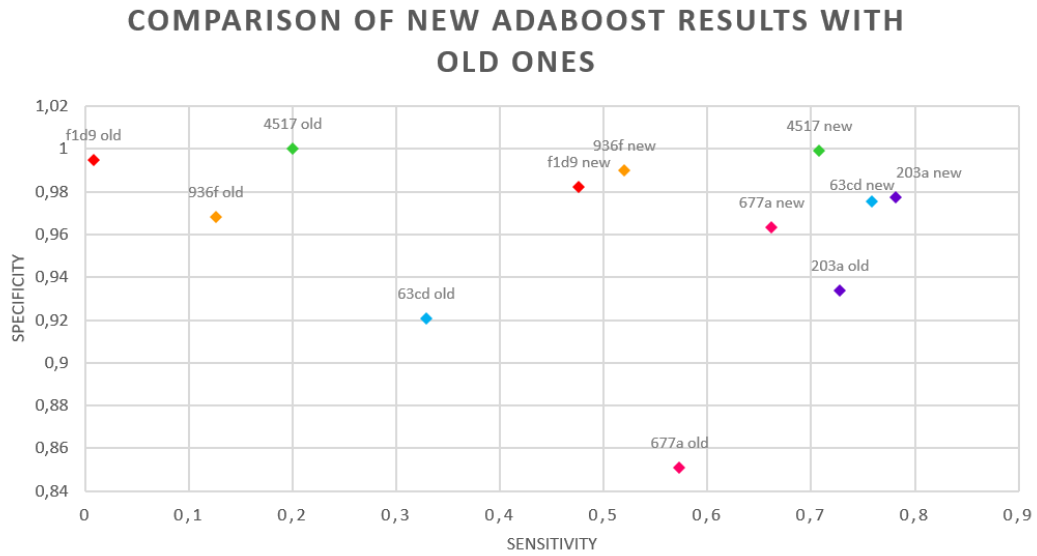
A few years ago, AdaBoost had already used it to try to predict WiFi cuts. The results are shown below for six different IMEIs.



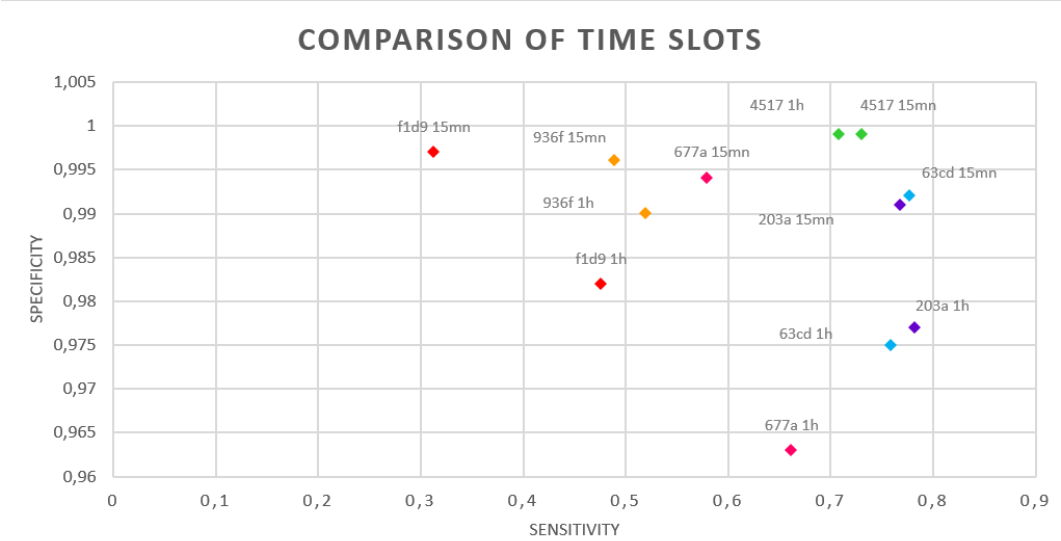
The results are not convincing since the sensitivity almost never exceeds 0.5.

3.2 New Results

We use as estimator DecisionTreeClassifier and a learning rate of 0.5.
We compare the results obtained with those obtained before.

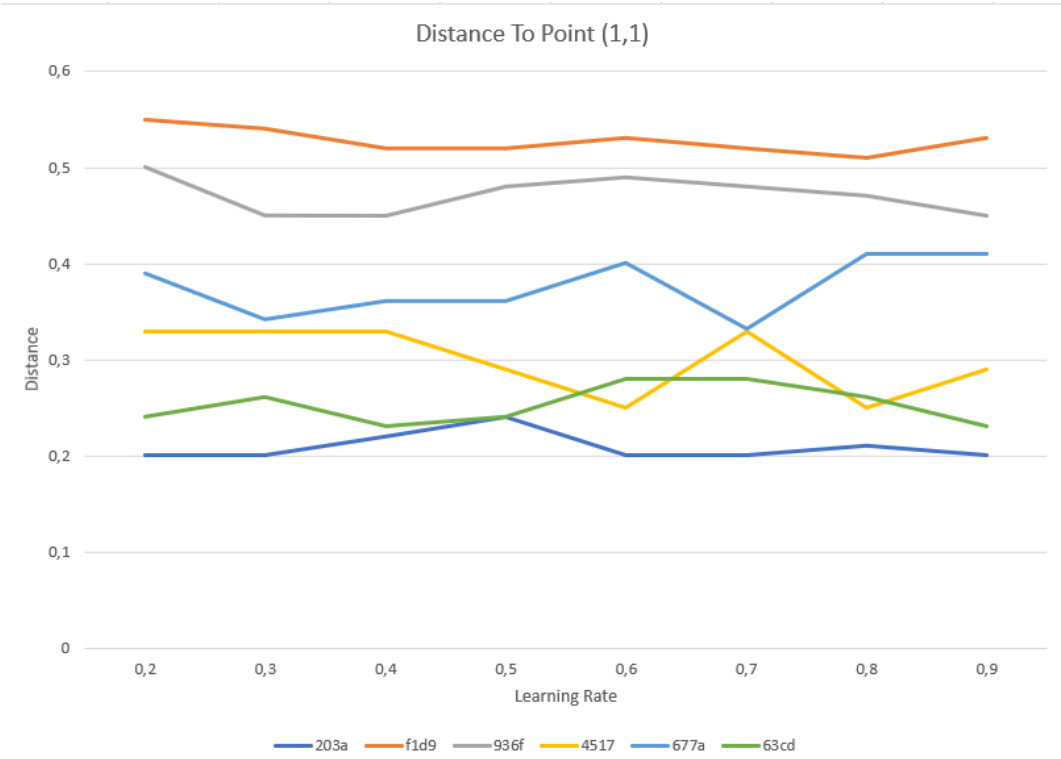


We can see that the results have improved. The lowest sensitivity is slightly below 0.5.
To try to improve the results even further, we compare the results with one hour slots and fifteen minute slots.



Depending on the IMEI, the duration of the slots that optimize the results change.

We have tested different learning rate values and display the distance to the best expected point (1.1). We can notice that according to the IMEI, the minimum distance is found for a learning rate between 0.4 and 0.7.



4 AdaCost

After having obtained conclusive results with AdaBoost, we are trying to further improve the prediction with a new implementation of AdaBoost named AdaCost.
The initial code can be found at the following address: https://github.com/joelprince25/adacost

5 Conclusion

5.1 Neural Networks

No matter what approach we put in place, the results are not actionable.
The loss function used instead of the MSELoss or L1Loss made it possible to observe certain phenomena more efficiently.
Several avenues for improvement were considered. first, we could try to work with a batch size greater than 1. Then, we should continue to explore the path of reducing the size of the network (with different combinations of loss functions, inputs and prediction). Finally, try to find the most exploitable entries by the network (add, remove or transform some).

5.2 AdaBoost

After testing a number of combinations of parameters, we seem to have arrived at the optimal combination of parameters. We managed to improve the results and make a correct prediction of the cuts, provided we had enough data and a fairly high cuts / no-cuts ratio.

5.3 Continuation

By following these instructions, you will be able to start retrieving new data and start training on that data.

- Create a "phones.txt" file that contains the IMEIs of the phones from which you want to recover data.
- Launch : /usr/bin/python2 /home/cgilet/Reseau1/Codes/GlobalPredict.py "Path to phones.txt" ADA
- You can choose the length of the slots in the GlobalPredict.py file
- Rename the created files (add in home / cgilet) in the form ADA_cuts_IMEI_TimeSlots (as in the AdaBoost / Data folder) and move them to the AdaBoost / Data folder.
- In the AdaBoost_set1.py file, change the IMEI variable to give it the value of the IMEI to test and slot over the chosen duration.
- Start the program, the results displayed are: the percentage of cuts among the measurements, the results with AdaBoost (old estimator) and RandomForest (new estimator)

Other estimators can be tested on the data to try to find the best possible combination. The code below groups together the main estimators. We must adapt this code to our case (change the name of the variables and calculate the sensitivity and specificity) to compare these estimators with our previous results.

```
from sklearn.datasets import make_blobs
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostC
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.cross_validation import train_test_split


datax,datay=make_blobs(n_samples=2000,random_state=1)
trainx, testx, trainy, testy = train_test_split(datax,datay,test_size=0.35, random_state=1)


#################SVC with probability=True
clf=SVC(probability=True,random_state=1)
clf.fit(trainx,trainy)
y_pred=clf.predict(testx)
print 'SVC ',accuracy_score(testy,y_pred)

#################AdaBoostClassifier with SAMME.R
clf=AdaBoostClassifier(algorithm='SAMME.R',base_estimator=SVC(probability=True,random_state
clf.fit(trainx,trainy)
y_pred=clf.predict(testx)
print 'AdaBoostClassifier',accuracy_score(testy,y_pred)



#################SVC with probability=False
clf=SVC(random_state=1)
clf.fit(trainx,trainy)
y_pred=clf.predict(testx)
print 'SVC ',accuracy_score(testy,y_pred)
#################AdaBoostClassifier with SAMM
clf=AdaBoostClassifier(algorithm='SAMME',base_estimator=SVC(random_state=1),random_state=1
clf.fit(trainx,trainy)
y_pred=clf.predict(testx)
print 'AdaBoostClassifier',accuracy_score(testy,y_pred)
```