Lab Book

Sherlock and Moriarty

Alfonso Bárragan Carmona
Javier Monescillo Buitrón
Roberto Plaza Romero

alfonso.barragan@alu.uclm.es
javier.monescillo@alu.uclm.es
roberto.plaza@alu.uclm.es

Start October 6, 2018

Contents

Getting started

In this section we will give all the necessary elements for a quick start in our project, first we present the place where the repository of the project is and secondly the video milestone 1.

1 Repository of the project

To start working on the project of the subject "Machine Learning Techniques", we have created a repository in GitHub for collaborative work, you can see the repository here:

```
https://github.com/RoberPlaza/MachineLearningLAB
```

2 Video of milestone 1

Here we have the video where it is explained from the beginning to the end of milestone 1 and its development.

```
https://www.youtube.com/watch?v=Bx8MOWg44h0&feature=youtu.be
```

Milestone 1

Problem
Sherlock and Moriarty is an iranian problem whose purpose is to learn, prepare and protect about several types of cyber-hazards. This database contains information gathered from several smartphones, and the goal is to make a program that could detect how many of those terminals were infected by the virus of Moriarty.

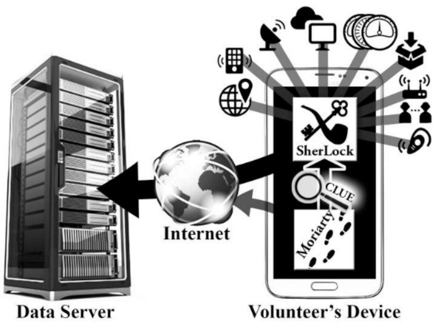

Figure 1: Problem

What's the problem with milestone 1?
Analize the routine of an iranian citizen via movile phone in order to learn about how to perform unsupervised learning

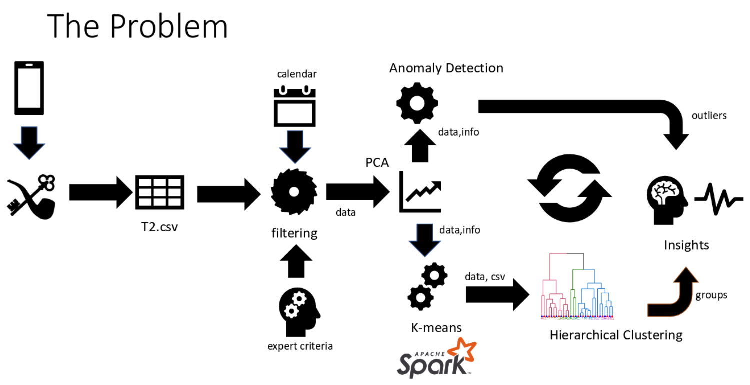

Figure 2: Milestone 1 problem

Monday, 15 - 10 - 2018

1 Approach to the BIG dataset

Today, we met in the library to begin our research over the sherlock and moriarty dataset, in the early, we make a lighty reading of the description and the article of sherlock and moriarty, we a less we try to open with excel (we know that we can't open the file by that way, but we wanna try).
We are thinking other kind of ways to explore the dataset, the most posible options are to take a proportion of the elements from the dataset (1 element of every 10, in a regular method or with a random method, for example) or making a bash script because our teacher says that bash can do wonders when dealing with files and the management associated with them.
After the first hours of contact, we have opted to cut the database via python programming. For reasons of efficiency and speed we will do it staying with the tenth line of every ten, and once we have a csv of a more manageable proportions, we will proceed to a more internal analysis to see how to manage the large database really. In order to make it as fast as possible, we are even trying to make programming in threads, for the future could be very profitable.
We did it and with a beer we're done for the day.

Thursday, 23 - 10 - 2018

1 Begining the milestone one

We begin with the treatment of the T2 dataset of sherlock and moriarty, our beginning is partly directed by the indications given by Francisco, we are beginning with doing a sampling of a single day instead of putting ourselves first with the complete dataset.
Once this has been done, we have decided to take rows of three in three (one minute), and make the average of the values poured by the sensors. We do this because a lot of the sensors are dependent on the user's situation and we assume that it won't vary substantially at twenty-second intervals.
In order to undertake a first approach in an efficient way, we are going to take the sampling of a specific day. We will deal with the choice of that day, randomly or by some specific criterion. But that is a problem for the machine learning laboratory group of the future.

Saturday, 27 - 10 - 2018

1 Some problems with missing values

Today we have gathered as a team to try to treat the lost values in T2 csv, and the result has been a failure, we have had problems with the fields that had "NULL" values. This is because when we try to do the filtering we replace these fields with basically "", which is nothing. So we have done several tests with pd.dropna(), and we have not been able to remove them.

2 Success in normalizing csv

In the end we were able to eliminate the lost values, thanks to the following code:

```python
def normalize_filtered_data(path):
    file = pd.read_csv(path, low_memory=False)

    exclude = ['UserID', 'UUID', 'Version', 'TimeStemp', "
        RotationVector_cosThetaOver2_MEAN", "
        RotationVector_cosThetaOver2_MEDIAN", "
        RotationVector_cosThetaOver2_MIDDLE_SAMPLE"]
    df_ex = file.loc[:, file.columns.difference(exclude)]
    df_ex = df_ex.replace(" NULL", np.NaN)

    df_ex = df_ex.dropna()

    min_max_scaler = preprocessing.MinMaxScaler()
    df_norm = min_max_scaler.fit_transform(df_ex)

    return df_norm
```

We couldn't remove the fields to "NULL" because they had a space in front of them, so using dropna(), the np.NaN values are replaced and deleted.

As you can see we remove the columns that are not necessary for the PCA, and then we use the normalized min max scaler. The next job we'll have to do is to think what kind of clustering algorithm we'll use. We will also request a meeting with the teacher to see if all the work is correct.

And now it's time to drop ourselves to sleep, and continuing tomorrow.

Thursday, 1 - 11 - 2018

1 Deciding which features we should delete

After a long period of work, thinking about what we had to do and meeting with the subject teacher in his office, we met to do the following activities:

- Update the organization of the project
- Treat features in preprocessing
- Decide which features to remove, according to the description of the csv file

At this moment, we are deliberating which are the most appropriate characteristics depending on the correlation matrix, correlation is something we have also done in this meeting.

We have updated the **.py** files and this is the following organization of the project:

- MachineLearningLAB
  - docs
  - milestone1
    - plots
    - data
    - 1_preprocessing.py
    - 2_normalize.py
    - 3_clustering.py
  - milestone2

In preprocessing.py we have the preprocess of both rows and columns, in normalize.py we have normalized the csv file, deleting categorical variables and applying minMaxScaler and finally in clustering.py we will try to apply several clustering algorithms.
After several hours of deliberation, we have decided to discard the following columns:

- Fast Fourier Transform.
- All columns of the axes of the y.
- All summary columns for medians, variance, middle sample.
- Columns that have to do with the rotation vector and the orientation probe.

(Reasons why here)

After these modifications, we have made a dimensional reduction by principal component analysis, obtaining two principal components of [0.52] and [0.27] respectively.

Once reached this step, we reached the point of choosing which could be the most suitable clustering for our data, at the moment in view of the PCA, we have tested first with the k-means and more or less does it correctly, but we are open to a dbscan because we have a large density of data and may group better than the k-means.

But we will arrange that tomorrow, for today we have already had enough Principal Components that represent 100 percent of our hard work.

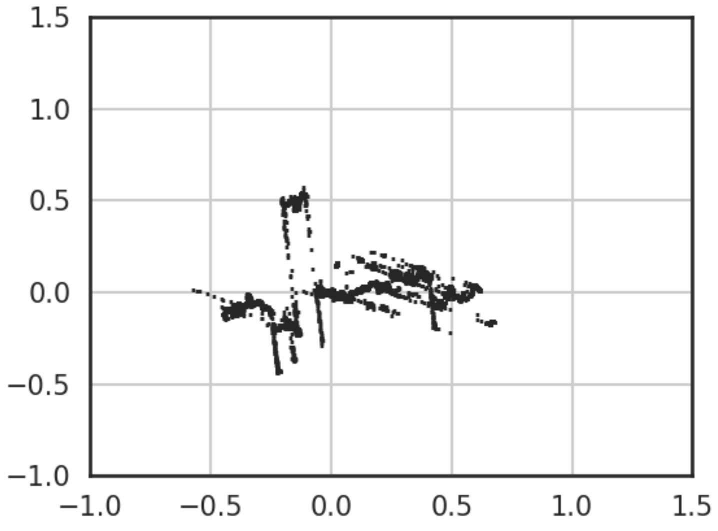

Figure 1: Principal Components Analysis

Friday , 2 - 11 - 2018

1 At the end of milestone 1

It should be noted that in recent days of meetings, specifically the previous and today, have been online.
At the beginning of the afternoon we have created the powerpoint related to the video and we have decided which parts are going to occupy each one and what is going to explain.
In terms of code, we moved the replacement of null values to preprocessing.py, as we were doing it in normalize.py.

Then we have saved the images of the similarity matrix and the correlation tests using plt.savefig().
And now comes the most important thing: We have added the DBSCAN clustering
After many deliberations we have decided to discard the accelerometer readings because they were redundant with the linear acceleration readings.
We are also interpreting the groups that pour the different clustering algorithms, at the moment we have KMeans, KMeans++ and DBSCAN.

For the next meeting that will probably be tomorrow we will finish to interpret the results, choose which clustering we are going to stay.

We can portray all our effort and hard work, with the next data scientist spell: Skidaddle skadoodle your data is now knowledge.

Why don't we use an hirearchical clustering algorithm?
We have decided not to apply a clustering algorithm because due to the volume of data, because we consider that we need that volume of data and the hierarchy would take too much time. And to filter more, the dataset could lead us to commit too much bias in the analysis of the data.

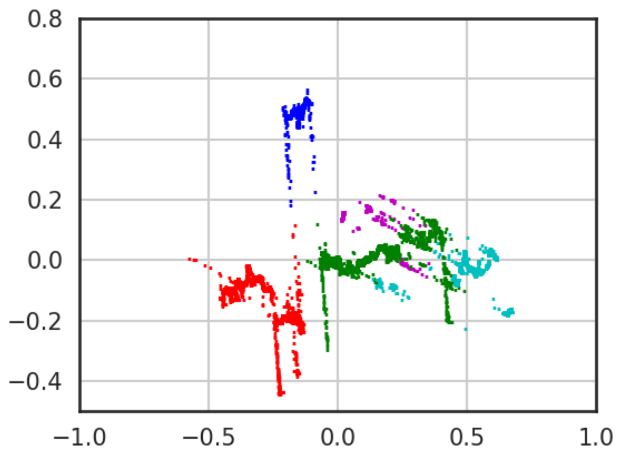

Figure 1: KMeans

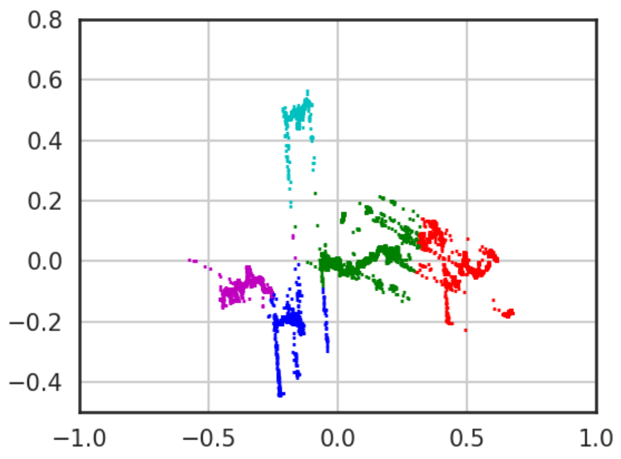

Figure 2: KMeans++

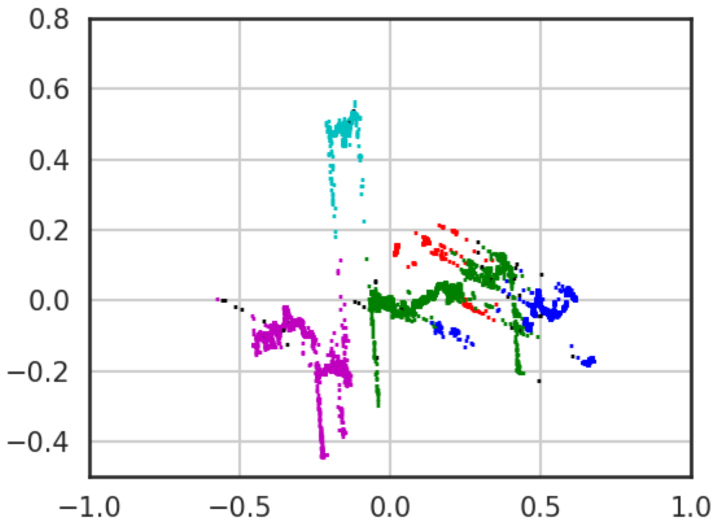

Figure 3: DBScan

Saturday , 3 - 11 - 2018

1 Fixing some problems and the video

As the last day of work pass by there were a few milestones to finish.

At first we started by formating our code, to make it more readable and sustainable. We also made a ConcurrencyUtils.py module for the next milestones. All the code, everything that had to be coded is now coded and nothing was added since yesterday.

While trying to understand the code we consider the idea of this milestone being a bait, because since the column filtering we lack a goal, we lack an objective, therefore the information we discovered was somewhat obvious. Despite the previous statement, we manage to extract quite the interesting information.

Looking in the normalize data frame with the help of Spyder we discovered that the linear acceleration in the Z axis was the variable that impacts the most in the data set.
Given that the optimal number of clusters was 5, we could interpret those clusters as 5 different patterns of displacement this particular individual have.
We could label them as:

- Static.
- Walking.
- Running.
- Car driving through city.
- Car driving through highway.

Additionally we wrote some Readme markdowns to make the repository more appealing. There is one Readme markdown for the general project and one Readme per milestone, they have information regarding the directory (goals, description, ...) and also photos of the plots and code we used through that part of the work.

Sunday , 4 - 11 - 2018

1 Milestone 1 conclusions

In the end we got the following correlations and heat map.

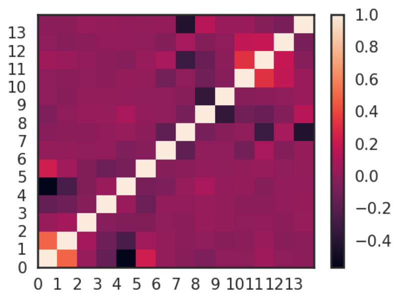

Figure 1: Correlation

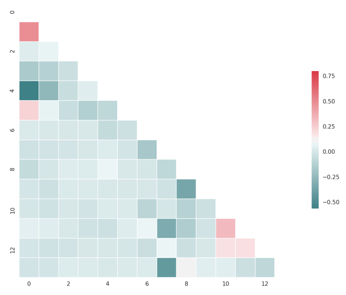

Figure 2: Heat map

Final conclusions
Given that the linear acceleration in the Z axis is the variable that influences the dataset the most we can guess that the 5 clusters means 5 displacement patterns over the transcourse of the time period the project took place.

Milestone 2

Problem

Having information regarding to several devices we need to realize an study about cyber-security. We know there is a virus called "Moriarty" and several of the terminals in a given dataset are infected with it. We need to explore the data in order to predict were Moriarty is about to "attack". Those "attacks can be different things, different behaviours that can harm the user experience or turn his or her phone in part of a botnet.

In order to accomplish that the original idea is to first label the UUID's in order to know what have been the times in with Moriarty has attacked. From then on, carry on.

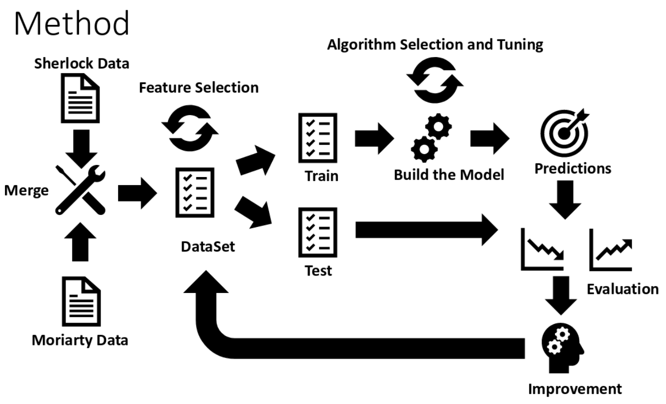

Figure 1: Milestone 2

Tuesday , 27 - 11 - 2018

1 Begining milestone 2

Today we started by upgrading the repository. Several features were added. Features such as:

- Project guide: a guide in order to take part in the project, all the members must accept it and work according to it.
- Labels for milestone 1. A release was created with with the version code 1.0, referring to the first executable.
- Lab-Book has been remodeled according to the advises and the recommendations given at the end of the evaluation of the first deliverable.
- Remodeled the repository in order to make it more suitable for development. Also advices were given in class and we want it to be prettier. The braches are now as follows:
  - Master: the main branch of the project. Useless code will never be part of this branch and it grow by means of merging with the Development one.
  - Development: with free commits and pushes. Bad code can be uploaded to this, but it won't be merged to master until everything works fine.
  - Experimental: the trash bin of Roberto's concurrency experiments. // Line written by Roberto Plaza

Wednesday, 28 - 11 - 2018

1 Making a method to label datasets

At the start, we need to make an good enought model, to label the data target that we dispose as good for our attack classification system. This reason pull us, to make an automatized method to label the data and when we have that, implementing the model, train it and evaluate it.

In the beggining of that approach, we started by using hadoop and pig, because we have a test of our compentences with pig soon, and we can use this easier, than other tools. Sadly, this approach ending with us trying to make a cartesian product of 70000 elements with 187 elements, and loosing 50 minutes of our life in that.

So we try, by an advice, to make a (far away as hell) more efficient method, based on a route about the 187 elements, and via pandas accesing to the dataset, changing an adding row called "attack" from 0 to 1 if its verified that the UUID from moriarty falls in a range between the element's UUID from the dataset to label - the time that tooks sherlock to make a lecture and the element's UUID itself. We also need to prepossess a little bit moriarty, because it has some columns with the character "," and can be troubled.

```python
def preprocess_moriarty(path_to_csv, path_to_destiny):
  input_file  = open(path_to_csv, 'r')
  output_file = open(path_to_destiny, 'w')

  output_file.write(input_file.readline()) # Column name
  line        = input_file.readline()

  while line:
    line_to_write = line.replace('],', '];')

    output_file.write(line_to_write)
    line = input_file.readline()

def label_dataset(df_to_label, df_labels):
  for i in df_labels["UUID"]:
    aux = df_to_label[(int(i) <= df_to_label.UUID ) & (int(i) >= (
        df_to_label.UUID - 20000))]
    print(aux)
    if not (aux.empty or aux.attack == 1):
      df_to_label.loc[aux.index, 'attack'] = 1
```

Friday, 30 - 11 - 2018

1 Balancing the occurrences

Once the data has been labeled after a few moments of discussion we came to the conclusion there is no point in trying to come in and kissing the saint. Therefore, in order to create a model we took T2.csv and started by Balancing the occurrences between Moriarty's attacks and safe timestamps. First of all we are doing clustering over the previously explained dataset (heavily based on the dataset of Millestone 1).

The data cleaning we did back in Milestone 1 helped us with the development of the model, because all the redundant features were dropped during the process. That means once we spot the centroids we can use them to represent the groups and clasify them quickly.

In the end we used the algorithm of Kmeans++ in order to discover the 32 centroids.

Saturday, 1 - 12 - 2018

1 Creation of the model

In order to create the final model we run several scripts.

First thing we wanted to do was to split the dataframe between the data we are going to use for training and the data we are going to use for testing. In order to accomplish that we took the centroids generated by a second round of clustering and merged it with the 36 registries of attacks we already had.

This new wave of clustering was made so the final number of centroids are 36, in order to have a 50/50 distribution between rows of attack and rows of no attack in the merged dataframe and keep the ocurrencies balanced.

Now we have a completely balance dataset in order to start looking for a model with different algorithms.

The selected distribution for this dataframe was a 33/66 distribution between testing-training. The algorithm we chose to apply was a random forest search after a few minutes of discussion.

After several attempts to parameterize, we do not know why spyder did not leave us and we would see enough bugs, so simply for this milestone we decided to implement the model as it is and train it.

With quite good results, but we do not rule out that when facing a real environment, our model is quite weak, although in principle remain faithful to our first preprocess in this subject, keeping much of the decisions we make, this model is quite good.

And in the case, that it isn't a good model for our porpouse, we can use it to make a better feature selection for future models and solutions.

```
Random Forest:
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00        12
        1.0       1.00      1.00      1.00        13

avg / total       1.00      1.00      1.00        25

Confusion Matrix:

preds   0.0  1.0
actual
0.0      12    0
1.0       0   13

Variable’s relevance:

                     Indicador  Relevancia
0         GyroscopeStat_x_MEAN         0.0
1         GyroscopeStat_z_MEAN         0.0
2        GyroscopeStat_COV_z_x         0.1
3        GyroscopeStat_COV_z_y         0.2
4         MagneticField_x_MEAN         0.2
5         MagneticField_z_MEAN         0.0
6        MagneticField_COV_z_x         0.3
7        MagneticField_COV_z_y         0.0
8                Pressure_MEAN         0.0
9   LinearAcceleration_COV_z_x         0.0
10  LinearAcceleration_COV_z_y         0.0
11   LinearAcceleration_x_MEAN         0.0
12   LinearAcceleration_z_MEAN         0.2

Relevance Maximum RF : 0.3
```

Milestone 3

Problem
Mainly, in this final milestone, we will try to finalize the model, extract all possible conclusions in reference to the project and make an attempt to be able to use Apache Spark.

We start from the base that we had in the previous milestone, that is to say, our model deployment is efficient, therefore, we will not have any problem testing decision trees or random forest.

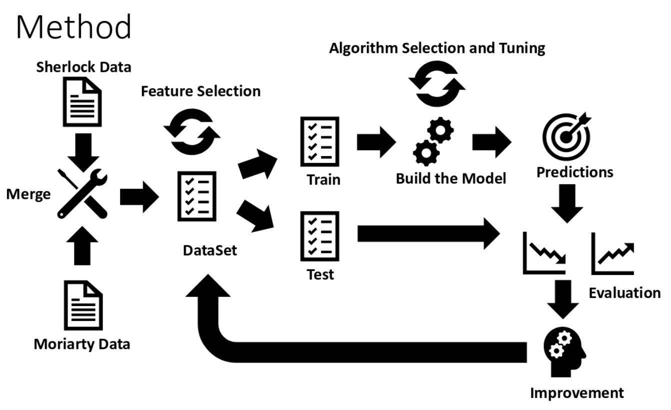

Figure 1: Milestone 3

Thursday, 06 - 12 - 2018

1 Testing our process to make sure that everything is alright

In order to probate that our process is correct in order to make the best model as we can.

At first what we do is cut the dataset, and stay only with the range of records in which there is an attack, to shorten as much as possible the search for attacks.
Then we add the feature **"attack"** to the dataset so that the method of labeling puts to one the records that must.

```
    df_no_labeled          = read_df("data/preprocessed.csv")
    df_labeled             = df_no_labeled[(df_no_labeled.UUID >=
       1461851815453) & ((df_no_labeled.UUID <= 1463565596193))]

    df_labeled['attack'] = 0
    label_dataset(df_labeled, df_Mor, 60000)
```

After doing this, we save the dataset full labeled in a new record.

Monday, 10 - 12 - 2018

1 Fixing the model

After beeing wondering for days why the heck we manage to achieve such a high-ass success rate. We spotted a mistake in the clustering goal. An error in the code make the clustering dataframe to be labeled. After having corrected the bug the system keeps on giving 100% success rate.

2 Adding path file

We also manage to add a routes file as the project guide suggests.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# 1_moriarty_labeling #
path_raw_data       = "data/preprocessed.csv"
path_raw_attacks    = "data/data_raw_attacks.csv"
path_raw_no_attacks = "data/data_raw_no_attacks.csv"
moriarty_to_fix     = "data/Moriarty.csv"
moriarty_fix        = "data/Moriarty_fix.csv"

# 2_preprocessing_data #
path_norm_attacks     = "data/data_norm_attacks.csv"
path_norm_no_attacks  = "data/data_norm_no_attacks.csv"

# 4 Clustering_data_no_attacks #
path_plot           = "plots/"
path_norm_centroids = "data/data_no_attacks_centroids.csv"

# 5 build_model #
path_prod_aux                 = "data/prod_sample_aux.csv"
path_prod_sample              = "data/prod_sample.csv"
path_processed_no_attacks     = "data/data_attacks_definitive.csv"
path_processed_attacks        = "data/data_no_attacks_definitive.csv"
path_merged_data              = "data/data_merged.csv"
path_train_aux                = "model/data_train_aux.csv"
path_test_aux                 = "model/data_test_aux.csv"
path_train                    = "model/data_train.csv"
```

Wednesday, 12 - 12 - 2018

1 Introducing Apache Spark and finishing Milestone 3

Yesterday we arraganged a meeting with Julio. After the meeting we decided to use spark in order to proof we know what to do.

After a bit of struggle we managed to replicate the random forest algorithm. We discovered that the amount of train/test is quite relevant, and with 80% of the data as train the algorithm behaves in a very alarmist way.

A jupyter notebook has been added to the suit, and yes, it has 100% success rate.

```
    from pyspark.ml.linalg import Vectors
    from pyspark.ml.feature import VectorAssembler

    assembler = VectorAssembler(inputCols = features, outputCol="
       features")
    assembled = assembler_train.transform(data_merged)

    (trainingData, testData) = assembled.randomSplit([0.67,0.33],
       seed=13234)
    trainingData.count(), testData.count()

    from pyspark.ml.classification import DecisionTreeClassifier

    d_tree = DecisionTreeClassifier(labelCol = "attack", featuresCol
       = "features", maxDepth=5,
                                minInstancesPerNode = 20, impurity =
                                   "gini")
    from pyspark.ml import Pipeline
    pipeline = Pipeline(stages=[d_tree])
    model = pipeline.fit(trainingData)

    predictions = model.transform(testData)
    predictions.select("prediction","attack").show()
```

After a ton of work we managed to create visual representation of the decision trees using sklearn plot library, sure they are going to be a great addition to the presentation.

Thursday, 13 - 12 - 2018

1 Milestone 3 conclusions

Giroscope rotation and Magnetic field is oddly important, is the most weighted one in order to predict attacks. Perhaps it's due to high resource consumption, or perhaps it's because the virus attacks when the user is in a phone call. Thinking it carefully we could study if the virus attack is ordered using low wave spectrum as the main channel, sounds like an interesting idea.

Anyway, after an attack, the magnetic field is SO altered by means of every wireless communication it's somewhat easy to guess if there's been an attack.

Final conclusions of the project

At first we had a big discussion about the idea of "packing" the data. If we group the data 3 by 3 we are predicting if an attack is going to happened in a minute, but the data says attacks happen every 20 seconds. That means the attack bracket is 3 times bigger, therefor is WAY easier to score. After do a test with a barebones version of T2. 100% accuracy, nothing to say.

Whereas our work is theoretically quite good we would like to see a system with this knowledge in a production environment, because a 100% success rate is a lot doesn't matter how good is the work.

And, in the end here is some visual candy of the generated decission tree.

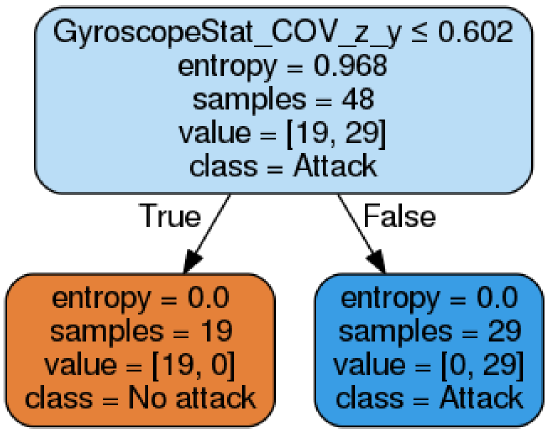

Figure 1: Milestone 3