RAPPORT TD LOG ANGLAIS  
GREGOIRE CAMPOS

BADR BELKEZIZ

**Initial Objective :**

The final goal of this project is to develop an intelligent software to automate the decisions of sports bettors. We limited ourselves to betting on football matches, and in particular the French first division: *Ligue 1*.

Within the academic framework of the TDLOG course, we set ourselves three objectives:

- To discover the different classification techniques, including machine learning algorithms.

- To implement these models in Python, both in terms of data processing and of the classification itself.

- Create a user interface to illustrate the algortihmes and play with the different possibilities.

We have distributed the tasks within the group as follows:

Belkeziz Badr: Data processing and implementation of machine learning algorithms

Campos Grégoire: Creation of the user interface and link between these algorithms.

**Machine Learning :**

**Data Collection :**

The first difficulty we encountered was collecting data on the matches. Where to find them? How to retrieve them? We first considered using a web crawler using the Python selenium library, but we ran into the problem of copyrights which forbade us to do so.

So we decided to use available and royalty-free data available for download at https://www.football-data.co.uk/francem.php. The .csv files provide us with statistical data on hundreds of games. Note that these CSV files are necessary for the proper execution of the python script. They must be placed in "Training/Files ...". **Corriger ici**

The ideal would have been to add a feature to regularly update the statistical data, for example via a crawler that would download new excel files every week, but we have not yet developed this feature.

**Data processing.**

The CSV file provides approximately 60 numeric data per match. We decided not to provide the 60 raw data to the algorithms for several reasons:

* Complexity of the algorithms: over several hundreds of matches, we would provide

1000\*60 data matrices, which greatly increases the processing time by the algorithms.

* Relevance of the data: Some of the data we would have liked to have was not in the

Csv file, others were not of interest to us.

It was therefore necessary to first select the relevant data (via the "Keys\_to\_keep" list) and then, from this list, to create other data that seemed to us relevant to the prediction of the final result. For example Home Team Goal Difference Before Game (HTGDBG) , Away Team Goal Difference Before Game (ATGDBG), Home Team Points Difference Before Game ...

**Implementation of algorithms :**

**A first model:**

We created a first very simple model returning the probability p that the home team wins by computing p as p = a\*HTGDBG + b\*HTPBG ???

Then we had to determine a and b manually. **Partir à compléter**

**Machine learning models:**

We used several supervised machine learning algorithms, meaning that they need the "Training\_X" and "Training\_Y" data to learn how to find a relation between X (the data of a game) and Y (the result of a game) , then test the efficiency of this relation by comparing the results of the algorithm (applied to "Testing\_X") to the "Testing\_Y" data.

We first trained the algorithms on the first 80% of the dataset before randomly selecting 80% of the dataset, thanks to the random.shuffle function of the random library. This random selection allows us to eliminate specificities such as for example training exclusively on games in the beginning of the season then testing on games at the end of the season …

We used 4 classifiers from the Sklearn library:

KNeighborsClassifier - DecisionTreeClassifier - GaussianNB - SVMClassifier.

By using the functions .fit(training\_X, training\_Y) then .predict (testing\_X) for each of these classifiers, we obtained a classification in the form of a C vector of size (1,3)

C=[1,0,0] for "Home team will win", C=[0,1,0] for "Draw", C=[0,0,1] for “away team will win”.

We decided to go a step further by implementing the functions predict\_proba() which now returns C in the form C=[x,1-(x+y),y] where x is the probability of "Home team will win", y the probability of the event "Away team will win".

The user will then have the possibility to set a threshold, the probability from which he accepts the bet.

Compared to the first model we considered, the advantages of these machine learning algorithms are multiple:

- automatic learning, no need for manual manipulation to find a and b

- Allows a multilabel classification in three categories and not binary "Home team will win or not".

**Why aren't we making a lot of money?**

As you will notice when using the software, we don't win 100% of our bet all the time. Even worse, the algorithms can predict that we will lose money on average.

Indeed, we are always aware that we still have several points to study in order to perfect our algorithms:

- Review the relevance of the data : Is the difference in goals and points really decisive for the final outcome? In spite of the interest of the data we have collected, we think that others can bring important information (Ex: team dynamics over the last 3 games, comparison of home vs. away results ...).

- Is the amount of data sufficient? Does the training require a larger database?

- Is there an overfitting problem, common with machine learning algorithms?

- Aren't there any other algorithms we could test? A neural network, for example, is another possible classification algorithm.