Big Data
Sport Analytics

Lecturer: FALIH Issam

He NI
Bader GUETARI

BI – 2

SUMMARY

Introduction

The aim of this project is to make a study about a specific dataset. Technically, when we talk about a study in the data analytics field, we think about every step of the process of data analysis. From the first step of data processing to the last step of predictions or visualization.

The topic chosen for this project was sport. Given a list of datasets concerning sport (basketball, baseball, soccer, hockey), we choose a set of data related to soccer. For this field of sport, we focused on the English soccer championship (the Barclays Premier League). The final goals of this project are to make a clear representation of the data by organizing different teams by games and so on, and mainly to predict the results of different games. Optionally we could have done more advanced predictions as suggested in the proposal subject of this project, but we chose mainly to focus on basic prediction.

This report will answer to the project proposal by first, presenting how we manage the data cleaning and processing step (what type of processing method we use), how we did our predictions (which model and type of classifying algorithms we used), what visualizations we did and finally we will discuss about the link between this problem and the use of big data techniques.

I – Data cleaning & pre-processing

1) *Selection of the dataset*

We have selected the football dataset from: http://openfootball.github.io/

According to the image on the right, this dataset divides the match data by year and match level. Because of the large amount of data, we only extracted the premier league data for each year, if we still have enough time, we will use the others like "The Championship"dataset.

Step 1: Download eng2020-21.rb Datafile (from GitHub) to your working folder as ./Datafile

Step 2: Run the sportdb build command

Step 2.a: Download all datasets listed in the Datafile as zip archives (from GitHub) to ./tmp

Step 2.b: Create the "empty" database, that is, table structure, indexes, etc... (schéma)

Step 2.c: Read in all datasets from the zip archives in ./tmp (no need to unpack)

Then we can get the data.csv file:

Div	Date	HomeTear	AwayTean	FTHG	FTAG	FTR	HTHG	HTAG	HTR
E0	13/08/11	Blackburn	Wolves	1	2	A	1	1	D
E0	13/08/11	Fulham	Aston Villa	0	0	D	0	0	D
E0	13/08/11	Liverpool	Sunderlanc	1	1	D	1	0	H
E0	13/08/11	Newcastle	Arsenal	0	0	D	0	0	D
E0	13/08/11	QPR	Bolton	0	4	A	0	1	A
E0	13/08/11	Wigan	Norwich	1	1	D	1	1	D
E0	14/08/11	Stoke	Chelsea	0	0	D	0	0	D
E0	14/08/11	West Brom	Man Unite	1	2	A	1	1	D
E0	15/08/11	Man City	Swansea	4	0	H	0	0	D

2) *Feature Engineering*

Step 1: Data cleaning - clean up abnormal samples and remove null values

The English Premier League is the top league of professional soccer in England, with 20 teams and a double round robin home and away system, with 38 rounds of matches per season, 10 matches per round, and 380 matches per season. So we drop the csv document which are not composed of 380 lines.

```
for i in range(len(res_name)):
    res_name[i] = pd.read_csv(loc+filecsv_list[i],error_bad_lines=False)
    print('the%2s document is %s,size is %s'%(i+1,filecsv_list[i],res_name[i].shape))
```

```
the 1 document is 2010-11.csv,size is (380, 71)
the 2 document is 2011-12.csv,size is (380, 71)
the 3 document is 2012-13.csv,size is (380, 74)
the 4 document is 2013-14.csv,size is (380, 68)
the 5 document is 2014-15.csv,size is (381, 68)
the 6 document is 2015-16.csv,size is (380, 65)
the 7 document is 2016-17.csv,size is (380, 65)
the 8 document is 2017-18.csv,size is (380, 65)
the 9 document is 2018-19.csv,size is (380, 62)
the10 document is 2019-20.csv,size is (380, 106)
the11 document is 2020-21.csv,size is (170, 106)
```

Here we drop the csv which are not composed of 380 lines

```
for i in range(len(res_name),0,-1):
    if res_name[i-1].shape[0] != 380:
        key = 'res_name[' + str(i) + ']'
        print('the drop data is: %s, csv: %s size: %s'%(time_list[i-1],key,res_name[i-1].shape))
        res_name.pop(i-1)
        time_list.pop(i-1)
        continue
```

```
the drop data is: 2020, csv: res_name[11] size: (170, 106)
the drop data is: 2014, csv: res_name[5] size: (381, 68)
```

Step 2: Data pre-processing:

	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTGD	ATGD	HTP	ATP	HM1	AM1	HM2	AM2	HM3	AM3	MW
3415	Leicester	Man United	0	2	A	0.736842	0.736842	1.631579	1.657895	L	D	W	W	L	D	38.0
3416	Man City	Norwich	5	0	H	1.631579	-1.157895	2.052632	0.552632	W	L	W	L	W	L	38.0
3417	Newcastle	Liverpool	1	3	A	-0.473684	1.315789	1.157895	2.526316	D	W	L	L	L	D	38.0
3418	Southampton	Sheffield United	3	1	H	-0.289474	0.052632	1.289474	1.421053	W	L	D	L	D	W	38.0
3419	West Ham	Aston Villa	1	1	D	-0.342105	-0.684211	1.000000	0.894737	D	W	W	D	W	W	38.0

Feature explanation：HTGD,ATGD,HTP,ATP has already been divided by weeks and get the mean.

•*FTHG & FTAG:* Full Time Home/Away Team Goals

•*FTR*: Full Time Result (H=Home Win, D=Draw, A=Away Win)

•*HTGD & ATGD:* Till current match, home/away team goals difference.

Example: For the row 3415, the hometeam Leicester get -2 goals difference in this match, and we add -2 to the previous Leicester goals difference(all past matches of Leicester) and divide it by weeks(38) and get the HTGD = 0.736842, same way with ATGD

•*HTP & ATP:* Win = 3 point, Draw = 1 point, Lose = 0 point. Home/Away Team cumulative score as of the current competition week

•*HM123 & AM123:* Represents the Home/Away Team's last 3 match win or loss.

•*MW* : Weeks

We can see the HTGD, ATGD, HTP, ATP, HM123, AM123, MW are all the created features, so we are going to explain how we create them.

- **HTGD & ATGD**

(Calculate the **cumulative** goal difference per team per week)

First create a dictionary (get_goals_diff) with the name of each team as the key.

The goal difference of the home team is added to the corresponding home team in the dictionary.
Add the goal difference of the away team to the corresponding away team in the dictionary.
Create a Goals Different data frame that returns Goals Different.
Then create a method (get_gss) to iterate the goal difference by 380 games and append it to HTGD and ATGD.

	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTGD	ATGD
375	Newcastle	West Brom	3	3	D	-1	-15
376	Stoke	Wigan	0	1	A	-1	-22
377	Tottenham	Birmingham	2	1	H	8	-20
378	West Ham	Sunderland	0	3	A	-24	-14
379	Wolves	Blackburn	2	3	A	-19	-14

- **HTP & ATP**

(Statistics of cumulative points scored by home and away teams up to the current game week)

Create a method to convert scores (get_points) We convert the result of a game into points, three points for a win, one point for a tie, and zero points for a loss.

Create a method to get the cumulative points per game (get_cuml_points)

Create a dictionary (get_matchres) with each team's name as the key, record the match results, FOR loop each match, H for home win, A for away win, D for draw.

Create a method (get_agg_points) to iteratively calculate the scores of each match and add the cumulative home and away scores to the HTP and ATP respectively.

	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTGD	ATGD	HTP	ATP
375	Newcastle	West Brom	3	3	D	-1	-15	45	46
376	Stoke	Wigan	0	1	A	-1	-22	46	39
377	Tottenham	Birmingham	2	1	H	8	-20	59	39
378	West Ham	Sunderland	0	3	A	-24	-14	33	44
379	Wolves	Blackburn	2	3	A	-19	-14	40	40

- **HM123 & AM123**

(Statistics of a team's performance in the last three matches)
Create a method (get_form) to get the results of the matches in the dictionary get_matchres.
Create a method (add_form) to put the results of the first 3 matches into

Create a method (add_form_df) to add the results to the table.

	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTGD	ATGD	HTP	ATP	HM1	AM1	HM2	AM2	HM3	AM3
375	Newcastle	West Brom	3	3	D	-1	-15	45	46	D	W	W	L	L	W
376	Stoke	Wigan	0	1	A	-1	-22	46	39	L	W	W	D	D	D
377	Tottenham	Birmingham	2	1	H	8	-20	59	39	W	L	L	L	D	D
378	West Ham	Sunderland	0	3	A	-24	-14	33	44	L	L	D	W	L	L
379	Wolves	Blackburn	2	3	A	-19	-14	40	40	W	D	W	D	D	W

- **MW**

Join the competition week feature (the first few competition weeks)

	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTGD	ATGD	HTP	ATP	HM1	AM1	HM2	AM2	HM3	AM3	MW
0	Blackburn	Wolves	1	2	A	0	0	0	0	M	M	M	M	M	M	1
1	Fulham	Aston Villa	0	0	D	0	0	0	0	M	M	M	M	M	M	1
2	Liverpool	Sunderland	1	1	D	0	0	0	0	M	M	M	M	M	M	1
3	Newcastle	Arsenal	0	0	D	0	0	0	0	M	M	M	M	M	M	1
4	QPR	Bolton	0	4	A	0	0	0	0	M	M	M	M	M	M	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
375	Sunderland	Man United	0	1	A	0	55	45	86	L	W	D	L	D	D	38
376	Swansea	Liverpool	1	0	H	-8	8	44	52	L	W	D	L	D	W	38
377	Tottenham	Fulham	2	0	H	23	-1	66	52	D	W	W	W	W	L	38
378	West Brom	Arsenal	2	3	A	-6	24	47	67	D	D	D	D	W	D	38
379	Wigan	Wolves	3	2	H	-21	-41	40	25	W	D	W	D	L	L	38

Combining the information of the games

We are going to merge all the information of the games in the dataset into one table, and then we divide these scoring data we just calculated, the goal difference data by the number of weeks to get the value after the weekly average. The result can be seen by looking at the last 5 data items of the dataset after constructing the features.

	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTGD	ATGD	HTP	ATP	HM1	AM1	HM2	AM2	HM3	AM3	MW
3415	Leicester	Man United	0	2	A	0.736842	0.736842	1.631579	1.657895	L	D	W	W	L	D	38.0
3416	Man City	Norwich	5	0	H	1.631579	-1.157895	2.052632	0.552632	W	L	W	L	W	L	38.0
3417	Newcastle	Liverpool	1	3	A	-0.473684	1.315789	1.157895	2.526316	D	W	L	L	L	D	38.0
3418	Southampton	Sheffield United	3	1	H	-0.289474	0.052632	1.289474	1.421053	W	L	D	L	D	W	38.0
3419	West Ham	Aston Villa	1	1	D	-0.342105	-0.684211	1.000000	0.894737	D	W	W	D	W	W	38.0

Deleting some data

In the previous section, we constructed many features based on the initial features. Some of these features are intermediate features, and we need to discard these intermediate features. Because there is not enough information about each team's

history of winning and losing in the first three weeks of the game, we intend to discard the data of the first three weeks.

```
#Remove intermediate features
playing_stat = playing_stat[playing_stat.MW > 3]
playing_stat.drop(['HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'MW'],1, inplace=True)
```

Solving the sample imbalance problem

After constructing the features, it is found that the percentage of home field wins is close to 50%, so for this three-classification problem, the label proportion is unbalanced.
We simplify it to a binary classification problem, that is, whether the home team will win, which is also a way to solve the problem of unbalanced label proportion.

Dividing data into feature values and label values :

```
def only_hw(string):
    if string == 'H':
        return 'H'
    else:
        return 'NH'
playing_stat['FTR'] = playing_stat.FTR.apply(only_hw)
# Dividing data into feature values and label values
X_all = playing_stat.drop(['FTR'],1)
y_all = playing_stat['FTR']

len(X_all)
```

```
3150
```

Data normalization and standardization

```
def convert_1(data):
    max=data.max()
    min=data.min()
    return (data-min)/(max-min)
r_data=convert_1(X_all['HTGD'])

# Data standardization
cols = [['HTGD','ATGD','HTP','ATP']]
for col in cols:
    X_all[col] = scale(X_all[col])
```

Converting feature data types (Dummy)

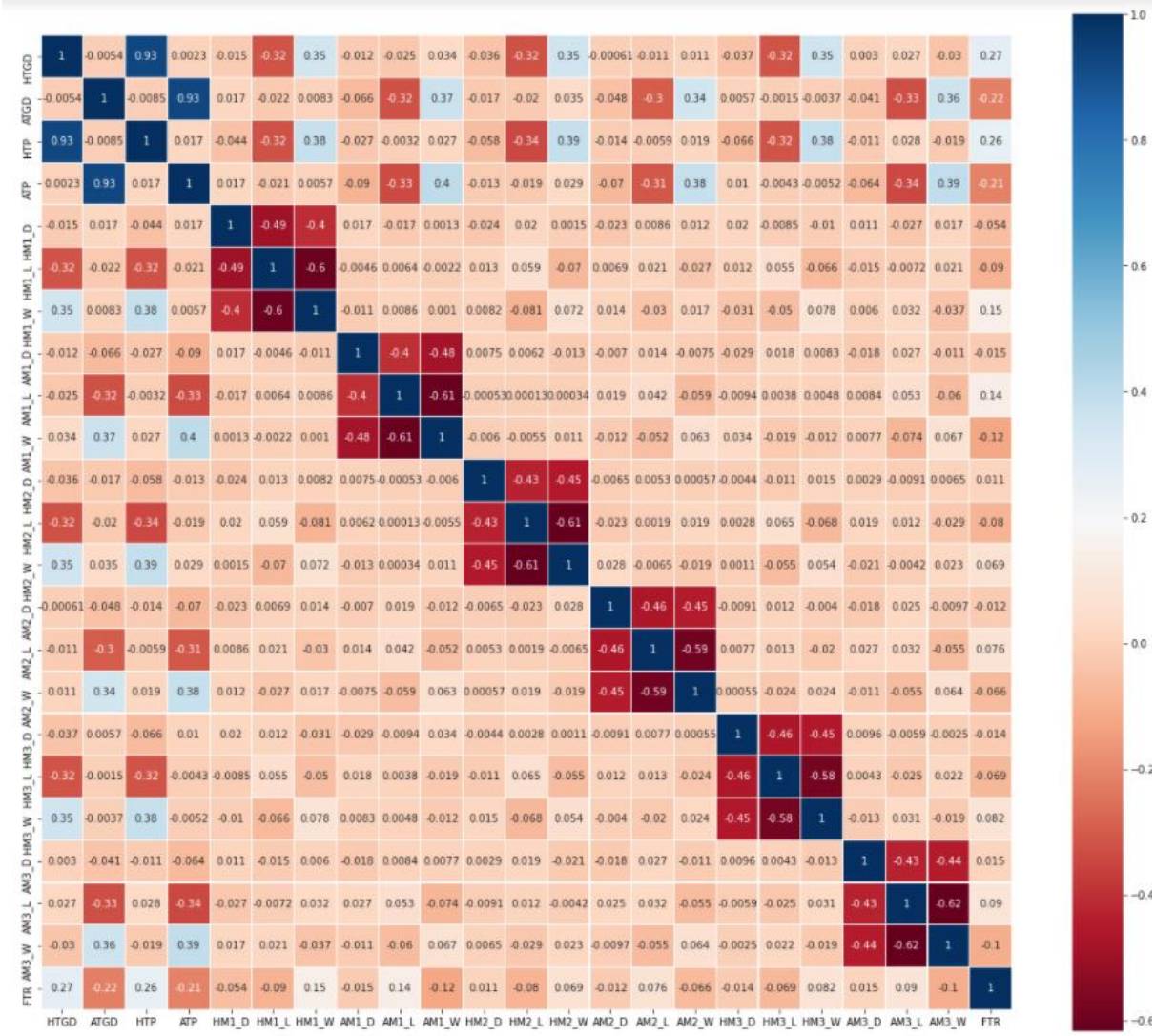
```
X_all.HM1 = X_all.HM1.astype('str')
X_all.HM2 = X_all.HM2.astype('str')
X_all.HM3 = X_all.HM3.astype('str')
X_all.AM1 = X_all.AM1.astype('str')
X_all.AM2 = X_all.AM2.astype('str')
X_all.AM3 = X_all.AM3.astype('str')

def preprocess_features(X):
    '''Converting discrete type features to dummy encoded features  '''
    output = pd.DataFrame(index = X.index)
    for col, col_data in X.iteritems():
        if col_data.dtype == object:
            col_data = pd.get_dummies(col_data, prefix = col)
        output = output.join(col_data)
    return output

X_all = preprocess_features(X_all)
print("Processed feature columns ({} total features):\n{}".format(len(X_all.columns), list(X_all.columns)))
```

```
Processed feature columns (22 total features):
['HTGD', 'ATGD', 'HTP', 'ATP', 'HM1_D', 'HM1_L', 'HM1_W', 'AM1_D', 'AM1_L', 'AM1_W', 'HM2_D', 'HM2_L', 'HM2_W', 'AM2_D', 'AM2_
L', 'AM2_W', 'HM3_D', 'HM3_L', 'HM3_W', 'AM3_D', 'AM3_L', 'AM3_W']
```

3) Pearson related heat map



The above graph shows that the HTP and HTGD features are strongly correlated, and the ATP and ATGD features are also strongly correlated, indicating a case of multicollinearity. It is also easy to understand that the higher the average home weekly score, the higher the average home weekly goal differential.

If we consider these variables, we can conclude that they give almost the same information and therefore multicollinearity occurs, and here we would consider removing the two features HTP and 'ATP' and keeping the two features HTGD and ATGD.

Pearson heat maps are well suited to detect this situation and they are an essential tool in feature engineering. Also, we can see that the results of the last team's last match have less influence on the results of the current match, and here we consider keeping these features.

- Considering that the correlation between the sample set features HTP and HTGD, ATP and ATGD is over 90%, we remove the features HTP , ATP.
- Take a look at the 10 most relevant characteristics of FTR

We can see that the most relevant feature is HTGD, which indicates that the higher a team's weekly average goal difference at home, the higher their probability of winning.

To summarize:

Single features: We normalize normalization and dummy variables.

Multiple features:

1. We reduce the dimensionality: PCA or LDA (we do not have to reduce the dimensionality, we end up with 4 features, so there is no dimensional disaster, no need to reduce the dimensionality)
2. Feature selection: we have to consider the relationship between variables first, such as which features are related to home team or away team winning, then we need to calculate the correlation coefficient (hotspot graph), or information gain (cross entropy). But we do not choose to use decision trees, so we just need to calculate the correlation coefficients, find the most relevant few features, and then use several supervised learning methods for analysis and prediction.
3. Here we choose SVM, Logistic Regression, and XGB Classifier. We will discuss the prediction methods in part2
4. We need to choose the useful features from the dataset, and do the normalization and standardization or even dummy variables, to make our dataset become more

II – Predictions with machine learning

1) *Logistic Regression*

The logistic regression model assumes that the data obeys Bernoulli distribution, and uses gradient descent to solve the parameters by maximizing the likelihood function to achieve the purpose of dichotomizing the data. The main advantages of this model are that it has good explanations; if the feature engineering is done well, the model effect is also very good; the training speed is also relatively fast; and the output results are easily adjustable. However, the disadvantages of this model are also prominent, such as: the accuracy rate is not very high, and it is difficult to deal with the data imbalance problem.

2) *SVC Classifier*

SVM (Support Vector Machine) is a two-class classification model. Its basic model is to find linear classifiers in the feature space with separated hyperplanes with maximized intervals.

(1) When the training samples are linearly divisible, a linear classifier, i.e., a linearly divisible support vector machine, is learned by hard interval maximization.

(2) When the training data are approximately linearly divisible, a linear classifier, i.e., a linear support vector machine, is learned by soft interval maximization by introducing relaxation variables.

(3) When the training data are linearly indistinguishable, a nonlinear support vector machine is learned by using kernel tricks and soft interval maximization.

3) *XGB Classifier*

XGBoost is one of the Boosting algorithms. The idea of Boosting algorithm is to integrate many weak classifiers together to form a strong classifier, and the basic principle is that the next decision tree input samples will be correlated with the training and prediction of the previous decision tree. The idea is that XGBoost is a boosted tree model, so it integrates many tree models together to form a strong classifier. And the tree model used is the CART regression tree model.

Code processing split data :

```
import joblib
import xgboost as xgb
from time import time
from sklearn.svm import SVC
from sklearn.metrics import f1_score
from sklearn.metrics import make_scorer
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_all, y_all,test_size = 0.3,random_state = 2,stratify = y_all)
```

3-1) Build machine learning models and evaluate

A. *Modeling*

```
def train_classifier(clf, X_train, y_train):
    ''' Training models '''
    # Record training hours
    start = time()
    clf.fit(X_train, y_train)
    end = time()
    print("train time {:.4f} second".format(end - start))

def predict_labels(clf, features, target):
    ''' Using models for forecasting '''
    # Recording of forecast hours
    start = time()
    y_pred = clf.predict(features)
    end = time()
    print("predict time in {:.4f} second".format(end - start))
    return f1_score(target, y_pred, pos_label=1), sum(target == y_pred) / float(len(y_pred))

def train_predict(clf, X_train, y_train, X_test, y_test):
    ''' Train and evaluate models '''
    # Indicate the classifier and the training set size
    print("Training {} model, number of samples {}。".format(clf.__class__.__name__, len(X_train)))
    # Training models
    train_classifier(clf, X_train, y_train)
    # Evaluating the model on the test set
    f1, acc = predict_labels(clf, X_train, y_train)
    print("The F1 scores and accuracies on the training set are: {:.4f} , {:.4f}。".format(f1 , acc))

    f1, acc = predict_labels(clf, X_test, y_test)
    print("The F1 scores and accuracies on the test set are: {:.4f} , {:.4f}。".format(f1 , acc))
```

B. *Initialize, train and evaluate the model separately*

Get the F1 score and accuracies for each model on train/test set:

```
Training LogisticRegression model, number of samples 2205。
train time 0.0098 second
predict time in 0.0010 second
The F1 scores and accuracies on the training set are: 0.6151 , 0.6658。
predict time in 0.0010 second
The F1 scores and accuracies on the test set are: 0.5621 , 0.6307。

Training SVC model, number of samples 2205。
train time 0.1786 second
predict time in 0.1035 second
The F1 scores and accuracies on the training set are: 0.6098 , 0.6807。
predict time in 0.0449 second
The F1 scores and accuracies on the test set are: 0.5210 , 0.6265。

Training XGBClassifier model, number of samples 2205。
[17:54:29] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
train time 0.1376 second
predict time in 0.0039 second
The F1 scores and accuracies on the training set are: 0.9538 , 0.9578。
predict time in 0.0020 second
The F1 scores and accuracies on the test set are: 0.5457 , 0.6159。
```

3-2) Hyperparameter adjustment

We found that XGB Classifier performs very well in the training set, so we decided to use XGB Classifier and use Grid Search for tuning, and finally get the best model.

```
F1 score and accuracy score for training set: 0.8834 , 0.8952。
predict time in 0.0020 second
F1 score and accuracy score for test set: 0.5279 , 0.5968。
```

3-3) Saving models and loading models

```
#SaveModel
joblib.dump(clf, 'xgboost_model.model')
#ReadModel
xgb = joblib.load('xgboost_model.model')
```

Then we try to make a prediction, we choose 2019-2020 Season matches as the sample and we get a 79.2% Correct Rate. Because to reduce errors, we drop the first 30 matches of each season. So we have 350 matches as the sample.

```
The True FTR is:
      FTR        HomeTeam          AwayTeam        Date
3070    0     Southampton        Man United  31/08/2019
3071    0         Chelsea  Sheffield United  31/08/2019
3072    1  Crystal Palace       Aston Villa  31/08/2019
3073    1       Leicester       Bournemouth  31/08/2019
3074    1        Man City          Brighton  31/08/2019
...   ...             ...               ...         ...
3415    0       Leicester        Man United  26/07/2020
3416    1        Man City           Norwich  26/07/2020
3417    0       Newcastle         Liverpool  26/07/2020
3418    1     Southampton  Sheffield United  26/07/2020
3419    0        West Ham       Aston Villa  26/07/2020

[350 rows x 4 columns]
The Prediction FTR is:[0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 1 0
 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0
 1 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 0 1 1 0 0
 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0
 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1
 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0
 1 0 0 0 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 1 0
 1 0 0 1 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 1 0
 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0
 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0]

accurancy is 0.7914285714285715
```

1 : Home team win

0 : Away team win

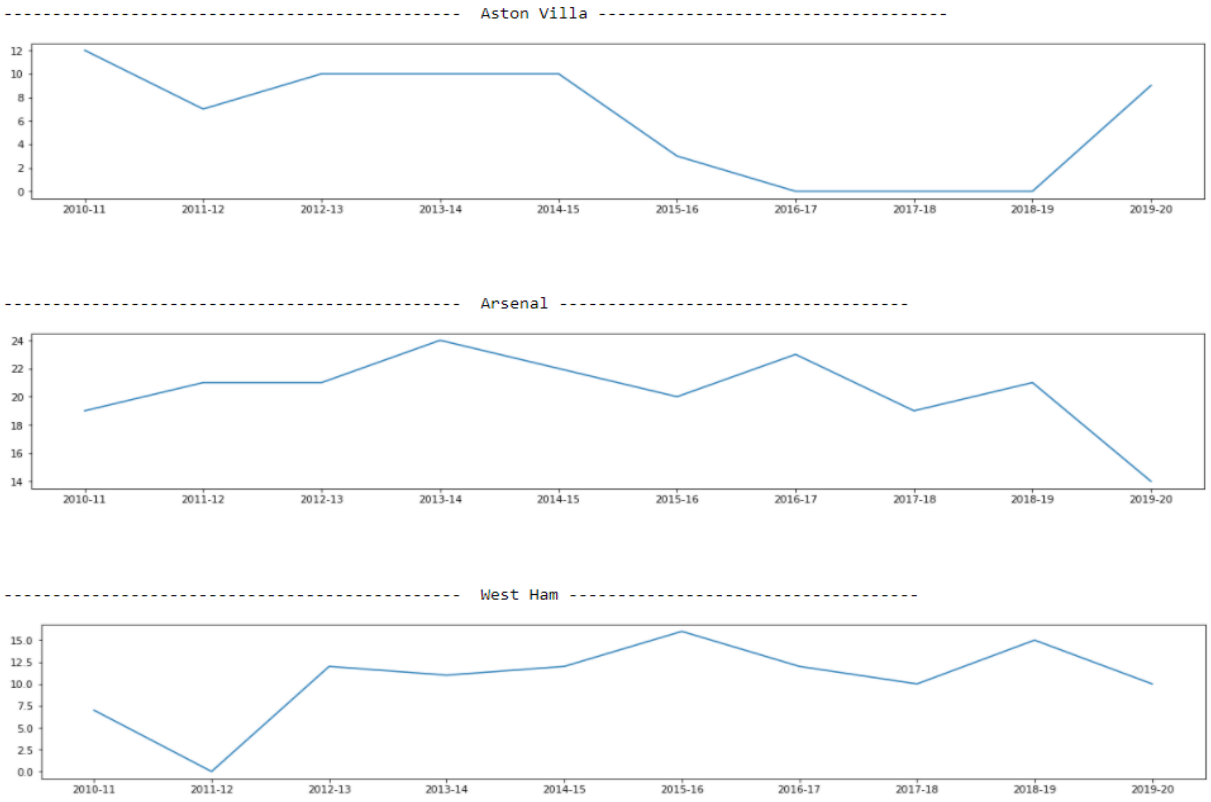
III – Data visualization

For this part, we are looking to visualize some aspect about the different teams of our dataset. The main thing that we can display and that can be useful for a team to have these informations concern their basic statistics. Those statistics given in the form of pie, plots or charts and so on that give information about the performance of a team can help to figure what is going on in order to improve their efficiency. Those aspects of analysis to improve efficiency are called insights.

1) *Insights*

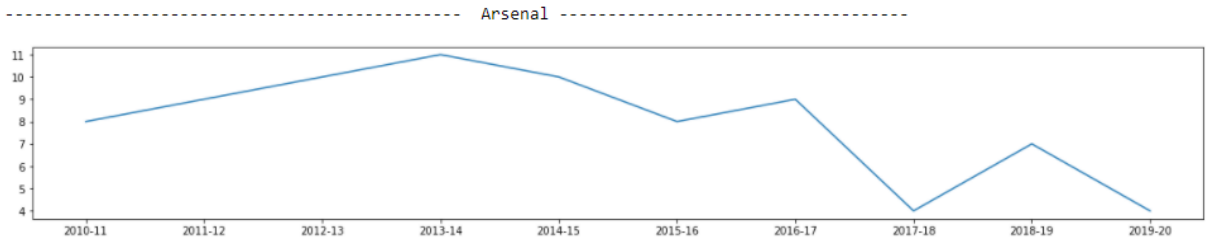
For basic information, we chose to retrieve the efficiency of a team. That is to say how they win, when they win, and their global efficiency during one season for every season of the dataset that we previously cleaned. To do so, we have created some functions that calculate for a given data frame the number of wins of a specific team at home and away.

Number of wins during 10 season for the teams Aston Villa, Arsenal and West Ham clubs

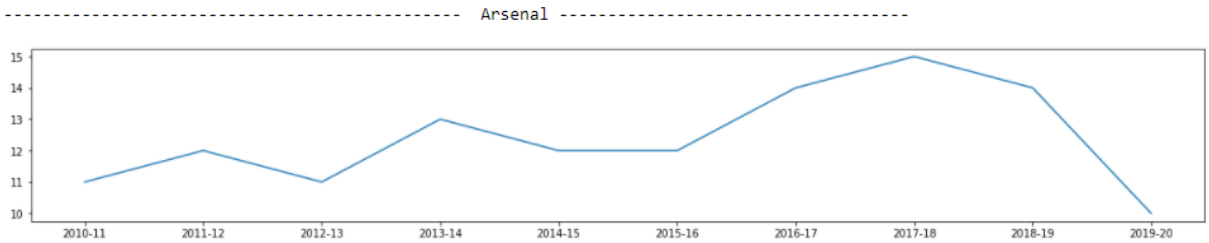


With this kind of visualization we can see the evolution of the efficiency from a season to another. We can see that Arsenal is quietly decreasing its amount of wins from one season to another. With that, we can logically think about how this plot will evolve if we add the current season of 2020-2021. We can also push this analysis for Arsenal for example to see how the efficiency of the team is impact by the fact that they play away or at home :

Number of outside wins during 10 season for Arsenal club



Number of inside wins during 10 season for Arsenal club



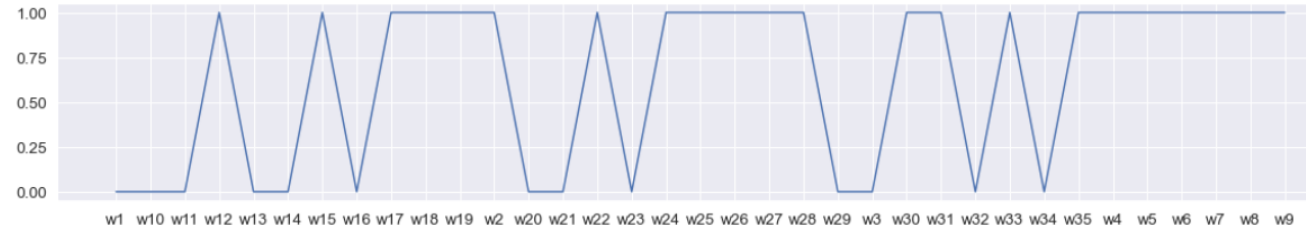
With this kind of visualization, we can easily see that the numbers of wins outside for the previous seasons were way higher than outside. We can think that the reason why Arsenal has lost some ranking place is due to the fact that they are not as efficient, outside, as they used to be in the beginning of the decade. This leads to the interpretation that this club should figure why they are winning less outside than before to be more efficient. These kinds of interpretations are what data visualization is made for.

2) *Predictions*

Here for the visualization part, we also have added some charts for our prediction models. As the previous part is showing, we can predict a win or a loss of a team. We did that for 5 random games. We also did that to see how a specific team is going to perform, week by week for the season 2019-2020 for example:

Number of wins that we predict for Chelsea in

2019-2020 season



1.00 corresponds to a win and 0.00 to a loss of Chelsea, and w corresponds to week. We can see that they are going to be in the best shape from week 24 to 29 and week 4 to 5 for a total of 10 games.

IV – Problem that we encountered

When downloading original data from GitHub, we need to use sportdb. it's a new tool for us , and more about SQL commands. It took us a long time to figure out how to use it to get the data we needed.

XGB: XGB has never been used before, and we were considering whether to use other classification methods, such as knn, but we saw an article online that showed that xgb, as a collection of tree models, brings together many weak classifiers to form a strong classifier, and then the classification accuracy is higher.

If you are more aware of the English soccer game, there are several levels. In addition to the Premier League, which is the highest level of the game, there is also the Championship below it. Then the top 4 teams in the Championship every year are automatically promoted to the Premier League, so join us to predict the match of the Premier League in 2021, when there will be 4 teams joined without past data, then we can't predict their winning percentage, and this is one of the problems we have to solve in the future.

Conclusion

To conclude with this project, we can say that this one was really enriching due to its technical notions. Indeed, during our studies, we have rarely done a project that starts from a non-cleaned dataset to the last part of predictions. We always have had to make visualization or predictions for a pre-cleaned given dataset. Here the fact that we should apply data processing, with normalization correlation matrix and so on, was really challenging. Indeed, all the project is based on that.

On the other hand, it was also a concrete project that can be useful in real life. We can take the example of a team staff by making basic statistics of their previous seasons and games, by analyzing what are their weak points and strong points. This is a case when we really find the help of IT and specifically data science in the sport fields.

Thanks to this project, our way of looking at data science has really evolved, and we can’t wait to see more about it in professional life.