
# FINAL REPORT

## Data Analysis in Python

Topic: **OUTCOME OF MATCHES PREDICTION MODEL**

**Lecturer** *Nguyen Ngoc Tu*

**Student** *Nguyen Tuan Hung*

*Ho Chi Minh City, 28th February 2022*

# Contents

## 1. Introduction

The goal of this project is to predict outcomes of European football matches in Python. Based on the dataset provided on https://www.kaggle.com/hugomathien/soccer that includes basic match data, FIFA player statistics and bookkeeper data, I built a model to predict the probability of each match outcome: home win, draw, or home lose.

Two of most important things I used are feature transformation and dimensionality reduction techniques that in order to improve the quality of the features. This helped me to make some simple analyses from the actual key features of each match and what relationships between them and the outcomes. Lastly, I kept the 10 most useful features to predict our output and given them into machine learning model.

This European Soccer Database has more than 25,000 matches, more than 10,000 players for European professional soccer seasons from 2008 to 2016, 11 European Countries with their lead championship, players and Teams's attributes sourced from EA Sports' FIFA video game series, team line up with squad formation, betting odds from up to 10 providers and detailed match events (goal types, possession, corner, cross, fouls, cards etc...) for more than 10,000 matches.

Analytical process: some steps for exploring and cleaning the dataset, some steps for analyzing relationships between features and label using basic statistics, and some steps for predicting the outcome using Random Forest algorithm.

## 2. Process (Step-by-step)

### 2.1. *Loading required data*

Loading matches data from CSV file to Dataframe that will give us the scores and opposing teams. I used also player’s attributes in order to filter player's ratings and compare overall team ratings. The following is all complete dataframes of the features uses in the analysis:

| Index | id | eam_api_ic | am_fifa_api_ | team_long_name | m_short_na |
|---|---|---|---|---|---|
| 48 | 4996 | 8483 | 1926 | Blackpool | BLA |
| 85 | 11822 | 4087 | 111271 | Évian Thonon Gaillard FC | ETG |
| 115 | 16848 | 8350 | 29 | 1. FC Kaiserslautern | KAI |
| 283 | 48358 | 8306 | 472 | UD Las Palmas | LAS |
| 259 | 43043 | 8696 | 456 | Racing Santander | SAN |
| 270 | 43054 | 9864 | 573 | Málaga CF | MAL |
| 112 | 16237 | 8358 | 25 | SC Freiburg | FRE |
| 193 | 31455 | 8322 | 111082 | Arka Gdynia | ARK |
| 8 | 9 | 7947 | nan | FCV Dender EH | DEN |
| 133 | 20522 | 9885 | 45 | Juventus | JUV |
| 297 | 50204 | 7730 | 1862 | FC Lausanne-Sports | LAU |

**Figure 1: Team data**

| id | country_id | league_id | season | stage | date | natch_api_i | ne_team_ap | ay_team_ap |
|---|---|---|---|---|---|---|---|---|
| 6970 | 4769 | 4769 | 2013/2014 | 37 | 2014-05-10 00:00:00 | 1468561 | 9851 | 9874 |
| 2146 | 1729 | 1729 | 2009/2010 | 12 | 2009-11-07 00:00:00 | 658726 | 10252 | 8559 |
| 15749 | 15722 | 15722 | 2008/2009 | 12 | 2008-11-08 00:00:00 | 506624 | 8025 | 8028 |
| 20175 | 19694 | 19694 | 2010/2011 | 13 | 2010-11-13 00:00:00 | 840302 | 9938 | 8597 |
| 5624 | 4769 | 4769 | 2010/2011 | 18 | 2010-12-18 00:00:00 | 830119 | 9847 | 9829 |
| 7240 | 4769 | 4769 | 2014/2015 | 27 | 2015-02-28 00:00:00 | 1709958 | 8588 | 9851 |
| 22804 | 21518 | 21518 | 2011/2012 | 22 | 2012-02-04 00:00:00 | 1051855 | 8581 | 8696 |
| 9147 | 7809 | 7809 | 2012/2013 | 20 | 2013-02-02 00:00:00 | 1239645 | 8721 | 8406 |
| 12978 | 10257 | 10257 | 2015/2016 | 17 | 2015-12-20 00:00:00 | 2060588 | 9891 | 8564 |
| 15548 | 13274 | 13274 | 2015/2016 | 22 | 2016-02-07 00:00:00 | 1983467 | 8526 | 8464 |
| 12991 | 10257 | 10257 | 2015/2016 | 18 | 2016-01-06 00:00:00 | 2060612 | 8540 | 8535 |

**Figure 2: Match data**

| Index | id | yer_fifa_api | layer_api_i | date | verall_ratin | potential | eferred_fo | cking_work_ | nsive_work_ | crossing |
|---|---|---|---|---|---|---|---|---|---|---|
| 59994 | 59995 | 194149 | 180283 | 2012-08-31 00:00:00 | 62 | 74 | right | medium | medium | 33 |
| 114808 | 114809 | 684 | 30633 | 2013-04-12 00:00:00 | 79 | 79 | right | medium | medium | 12 |
| 99109 | 99110 | 191061 | 164210 | 2014-03-14 00:00:00 | 67 | 67 | right | medium | medium | 54 |
| 78338 | 78339 | 192350 | 172324 | 2015-10-09 00:00:00 | 74 | 76 | right | None | 7 | 71 |
| 53074 | 53075 | 53036 | 26141 | 2008-08-30 00:00:00 | 65 | 73 | right | medium | medium | 58 |
| 140205 | 140206 | 186386 | 164017 | 2011-08-30 00:00:00 | 61 | 64 | left | medium | medium | 70 |
| 40244 | 40245 | 24248 | 37545 | 2009-02-22 00:00:00 | 82 | 85 | right | medium | medium | 81 |
| 111010 | 111011 | 176894 | 23675 | 2013-06-07 00:00:00 | 78 | 78 | right | high | medium | 70 |
| 43695 | 43696 | 219693 | 570434 | 2015-10-30 00:00:00 | 68 | 75 | right | medium | high | 35 |
| 174846 | 174847 | 183125 | 72436 | 2014-09-18 00:00:00 | 71 | 75 | right | low | medium | 39 |
| 18488 | 18489 | 163419 | 26111 | 2015-10-02 00:00:00 | 79 | 79 | right | high | medium | 74 |

**Figure 3: Players's attributes data**

2.2. *Raw data preparation*

I decided to remove useless columns so as to not analyze with meaningless informations. I have kept id of player who have played, id for each team (key), the date of the match and number of goals scored by each team (label).

```
home_players = ["home_player_" + str(x) for x in range(1, 12)]
away_players = ["away_player_" + str(x) for x in range(1, 12)]

matches_kept_columns = ["id","match_api_id", "date", "home_team_api_id", "away_team_api_id",
                        "home_team_goal", "away_team_goal"]
matches_kept_columns = matches_kept_columns + home_players
matches_kept_columns = matches_kept_columns + away_players

matches_df = matches_df[matches_kept_columns]
```

Real data is never clean. We need to make sure we clean the data by converting or getting rid of null or missing values. The next code cell would show you if any of the rows have null value.

```
In [9]: matches_df.isnull().any().any(), matches_df.shape
Out[9]: (True, (27000, 29))
```

Then, I would find how many data points in each column are null.

| Index | 0 |
|---|---|
| id | 0 |
| match_api_id | 0 |
| date | 0 |
| home_team_api_id | 0 |
| away_team_api_id | 0 |
| home_team_goal | 0 |
| away_team_goal | 0 |
| home_player_1 | 1277 |
| home_player_2 | 1372 |
| home_player_3 | 1332 |
| home_player_4 | 1375 |
| home_player_5 | 1370 |
| home_player_6 | 1380 |
| home_player_7 | 1278 |
| home_player_8 | 1367 |
| home_player_9 | 1320 |
| home_player_10 | 1496 |
| home_player_11 | 1621 |
| away_player_1 | 1286 |
| away_player_2 | 1328 |
| away_player_3 | 1353 |
| away_player_4 | 1373 |
| away_player_5 | 1394 |
| away_player_6 | 1360 |
| away_player_7 | 1283 |

I decided to fix null values by deleting them. Again, I checked the null values and number of rows, we would see that there are no null values and number of rows decreased accordingly.

| | |
|---|---|
| away_team_api_id | 0 |
| home_team_goal | 0 |
| away_team_goal | 0 |
| home_player_1 | 0 |
| home_player_2 | 0 |
| home_player_3 | 0 |
| home_player_4 | 0 |
| home_player_5 | 0 |
| home_player_6 | 0 |
| home_player_7 | 0 |
| home_player_8 | 0 |
| home_player_9 | 0 |
| home_player_10 | 0 |
| home_player_11 | 0 |
| away_player_1 | 0 |
| away_player_2 | 0 |
| away_player_3 | 0 |
| away_player_4 | 0 |
| away_player_5 | 0 |
| away_player_6 | 0 |
| away_player_7 | 0 |
| away_player_8 | 0 |
| away_player_9 | 0 |
| away_player_10 | 0 |
| away_player_11 | 0 |

```
In [14]: matches_df.isnull().any().any(), matches_df.shape
Out[14]: (False, (22224, 29))
```

Then, I created some new features in order to serve for analysis:

- **goal_difference** -> the difference of number of goals between home and away teams
- **home_status** -> HL if home team loses, HW if it wins else it stores D. It will be used as label to predict.
- **overall_rating_home** -> total rating of each player in the home team
- **overall_rating_away** -> total rating of each player in the away team
- **min_overall_rating_home** -> minimum rating of each player in the home team
- **min_overall_rating_away** -> minimum rating of each player in the away team
- **max_overall_rating_home** -> maximum rating of each player in the home team
- **max_overall_rating_away** -> maximum rating of each player in the away team
- **mean_overall_rating_home** -> mean rating of each player in the home team
- **mean_overall_rating_away** -> mean rating of each player in the away team
- **std_overall_rating_home** -> standard deviation rating of each player in the home team
- **std_overall_rating_away** -> standard deviation rating of each player in the away team
- **overall_rating_difference** -> difference of total rating between the two teams

After merging new features with the initial dataframe, I dropped player's rating, removed the matches that have the same **match_api_id** (key) and filter the latest dates of each unique matches. Finally, to perform further analysis, I merged home and away team's names with the main dataframe. We can see a sample of our resulting DataFrame in below image:

| Index | id | natch_api_i | date | ne_team_ap | ay_team_ap | ne_team_g | ay_team_g | al_differen | ome_statu | all_rating_h |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4770 | 483130 | 2008-08-09 00:00:00 | 9827 | 7819 | 2 | 1 | 1 | HW | 759 |
| 1 | 4771 | 483131 | 2008-08-09 00:00:00 | 9746 | 9831 | 1 | 0 | 1 | HW | 763 |
| 2 | 4773 | 483133 | 2008-08-10 00:00:00 | 9748 | 9941 | 3 | 0 | 3 | HW | 770 |
| 3 | 4774 | 483134 | 2008-08-09 00:00:00 | 9829 | 9847 | 1 | 0 | 1 | HW | 751 |
| 4 | 4775 | 483135 | 2008-08-09 00:00:00 | 8481 | 8639 | 0 | 0 | 0 | D | 775 |
| 5 | 4777 | 483137 | 2008-08-09 00:00:00 | 9874 | 9855 | 1 | 2 | -1 | HL | 818 |
| 6 | 4778 | 483138 | 2008-08-09 00:00:00 | 9873 | 9853 | 1 | 0 | 1 | HW | 743 |

## 3. Exploratory Data Analysis

3.1. *Overall Rating HOME/AWAY*

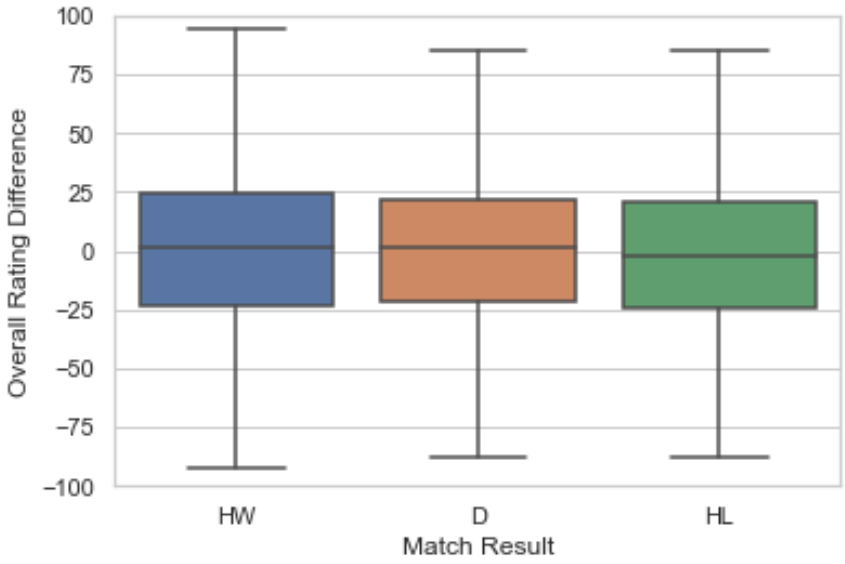
*Insight #1: The win rate of home team is higher if home team's overall rating is higher than away team's overall rating, however, that are not much different. If they are the same, the win rate of home team is lower.*

| mpare_overall_rat | D | HL | HW | win_rate |
|---|---|---|---|---|
| home_is_equal | 20 | 20 | 28 | 41.18% |
| home_is_higher | 762 | 752 | 1388 | 47.83% |
| home_is_lower | 709 | 815 | 1334 | 46.68% |

**Graph 4: Overall Rating vs Home's Win Rate**

3.2. *Overall Rating Difference*

*Insight #2: There is a difference in distributions of overall rating difference when the home team lose and the home team didn't win, yet that are not much different.*



**Graph 5: Overall Rating vs Match Outcome**

Most of the matches in which the home team didn't lose, the home team tended to have a higher overall rating than the away team. Conversely, when the home team lost, the away team tended to have a higher overall rating.

3.3. *Min Overall Rating HOME/AWAY*

*Insight #3: The win rate of home team is higher if home team's min overall rating is higher than away team's min overall rating, but not much different. If they are the same, the win rate of home team is highest.*

The below table compares min rating of each player in the home teams with away teams's minimum rating of each player. We can see that a little bit of different between the win rate of home team if home team's min overall rating is higher than away team's min overall rating. Furthermore, When the result is draw, the home team's win rate is higher than the other cases.

| ɔare_min_overall_r | D | HL | HW | win_rate |
|---|---|---|---|---|
| home_is_equal | 87 | 89 | 186 | 51.38% |
| home_is_higher | 700 | 728 | 1298 | 47.62% |
| home_is_lower | 704 | 770 | 1266 | 46.20% |

**Graph 6: Overall Rating vs Home's Win Rate**

3.4. *Max Overall Rating HOME/AWAY*

*Insight #4: There is a slight difference in how the win rate of home team is when home team's max overall rating is higher than away team's max overall rating. The win rate of home team is lowest when the difference of max overall rating between the two teams is 0.*

| ɔare_max_overall_r | D | HL | HW | win_rate |
|---|---|---|---|---|
| home_is_equal | 121 | 132 | 203 | 44.52% |
| home_is_higher | 703 | 692 | 1303 | 48.30% |
| home_is_lower | 667 | 763 | 1244 | 46.52% |

**Graph 7: Max Overall Rating vs Home's Win Rate**

The above table compares min rating of each player in the home teams with away teams's minimum rating of each player. Looking at win rate, it is clear that *the win rate of home team if home team's min overall rating is higher than away team's min overall rating* is relatively high when compared to other cases.

3.5. *Mean Overall Rating HOME/AWAY*

*Insight #5: The win rate of home team is higher if home team's min overall rating is higher than away team's min overall rating, but not much different. If they are the same, the win rate of home team is highest.*

| are_mean_overall_ | D | HL | HW | win_rate |
|---|---|---|---|---|
| home_is_equal | 20 | 20 | 28 | 41.18% |
| home_is_higher | 762 | 752 | 1388 | 47.83% |
| home_is_lower | 709 | 815 | 1334 | 46.68% |

**Graph 8: Mean Overall Rating vs Home's Win Rate**

The above table compares min rating of each player in the home teams with away teams's minimum rating of each player. Looking at win rate, it is clear that *the win rate of home team if home team's min overall rating is higher than away team's min overall rating* is relatively high when compared to other cases.

3.6. *Standard Deviation Overall Rating HOME/AWAY*

*Insight #6: The win rate of home team is higher if home team's std overall rating is higher than away team's std overall rating, but not much different. If they are the same, the win rate of home team is highest.*

| pare_std_overall_r | D | HL | HW | win_rate |
|---|---|---|---|---|
| home_is_equal | 1 | 0 | 0 | 0.00% |
| home_is_higher | 767 | 787 | 1414 | 47.64% |
| home_is_lower | 723 | 800 | 1336 | 46.73% |

**Graph 9: SD Overall Rating vs Home's Win Rate**

The above table compares standard deviation rating of each player in the home teams with away teams's standard deviation rating of each player. Looking at win rate, it is

clear that the win rate of home team if home team's min overall rating is higher than away team's min overall rating is relatively high when compared to other cases. Especially, the case with the no difference of standard deviation rating of each player between two teams is extremely rare.

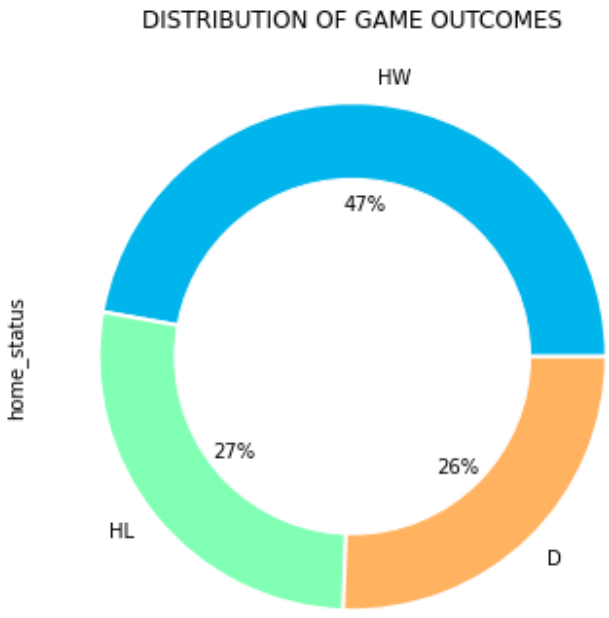
3.7. Further analysis

*Insight #7: Playing at home boosts the chances of a team to win the match.*

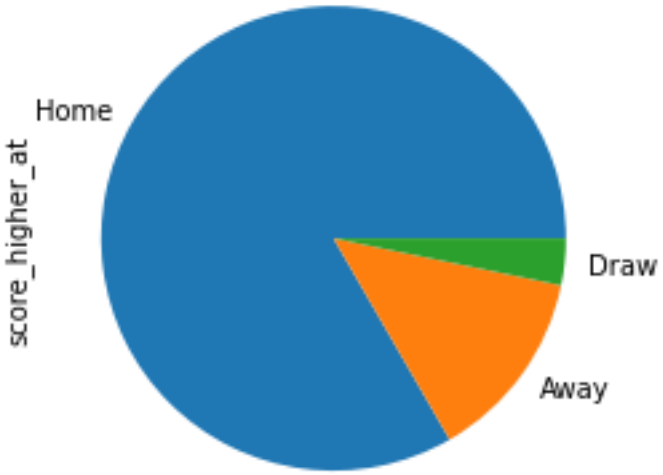
Lets look at big picture:

| Index | home_team_goal | away_team_goal |
|---|---|---|
| count | 5828 | 5828 |
| mean | 1.5415236787… | 1.1350377487989 |
| std | 1.2625856041… | 1.12316248055681 |
| min | 0 | 0 |
| 25% | 1 | 0 |
| 50% | 1 | 1 |
| 75% | 2 | 2 |
| max | 8 | 8 |

It is seen from the mean that home teams generally score higher than away team, in turn win the match. Let's try to look at distribution of game outcomes:



The above composition suggests that Home teams won around 47% of matches, while away team won 29%. Furthermore, a quarter of all matches were a draw.



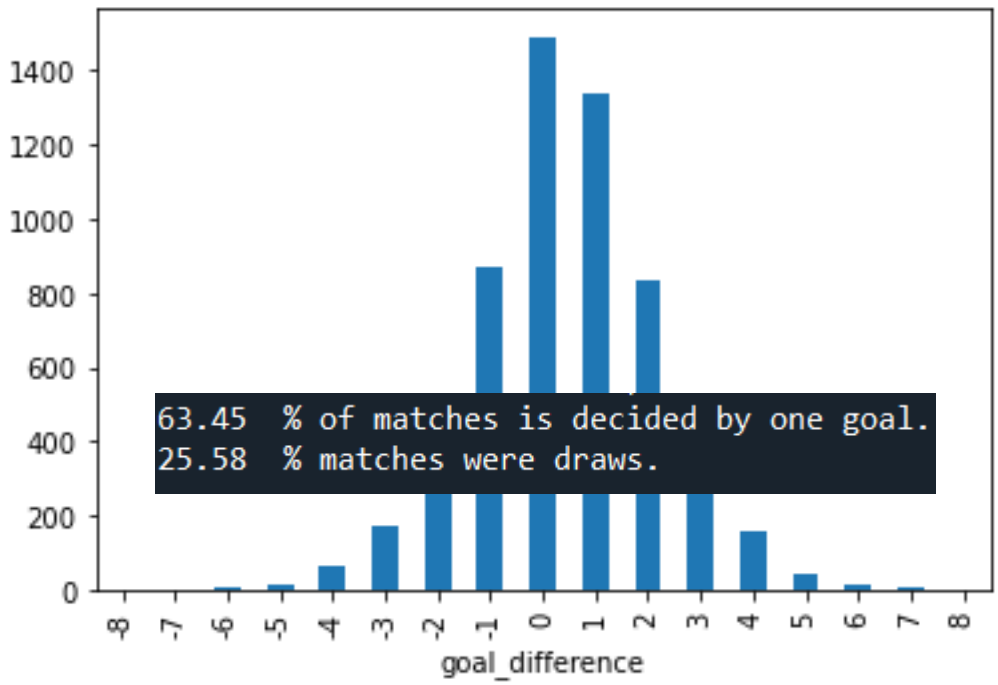

| | |
|---|---|
| Home | 83.33 |
| Away | 13.43 |
| Draw | 3.24 |

It was seen that 83% teams have scored higher at home location in comparison to their away scores. Furthermore, there is a quite strong correlation between a team's home scoring and away scoring

Out[10]:

| | home_team_goal | away_team_goal |
|---|---|---|
| home_team_goal | 1.000000 | 0.563331 |
| away_team_goal | 0.563331 | 1.000000 |

*Insight #8: There're just small differences decide matches.*

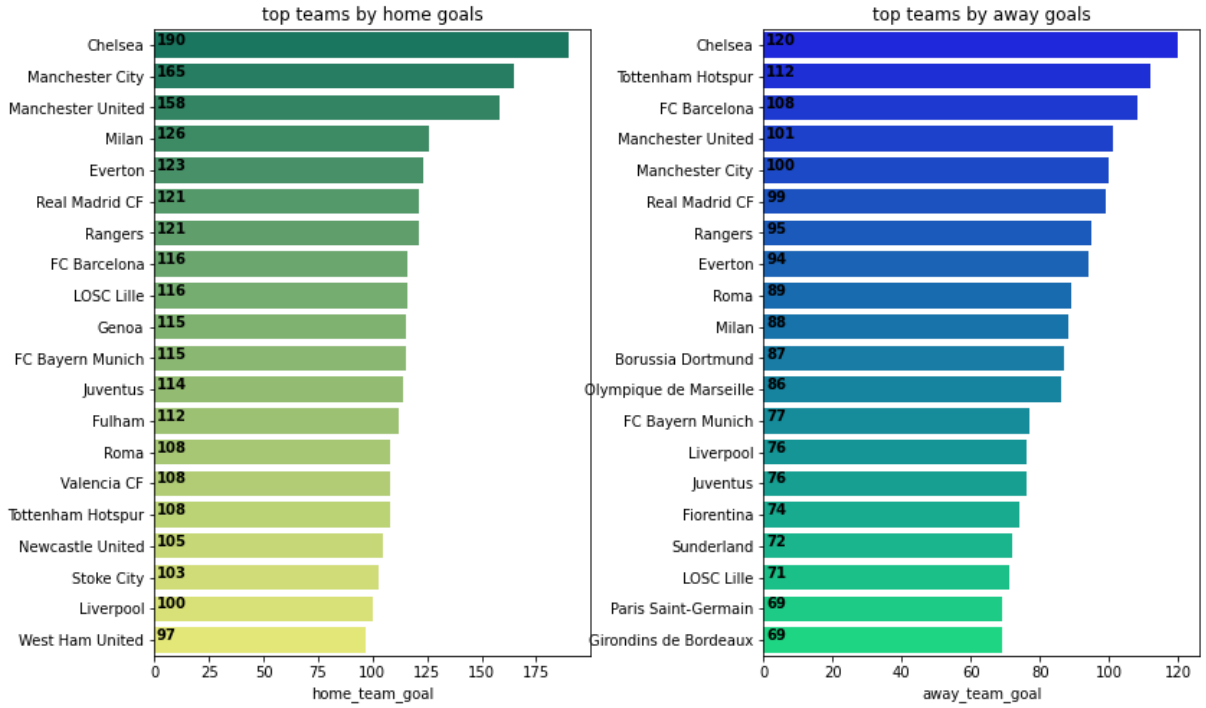



63.45 % of matches is decided by one goal.
25.58 % matches were draws.

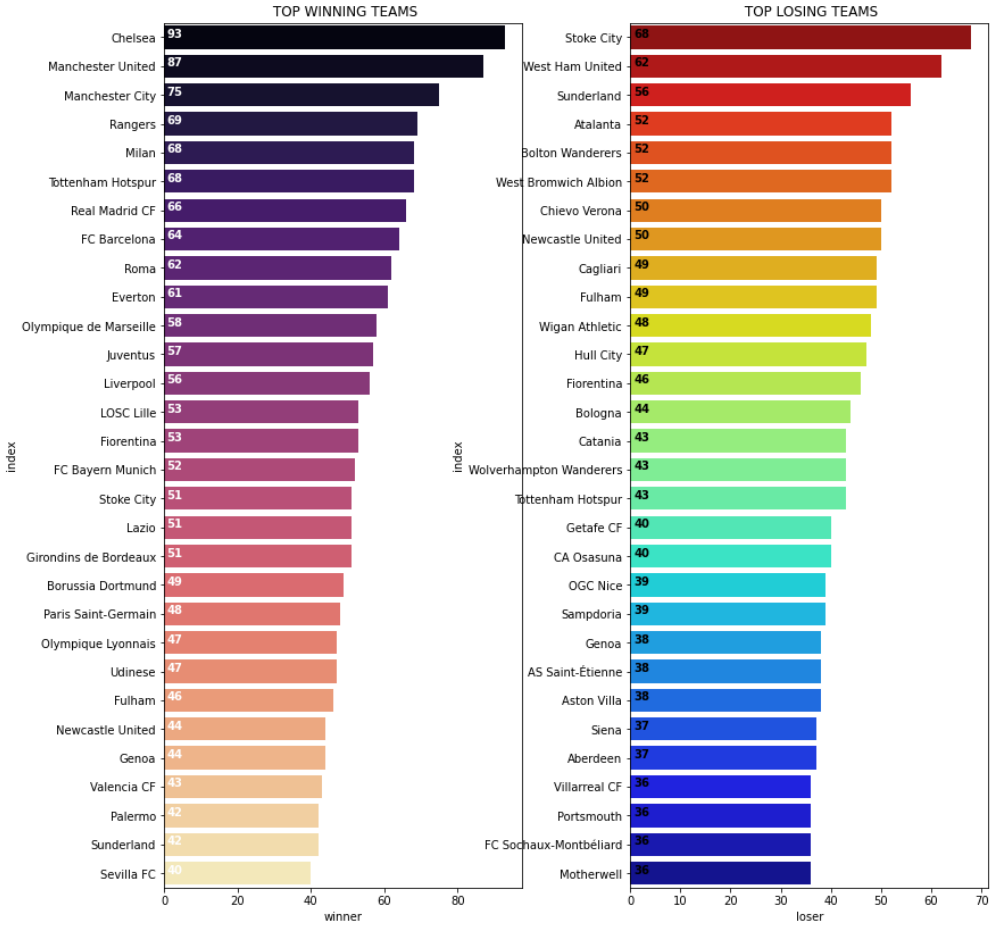
So, ches are decic goal,

which means that most matches are tough to predict, and that small differences decide matches.

*Insight #9: Chelsea is the only club that leads the team score list even when it's home or away team.*




*Insight #10: Chelsea is the club that leads the list of clubs with the most wins. Stoke City is the club that leads the list of clubs with the most defeats.*



## 4. Prediction phase

First, I kept the 10 most useful features to predict my output. It's pretty logical, the best features are linked with goal differences and ratings of teams.

In my model of machine learning, I decided to use Random Forest algorithm:

```
# Removing useless columns to process classification
df_final = matches_df.copy()
columns_to_drop = ["id", "match_api_id","date", "away_team_name","home_team_name",
                   "home_status", "home_team_goal","away_team_goal",
                   "home_team_api_id", "away_team_api_id",
                    "goal_difference"]

y = df_final["home_status"]
df = df_final.drop(columns_to_drop, axis=1)
df = df.fillna(0)

# Split X and y into a train and test set
X_train, X_test, y_train, y_test = train_test_split(df, y, shuffle=True, random_state=42)

# Select features using RFE
clf = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=42, n_jobs=-1)
estimator = clf
selector = RFE(estimator, n_features_to_select = 10, step=1)
selector = selector.fit(X_train, y_train)
test = X_train.iloc[:, selector.support_].tail()
clf.fit(selector.transform(X_train), y_train)
```

The following table would show you 10 final features I used in my model:

test - DataFrame

| Index | all_rating_h | all_rating_a | _rating_diff | erall_rating | erall_ratinc | erall_ratinc | ean_overall_rating_hon | ean_overall_rating_awa | td_overall_rating_hom | std_overall_rating_away |
|---|---|---|---|---|---|---|---|---|---|---|
| 3772 | 741 | 738 | 3 | 59 | 51 | 78 | 67.3636363636364 | 67.0909090909091 | 5.02539007983912 | 10.0145348913921 |
| 5191 | 768 | 769 | -1 | 56 | 65 | 81 | 69.8181818181818 | 69.9090909090909 | 8.15865407794915 | 5.37502642699634 |
| 5226 | 739 | 774 | -35 | 53 | 60 | 77 | 67.1818181818182 | 70.3636363636364 | 8.56525751881614 | 7.01815826656435 |
| 5390 | 737 | 708 | 29 | 62 | 49 | 74 | 67 | 64.3636363636364 | 3.63318042491699 | 6.42297014274124 |
| 860 | 764 | 775 | -11 | 54 | 65 | 80 | 69.4545454545455 | 70.4545454545455 | 6.63872934172853 | 4.13191569041858 |

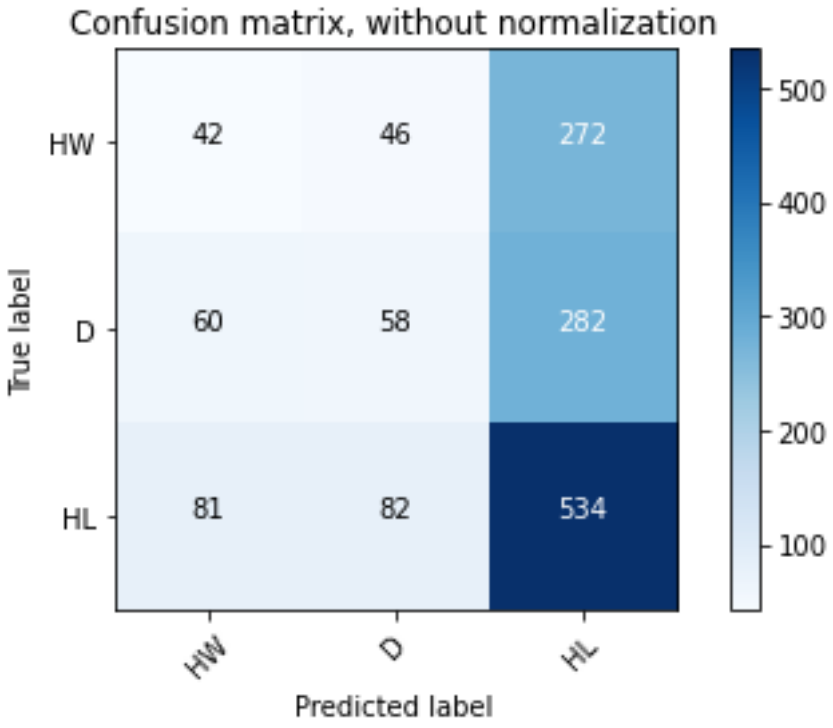
## 5. Final Conclusions

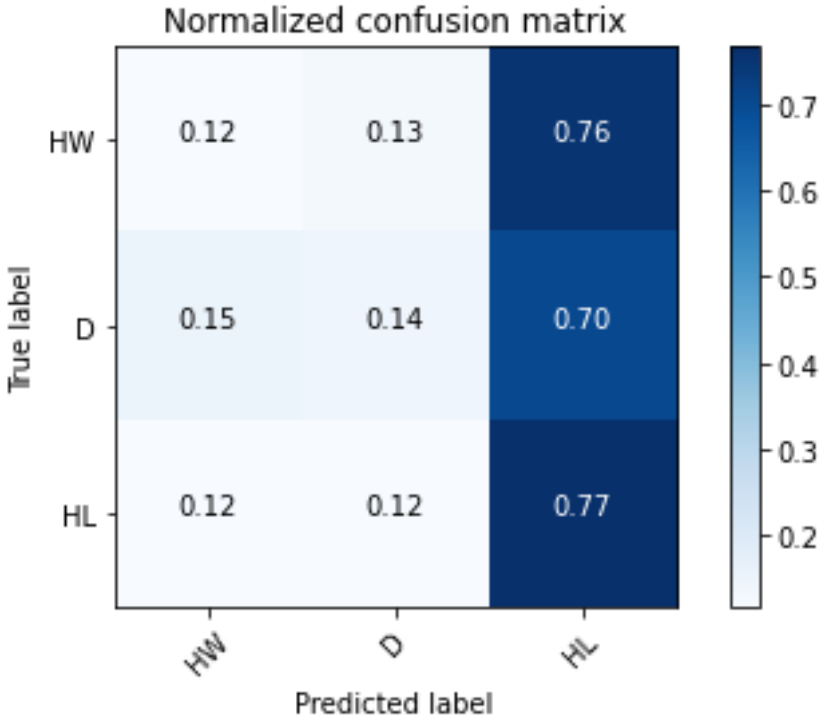
5.1. *43,5% accuracy of predictions on test dataset.*

```
In [28]: score = clf.score(selector.transform(X_test), y_test)
    ...: y_pred = clf.predict(selector.transform(X_test))
    ...: score
Out[28]: 0.4351407000686342
```

5.2. *Plotting results for each categorical output.*

In conclusion, I have relative predict results when victory or draw happen (approx. 68% accuracy) but my model is not able to predict defeats (only 50,8%).


Confusion matrix, without normalization
HW
D
HL
True label
42
46
272
60
58
282
81
82
534
500
400
300
200
100
HW
D
HL
Predicted label


Normalized confusion matrix
HW
D
HL
True label
0.12
0.13
0.76
0.15
0.14
0.70
0.12
0.12
0.77
0.7
0.6
0.5
0.4
0.3
0.2
HW
D
HL
Predicted label