Predicting the Outcome of Soccer Matches

Thomas Forest

Table of Contents

I. Description of the problem

Soccer is one of the most unpredictable sport. In facts, there are so many variables that can be taken into account that you can never be sure of the outcome of the match. Our goal is to take up this challenge and to find the best Machine Learning model to predict the outcome of football matches. In football matches, the home team wins about 46% of the time so our model needs to do better than this.

II. Description of the dataset

Therefore, we have a dataset from Kaggle to help us in this task. This dataset contains a lot of information such as the following:

- +25,000 matches
- +10,000 players
- 11 European Countries with their lead championship
- Seasons 2008 to 2016
- Players and Teams' attributes* sourced from EA Sports' FIFA video game series, including the weekly updates
- Team line up with squad formation (X, Y coordinates)
- Betting odds from up to 10 providers
- Detailed match events (goal types, possession, corner, cross, fouls, cards etc…) for +10,000 matches

However, a large part of the available data won't be used as we will mainly focus on the match table. Furthermore, I have decided to focus on a single football league that will be the England Premier League.

a. Data exploration

So as we said, we will only use the match table to realize our task. The match table contains a lot of information and its dimension is 115. Of course we will not use everything from this table because some of them are not relevant. Nevertheless, here are the attributes that I found that would be useful:

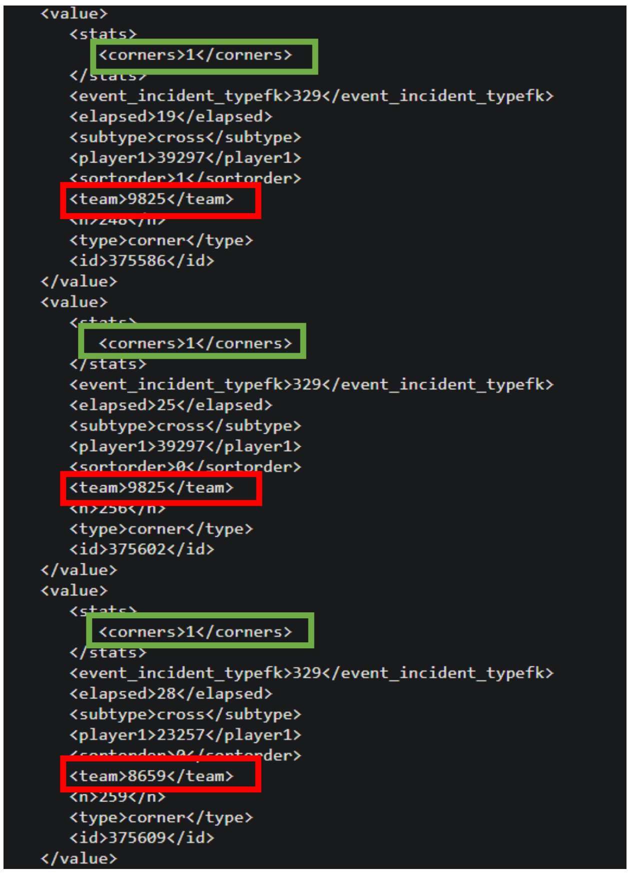
- season
- date
- home_team_api_id
- home_team_name
- away_team_api_id
- away_team_name

- home_team_goal
- away_team_goal
- shoton
- shotoff
- foulcommit
- card
- cross
- corner
- possession
- (betting odds from different bookmakers)

b. Data preprocessing

First of all, I started by averaging all the betting odds into three columns (home win – draw – away win). By doing this, it would allow me to have a global vision of what the bookmakers are thinking about the outcome of the match.

Then, I needed to extract the data from the XML string we have for each statistic in the match (shoton, shotoff, corner, possession…). This was by far one of the most complicated task to do as I didn't know how to extract the data from an XML format in Python. Moreover, I didn't only have to extract a single information but several in different variables. Take the following as an example from a corner xml string:



```
<value>
   <stats>
      <corners>1</corners>
   </stats>
   <event_incident_typefk>329</event_incident_typefk>
   <elapsed>19</elapsed>
   <subtype>cross</subtype>
   <player1>39297</player1>
   <sortorder>1</sortorder>
   <team>9825</team>
   <n>248</n>
   <type>corner</type>
   <id>375586</id>
</value>
<value>
   <stats>
      <corners>1</corners>
   </stats>
   <event_incident_typefk>329</event_incident_typefk>
   <elapsed>25</elapsed>
   <subtype>cross</subtype>
   <player1>39297</player1>
   <sortorder>0</sortorder>
   <team>9825</team>
   <n>256</n>
   <type>corner</type>
   <id>375602</id>
</value>
<value>
   <stats>
      <corners>1</corners>
   </stats>
   <event_incident_typefk>329</event_incident_typefk>
   <elapsed>28</elapsed>
   <subtype>cross</subtype>
   <player1>23257</player1>
   <sortorder>0</sortorder>
   <team>8659</team>
   <n>259</n>
   <type>corner</type>
   <id>375609</id>
</value>
```

Here we can see that for each value we have different information such as the number of corners (which is always 1, 1 value = 1 corner), the time elapsed in the match at the time of the corner etc.. And in red we can see the team who actually got the corner. Therefore, for each cell in the corner columns, I had to check each value in the XML string to check which team actually got the corner and then store the total value for each team. And this for every feature concerned.

After obtaining the concerned value for each team, I subtract them to obtain a single value which will be easier to use in the future.

This step took me a lot of time and it was sometimes painful not understanding why I got NaN or wrong values back from the function as I had never manipulated XML in Python before.

Fortunately, once this step was over, I had done the most of preprocessing the data. I added a result column containing three possible values:

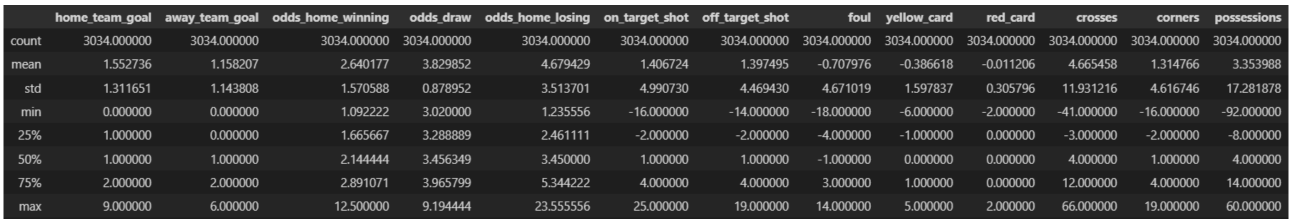
- "Win" if home_goals > away_goals
- "Draw" if home_goals = away_goals
- "Lose" if home_goals < away_goals

This column will be our future target for our model.

Finally, I removed any rows containing any N/A value to avoid any problem when building the model. I decided to do this because there was only 6 missing values (out of 3040) so I was actually not losing a lot of data.

c. Analysing Data

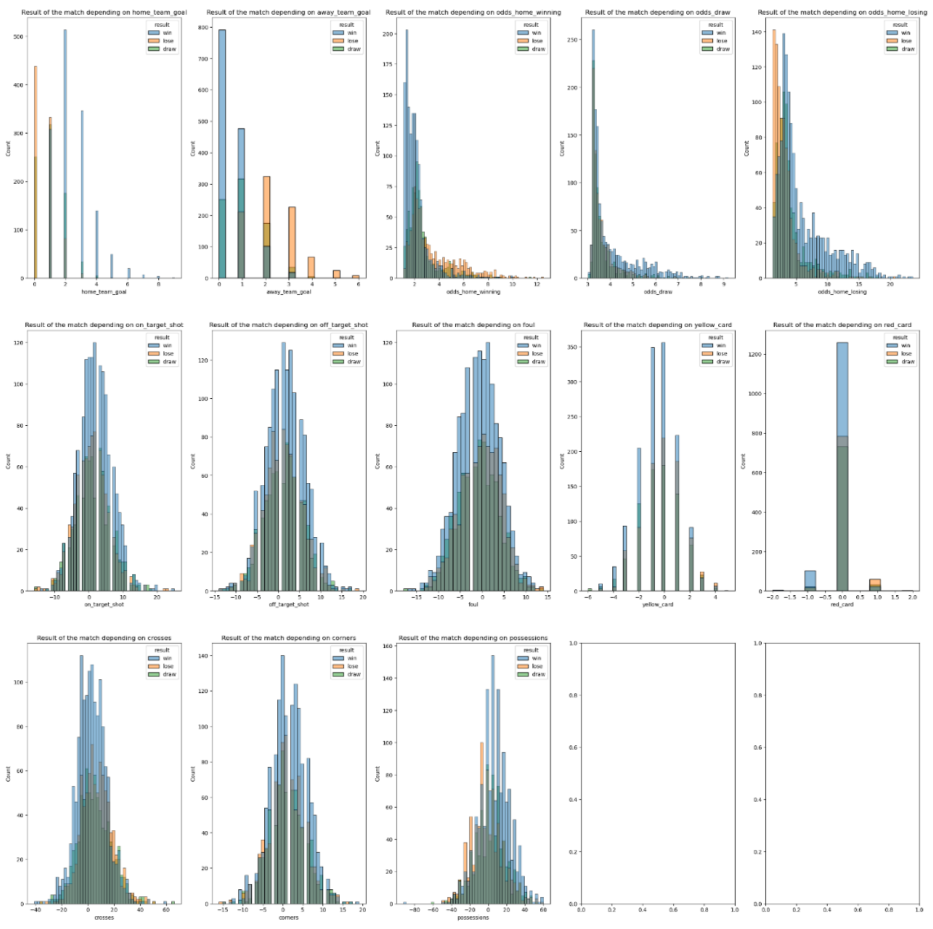
Now that we have a clean dataframe to work with, we can analyse it. Here is the description of the actual data:



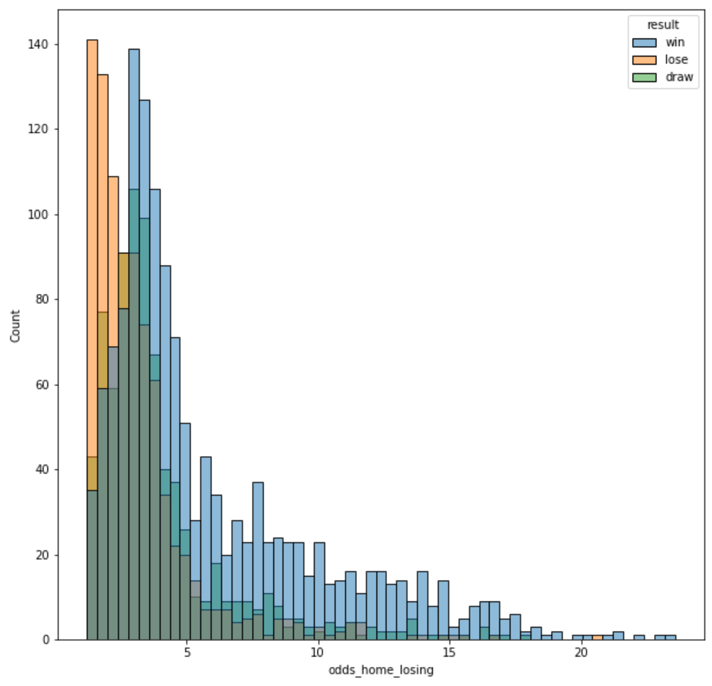
	home_team_goal	away_team_goal	odds_home_winning	odds_draw	odds_home_losing	on_target_shot	off_target_shot	foul	yellow_card	red_card	crosses	corners	possessions
count	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000	3034.000000
mean	1.552736	1.158207	2.640177	3.829852	4.679429	1.406724	1.397495	-0.707976	-0.386618	-0.011206	4.665458	1.314766	3.353988
std	1.311651	1.143808	1.570588	0.878952	3.513701	4.990730	4.469430	4.671019	1.597837	0.305796	11.931216	4.616746	17.281878
min	0.000000	0.000000	1.092222	3.020000	1.235556	-16.000000	-14.000000	-18.000000	-6.000000	-2.000000	-41.000000	-16.000000	-92.000000
25%	1.000000	0.000000	1.665667	3.288889	2.461111	-2.000000	-2.000000	-4.000000	-1.000000	0.000000	-3.000000	-2.000000	-8.000000
50%	1.000000	1.000000	2.144444	3.456349	3.450000	1.000000	1.000000	-1.000000	0.000000	0.000000	4.000000	1.000000	4.000000
75%	2.000000	2.000000	2.891071	3.965799	5.344222	4.000000	4.000000	3.000000	1.000000	0.000000	12.000000	4.000000	14.000000
max	9.000000	6.000000	12.500000	9.194444	23.555556	25.000000	19.000000	14.000000	5.000000	2.000000	66.000000	19.000000	60.000000

Here we can see that every columns as the same number of information (as I removed missing values just before). We can also notice that the median for "away_team_goal" is 1 which means that in a lot of matches, the away team does not score any goals.

After that I decided to plot the result of the matches depending on the others features. Here is the figure that I obtained:



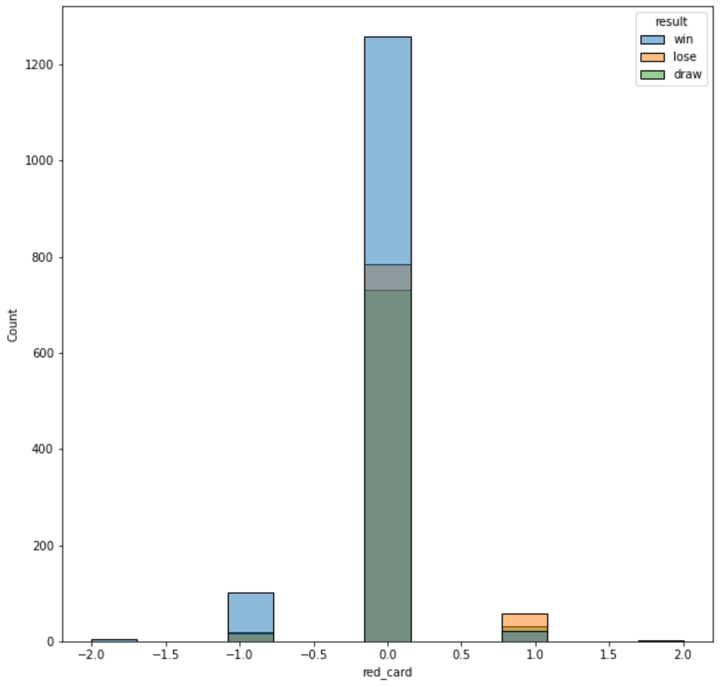
Here we can notice a few things. For example, we can see that betting odds is a really good indicator of the result of the match. If we take a look at the upper right plot which is the following:



Result of the match depending on odds_home_losing

We can see that the bigger is the odd for the home team to lose, the more there are winning.

Here we can also notice that red cards have a huge impact on the result of the match:



Result of the match depending on red_card

III. Finding the best model

To find the best model for this dataset, I decided to use KFold and Cross Validation in order to get an idea on how each model would perform. Here are the models that I tested in the first place:

- Logistic Regression
- Linear Discriminant Analysis
- Decision Tree
- Gaussian Naive Bayes
- Support Vector Machine
- Random Forest

However, I figured out that several of those models were not adapted for this task.

Therefore, I decided to choose Linear Discriminant Analysis and Support Vector Machine as they seemed to be the best models for the data I was using. I tuned both models to obtain the best parameters to work with.

IV. Linear Discriminant Analysis (LDA)

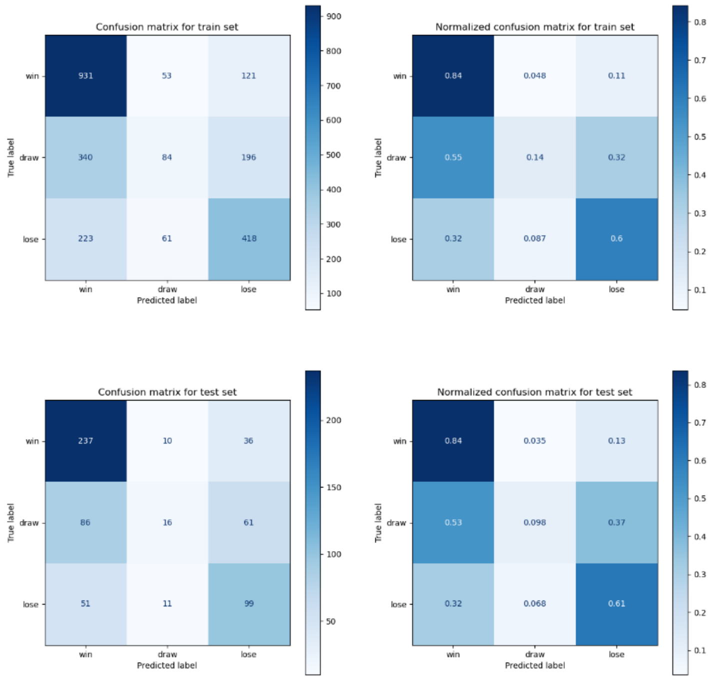
Linear Discriminant Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups (separating two or more classes). It is used to project the features in higher dimension space into a lower dimension space. It is quite easy to use and fast to execute. That’s why I chose to use this model at first.

Using LDA, I made several versions by changing the number of data or the features. Every version was trained with a 80/20% split of the data.

a. Using the whole dataset

For this first iteration, I took the data as I prepared it before without any changes. The result was already quite good with an accuracy score of nearly 58% on the test set but I knew I could do better.

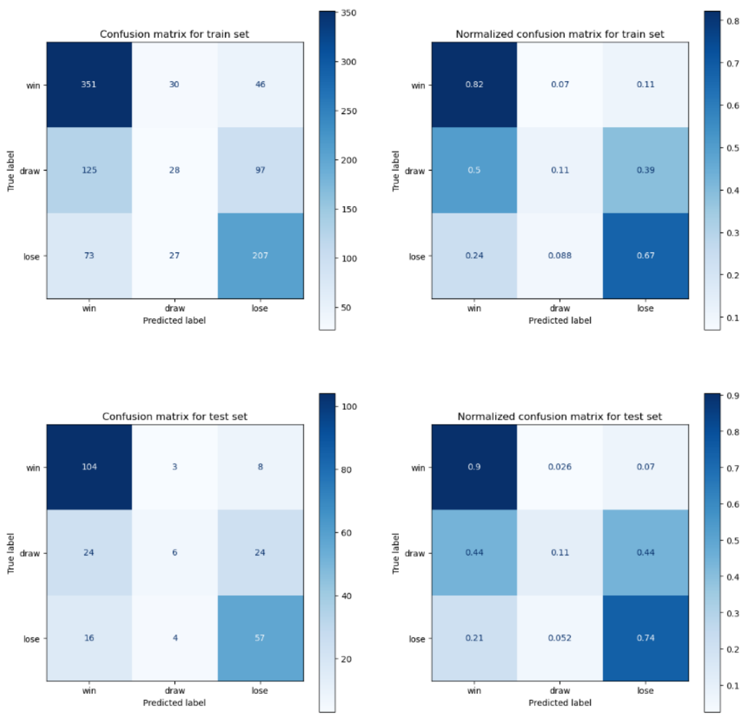
Here are the confusions matrixes for this first iteration:



Confusion matrixes for LDA with the whole dataset

b. Using the best number of matches

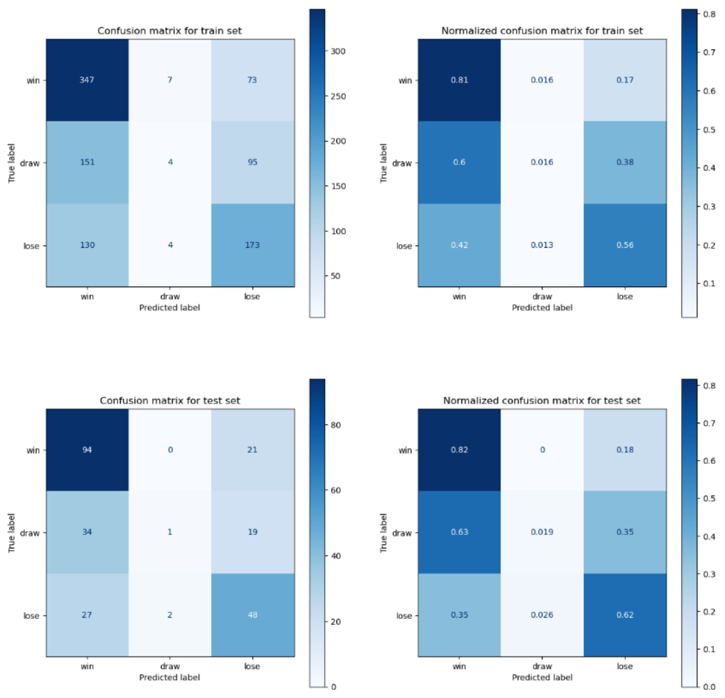
One of the thing than can be improved is the number of matches that we are taking into account to predict the outcome. That’s why I made a function searching for the best number of past matches to take into account. This function was actually really useful and I found that for this model (LDA), the best number of matches was 1230 (out of 3034). Retraining the model with this new split of the data gave me insane result with an accuracy score of 67.89% on the test set. At this moment, I thought that this result was a clearly higher than what I expected. Thus, I started to search where I could have done something wrong.



Confusion matrixes for LDA with the best number of matches

c. Removing the betting odds features

After having such a good result, I decided to remove those features that came from bookmaker. Those numbers were not directly depending on the match and I was afraid that it would have a negative impact on the model. In this third iteration, I trained the model only with statistical data coming from matches (corners, shot on target etc..) and using the best number of matches. This time, the accuracy score was about 58.13% and I was pretty happy with that.



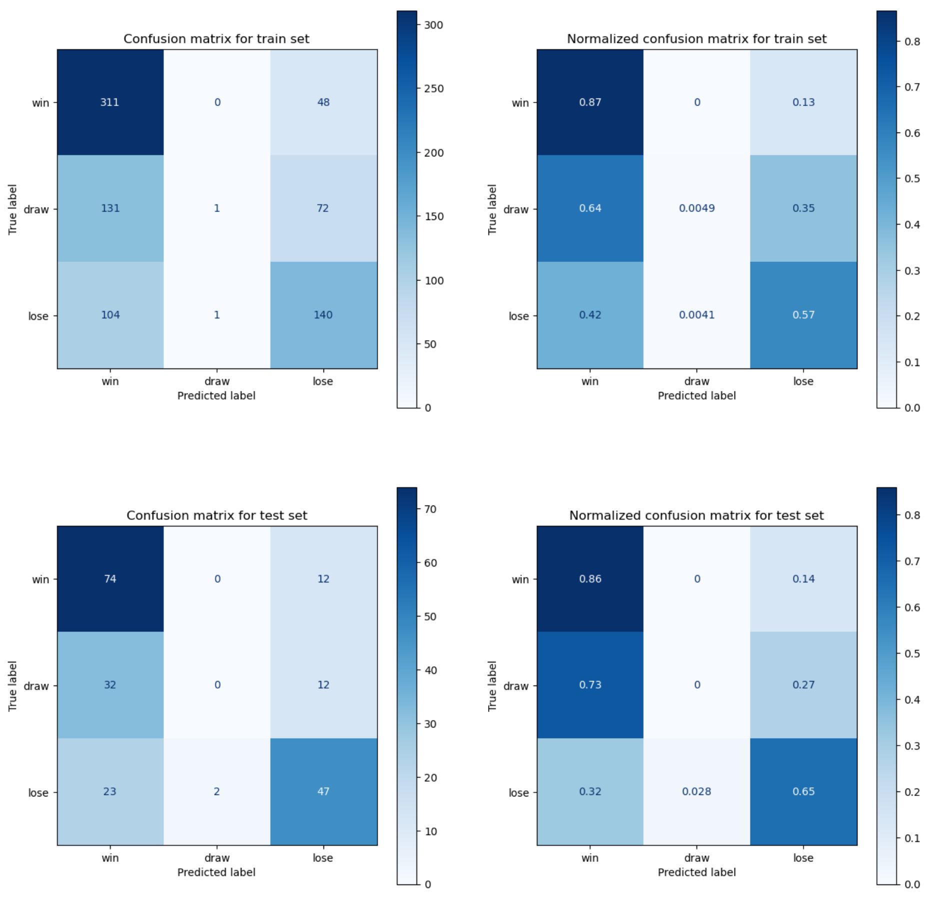
Confusion matrixes for LDA without betting odds

V. Support Vector Machine (SVM)

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

Using this model, I only took the data without betting odds and using the best number of matches which this time was 1010.

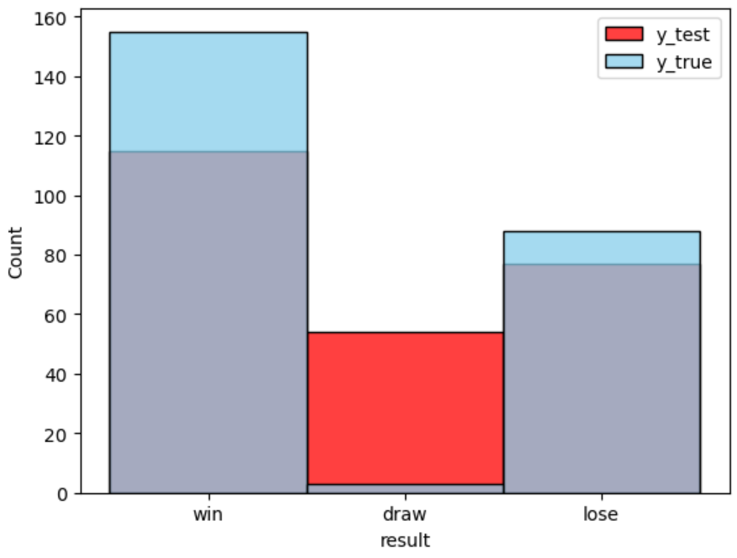
Therefore, using SVM improved a bit the result compared to LDA and I obtained an accuracy score of 59.9% on the test set. Here are the confusions matrixes for this last iteration.



Confusion matrixes for SVM without betting odds

VI. Conclusion

For every test I made, the thing that always remained quite the same is this histogram showing what the model actually predicted.



We can see that, every time, the model has trouble to predict draw matches. It is understandable because it is really hard to predict the winner of a football match but it is even harder to predict that there will be no winner. Even for us humans, if we were asked “who will win this match ?”, we would tend to answer either a team or the other one but rarely “neither of them”.

This project allowed me to discover new aspects of Data Science like managing bigger datasets. Furthermore, I understood how complicated it can be to get and clean the data. I also improved my comprehensions of Machine Learning models by using LDA and SVM.

I feel really lucky to have been able to take this course because I’m only 19 and I did not start my master yet so it's really a benefit for me.