MIDA Project

Toni Cifre Vicens, Daniel Beltrán Drago, Nastasia

December 21, 2020

Abstract

In this project, we will try to find similarities between the multiples positions from all *LaLiga Santander*[1], *LaLiga Smartbank*[2] and *LaLiga Iberdrola*[3], trying to classify all the players to these 3 main categories: defences, midfielders and attackers.

[1] *LaLiga Santander* is the men first tier in Spanish football.
[2] *LaLiga Smartbank* is the men second tier in Spanish football.
[3] *LaLiga Iberdrola* is the women first division in Spanish football.

Contents

Chapter 1

Project explanation

1.1 Data description

All of the data was obtained from the official *LaLiga* webpage[1], where you can find all the stats and advanced stats used in this project.

For obtaining this stats we used the command `wget`[2], which is in fact a request to the *API* provided from *LaLiga*, whose return was a *JSON* file with all the stats demanded.

1.2 Pre-processing data description

Since we work with *.CSV* files, we needed to convert the *.JSON* file obtained from *LaLiga API*.

All the stats with 0 value were missing (for weight purposes), and we had to fill all those with the 0 value. Also, in there were a lot of useless information, for example, the position value was a dictionary were it stored the: id, the feminine name, the masculine name, short (both genders) name,... Since we only wanted the a name for the value, we had to delete and change the dict for the corresponding name.

We're going to list all the pre-changes we made to the *.JSON* file for converting to a *.CSV*:

- All the coaches we're removed from the dataset
  - They not are a football player

[1] LaLiga Advanced statistics - https://www.laliga.com/estadisticas-avanzadas

[2] We used a program utilising this command because we needed to first of all get a key from the API and after you had to request all the information

- Anyone with less than 100 minutes of played time
  - The data sample is very low to distinguish any position

Also we've normalized our values for better performance and results, with the function `Normalizer().fit_Transform()` from the package `sklearn.preprocessing`

1.3 Evaluation criteria

For the evaluation we've selected the *Accuracy* parameter, the use also of the *Confusion Matrix* (both for *KNN* and *Naïve-Bayes* method) and the *Classification Report* (for the *Decision Trees*, *SVM*[3], *Boosting*, *Random Forest*, *Bagging* and also the first ones).

This types of evaluation will make it more simple and easy the tasks of picking and deciding which of the methods mentioned above are valid for our purposes and objectives.

1.4 Execution of different machine learning methods

1.4.1 Naïve Bayes

Our set of players available are about 1073 players, at first we could say that we have a set large enough to obtain adequate reliability. But once the model is generated and we look at the results we can conclude that with our classification problem, to achieve a better accuracy a larger set would be needed because the data are very grouped and the noise between them is very high

1.4.2 KNN

Since we have done a resize of our data, we don't need any type of analysis to found the best K possible, the maximum K we've got is 3.

In this first figure we can see how from k = 20 it stabilizes and we no longer achieve a result greater than 0.72, but when doing a resize of the data, we can see how the value reaches 0.8 with k = 3 such and as we see in the second graph.

Using the 'GridSearchCV' function we can get the best parameters for KNN, both the number of neighborhoods and the weights. From the results you return to us for these parameters we redefine the model but using the indicated parameters obtaining a better result.

[3]SVN: Support Vector Machines

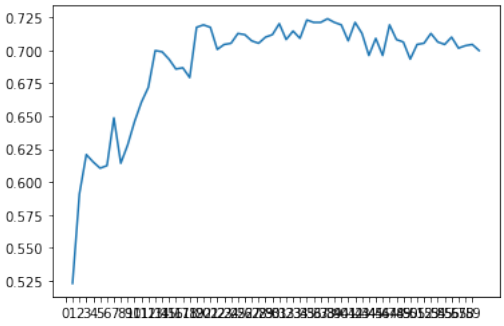


Figure 1.1: Best K graph

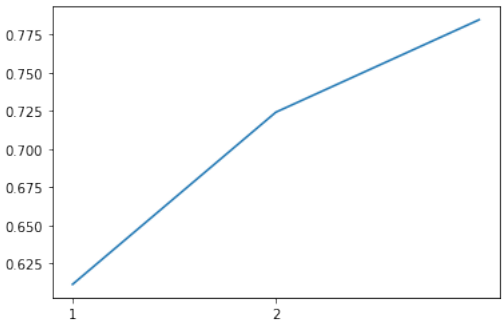


Figure 1.2: Best K normalized graph

In our case, we haven't deleted irrelevant columns, just because we've remodeled our data set for a better fitting and prediction for our model, since the results were much better.

1.4.3 Decision Trees

els parametres escollits per el criteri de la classificació es l'entropia i el nombre minim d'elements per a escollir el cami son 2

un cop executat el conjunt obtenim una presició del 0.788, mentres que la matriu de confusió es bastant ajustada al que nosaltres desitjem.

Al utilitzar una reducció en les dimensions, no podem observar quines son les columnes més rellevant a l'hora de prendre les desicións degut a que les columnes s'han unificat. En aquest cas podem observar com depenen de l'index de cada columna selecciona u conjunt o un altre. En la primera desició podem observar com la majoria de centrocampistes en relació als delanters i defenses son dividits en dos camins diferents. Es a dir que la podriem seleccionar com una norma molt rellevant

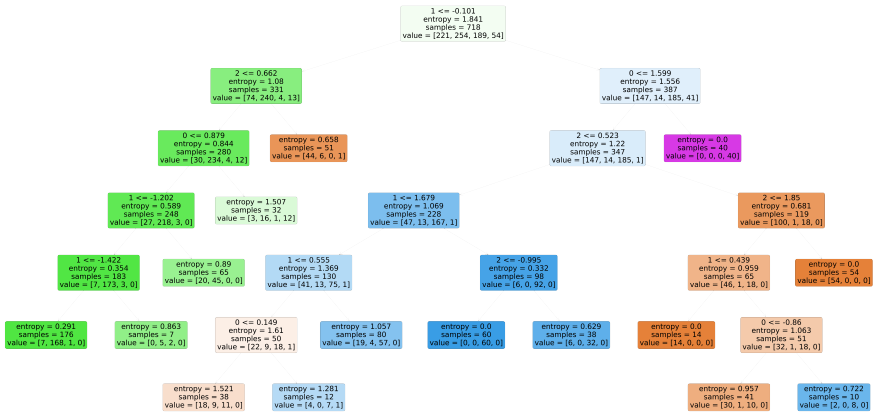


Figure 1.3: Best K normalized graph

1.4.4 Support Vector Machines

First of all we plotted our data into a 2d plane to make a preview and think how the main blocks will end.

We can observe in this plot a main 2 blobs of points, one for the field players and another for the goalkeepers (the yellow dots). Since almost all of the yellow dots are separated by a line (parallel to the main figure at the left), we can predict this method will be very usefully to distinguish between those.

Since this problem isn't about differentiating goalkeepers and field players, we can estimate that this won't be a successful method of solving our problem.

From our three main types of kernel (linear, polynomial and radial) we made a first approximation of the performance of each one with this results:

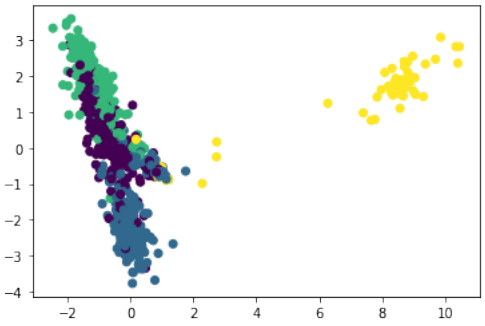


Figure 1.4: SVM preview

Linear Vector Machine

Group	Precision	Recall	f1	support
Midfielders	0.67	0.71	0.69	123
Defenders	0.83	0.85	0.84	130
Strikers	0.72	0.67	0.69	81
Goalkeepers	1.00	0.81	0.89	21

Confusion Matrix

	Midfielders	Defenders	Strikers	Goalkeepers
Midfielders	86	18	19	0
Defenders	17	110	3	0
Strikers	28	1	52	0
Goalkeepers	0	4	0	17

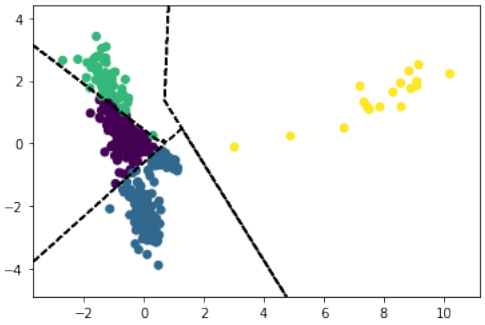


Figure 1.5: SVM Linear

Polynomial Vector Machine

Group	Precision	Recall	f1	support
Midfielders	0.57	0.85	0.68	123
Defenders	0.92	0.70	0.79	130
Strikers	0.80	0.58	0.67	81
Goalkeepers	1.00	0.76	0.86	21

Confusion Matrix

	Midfielders	Defenders	Strikers	Goalkeepers
Midfielders	104	8	11	0
Defenders	38	91	1	0
Strikers	34	0	47	0
Goalkeepers	5	0	0	16

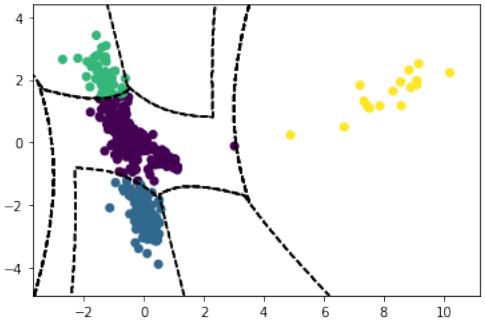


Figure 1.6: SVM Linear

Radial Vector Machine

Group	Precision	Recall	f1	support
Midfielders	0.66	0.71	0.68	123
Defenders	0.83	0.84	0.83	130
Strikers	0.73	0.67	0.70	81
Goalkeepers	1.00	0.81	0.89	21

Confusion Matrix

	Midfielders	Defenders	Strikers	Goalkeepers
Midfielders	87	18	18	0
Defenders	19	109	2	0
Strikers	26	1	54	0
Goalkeepers	0	4	0	17

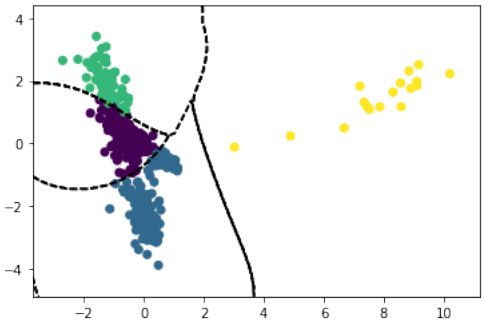


Figure 1.7: SVM Linear

After reviewing the results, we've selected to implement the radial one.

We're going to calculate the C coefficients and the gamma ones for improving our results and accuracy. First of all we started with exponential values for gamma and exponential values for C. After that the plotted the best results we found implementing what we had learned in class:

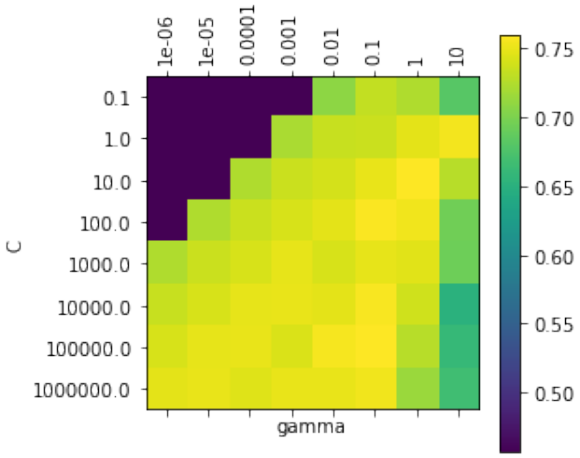


Figure 1.8: SVM Gamma

The best combination proportionate by the SVM kernel is `C = 10.0` and `gamma = 1` .

We've applied that to the SVM Radial, and this are the results:

Group	Precision	Recall	f1	support
Midfielders	0.66	0.71	0.68	123
Defenders	0.83	0.84	0.83	130
Strikers	0.73	0.67	0.70	81
Goalkeepers	1.00	0.81	0.89	21

Confusion Matrix

	Midfielders	Defenders	Strikers	Goalkeepers
Midfielders	87	18	18	0
Defenders	19	109	2	0
Strikers	26	1	54	0
Goalkeepers	0	4	0	17

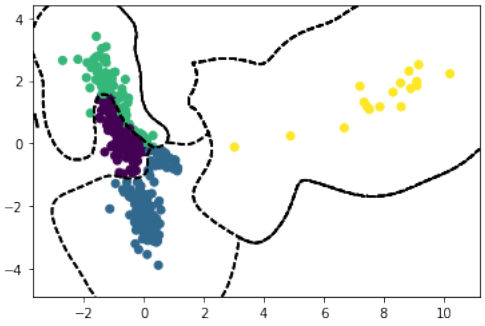


Figure 1.9: SVM Radial with gamma and C calculated

As we can see, we've obtained worse values than the Radial Kernel without the C and gamma calculated. But we've got a much better divisions and line making. If you look up to the *SVM Preview* figure, you could see a similar line between the green and purple dots, and also a similar curved line between the purple and blue dots. Lastly, a line (almost parallel to the main blob of pitch players) is spotted.

Taking a closer look to the confusion matrix, we could see an almost innexistential

1.4.5 Meta-learning algorithms

In this last section we have used a fairly broad set of estimators. among them are the following: ExtraTreesClassifier, AdaBoostClassifier, RandomForestClassifier, LogisticRegression i BaggingClassifier.

In this set of models, we have used a 200 estimators to be able to increase its accuracy while in the LogisticRegression we have set the number of iterators in 10000 to maximize their efficiency.

Once we have had all this set tested and observed that its accuracy was adequate we can create a voting model from all the classifiers created previously,

in this case we used the 'VotingClassifier' using only the classifiers with an accuracy higher than 0.75.

1.5 Comparison

Once all the tests and possible improvements have been made, we have obtained the following table with the precision of each of the classifiers.

Model	Accuracy	F1	Recall	Precision
K Neighbors	0.8422	0.8260	0.8317	0.8281
Decision Tree	0.7718	0.8098	0.8138	0.8084
Gaussian NB	0.8084	07787.	0.7887	0.7830
Support Vector Machines	0.8309	0.8176	0.8228	0.8197
Extra Trees	0.7971	0.7906	0.7930	0.7915
Random Forest	0.8056	0.7990	0.8014	0.8
Logistic Regression	0.8084	0.8206	0.8252	0.8225
Voting Classifier	0.8281	0.8235	0.8288	0.8253

A continuació podem observar un conjunt de gràfiques en les quals veim com cada clasificador crea una frontera entre cada conjunt ajustada a les seves caracteristiques. Per exemple, com es pot observar ente el k neighbors i el Random forest existeix una gran diferencia. En k neighbors te les fronteres ben definides mentres que el forest es pot dir que no te fronteres entre els conjunts acceptant i controlant millor l'ambibalencia de les dades. Per altra banda, ente el decision tree i el support vector machines, es pot veure com la frontera en el cas del suport vector es molt més lineal en relació al decisoin tree degut al kernel que hem utilitzat per el SVM es el radial.

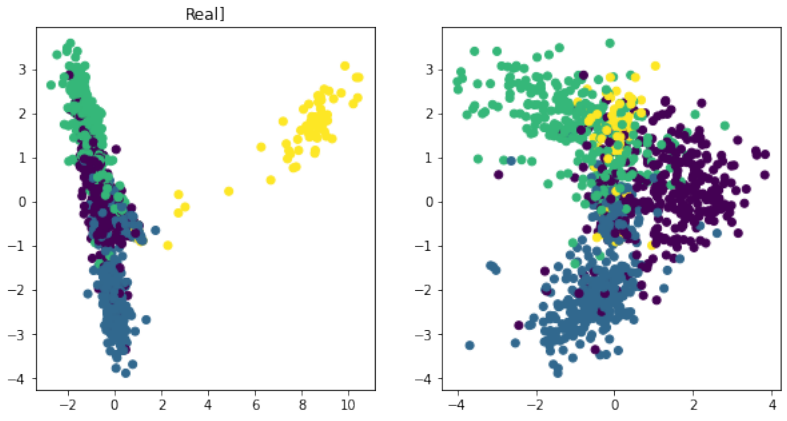
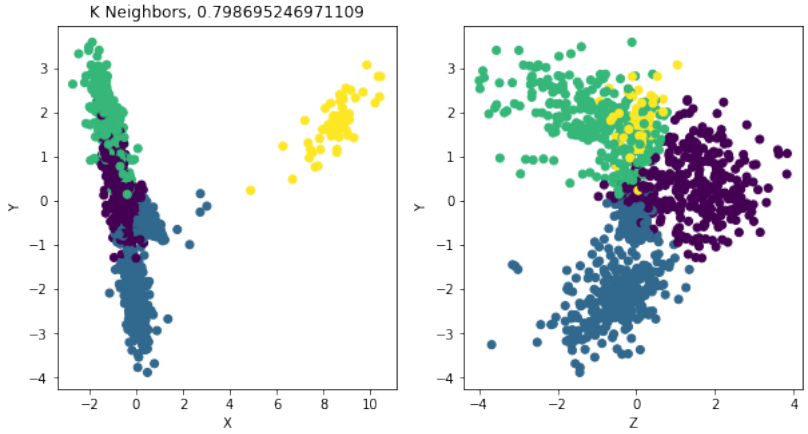
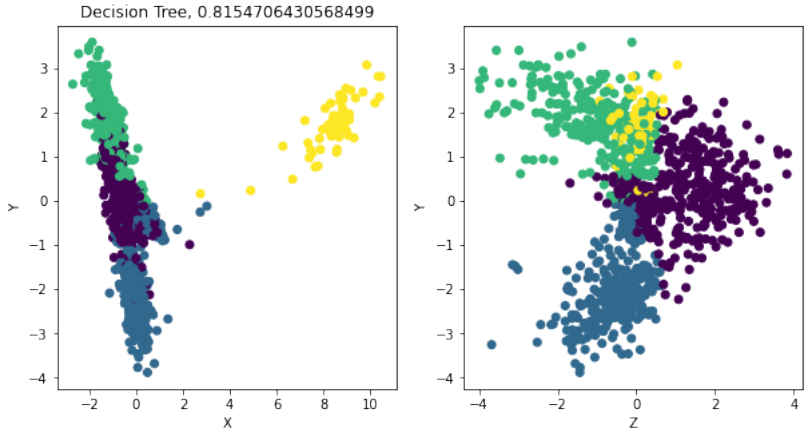
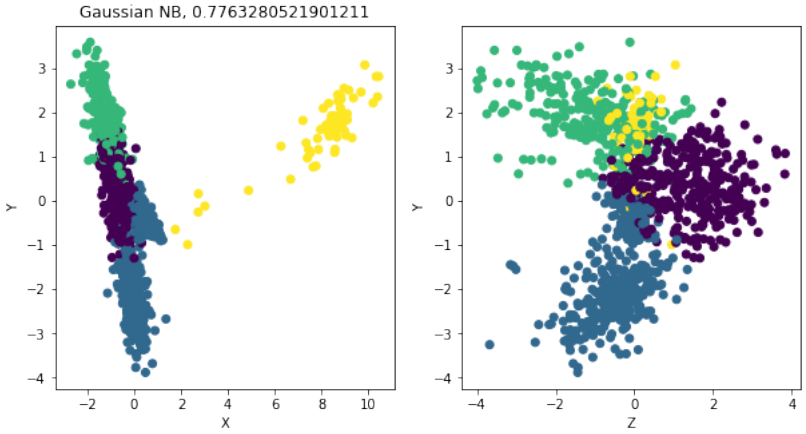
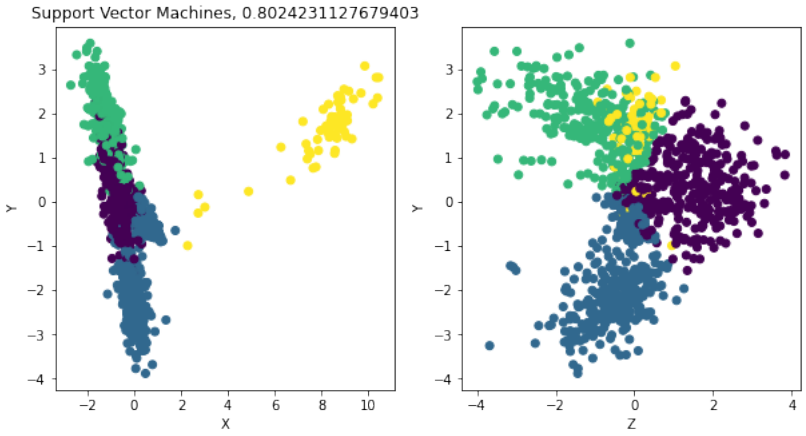


Figure 1.10: f1


K Neighbors, 0.798695246971109
X
Y
Z
Y


Decision Tree, 0.8154706430568499
X
Y
Z
Y


Gaussian NB, 0.7763280521901211
X
Y
Z
Y


Support Vector Machines, 0.8024231127679403
X
Y
Z
Y

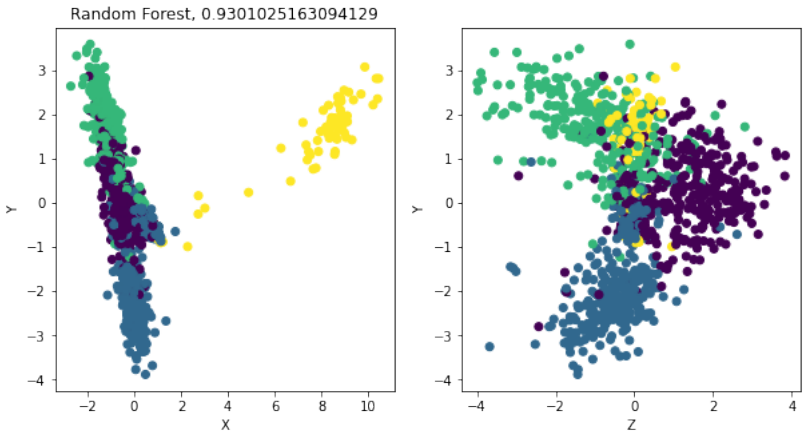


Figure 1.11: f6

1.6 Conclusions

As we can see, this problem wasn't suited for any of the data mining models we've used and implemented. In our opinion, the best and most interesting would have been clustering, because football is such a dynamic sport and a lot of players doesn't fulfill just one category.

The main example of this type of dynamic football is the tactical approach of possession football, where a lot of players act like midfielders gathering a lot of passes and a high pass completions, like the goalkeepers that act like modern defenders, being in a higher up position of the pitch and sometimes, acting like a 5Th defender on a 4 back-line.

But the results obtained with the implemented methods where not bad at all. Almost any of the methods have accuracy and precision of $78\% \sim 83\%$, a respectable percentage for being implemented with this such variety of footballers from the leagues mentioned above.