Machine Learning Engineer Nanodegree

Capstone Project

Artur Pereira

May 12th, 2017

Keywords: Machine Learning, CartolaFC, Linear Regression, SVR, XGBoost, Random Forest, Multi-Layer Perceptron, Neural Network, K-Fold Cross Validation, GridSearchCV, RandomizedSearchCV

I. Definition

Project Overview

CartolaFC[1] is the name of the biggest fantasy football league in Brazil, with millions of participants around the country, having shown continuous growth each year. The participants choose 11 players plus a coach for each round of the Brazilian National Football League, the Brasileirão, given a limited amount of virtual cash, and score based on each player's performance. The league has been growing in popularity every year, for round 10 of the 2017 season, there were a total of 5,540,835 teams picked[2]. The usage of Machine Learning in the field of Fantasy Sports Leagues has grown in popularity over the last few years, for example, Kaggle has a yearly competition for the NCAA March Madness[3] [4], which includes cash prizes and several participating teams. However, there have been few attempts at using Machine Learning for the CartolaFC League, namely, research showed only two instances, André Sakata attempted to use a Linear Regression model[5], while Arnaldo Gualberto used a Neural Network[6]. However,

[1] GloboEsporte. CartolaFC. http://globoesporte.globo.com/cartola-fc/
[2] GloboEsporte. CartolaFC. Virou rotina! Cartola bate novo recorde de times na rodada #10: 5,5 milhões. June 24, 2017. http://globoesporte.globo.com/cartola-fc/noticia/2017/06/virou-rotina-cartola-bate-novo-recorde-de-times-na-rodada-10-55-milhoes.html
[3] Kaggle. Google Cloud & NCAA® ML Competition 2018-Men's. Apply Machine Learning to NCAA® March Madness®. https://www.kaggle.com/c/mens-machine-learning-competition-2018
[4] Kaggle. Google Cloud & NCAA® ML Competition 2018-Women's. Apply Machine Learning to NCAA® March Madness®. https://www.kaggle.com/c/womens-machine-learning-competition-2018
[5] Sakata, André. Aplicando machine learning no CartolaFC. September 30, 2017. https://medium.com/@andresakata/aplicando-machine-learning-no-cartolafc-4ebb5aa0a531

it seems both attempts were incomplete and did not seem to produce substantial results, nonetheless, both showed potential and that there was a lot of room for improvement. Lutz(2015)[7] makes an attempt at the Fantasy League for the NFL, using both SVR and Neural Networks, although for a different sport, and only for the Quarterback position, the approach and results show that in a Fantasy League setting, Machine Learning techniques show promising results, especially for Neural Networks. Matthews, Ramchurn, Chalkiadakis (2012)[8] used a Bayesian Reinforcement Learning Algorithm to compete on the Premier League's Fantasy Competition, the Fantasy Premier League. Although the approach for this project is different, their success in ranking among top players of the Fantasy League shows the potential of Machine Learning techniques in these scenarios.

Given the popularity of the League, the few attempts of using Machine Learning for it, and the demonstrated relative success of its usage in other Fantasy League settings, this project aimed at attempting different machine learning techniques on the available data in order to obtain competitive 2017 season scoring averages and match the results against other league players. In addition to finding efficient algorithms, this project compares its scoring performance to the best player for the 2017 season, who scored an average of 67.4 points per round. The data source used for this project was obtained from the CartolaFCDados Github repository[9], which includes data for the Fantasy League from 2014 to 2017.

Problem Statement

Given the goal of this project is to get the maximum amount of points possible for each round, the model will attempt to predict the scores for each participant player in a round and make a selection of the top predicted scoring players for each position for the 2017 season, which also implies that this is a Regression Problem. The player scores for each round of the CartolaFC are a result of a sum of the points achieved for specific criteria, namely statistics such as goals scored, yellow and red cards received, missed passes, and scoring attempts. Each variable is associated with a specific amount of points awarded, positive or negative. Thus, the best fitting model will be the one that most accurately predicts the scores for every participating player in the round, as

[6] Gualberto, Arnaldo. Github. Análise dos Dados. Last Update: December 10, 2017. https://github.com/henriquepgomide/caRtola/blob/master/src/python/An%C3%A1lise%20dos%20Dados.ipynb
[7] Lutz, Roman. Fantasy Football Prediction. University of Massachusetts Amherst. Amherst, United States. May 26, 2015. https://pdfs.semanticscholar.org/2e04/f58fa272a6eea79d23113f6a4743f8f53ac0.pdf
[8] Matthews, Tim; Ramchurn, Sarvapali; Chalkiadakis, Georgios. Association for the Advancement of Artificial Intelligence (www.aaai.org). Competing with Humans at Fantasy Football: Team Formation in Large Partially-Observable Domains. University of Southampton; Technical University of Crete. Southampton, United Kingdom; Crete, Greece. 2012. http://www.intelligence.tuc.gr/~gehalk/Papers/fantasyFootball2012cr.pdf
[9] T M, Vinícius. Github. Cartola FC Dados. Last Update: 12 January, 2018. https://github.com/thevtm/CartolaFCDados

trained on the previous seasons, based on the fact that the predicted top scorers likely perform well and result in good overall final scores.

This implies that there are two main problem layers being considered, the first one corresponds to the model tuning and selection part, the second one regards the actual CartolaFC score. In the first stage, different models are defined and paired against each other, the performances will be based on the difference between the predicted and actual scores for each player, for each position. As for the second stage, the best performing algorithms will be used to select the top predicted scoring players for each round, and the final average score for the season will then be compared to other model average scores and ultimately the top 2017 performer for the fantasy league.

Metrics

The solution to the problem will be the final average score obtained for the 2017 CartolaFC Season. The Machine Learning models used will predict the scores for each player for each round, given a set of pre-round available statistics, and from these predictions, the team positions will be filled with the highest scoring players, and the score obtained will be the sum of the effective scores for each chosen player for that round. This process, repeated for every round of the season, allows for a final total score and average, which will then be compared to the highest scoring player for the 2017 CartolaFC Season.

Given this solution proposition, the evaluation metric used for this project will be regarding a simple difference between the predicted player score and the actual score obtained for the round. Given the difference can be either positive or negative and what matters is how far from the true scores the predicted value is, models will be evaluated given two main difference evaluation scores, the mean absolute error (MAE) and the mean squared error (MSE). Although both scores are used to compare the different model performances, ultimately the MAE will be the focus of the model refinement processes. The preference for MAE stems from the fact that outliers and values that are further from the mean do not influence as much as the case for MSE. The interpretation made for this dataset was that the larger values found were stronger indicators of some outstanding individual performances rather than the underlying implications of the different parameter values, which, ultimately, is what is trying to be modeled for this project.

An important distinction for this specific project is that the model scores will follow a second evaluation stage, in which the highest predicted scores are then selected and the actual scores received are then counted towards a 2017 season scoring average,

which is ultimately the goal of the project. This final average score is then compared to the best scoring player as a means to see whether the Machine Learning models used for this project were able to have a good final score. Given there is a fairly high likelihood of variance between the players chosen for each of the models used, it is very possible that a model which had a higher predictive error could potentially score higher than a model with lower errors. However, choosing a winning model with higher errors is a “hindsight” decision, as it takes advantage of information that is not available a priori, thus it could be that the best fitting model will not necessarily be the highest season scoring average model.

II. Analysis

Data Exploration

The datasets used for the project are available through the CartolaFC Dados Github repository[10]. The data is publicly available and obtained via a scraper[11]. The repository contains 6 spreadsheets for each year from 2014 to 2017, with the following data features:

Positions: Position ID, Position Name, Shorthand Name;

Status: Status ID, Status Name;

Teams: Team ID, Team Name, Shorthand Name, Name Code;

Matches: Match Id, Round, Home Team ID, Away Team ID, Home Score, Away Score, Result ('Home', 'Away', 'Tie')

Athletes: Athlete ID, Nickname, Team ID, Position ID;

Scouts: Round, Team ID, Athlete ID, Participated (True or False), Points, Average Points, Price, Average Price, Fouls Received, Missed Passes, Assists, Shots on Post, Saved Shots, Missed Shots, Goals, Offsides, Missed Penalties, Stolen Balls, Fouls Conceded, Own Goals, Yellow Cards, Red Card, Game without being scored, Difficult Saves (Goalie), Penalty Saves (Goalie), Goals Conceded (Goalie).

The Scouts spreadsheet has a data entry for each athlete, for each round of that season. This dataset is used as the basis for the project, and the other tables are used

[10] T M, Vinícius. Github. Cartola FC Dados. Last Update: 12 January, 2018. https://github.com/thevtm/CartolaFCDados
[11] T M, Vinícius. Github. CartolaFCScraper. Last Update: 3 May, 2017. https://github.com/thevtm/CartolaFCScraper

for referencing Team, Athlete, and Position names, and whether the game played for each athlete was 'home' or 'away'. Once filtering out all players that did not participate in a given round, and therefore are not scored, there are a total of 43,927 entries. 10,938 are from 2017, the testing set, and 32,989, from 2014-2016, constituting the training set.

The target variable, the Score/Points are a direct result from the specified scouts for the match with the following attributed weights[12]:

Defense Scouts:

Feature	Game without being scored	Penalty Saves*	Difficult Saves*	Stolen Balls	Own Goal	Red Card	Yellow Card	Goals Conceded*	Fouls Conceded
Points	5	7	3	1.7	-6	-5	-2	-2	-0.5

Offense Scouts:

Feature	Goal	Assist	Shot on Post	Saved Shot	Missed Shot	Foul Received	Missed Penalty	Offside	Missed Pass
Points	8	5	3.5	1	0.7	0.5	-3.5	-0.5	-0.3

Therefore, the score for a player in a round is the sum of occurrences of these features times the different associated weights.

These stats are calculated after the game is finished, hence, an average is taken for the previous participated rounds (maximum of 3) of each of these features for player, team, and opponent statistics that influence the scoring for each data entry. This is a way of avoiding look ahead bias, as the scouts for the present round, which calculate the player score, are not used in the prediction model for that round. In addition to these averages, using the Matches dataset, a ‘home/away’ feature is included for each instance of the Scouts dataset.

The following table is a description of the target variable for the training set, Points, for each position for the cleaned up dataset:

PositionID	**Count**	**Mean**	**Std. Dev.**	**Min**	**25%**	**50%**	**75%**	**Max**
1 (Goalie)	2015	3.4151	5.5686	-10.5	-1	3.4	7.7	22.7

[12] GloboEsporte. CartolaFC. Entenda as pontuações do Cartola FC e defina como escalar a sua equipe. July 7, 2017. http://globoesporte.globo.com/cartola-fc/noticia/2017/06/virou-rotina-cartola-bate-novo-recorde-de-times-na-rodada-10-55-milhoes.html

2 (Right/Left Defender)	4040	3.1455	4.1492	-9.7	0.1	2.7	5.6	27.1
3 (Defender)	4126	2.795	3.9437	-11.2	0	2.3	5.2	20.5
4 (Midfield)	9736	2.3156	3.8018	-10.8	-0.1	1.4	4	31.9
5 (Forward)	6516	2.6958	4.5543	-7.3	-0.1	1.2	4.4	32.7
6 (Coach)	2022	3.3343	2.1254	-1.52	1.68	3.095	4.8075	10.54

The table above shows that the variation between positions can be quite significant, in addition to that, it seems reasonable to assume that different features have varying implications for each position, that is, the influence that different features have on the amount of points obtained for each position will not be uniform, therefore the choice was made of developing models for each position, thus allowing for the idiosyncratic analysis of each position. This, of course, means that the data available is segmented and fewer data points are available for training each separated position, which could hinder performance, however, this trade-off seems reasonable for this specific scenario.

Something that stands out from this initial data description is the difference in standard deviations for each position. The interpretation for this discrepancy can be characterized as the greater effect that random chance has on the construction of the scores for each position. For example, the goalie position has the highest standard deviation, which implies that the player performances and scores will likely be harder to predict. This is intuitive, as the goalie position is likely the most dependent on luck for a good performance, the position's score is greatly defined by the difficult saves, which has inherently several layers of randomness involved.

Problems regarding the dataset

On the process of analyzing the dataset, some problems were found, likely resultant of the Cartola FC API, used to obtain the datasets.

The dataset contains missing values for some of the data points. Given the number of instances in which this happens is not substantial, the strategy chosen for dealing with these missing values was to remove the entries.

The 2015 and 2017 season scouts were cumulative, meaning there was a pre-processing stage that addressed this problem. In addition to this, for 2017, there were no scouts available for round 7, and given the previous round averages were taken as additional features, the missing data directly affects this calculation.

Unfortunately, there were no easy ways to circumvent this issue, and therefore the round was simply skipped, both in terms of predictions, as the points for that round were unavailable for comparison, and in the calculation of individual player averages.

There is no indication of whether the player was substituted, neither whether the player came from the bench, which surely bare influence on the amount of points obtained. However, although this is an important aspect, and data on this would be very valuable, for example, the amount of minutes each player played for each participating game, there was no easy way of circumventing this issue, and no changes were made to deal with this.

One other problem regarding this dataset is that there are cases in which a benched player receives a yellow card, without actually getting on the field, thus ending the round with a negative score. Identifying all instances that this occurred was not feasible, however, generally speaking, this happens fairly rarely and therefore no changes were made to the dataset.

There are other datasets with richer feature sets, however, there is a cost for obtaining them, thus using them for this project was not possible. For this reason, only this open source dataset was used and the choice of attempting different feature engineering strategies was made.

Exploratory Visualization

Below are a few visualizations of important characteristics for the dataset.

Importance of playing at home turf

The feature that has the most straightforward impact on the different scores and follows the same tendency for all positions is the feature that represents whether the player was playing at home turf or away. Below is the effect that this feature represents on the Points awarded for position 6 as represented by a boxplot:

Considering the amount of randomness that constitute the points awarded for each player, there is a very clear positive implication on the scores for players that play at home turf.

Relevant Correlations

Not all features demonstrate strong correlation with the Points target variable, however, some of the features show some important characteristics in relation to the target variable that pose some interesting interpretations.

The graph above shows the correlation between the position 1 Points and its relationship to the player's average missed passes for position 3, the position attributed to defenders. Although the points are fairly equally scattered, there is a tendency for higher scores for goalies that have defenders that miss more passes. This is quite interesting as a negative characteristic for the team implies a better performance for the goalie. This could be explained by the fact that missed passes can lead to more shooting chances for the opponent, and since goalie scores are heavily reliant upon their saves, this would allow them more chances, despite also possibly allowing more goals.

For all positions, there is fairly straightforward correlation between the player's average price and points for that round. This is expected as the price varies as the rounds progress for the seasons and this variation per round is dependent on the general performance that player had on the previous round. This means that the player price can be seen as a sort of player quality indicator, however, as seen from the graph below, this correlation is not as strong as one might expect.

It is clear that there is a tendency of higher scores for increased average price for the player, however, as seen by the scattered distribution, this relationship is not as strong. This is another indicator of the effect that randomness has on the points obtained by each player.

Several other features follow this same characteristic, with varying degrees of correlation. This implies that the models constructed for predicting the Points for these positions will likely not be able to explain the full variance of the different scores, as it is clear that randomness still plays an important role on this scenario.

Below is the heatmap of feature correlations for Position 4, including the target variable. Some of the feature correlations are straightforward, for example, the average goals scored for the player is heavily correlated with the player's average points. Similarly, for position 4, a high stolen balls average correlate with a higher point average as well.

For brevity the correlation maps for the other positions are omitted and can be found on the supporting notebook. That being said, for the other positions, similar relationships can be found between the different features. However, these relationships are not exactly linear between the positions, which is another example that the different positions have idiosyncratic relationships with the same set of features, posing different correlation dynamics.

Algorithms and Techniques

For the training/validation/testing sets, the following strategy is used: as the objective of the project is to ultimately score well on the 2017 CartolaFC Season, the 2017 dataset will be used as the testing set. For the separation into training and validation sets, a cross validation strategy will be used to partition the training data into K-Fold Splits. Given the data is distributed progressively through time, a time series K-Fold split was considered, however, given points obtained previously are not necessarily contingent to

future points and that each entry is comparable to each other, independent of season or tournament round, it was understood that there was no need for this time series modification on the fold splits. This also means that there is a simple shuffling of the data entries previous to the K-Fold splits.

Regarding the algorithms used, the project was split in 5 stages.

Linear Regression

Firstly, given the absence of a benchmark for the problem at hand, a model was created using a simple Linear Regression Model to act as a baseline performance for further comparisons. Linear Regression models are usually regarded as the simplest methods of implementing machine learning technique for analyzing regression problems, and therefore seems like a reasonable initial choice for an initial baseline implementation.

Linear Regression follows the Ordinary Least Squares (OLS) implementation, fitting the model by minimizing the residual sum of squares between the original target values and the values predicted by the linear model[13]. Given “w” corresponds to the associated weights of the different features, the model solves the following function:

SVR

As a second part of this project, an SVR model is formulated and tested against the baseline Linear Regression performance. This model undergoes two steps; first an initial baseline implementation, with no parameter tuning; second, each of the position models undergo tuning through the usage of different hyper-parameters, as a consequence of a Grid Search, in order to obtain the best fitting parameters and improving upon a more sophisticated model. The choice for SVR follows the previously mentioned paper on NFL Quarterbacks, and is a step up from a simple Linear Regression model in terms of hyper-parameterization, and allows for the study of increase in efficiency in improving the model with a different parametrization alternative via Grid Search.

SVR follows the logic behind Support Vector Machines, constructing hyper-planes in a high dimensional space, which attempts to find vectors that are at a maximum distance between the closests values of all different features. Below is a representation of a simple example of a support vector that is splitting two sets of features for some

[13] http://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares.

dataset. The dotted lines are the distances from the vector to the closests values for the given features, and the straight line at the middle would be the actual support vector[14].

XGBoost

For a third stage, an XGBoost model is implemented, following the idea behind the SVR implementation, both a baseline performance is measured, without tuning, and then the performance is compared with a tuned model via Grid Search as well.

The XGBoost model has grown in popularity recently as it has been the go-to model for Kaggle competitions, as it has been a constant presence among top performers. XGBoost stands for Extreme Gradient Boosting, and follows the idea of Gradient Boosting as proposed by the paper *Greedy Function Approximation: A Gradient Boosting Machine*, by Friedman[15]. Boosting is an ensemble technique in which different models are used to improve upon existing models, when used with a gradient descent algorithm to minimize loss, the new models predict the residuals from the previous models and are then added together to make a final prediction[16].

Simple example of XGBoost trees taken from the original documentation[17]:

[14] http://scikit-learn.org/stable/modules/svm.html#implementation-details
[15] https://xgboost.readthedocs.io/en/latest/model.html
[16] https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
[17] https://xgboost.readthedocs.io/en/latest/model.html

Random Forest

Following suit, as a fourth step, a Random Forest Regressor is implemented, also contemplating an initial baseline implementation followed by some model tuning. However, for this tuning step, the method used was of a Randomized Grid Search, which allows for the study of an alternative type of model tuning process.

A random forest is a meta estimator in which each tree from the forest ensemble is built from a sample of the training data set. For each of these trees, as an additional layer of randomness, the splits are chosen based on a random subset of the features, which causes each tree to have a worse performance in general, however, it constructs a fairly diverse set of estimators. The finalized model is then a combination of all of these estimators, as the final results of each tree are averaged out, thus improving upon the overall accuracy and at the same time controlling for over-fitting[18].

[18] http://scikit-learn.org/stable/modules/ensemble.html#forest

Simple Random Forest Example[19]

MLP

For the fifth stage for the project, a neural network model is used, an MLP Regressor will be set as a baseline neural network performance and will also be tuned via a Grid Search mechanism. The choice of using a neural network follows the idea presented in the NFL Quarterbacks study, in which a neural network was created and tested for different architectures and parameters, and also works in line with the fundamental idea of this project which is in essence a study of different machine learning techniques.

The MLP, or Multi-layer Perceptron is a neural network that learns a loss minimizing function through the means of different processing layers with sets of neurons. The input layer consists of a set of neutrons that represent the feature set of the data. The middle layers, called hidden layers, transform the incoming values with a weighted linear summation, followed by a non-linear activation function, which can take different forms. The output layer then takes on the last hidden layer values and transforms them into output values[20]. The figure below shows a one hidden layer MLP with scalar output:

[19] https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/
[20] http://scikit-learn.org/stable/modules/neural_networks_supervised.html

Benchmark

Unfortunately, given the few and limited attempts at using Machine Learning for this specific scenario, there is no readily available benchmark model, therefore, as discussed previously, a Linear Regression Model is implemented with some minimal feature engineering of the available data as a way of benchmarking the performance of the other chosen techniques such as the SVR Model and the Neural Network implementation. Given the structure of the proposed problem, a different model will be made for each of the available positions, 6 in total. A standard version of the Linear Regression Benchmark Model will be used for each of the positions with a set of relevant feature variables. The benchmark model will make predictions of the players' score for each round, this can then be compared with the actual score obtained for each player, thus giving an error that can be compared to the other proposed models, using the proposed metrics.

Given the results obtained from the benchmark models, the players with the highest predicted scores will be chosen for each position, following the proposed formation scheme. This will yield a score for each round and thus give a final season average score, which can also be compared to the results obtained from the other models and ultimately compared to the highest scoring player for the Fantasy League. Therefore, there are two kinds of benchmarks, one based on the prediction errors for the different models, benchmarked with the Linear Regression Models, and the second one is the

final average total score for the actual CartolaFC season, which will be also benchmarked with the top scoring player.

III. Methodology

Data Preprocessing

As previously mentioned, the data obtained from the CartolaFCDados Repository is divided into 6 different spreadsheet types for each season (2014, 2015, 2016, 2017). The first preprocessing step is to create a column for the year for each of the relevant spreadsheets, in order to be able to distinguish them once grouped up. Given we are attempting to use the previous season data in order to predict 2017 scores, the scouts for 2014 through 2016 are grouped up.

Only the entries for players who participated matter for the total score, hence, players who did not participate are filtered out. Given the scouts do not include player positions, a new column is added with a categorical feature for each player position, from 1 through 6. This information is obtained from the Athletes spreadsheet, which includes the necessary data to identify each athlete and their associated position.

In addition to that, there is also no scout for whether the game was played at home turf or away, therefore a new column is added with a binary feature indicating whether the game was played ‘home’ or ‘away’. Although not explicitly informed, this information is contained in the Matches spreadsheet since it informs the home and away team ids. By identifying what the team id for the scout entry is, along with what year that match happened and on what round, it is a simple matter of identifying whether the home team id matches it or not for that specific year and round. In similar fashion, the adversary team ID and name are included for each entry, as well as the player’s own team name for ease of identification.

As aforementioned, the scouts for 2015 and 2017 are cumulative, therefore a function is run on these specific season scouts in order for them to be correctly distributed. Given the scouts are a simple linear summation as the rounds go, it is just a matter of running a function for each round, for the relevant seasons, getting the difference between the current and previous round scouts.

Feature Engineering

As mentioned previously, in order to use the scouts as features, some feature engineering needed to be made, as the scouts are only obtained post round, and there are no given scout averages. Three different sets of features were constructed, individual player averages, team position averages, and opponent team position averages.

I. Player Averages

One consideration that was taken into account when doing this implementation was how to construct features that could try to indicate somewhat of an individual player's strength for each feature while diminishing the importance of the opponent team's strength for each round, and also accounting for the randomness that is intrinsic to the available features and the game as a whole. Of course there are several different approaches that are available for tackling this issue, for this project I took the average for the previous 3 participating games for the player for each of the features. In order to maintain the data for the initial rounds and for players that played perhaps two are three times, the average was taken for the available rounds with a minimum of one and a maximum of three.

II. Team Position Averages

Another consideration that was taken into account was the idea of making features that could attempt to represent the strength of the player's team. Likewise, there are several possible approaches, in this case, the idea was to get the team averages for each position. These features would give a measure of the player's teammates general strength, which intuitively should be a factor in their own performance. This was done in a similar fashion as the player's own averages, although in two different steps. Firstly, the entries were grouped by team ids and positions, and averages were taken for each of the available scouts. This represented the team average per position for each round for that season, however, similar to the problem with the initial scouts, these are the post game averages. Therefore, as a second step, the averages for the last three matches are taken, the same way the player averages were obtained.

III. Opponent Team Position Averages

The final feature set construction consisted of the idea of making features that could attempt to represent the strength of the opponent team. There are also several possible approaches, for this feature set, the idea was to get the opponent averages for each position. The idea behind this set was to construct features that would help indicate the strength of the opponent team, partitioned by each position, which intuitively should also

influence the player's performance. This was obtained by cross referencing each player's opponent team for that round and their associated team averages.

Feature Scaling

As pointed out, the features used have very different characteristics and influence in different ways the target variable, hence the strategy used in order to deal with these discrepancies was to scale the features via a Standard Scaler function, which removes the mean and scales each feature to unit variance.

The process of scaling must be limited to each of the training sets, as using it for the entirety of the data implies an unwanted data leakage between training and testing sets. For this reason, and given a KFold strategy was used to divide the training sets, a pipeline which included the scaling was established for the creation of the different models where scaling is seen as an important step.

Missing Values

As a final preprocessing step, there are several entries that have missing values, mostly due to problems with the Cartola FC API, as mentioned previously, but with the addition of missing values as a result of the feature engineering step, for example the first round data. Since the number of missing values was not substantial, and the first round statistics were already expectedly unusable, the missing values were simply removed from the datasets.

The following table displays the final shapes of each of the data frames that are used:

Data frame (2014-2016)	Entries	Columns
Position 1	2015	202
Position 2	4040	202
Position 3	4126	202
Position 4	9736	202
Position 5	6516	202
Position 6	2022	202

Data frame (2017)	Entries	Columns
Position 1	663	202
Position 2	1370	202
Position 3	1373	202
Position 4	3277	202
Position 5	2059	202
Position 6	671	202

Models

As mentioned previously, this project is divided in six different implementation stages, a Linear Regression Model, as a means to formulate a baseline score and benchmark model performance, an Epsilon-Support Vector Regression (SVR) model, which builds upon the baseline implementation with a more complex structure and refinement capabilities, an XGBoost model, also with baseline and refined versions, a Random Forest Regressor Model, also with a baseline implementation followed by model tuning via Randomized Grid Search, and finally a Multi-layer Perceptron Neural Network (MLP), with a simplified architecture and implementation, followed by versions with more idiosyncratic model refinements.

Training / Testing Split - KFold

The KFold strategy for splitting between training and testing data for the 2014-2016 season data was used uniformly throughout the different model implementations. The chosen K value was 5, that is, the training set was shuffled then divided in 5 parts, each fold representing 20% of the data. For each of the 5 training steps, one of the folds was left out as the internal testing set, and the other 4 splits are used for training the model. This cross-validation is scored against the predefined metrics, namely MAE and MSE, for the five different KFold setups, resulting in 5 different scores for each metric. These scores are averaged out, defining a mean and standard deviation for each metric, which ultimately correspond to the performance metrics for each model.

Kfold Visual Representation[21]

[21] http://genome.tugraz.at/proclassify/help/pages/XV.html

Feature Selection

A preliminary note regarding feature selection for the different models; each position has idiosyncratic relationships to the different features, for example a player's average missed shots are fairly irrelevant for goalies, whereas for example, saved penalty kicks are only relevant for goalies. In order to deal with these different feature-position relationships, the following methodology was used: before setting the parameters to be used as features, a mean of the 2014-2016 (total training set) is taken for all features for that position, if that value is less than 0.01, which would imply a fairly irrelevant feature, that feature is not included as a parameter. This value, 0.01, is fairly arbitrary, and could be modified for other implementations. Using the mean of the whole 2014-2016 set could imply some data leakage, since the KFold split strategy was used for training the models, however, given this is a very small value, it was understood that this would not influence the results in any sensible manner.

Tuning

Following a baseline implementation of the algorithms, performance can generally be improved through the fine tuning of the model's associated hyperparameters. For this project two different methods were used, Grid Search Cross-Validation for SVR, XGboost, and MLP, and Randomized Search Cross-Validation for the Random Forest Regressor.

GridSearchCV

In a Grid Search Cross-Validation[22] method of parameter tuning, given a set of parameters that will be tuned for the specified model, a set of values are chosen for each parameter which will serve to construct and train all of the possible combinations available. Given a KFold split in the training procedure, each of the possible combinations of parameter values are then trained and tested K times. This implies a very computationally intensive task, hence the concept of a "brute-force" way of testing different parameter value combinations. For example, in the case of 5 different parameters, with 5 chosen values for each, a total of 3125 combinations would be possible, and on top of that, given a 5-Fold Split cross-validation, a total of 15,625 model instances would be trained. Depending on the model and parameters, the training time for such cross-validation procedure can take a fairly long time, hence, although this allows for very robust results, as all possible combinations are considered, it might not always be the most efficient alternative.

[22] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

RandomizedSearchCV

Different from the “brute-force” Grid Search method of parameter testing, in Randomized Search Cross-Validation[23], not all of the values are tested, but rather a set number of parameter combinations are tested from the distributions of the chosen parameters. Although this means that the best alternative might not end up being selected through this random procedure, the computational cost of this method is much lower and is an alternative for this cost-benefit relationship between model performance and computational cost.

Linear Regression

As aforementioned, the Linear Regression model is commonly used as a baseline implementation for machine learning regression problems as it usually entails a fairly simple and fast setup and implementation.

Implementation

Given the purpose of this initial implementation was to formulate a baseline performance, the training follows a simple “out-of-the-box” implementation of the algorithm. For each position, the algorithm was trained using a 5-Fold split, going through the Standard Scaler pipeline step, after selecting the relevant features following the strategy informed previously. As established earlier, two different scoring metrics are separated for analysis between models, the Mean Absolute Error (MAE) and the Mean Squared Error (MSE). Both MAE and MSE are recorded for each of the positions, these are the baseline performance metrics that ultimately will be used to choose which models had the best performances and consequently are the ones that should be chosen to predict the scores for the 2017 season.

As a second step of this implementation, the models were then used to predict the scores for the 2017 season, the result was a final average of **46.8447**. These results serve also as somewhat of a baseline performance for the Cartola FC scoring system, in addition to that, they serve as a way of identifying whether there is some link between model performance, through this error reducing methodology, and the actual scoring for the Cartola FC season. The expectation, of course, is that an increase in model efficiency leads to an increased Cartola FC Season average score.

SVR

[23] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

The SVR model serves as step up in model complexity from the baseline Linear Regression with some more intricate parameter tuning, which was realized via a Grid Search Cross-Validation algorithm.

Implementation

The SVR baseline implementation follows a very similar structure to the Linear Regression implementation. The chosen relevant features are selected for each position, as described earlier in order to get rid of the irrelevant features for each position. Then, the “out-of-the-box” models for each position are trained with the usage of a 5-Fold split, following the established Standard Scaling pipeline step, fit for each of the different fold partitions. The MAE and MSE results obtained are recorded so that they can be compared to the performance of the tuned version of each of the models.

Refinement

For the tuning of parameters for the SVR model, a GridSearchCV was used. Below, a table with the list of parameters, values, and the different obtained results for each model is displayed:

Parameter	Values	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6
C	[0.5, 1, 1.5, 2]	1	2	2	0.5	0.5	1
Epsilon	[0.5, 1, 1.5, 2]	0.5	0.5	0.5	0.5	0.5	0.5
Kernel	['rbf', 'linear']	‘rbf’	‘rbf’	‘rbf’	‘rbf’	‘rbf’	‘rbf’

Following the same idea presented in the Linear Regression Model implementation, these refined models were used to predict the scores for the 2017 Cartola FC season, the final average for the season was **49.5061**.

XGBoost

The XGBoost model is popular Machine Learning algorithm with a more complex structure which also allows for some parameter refinement, which was also realized via a Grid Search Cross-Validation algorithm.

Implementation

The XGBoost implementation follows the same format as the SVR, the standard strategy of feature selection was chosen, and the baseline model was applied to the

data following the 5-Fold split, as established previously. From this cross-validation, the different MAE and MSE results were obtained and used to compare to the parameter tuned models as well as general results from the other models.

Refinement

As previously mentioned, a GridSearchCV was used for the tuning of the XGBoost parameters. Below, a table with the list of parameters, values, and the different obtained results for each position is displayed:

Parameter	Values	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6
max_depth	[4,5,6]	6	6	6	6	6	4
min_child_weight	[1,2,3]	1	2	1	2	3	1
subsample	[0.5, 1]	0.5	0.5	0.5	0.5	0.5	0.5
colsample_bytree	[0.5,1]	0.5	1	1	1	1	0.5
n_estimators	[200,400,600]	200	400	600	200	200	400

Following the same idea presented in the previous implementations, these refined models were used to predict the scores for the 2017 Cartola FC season for a final season average of **52.6950**.

Random Forest

The Random Forest Regressor Model is another alternative used to test against the baseline performance. For this model's refinement process, a Randomized Search Cross-Validation alternative was used. During implementation, the attempts at cross-validation using the MAE criterion as the defining metric demonstrated exceedingly poor performance. After research[24] it was evident that the sklearn Random Forest function had inherently very poor performance, as the usage of the median and its constant updating demand a large amount of computing power, unavailable at hand. For this reason two main strategies were used; first, a lower limit of 200 'n_estimators' was used for the Randomized Search, as the performance is heavily taxed by the number of estimators attempted; second, the fitting used for the models was made with the MSE criterion, however, the cross-validation step through the 5-Fold split strategy was taken with an MAE scoring parameter. This, of course, implies that some model performance capabilities are restrained and better results could have been obtained

[24] Trees with MAE criterion are slow to train #9626. Github. https://github.com/scikit-learn/scikit-learn/issues/9626

were it not for these changes, nonetheless, all of the tuned models showed better results, hence, the refinement process was still a relevant and important step in improving the models.

Implementation

Similar to the previous models, the Random Forest implementation has the same structure. The standard strategy of feature selection was chosen, and the baseline model was applied to the data following the 5-Fold split. From this cross-validation, the different MAE and MSE results were obtained and used to compare to the parameter tuned models as well as general results from the other models. The only difference in this implementation was that the "out-of-the-box" model has a value of 10 for the "n_estimators" parameter, which yielded larger than normal error values, for this reason, a new standard value of 100 was used, which was then further tuned during the refinement process.

Refinement

As previously mentioned, a RandomizedSearchCV was used for the tuning of the Random Forest parameters, with the variation that the scoring mechanism of minimizing the MSE was used instead for the models instead, for performance reasons. Below, a table with the list of parameters, ranges, and the different obtained results for each position is displayed:

Parameter	Values	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6
n_estimators	[int(x) for x in np.linspace(start = 20, stop = 200, num = 10)]	140	180	120	40	20	120
max_features	['auto', 'sqrt']	'sqrt'	'sqrt'	'auto'	'auto'	'sqrt'	'auto'
max_depth	[int(x) for x in np.linspace(10, 110, num = 11)] , None	30	70	100	10	10	60
min_samples_split	[2, 5, 10]	5	2	2	10	5	10
min_samples_leaf	[1, 2, 4]	1	4	4	1	2	2

Following the same idea presented in the previous implementations, these refined models were used to predict the scores for the 2017 Cartola FC season for a final season average of **47.7181**.

MLP

The Multi Layer Perceptron model, which is part of the so called Neural Network part of Machine Learning, is an attempt at a different approach to the problem. The model implementation followed a similar procedure as the previous models, including a refinement step via Grid Search Cross-Validation.

Implementation

The MLP implementation follows the same format as the previously established models, the standard strategy of feature selection was chosen, and the baseline model was applied to the data following the 5-Fold split, as established previously. From this cross-validation, the different MAE and MSE results were obtained and used to compare to the parameter tuned models as well as general results from the other models. One issue that arose from the initial implementation was regarding the usage of a Standard Scaler step as part of the pipeline for the model. From general research regarding the topic, although not a consensus, a scaling procedure is usually recommended for Neural Network problems. However, upon using this procedure, the errors increased substantially, whereas when the StandardScaler step was skipped, the error ranged in a similar fashion to the previous models. For this reason, the decision taken was not to include the scaling step in the pipeline of the model definition.

Refinement

As previously mentioned, a GridSearchCV was used for the tuning of the Neural Network parameters. Below, a table with the list of parameters, values, and the different obtained results for each position is displayed:

Parameter	Values	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6
hidden_layer_sizes	[50,100,150,200]	200	200	200	150	200	150
activation	['logistic', 'relu']	‘logistic’	‘relu’	‘relu’	‘relu’	‘relu’	‘relu’
learning_rate_init	[0.01, 0.001, 0.0001]	0.001	0.0001	0.0001	0.01	0.01	0.01
alpha	[0.01, 0.001, 0.0001]	0.0001	0.01	0.01	0.01	0.001	0.01

Following the same idea presented in the previous implementations, these refined models were used to predict the scores for the 2017 Cartola FC season for a final season average of **52.3989**.

IV. Results

Model Evaluation and Validation

The tables below demonstrate the performances of each of the above described models through the comparison of the chosen metrics, MAE, MSE and their corresponding Standard Deviations. The best performing models are highlighted for each position.

Position 1		
Model	MAE	MSE
Linear Regression	4.6667 (0.1672)	33.5829 (2.1800)
SVR	4.5419 (0.1608)	31.1216 (2.2865)
SVR_tuned	***4.5414 (0.1599)***	31.0673 (2.2597)
XGBoost	4.6003 (0.1323)	32.1692 (2.0564)
XGBoost_tuned	4.5432 (0.1408)	31.1795 (2.1219)
Random Forest	4.5837 (0.1527)	31.3901 (2.2306)
Random Forest_tuned	4.5635 (0.1320)	31.1443 (1.9117)
MLP	4.5808 (0.1263)	31.7922 (1.8654)
MLP_tuned	4.5603 (0.1753)	30.9520 (2.1276)

Position 2		
Model	MAE	MSE
Linear Regression	3.2614 (0.1208)	17.1643 (1.6547)
SVR	3.1188 (0.1168)	16.2679 (1.5020)
SVR_tuned	3.0859 (0.1275)	16.0385 (1.5978)
XGBoost	3.1482 (0.1431)	16.3038 (1.6692)
XGBoost_tuned	3.0779 (0.1355)	15.8219 (1.5885)
Random Forest	3.1213 (0.1331)	16.4159 (1.7277)
Random Forest_tuned	***3.0537 (0.1332)***	15.7032 (1.6044)
MLP	3.3036 (0.0827)	17.8213 (1.2453)
MLP_tuned	3.2033 (0.1064)	16.6079 (1.4895)

Position 3		
Model	MAE	MSE
Linear Regression	3.0713 (0.0956)	15.2451 (0.8512)
SVR	2.9454 (0.1081)	14.4006 (0.8827)
SVR_tuned	2.8805 (0.1168)	14.0026 (0.9050)

Position 4		
Model	MAE	MSE
Linear Regression	2.7932 (0.0214)	13.9950 (0.0798)
SVR	2.7172 (0.0275)	14.6633 (0.1567)
SVR_tuned	***2.6927 (0.0279)***	14.5194 (0.1580)

XGBoost	2.9649 (0.1031)	14.3270 (0.8760)
XGBoost_tuned	2.8444 (0.1159)	13.7548 (0.9027)
Random Forest	2.8329 (0.1006)	13.9056 (0.9303)
Random Forest_tuned	***2.8033 (0.1061)***	13.5813 (0.8960)
MLP	3.1767 (0.1595)	16.7656 (1.2963)
MLP_tuned	3.0168 (0.1187)	14.8222 (0.8945)

XGBoost	2.7757 (0.0256)	13.8242 (0.0840)
XGBoost_tuned	2.7250 (0.0256)	13.8443 (0.0837)
Random Forest	2.8709 (0.0360)	14.5221 (0.1924)
Random Forest_tuned	2.7802 (0.0267)	13.9318 (0.1321)
MLP	2.9893 (0.0346)	15.7306 (0.6198)
MLP_tuned	2.7399 (0.0480)	14.0416 (0.1279)

Position 5

Model	MAE	MSE
Linear Regression	3.3341 (0.0866)	20.2631 (1.4445)
SVR	3.1183 (0.0989)	21.6829 (1.8096)
SVR_tuned	***3.0999 (0.1032)***	21.6861 (1.8521)
XGBoost	3.3082 (0.0885)	20.0875 (1.6018)
XGBoost_tuned	3.2222 (0.0992)	20.0498 (1.7313)
Random Forest	3.4274 (0.0730)	21.3634 (1.3978)
Random Forest_tuned	3.3189 (0.0908)	20.1786 (1.6739)
MLP	3.3911 (0.0960)	20.6485 (1.4861)
MLP_tuned	3.2272 (0.1655)	20.1524 (1.4867)

Position 6

Model	MAE	MSE
Linear Regression	1.6639 (0.0612)	4.2747 (0.3486)
SVR	1.6561 (0.0617)	4.2140 (0.3557)
SVR_tuned	1.6575 (0.0631)	4.2065 (0.3513)
XGBoost	1.6458 (0.0754)	4.1235 (0.3457)
XGBoost_tuned	***1.6219 (0.0668)***	3.9948 (0.3240)
Random Forest	1.6392 (0.0594)	4.0598 (0.3103)
Random Forest_tuned	1.6364 (0.0575)	4.0457 (0.2915)
MLP	1.6886 (0.0558)	4.3100 (0.3206)
MLP_tuned	1.6397 (0.0666)	4.1763 (0.3227)

Interesting to note that the best performing model, as measured by MAE, can be different if measured for MSE, however, since the tuning for the models was done through the MAE criteria, with the exception of the Random Forest Regressor as explained previously, the models that were considered best performing are the ones with the lowest Mean Absolute Errors.

One important characteristic that is shown from these results is that for all positions, the best performing models have MAE's that fall within one standard deviation from the points distributions. This shows that despite the randomness that is intrinsic to these

distributions, the obtained mean absolute errors fall within a reasonable practical margin, demonstrating that the models seem to be able to capture, at least to some extent, the variations that occur on the different point distributions. One other characteristic of the results is that there is a fairly substantial variation in errors between positions, corroborating the idea that each position has its own idiosyncratic properties. This is further evidenced by the fact that the best performing models do not all stem from the same algorithmic types, the variation in best performing algorithms further suggest this idea that separating the original project structure into different positions allows for a better identification of these unique and not immediately apparent characteristics. However, the interpretation of why each different model performs better or worse depending on the position is not straightforward, nonetheless, given the interpretability of the models are not a focus of this particular project, these considerations are not taken into account in the model choices or specifications.

Below are the MAE results for the 2017 CartolaFC Season obtained using the models to predict the scores and comparing them to the actual obtained scores:

	Linear Regression	SVR	XGBoost	Random Forest	MLP
Position 1	4.9055	4.7649*	4.7181	4.7304	***4.7097***
Position 2	3.3634	3.2539	3.2267	***3.2149****	3.3077
Position 3	3.0274	2.9648	***2.9276***	2.9410*	2.9872
Position 4	2.9625	***2.8842****	2.9028	2.9497	2.9947
Position 5	3.4895	***3.3149****	3.4340	3.5485	3.5071
Position 6	1.6600	1.6402	***1.5971****	1.6053	1.6226

* Best performing models on the training set.

An indication of the improved performance of the models relative to the baseline performance of the Linear Regression is the reduced MAE for the unseen 2017 test set for the best performing models. Although the best performing models on the training set did outperform the Linear Regression benchmark model, they were not the best performers overall for positions 1, and 3. However, when analyzing the performance of the best models for the test set and their performance for the training set, a few relevant results are found; for position 1, which as discussed previously is the position with biggest randomness dependency, SVR had the fourth best test set score, however, for example, XGBoost which had almost the same performance as SVR for the training set, was also the second best test performer, by a very small margin as well; for position 3, Random Forest had the second best test performance and the training performance of the XGBoost was actually the second best, after Random Forest.

The tables below demonstrate a second step in the model analysis procedure, which is to use the above mentioned models to select the players with top predicted scores for the CartolaFC 2017 season. As mentioned previously, the models trained on the 2014-2016 scouts are used to predict the scores for the unseen 2017 season data to see how well they perform in practice. The results contained in the tables include the season average points for each position, along with their associated model predictions.

Linear Regression

Position	Points	Prediction
1	0.6583	6.0887
2	4.1167	5.8020
3	3.3153	5.4860
4	4.3590	4.6861
5	4.9514	5.3300
6	3.9836	4.9493

SVR - Modified

Position	Points	Prediction
1	***2.4528***	***4.3550***
2	3.5583	4.5455
3	3.5972	4.1986
4	***4.6458***	***2.8355***
5	***5.1694***	***2.6826***
6	3.8200	4.1976

XGBoost - Modified

Position	Points	Prediction
1	3.3417	3.9688
2	4.1097	4.9126
3	3.5569	4.6853
4	4.7694	3.5644
5	5.0986	4.1828
6	***4.7450***	***4.3202***

Random Forest - Modified

Position	Points	Prediction
1	1.7278	4.6559
2	***4.1625***	***4.3182***
3	***3.9389***	***4.8774***
4	4.5722	4.3056
5	3.6625	4.9120
6	4.1736	4.6740

MLP - Modified

Position	Points	Prediction
1	3.3194	3.6799
2	3.5222	5.5187
3	3.7806	4.9651
4	4.7326	4.9114
5	5.1250	5.5174
6	5.2933	4.5881

The models that had the best training performances have been highlighted. Given the final model would be a combination of the best performing algorithms, then the final season 2017 Cartola FC result would have been **52.3226**. Although better than the Linear Regression benchmark, the result of the best model configuration would not have been better than different lower performing combinations, for example, the models that yielded the best season average score would have totaled an average of 54.2542. The worst score performance was for position 1, which, as expected, is the position with most randomness in its distribution and feature correlation. This is in line with the original expectations, given the intrinsic random nature of the sport and the limited amount of parameters used in the project, it would be natural that the variations in the

target variables would not be entirely explained by the available parameters and training data, and that variances in the predictions were expected.

That being said, from the total average results for 2017 presented earlier for each model, it is clear that although there seems to be a correlation between reducing the MAE's and improving on the final average score for the season, this is not a straightforward relationship. This is evidenced by the fluctuations in final average result for specific positions as opposed to the model's performance on the training set, for example, there are final position averages obtained from the Linear Regression model that ended up higher than the some of the different model predictions which had better performances on the training set, with lower MAE's. However, generally speaking, the majority of the models with lower MAE's on the training set ended up also with higher final season average scores. This varying dynamic was expected, as the parameters used are fairly limited and there is an intrinsic randomness to the sport, however, the overall performance of the models, as compared to an average fantasy league player, was fairly decent, with a final result which could be considered above average.

Sensitivity Analysis

In order to validate each model's robustness, a sensitivity analysis was made for each of the best performing models. In order to accomplish this, the following process was made; first, via the train_test_split function, 100 different splits were made, using a 80:20 training-test set ratio; second, for all of these splits, the model was trained and used to predict on the test set; third, the Mean Absolute Error was taken for all of these different sets, thus allowing for the comparison of the different values and identification of whether the average error was in line with the MAE obtained for that model. The model could then be considered fairly robust if the distribution of the MAE's obtained followed somewhat of a normal distribution centered around a mean that was similar to the one obtained originally. Below are the results:

From the above graphs it is possible to see that the distributions do in fact follow very closely a ‘bell-shaped’ normal distribution. The table below shows the values obtained from the average of all the MAEs for the 100 splits for each position as well as the original performance for the models. The proximity of the values is another indication of the relative robustness of the models.

MAE Results	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6
5-Fold	4.5414	3.0537	2.8033	2.6927	3.0999	1.6219
Split Averages	4.5645	3.0252	2.8220	2.6978	3.1245	1.6290

Justification

As previously discussed, two different benchmarking layers were proposed for this project. In the absence of a baseline model performance, a simple Linear Regression model was formulated and the associated MAE’s were used as a baseline benchmark for the other model performances. Below is a table with the performance for the Linear

Regression and the best performing combination of models for each position with the 2017 season test set:

Position	Linear Regression	Best Model*	Difference
1	4.6667	4.5414	-0.1253
2	3.2614	3.0537	-0.2077
3	3.0713	2.8033	-0.268
4	2.7932	2.6927	-0.1005
5	3.3341	3.0999	-0.2342
6	1.6639	1.6219	-0.042

* The best model is a combination of the different best performing models.

It is clear that there was an improvement from the baseline Linear Regression performance for all positions.

As a second benchmark, the comparison is made between the total average score for the 2017 CartolaFC season for the Linear Regression model and the top performing model configuration. The following table demonstrates these results:

Position	Linear Regression	Best Model*	Difference
1	0.6583	2.4528	1.7945
2	4.1167	4.1625	0.0458
3	3.3153	3.9389	0.6236
4	4.3590	4.6458	0.2868
5	4.9514	5.1694	0.218
6	3.9836	4.745	0.7614

* The best model is a combination of the different best performing models.

There was an increased average scoring for the combination of best performing models for all positions. Nonetheless, although for the best performing model configuration the improved performance resulted in an increased season average score, not all models that performed better than the Linear Regression model ended up with higher season averages. As previously discussed, this is in line with what was expected, there seems to be a general correlation between increased season average score as the models are improved via the MAE metric. However, this relationship is not linear, there is still a large amount of variance imbued on the data that is not explained by the best

performing models. That being said, given the initial objective of obtaining a decent score for the 2017 CartolaFC season, this goal was achieved. The final model is fairly robust and has good performance when applied in practice with the Cartola FC data, although the final score is still far from the best player for the Cartola FC 2017 season, with a 67.4 average, the score is still considered an above average performance, with still much room for improvement given a limited set of features available.

V. Conclusion

Free-Form Visualization

Below are the graphs that represent best performing models' predicted values as compared to the actual values obtained for each position:

Position 1

It is clear from the graph that what the model is predicting are small fluctuations in the score as a result from the relevant features, which means it ends up finding the scoring tendencies of the different players. This also demonstrate the limitation presented in capturing all of the variance for the position scores.

Position 2

For position 2, the variance in prediction is increased, which is a way of showing that the model is able to capture more of the variance in this position's score. Nonetheless, this is another example of how there is unknown information, be it randomness or missing features, that dictates a lot of the variance in player points for each round.

Position 3

Position 3 is another step towards reducing the gap in predictions as related to the true values. This is an indication that the model is able to explain more about the data, as compared to, for example, position 1.

Position 4

Position 4, which also had the greatest number of data entries available shows that the correlation between more data and increased model predictive power is not straightforward for this problem and dataset at hand. It seems that the greatest hurdle in better predicting the scores is the quality of the feature set.

Position 5

Position 5 had the second highest standard deviation for the points distribution of the training data, which intuitively would mean that the model would likely have difficulties in capturing all of this variation, as it seemed the position was more influenced by randomness. This position idiosyncrasy could be a result of goal scoring, as they provide the most points for each player, and position 5 is the Forward position, and more goals will end up coming from these players. This variation is evidenced by the graph above, showing how the predictions ended up tending around somewhat of a mean.

Position 6

Finally, for position 6, which had the best performing model seemed to have predictions that were much closer to the actual true scores. For the above graph it is possible to see the fluctuations in predictions following the fluctuations in true values in a closer way. Nonetheless, there is still limitation in the amount of variance that the model was able to catch.

Reflection

Idea

The motivation behind taking on this project was that it would allow for a practical application of Machine Learning in several different forms, using different models, algorithms, and tuning methods, allowing for a hands-on approach in a complex but fun endeavor. Using the CartolaFC environment imposed a clear initial difficulty which was the lack of a proper complete “end-to-end” study of the topic, thus imposing, in addition to the expected Machine Learning challenges, the necessity of novel programming structures to accomplish the different proposed tasks. Nonetheless, this served as a great incentive, as a lot would inevitably need to be learned during the process, especially not having a proper programming background.

There was also an additional challenge, which is imposed by one of the requirements of this project, which is that the data used should be open-sourced. Although there are a lot of alternative data sources for general football statistics, I did not find any well organized and easily accessible ones for the Brazilian National Tournament, relevant to the CartolaFC. There is a fairly famous football statistics database in Brazil called Footstats[25], however, recently they have “pay-walled” their data, which discards it as an

[25] Footstats. http://meu.footstats.net/

alternative. Therefore, the only dataset that ended up being used for the project was the ones directly tied to the CartolaFC database, which has a fairly limited set of features available. However, this was also a sort of motivation as this would imply the necessity of substantial data pre-processing and feature engineering steps, along with the challenge of scoring well on the 2017 Season with somewhat of a handicap.

Data Pre-Processing

The data was collected from the CartolaFCDados Github repository, which uses a Scrapper to collect the data from the CartolaFC API. This data was fairly neatly organized, however there were some vices, such as cumulative data for 2015 and 2017 scouts, which had to be addressed.

Passed this initial data cleaning stage, constructing what ended up being the relevant features for the models was quite a challenge. None of the features present in the scouts spreadsheets could be used “as-is”, the data was *a posteriori* information, that is, were only available post round, and thus, the associated player scores were a direct result from the data entries they were in. After the idea behind the different chosen feature sets was made, which would include player features, team features, and opponent features, the challenge was translating these concepts into different programming steps. After a lot of trial and error, a systematic way of including this data was created. Given the lack of familiarity with Python, several of the decisions made and different function structures could surely be improved and made more efficient, however, the final pre-processing result was in line with what I had aimed at obtaining.

Data Exploration

With the data at hand, some different data exploration techniques were used, visualizing the different feature relationships, the characteristics of the target variable and the correlations between them. One of my initial concerns was that there would be a lot of variation in the different feature averages for each season, which would greatly implicate in the predictive power of the 2017 season data. However, the data showed that the features had values that were fairly equal among the different seasons. The picture painted by the data showed that, although it seemed that the available features would be able to help explain the different variations of the target variable, the general randomness of the distributions and lack of visible straightforward tendencies pointed towards the fact that the final models would likely still have great difficulty in capturing all this variance. This gave the initial insight, which I thought was quite interesting, that the best performing model could very likely not be the model that best scored for the 2017 season. Hence, it would be important to delineate what the main objective was,

which was finding the best performing model or model configuration which best explained the 2014-2016 and consequently the 2017 dataset. A good 2017 total average score would be a result of a well established model and an unequivocal combination with luck.

Models

Before trying out different models, different training/test set configurations were contemplated. The choice of using the K-Fold Cross Validation strategy was made from general research regarding this type of implementation, the perception of robustness of the results obtained from it, and the fact that the data was not very large, and it would not pose a large computational burden. Different numbers of splits were tested, and considering the computational costs and results obtained, the chosen number of splits was 5.

The idea of using a Linear Regression model as a starting baseline/benchmark was fairly straightforward, the model is known as being commonly used as a benchmark performance for regression problems, but also its ease of implementation and efficiency made it a natural first model to target. The implementation was fairly straightforward, and the initial results showed that the R^2 metric would likely not be the most adequate way of measuring the different model accuracies. After studying on different available metrics and their characteristics, the Mean Absolute Error metric was chosen as the main reference point, and the Mean Squared Error metric was also included on the results as a way of further comparing the different model performances. One other issue that arose was the fact that not all features matter for all positions, which was expected. In order to solve this problem the different means for all features were taken, and a cutoff of 0.01 was made in order to remove the irrelevant features for each position. Important to note regarding this strategy, using the whole training set (2014-2016) to take the means implies some data-leakage as this information is obtained outside the K-Fold training splits, however, this was a simple way of programatically removing irrelevant features and the low cutoff value was understood to be reasonable.

After the initial results were obtained, some different models were chosen to see how their metrics compared; SVR, XGBoost, Random Forest, and MLP were tested without any hyper-parameter tuning. Given the repetition in different implementations and lengthiness of tests, as a different model had to be constructed for each position, a structure was created to support the different steps. After the initial results came out satisfactory, the different refinement strategies were structured. For each model, some initial fairly random values were tested, after these preliminary results, the

hyper-parameter values were narrowed down and a final set was consolidated. All of the model MAE results were improved after the different hyper-parameter refinements.

One note regarding the Random Forest implementation, as previously discussed, using MAE as a scoring metric for the cross-validation and even the fitting of the model drastically increases the time it takes for it to process, as compared to MSE. This huge increase in computational cost for the model made it simply impractical, therefore the MSE scoring metric was used instead.

Results/Validation

Following the training and refinement process of all models, 30 in total, 54 including the baseline implementations, the results for chosen metrics were all recorded and compared. This gave way to the formulation of a combination of the models that best performed for each position. With these models and results at hand a final average score was finally obtained. The results confirmed the fact that a good part of the variance of the target variable is not able to be explained by even the best performing models. This in turn results in the predicted notion that the best performing models not necessarily would correspond to the best scoring for the 2017 season. However, the models, especially when compared to the Linear Regression benchmark implementation showed that the increased performance of the models did in fact tend to lead to higher average scores for the season, thus, further improvements of these metrics would likely lead to even better final scoring averages.

In order to test for the best performing models' robustness, a Sensitivity Analysis was constructed. 100 different test-training splits were created and for each of these models, they were trained and used to predict the test target variables that were set aside. These results were then averaged out and compared to the original results obtained. The similarity between the values successfully demonstrated the robustness of each model.

Improvement

As discussed throughout the paper, there are several points that could be improved which ultimately would lead to better performing models and higher average CartolaFC season scores.

Data

Machine Learning algorithms are almost always very directly constrained to the quality of the data that accompanies it. For this project, unfortunately, only a limited number of features were available as open-source, and this heavily constrains the overall performance of the models. No matter how much the models are tuned and refined, the limited features only allow so much leeway. There are other datasets available online that could very likely improve the model performances for further iterations of this kind of project.

There is further data cleaning that could be done to the data used, for example, identifying players that came from the bench or were subbed before the end of the match, or even at what time a red card was given.

The feature engineering part of the project was chosen based on basic domain knowledge and some general personal intuition, different strategies could be chosen which could evidently improve upon the performance established for this project.

The feature selection step of this project was very simple and fairly superficial, some more idiosyncratic feature pruning for each position could certainly improve the models even more.

Models

Although a decent variety of models were attempted, there are many other types of algorithms that could be tried. One special type I would consider, which showed some initial promising results, are different Neural Network configurations. There are various different refinement possibilities available both parameter and architecture wise. Delving deeply into Neural Network formulations seems to me like very interesting improvement possibilities.

For refinement, only a subset of the possibilities were attempted, different combinations could be tried and some more fine tuning would certainly bring improvements to the models.

General

Different approaches could also be attempted, for example, 2017 season data was not used at all in the model training, when it seems reasonable to assume that, as rounds progress for the season, the previous rounds will help in the score predictions.

One other interesting approach that has been used for fantasy league Machine Learning studies is the idea of constructing a reinforcement learning algorithm that

iterates through the different seasons “playing” the game and learning what choices taken improve the final results and applying this algorithm to an unseen season.

Finally, one interesting extrapolation of this project would be to construct an app or website that would make predictions for each round and allow for different configurations.[26]

[26] Feel free to use this implementation as a reference for other projects. Reach out at arturcp@gmail.com for any questions regarding the project, different implementations, and results.