Machine Learning Engineer Nanodegree

Capstone Project

Nikolas Mesquita

Jun 21th, 2019

I. Definition

Project Overview

Electronic games move the billionaire market reaching 90 billion dollars in 2018 (Source). Industry giants like Sony, Nintendo, Microsoft and now Google make several monthly releases and of course the vast majority of these launches use at least one engine that employs the use of artificial intelligence techniques.A very successful video game is FIFA which is a very popular soccer simulator and sales leader over the years.FIFA Ultimate Team is a modality within the game where there are cards with the individual skills of the players who add up their overall maximum. When you receive these cards they are either won or bought with real or virtual money, you can set up your team (as if it were a coach) and play games that receive more cards or virtual cash prizes.

As electronic games are my favorite hobby and game since my 6 years of age, both in the virtual world and in the real world I am a soccer fan so I thought about doing this study (avoiding one of the problems of the corporate world which I work daily).

There are some studies about this background as follows:

Analyzing and Predicting European Soccer Match Outcomes
The ultimate Soccer database for data analysis and machine learning
Fifa 19 (Machine Learning)

Problem Statement

The FIFA Ultimate Team game is contained within FIFA 19. It consists of several types of cards that are bought or won at your club. For this capstone project, we will take a look at only the player cards Here are some examples of how these cards look like:

Player cards are categorized by feature quality, which can be bronze, silver and gold.

For each of these 3 categories rarity is included. In this way there are 6 possibilities.

- Bronze
- Bronze - Rare
- Silver
- Silver - Rare
- Gold
- Gold - Rare

At first sight, there are 6 features that are supposed to be the most important attributes of a player:

- PAC = pace
- SHO = shooting
- PAS = passing
- DRI = dribbling

- DEF = defending
- PHY = physicality

But if you do a simple verification the overall number cannot be measured only with this attributes:

`(88 + 91 + 88 + 96 + 32 + 61) / 6 = 76 != 94`

So how we can predict this target feature? (overall)

Reference: https://www.futbin.com/

The basic idea is to analyze a set of skills and other players information in order to predict their “overall” feature value. As this feature has a discrete variable characteristics (amounts or values) so we have/ a Regression Predictive Modeling problem (Supervised Learning).

Above are the steps that I took to complete the capstone project:

Data Exploration – I explored how the data was distributed by finding the min, max, median, mean and visualized skew data on a histogram chart. The correlated data also was discovered and visualized.

Data Cleaning – Each feature was explored and after that was determined how to manage outliers and missing data.

> Outliers: For some was used transformations, for other was applied some kind of filter. In this dataset there some interesting outliers like the Idols Players (category = Idols)
>
> Missing data: I removed some missing data when there’s no meaning for the prediction, or when will be harmful for the prediction. For this data set, where the players are 1/11 (relation among GK position and the others) there was a several missing data for GK position so i decide to clear this position from the data set.

Prepare data – the data was randomly split into train and test sets.

Feature treatments – for the categorical variables, one-hot encoding was used.

Metrics

For these sort of problem, regression on a supervised learning i chose the follow evaluation metrics:

Mean Absolute Error(MAE) is the average of the absolute differences between predictions and actual values. It gives an idea of how wrong the predictions were. It gives an idea of the magnitude of the error, but nothing about value over or under predicting.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - y_j|$$

Mean Squared Error(MSE) pretty similar MAE in that it provides only a gross idea of error magnitude. The difference lies in the conversion of units back to the original for the output variable so it can be more meaningful for data visualization. This measure is called the Root Mean Squared Error (RMSE).

$$MSE = \frac{1}{N}\sum_{i}^{n}(Y_i - y_i)^2$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

R^2 (R Squared) coefficient of determination: this measure provides an indication of how good was the fit of a set of predictions to the actual values. This is a value between 0 and 1 for no-fit and perfect fit respectively.

$$SQ_{\mathrm{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2,$$

Reference: [Metrics To Evaluate Machine Learning Algorithms in Python]

II. Analysis

Data Exploration

The dataset for this project was obtained from Kaggle and after reading the .csv, was loaded the data that contained 94 features,1 target variable and 18.831 observations

The goal for this project was to predict overall feature and to do that I explored all give features verifying:

Correlation, missing and zeros values and statistics metrics such as min, max, median, mean and skewness:

	age	pace	shooting	passing	dribbling	defending	physicality
count	18831.000000	16882.000000	16882.000000	16882.000000	16882.000000	16882.000000	16882.000000
mean	26.364027	69.072859	54.609762	59.218398	64.323540	52.701339	66.620187
std	5.223632	11.909037	15.107605	11.419965	10.906452	16.722049	9.684565
min	17.000000	24.000000	15.000000	24.000000	24.000000	15.000000	30.000000
25%	23.000000	62.000000	44.000000	52.000000	58.000000	38.000000	61.000000
50%	26.000000	70.000000	56.000000	59.000000	65.000000	57.000000	68.000000
75%	29.000000	77.000000	65.000000	67.000000	71.000000	65.000000	73.000000
max	89.000000	99.000000	99.000000	99.000000	99.000000	98.000000	97.000000

	overall
count	14396.000000
mean	67.061892
std	6.736791
min	47.000000
25%	63.000000
50%	67.000000
75%	71.000000
max	94.000000

Traits and Specialities features

These features has a peculiarity as we can see below:

```
data.traits.head()
```

```
0                                          Finesse Shot
1    Avoids Using Weaker Foot, Finesse Shot, Flair,...
2           Tries To Beat Defensive Line, Finesse Shot
3                                          Finesse Shot
4    Avoids Using Weaker Foot, Finesse Shot, Flair,...
Name: traits, dtype: object
```

```
data.specialities.head()
```

```
0    Speedster, Aerial Threat, Dribbler, Play Maker...
1    Speedster, Dribbler, Play Maker, Distance Shoo...
2    Speedster, Dribbler, Distance Shooter, FK Spec...
3    Speedster, Dribbler, Distance Shooter, Crosser...
4    Dribbler, Play Maker, Distance Shooter, Crosse...
Name: specialities, dtype: object
```

As we can see, for features traits and specialities both have some information that can be useful for the model but this information isn't in a good stand to be processed. This information shows some information about player's characteristics.

For this case, I created a special function (that separates the domain of a given information feature and after that generates a hot-encoding (dummy) for each new feature.

```
# feature 'traits' hot encoding
data = utl.hot_encoding(data, 'traits')
```

As a result we got:

For the feature traits was created 36 new columns on dataset, and for the feature specialities.

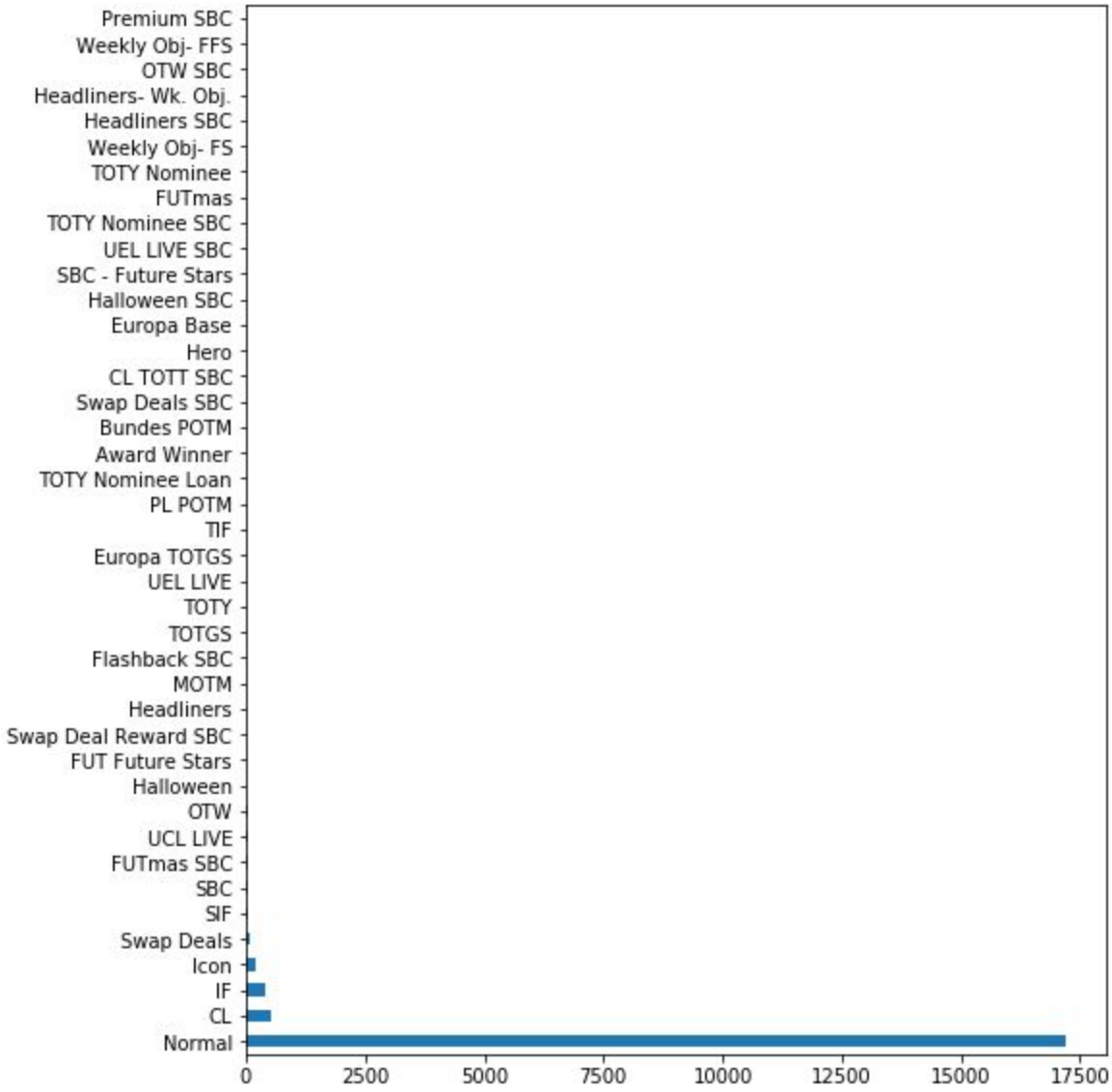
Example on the dataset:

```
data.head(1)
```

Finesse Shot	Avoids Using Weaker Foot	Flair	Takes Finesse Free Kicks	Tries To Beat Defensive Line	Team Player	Puncher	Power Free-Kick	Power Header	Speed Dribbler (CPU AI Only)	Playmaker (CPU AI Only)	Chip Shot (CPU AI Only)	Outside Foot Shot	Technical Dribbler (CPU AI Only)	Set Play Specialist	Solid Player
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Revision feature

Revision are upgrades made on cards such as best players on UCL or Icons that are retired legends of soccer such as Pelé, Maradona among others. But when we take a look at the data set the distribution are very skewed:



So i decided to filter only the normal revision

```
data = data.query('revision == "Normal"')
```

Duplicates

I found some duplicates on the data set that was cleaned.

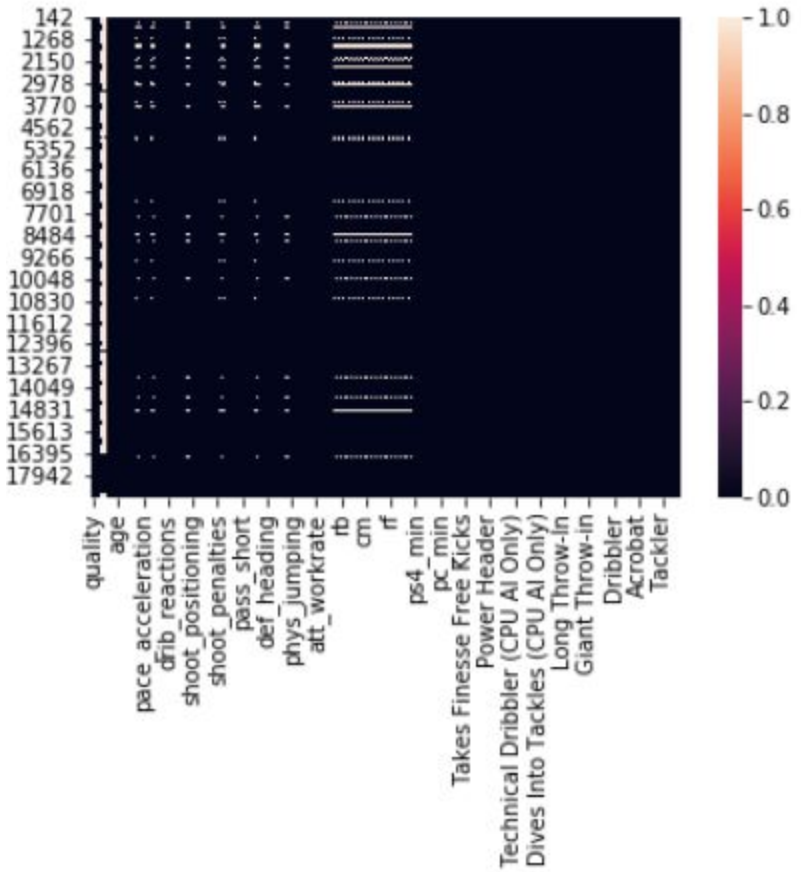
Null or NaN Values

I found some Null or NaN values. This values and its treatment will be demonstrated on the next section.

Exploratory Visualization

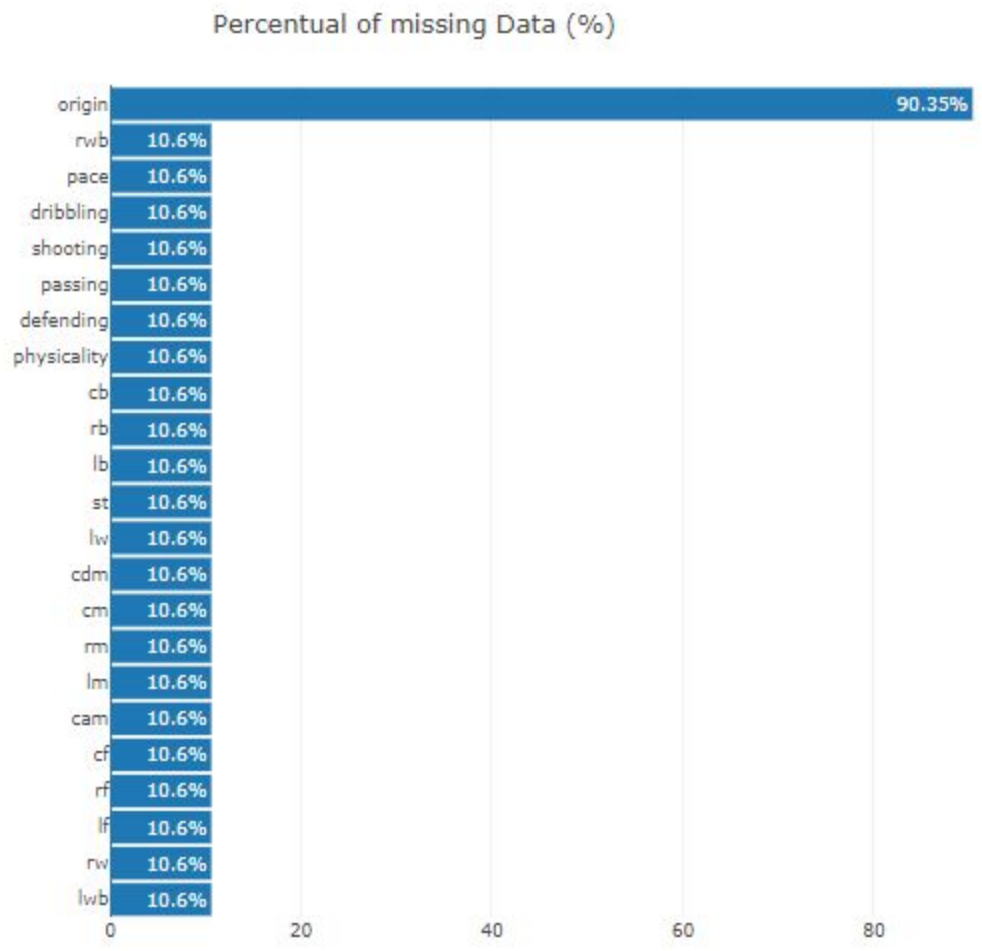
Missing Values

Percentage of missing data by a HeatMap

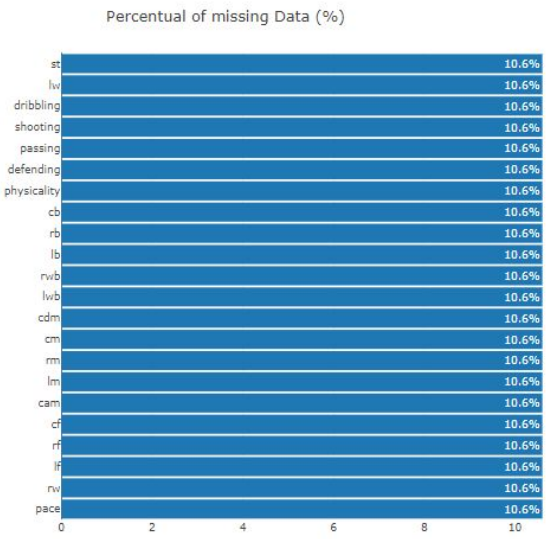


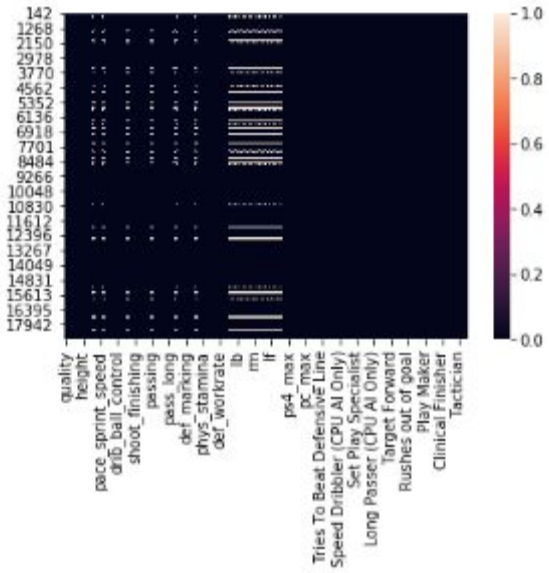
The white dots on the black background indicates missing values.

Percentage of missing data by a bar plot



The feature origin have a lot of missing values, so it was removed from the dataset by the rule 30% os missing values drop rate. After this process we can see the plots again:





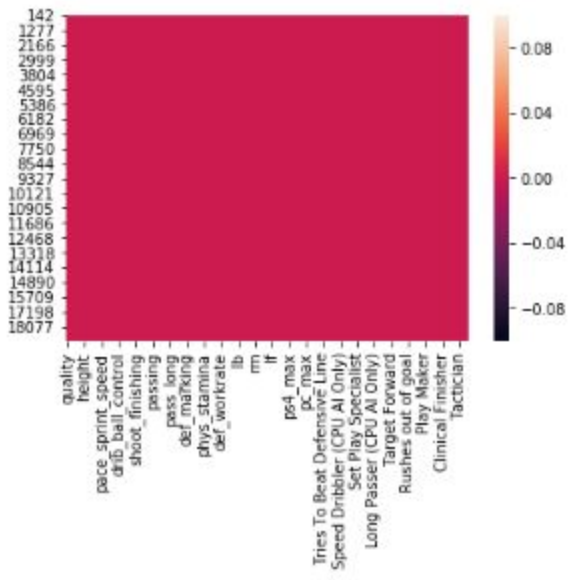
But as we can see we still have some missing features. Analyzing the data, I found a possible issue that occurs on the data set:

- DIV = dive
- HAN = handling
- KIC = kicking
- REF = reflex
- SPE = speed
- POS = position

- PAC = pace
- SHO = shooting
- PAS = passing
- DRI = dribbling
- DEF = defending
- PHY = physicality

As we can see, for the position Goalkeeper we don't have some features such as: pace, dribbling, shooting among other, in this case we need to do a separate analysis, so to not impact this analyst i decide to drop the position GK

After the drop of position GK, our data set are free of missing values



Outliers

Because of the feature revision, some players could be duplicated, in the following example we could se 3 versions for Messi, normal version is the golden one.

And below we can see 3 cards from Icons revision:

The 6 base attributes are so high and this sort of card are so unlikely to appear in the game that I classified it as an outlier and I removed it from the data set. Another characteristic that could generate some mistake is the duplicate players in different revisions.

Feature deletion

Meaningless features, such as Id, which indicated the record number in the dataset, should be removed from the dataset before feeding into the models. If the null data of a given feature is more than 30% will be deleted also.

Meaningless features

Exclusion of the following features due meaningless:

Dropped variables:

['player_id','player_name', 'player_extended_name', 'club', 'league', 'nationality', 'date_of_birth' ,
'added_date','base_id','resource_id','ps4_last','ps4_prp','xbox_prp',
'pc_last','pc_prp','GK Up for Corners','GK Long Throw','Sweeper Keeper','Saves with Feet', 'gk_diving',
'gk_reflexes','gk_handling','gk_speed','gk_kicking','gk_positoning','ps4_min',
'No traits','No specialities','revision','origin', 'intl_rep','ps4_max', 'xbox_last',
'Xbox_min','xbox_max','pc_min','pc_max']

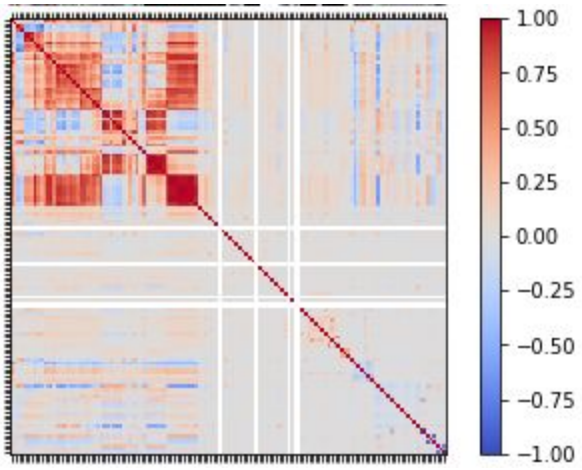
Drop Highly Correlated Features

When we have 2 or more highly correlated variables is a good choice to drop then because for the model it could consider the 2 variables carrying the same information or get confusing for a possible association with a third variable.

Drop the follow features due the high level of correlation:

Dropped variables: ['pace_acceleration', 'pace_sprint_speed', 'drib_ball_control', 'drib_dribbling', 'shoot_finishing', 'shoot_long_shots', 'pass_short', 'def_interceptions', 'def_marking', 'def_stand_tackle', 'def_slid_tackle', 'cb', 'rb', 'lb', 'rwb', 'lwb', 'cdm', 'cm', 'rm', 'lm', 'cam', 'cf', 'rf', 'lf', 'rw', 'lw', 'st', 'ps4_max', 'xbox_last', 'xbox_min', 'xbox_max', 'pc_min', 'pc_max', 'pref_foot_Right']

Below is the correlation matrix before treatment:



Algorithms and Techniques

There are four types of machine learning algorithms: supervised, semi-supervised, unsupervised and reinforcement. For this project we need to choose one algorithm capable of predicting the overall of a player given its attributes. As we have a label or dependant variable (overall feature) and independent variables (other attributes of the player), by this problem characteristics we need a solution that implement a Supervised Learning algorithm. For supervised learning we have 3 segmentations: classification, regression and forecasting. When we must estimate and understand the relationship among variables to predict the label we must use a regression algorithm. In this case we have some famous algorithms such as: decision trees, random forest, support vector machine regression and and linear regression. I chose the last one because the characteristics of the dataset: the certainty of the label, the independence of the variables and the data normality indicates to use it. Also linear regression is simple and it's easy to use - I prefer to initiate a solution for a problem for the more simple to more complex; for instance I dont see for this problem, the need to use a sophisticated deep learning algorithm at first sight!

Before applying the machine learning algorithm I'll use Pandas and Numpy to gain some understanding of the data, so I will try to get some new features based on the given features in order to train the model. For the model I will try Linear Regression Supervised Model.

In this way, I will do an overall players feature analysis, where I will randomly remove some overall data from the original data set among other analysis. These indices will

serve as a benchmark for model performance when compared to the predicted outcome.

Types of machine learning algorithms you should know
Machine learning algorithms
Machine learning algorithm use

Benchmark

I chose a supervised learning algorithm to solve the problem, for benchmark i will use a naive bayes predictor, and i will try to beat this benchmark model with my solution. So i'll train and test this benchmark model using the exact same data and conditions that I'll use for my solution.

```
# Counting the ones as this is the naive case. Note that 'overall' is the 'overall_raw' data
#encoded to numerical values done in the data preprocessing step.
TP = np.sum(overall)

# Specific to the naive case
FP = overall.count() - TP

TN = 0 # No predicted negatives in the naive case
FN = 0 # No predicted negatives in the naive case

accuracy = float(TP)/(TP+FP)
recall = float(TP)/(TP+FN)
precision = accuracy

fscore = (1+0.5**2)*(precision*recall)/(0.5**2*precision+recall)

# Print the results
print("Naive Predictor: [Accuracy score: {:.4f}, F-score: {:.4f}]".format(accuracy, fscore))
```

The results for the Naive Predictor was:
Naive Predictor: [Accuracy score: 66.9915, F-score: 4.7183]

III. Methodology

Data Preprocessing

For the linear regression algorithm we need some preprocessing steps. The follow steps were taken on the dataset:

“Special” one-hot encoding: the standard Pandas .get_dummies function was unable to treat the feature specialities and traits as I desired so I decided to create this function and I gave this name “special” because this treatment.

Duplicates: there was some duplicated rows on the dataset that need to be removed.

Null and NaN values: there was some Null and NaN values on the dataset that could implies on a bad algorithm performance.

Outliers: due some characteristics on the revision feature, some outliers was occurring on the data set. This also could imply bad algorithm performance.

Feature deletion: I decided to remove some meaningless features and high correlation features that could implies on a bad algorithm performance.

Hot encoding: after all steps above, I ran the last hot encoding pd.get_dummies(data)

All this preprocessing steps was fully discussed on the previous sections, so here i just remembered it. Additional, I executed some normalizing functions as follow:

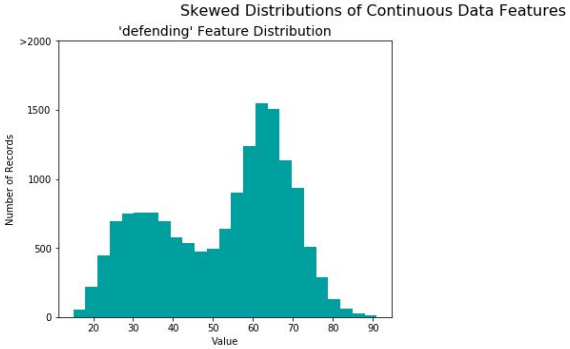
Normalizing Numerical Features

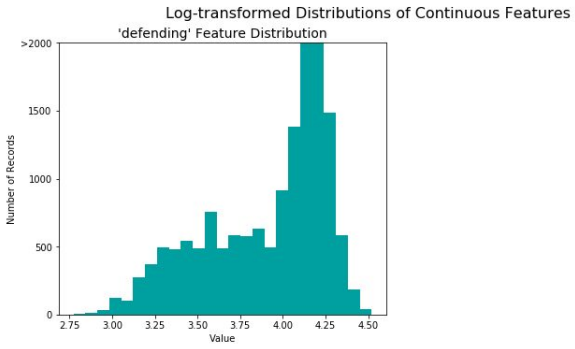
Min Max

It shrinks the range to a value between 0 and 1. The value will be -1 if there are negative values. When the standard scaler don't have a good outcome this scaler could be used. Be aware to outliers, its sensibel for then.

Log-transform

Skewed data is common. It's often desirable to transform skewed converting it into values between 0 and 1. Below we can for instance the feature defending before and after the log-transform application.





Implementation

The solution was supervised model capable of predicting the overall of a player given its attributes. I'll use Pandas and Numpy to gain some understanding of the data, so I will try to get some new features based on the given features in order to train the model.

The Linear Regression Predictor was used on the data set on a customized function as follows:

```
mdl.model_execution(features_final, overall)
```

```
def model_execution(X, y):

    X_train, X_test, y_train, y_test = training_testing(X,y)
    features_x_coefficients, lm = fit_model(X_train, X_test, y_train, y_test)
    error_valuation(X_test,y_test,lm)
    model_viz(X_train, y_train, X_test, y_test, lm)
```

There are 3 functions called: fit_model, error_valuation and model_viz that generates the metrics above:

Training set has 10708 samples.
Testing set has 4590 samples.
Estimated intercept coefficient -11936875152095.975
Number of coefficients: 103
coefficient of determination: -5378563791130.69

The error was:

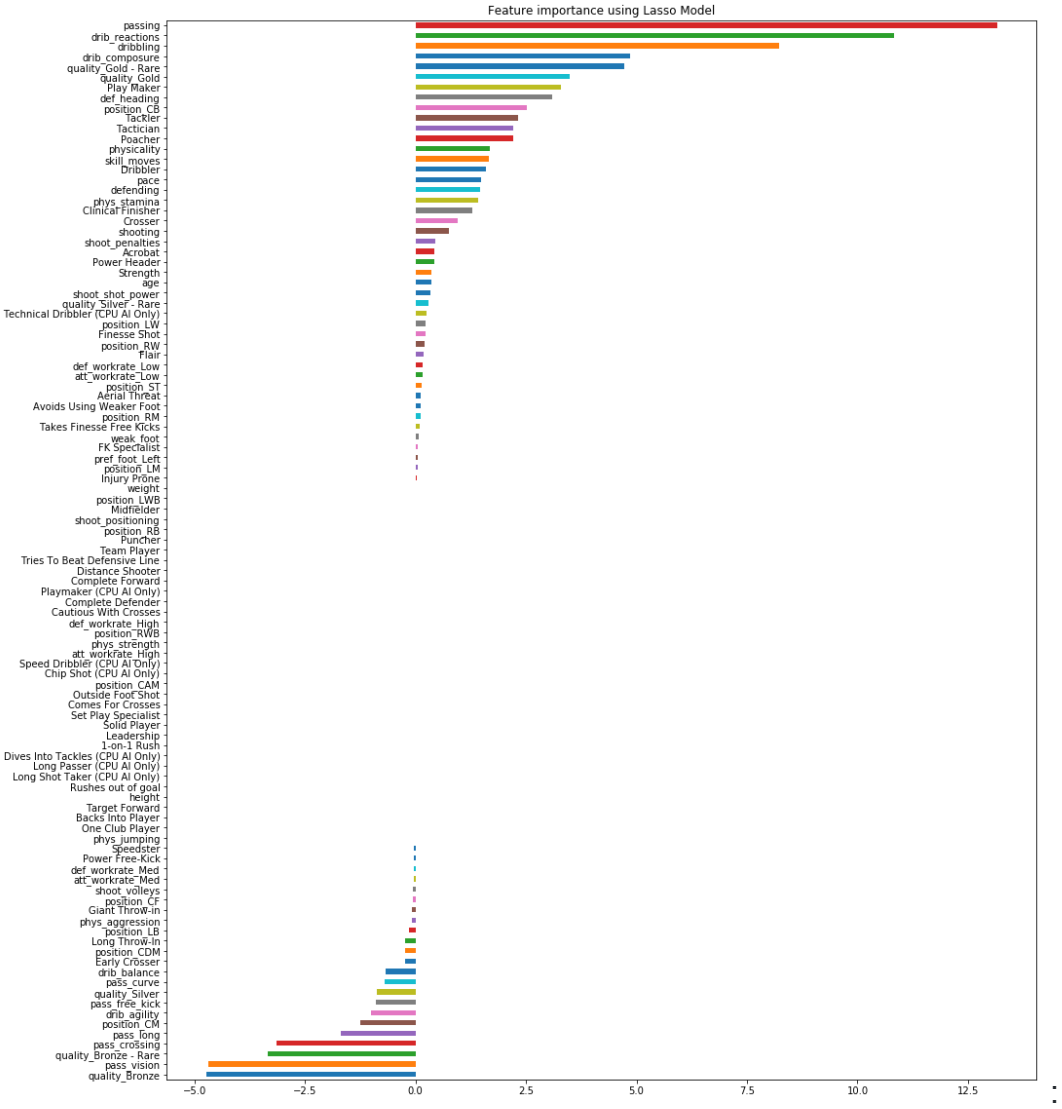
Mean Absolute Error MAE: 227924.18116489652
Mean Squared Error MSE: 238444937442912.16
R Squared: 15441662.392466433

Refinement

For the first execution of the algorithm was necessary 103 features getting a weird value of coefficient of determination (-5378563791130.69)

Its looks like that there are a lot of features for algorithm's understanding, so I used the Lasso and RFE - Recursive Feature Elimination technique to reduce the number of features:

Below, a graphic visualization of results using Lasso technique:



We can see on the middle of the chart, the features that we need to remove.

As we can see when the RFE algorithm hits 57 features its performance is almost the same as with 103 features. A negative side of this solution is the poor time performance

because we need to iterate feature by feature until get the sweet point, in this case with 57 features.

46	Dives Into Tackles (CPU AI Only)	47	46	0.928487	Solid Player, Avoids Using Weaker Foot, pace, ...
47	Long Passer (CPU AI Only)	48	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
48	Long Shot Taker (CPU AI Only)	49	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
49	Early Crosser	50	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
50	Injury Prone	51	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
51	Long Throw-In	52	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
52	Target Forward	53	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
53	Backs Into Player	54	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
54	One Club Player	55	47	0.928506	Solid Player, Avoids Using Weaker Foot, pace, ...
55	Cautious With Crosses	56	47	0.928506	Solid Player, One Club Player, Avoids Using We...
56	Giant Throw-in	57	47	0.928506	Solid Player, One Club Player, Avoids Using We...
57	Rushes out of goal	58	57	0.928722	Solid Player, One Club Player, Avoids Using We...
58	Comes For Crosses	59	57	0.928722	Solid Player, One Club Player, Avoids Using We...

IV. Results

Model Evaluation and Validation

The final model is reasonable and aligning with the solution expectation of predict the overall label, mainly when i used feature importance techniques: Lasso and RFE. The linear regression model was tested 4 times with different inputs - features and observations to guarantee its robustness even with 20% of data deletion. With this results presented below we can see that solution is trustworth:

1.Fully treated data set, with 103 Features.

2. Fully data set after Feature Importance using Lasso technique, with features reduction to 69.

3. Fully data set after Feature Importance using RFE, with features reduction to 68.

4. Data set with 20% of rows aleatory deletion (Thanos function) with 69 features. Was created a function called “Thanos” with the goal of doing some small perturbations on original data set by aleatory row deletion.

3 of 4 executions presented stable results, where using “Lasso” presented best time performance. The worst result was the first execution with no feature importance.

Justification

As a benchmarking for the project algorithm implementation, Naive Predictor was used with the following results:

Naive Predictor: [Accuracy score: 66.9915, F-score: 4.7183]

After Lasso or RFE techniques, the model’s performance was improved and now are better than benchmarking. As we can see it has a satisfactory result:

First execution Linear Regression :

Observations:

Training set had 10077 samples.
Testing set had 4319 samples.
103 Features.

Results:

coefficient of determination: -5378563791130.69
Mean Absolute Error MAE: 227924.18116489652
Mean Squared Error MSE: 238444937442912.16
R Squared: 15441662.392466433

**Without feature importance, the Naive Predictor was fairly better.*

Second execution - After Lasso, was removed 34 features.

Observations:

Training set has 10077 samples.
Testing set has 4319 samples.
69 Features.

Results:

coefficient of determination: 0.9258851400408786
Mean Absolute Error MAE: 1.422128634982639
Mean Squared Error MSE: 3.285693696834358
R Squared: 1.8126482551323513

Third execution - After RFE was removed 35 features.

Observations:

Training set has 10708 samples.
Testing set has 4590 samples.
68 Features.

Results:

coefficient of determination: 0.9308815095484677
Mean Absolute Error MAE: 1.3710499364211424
Mean Squared Error MSE: 3.064192370282628
R Squared: 1.7504834675833496

Forth execution - Using Thanos function (aleatory deletion).

Observations:

Training set has 8596 samples.
Testing set has 3685 samples.
69 Features.

Results:

coefficient of determination: 0.9278473331588036
Mean Absolute Error MAE: 1.3980833138992537
Mean Squared Error MSE: 3.15515674186044
R Squared: 1.7762760882983366

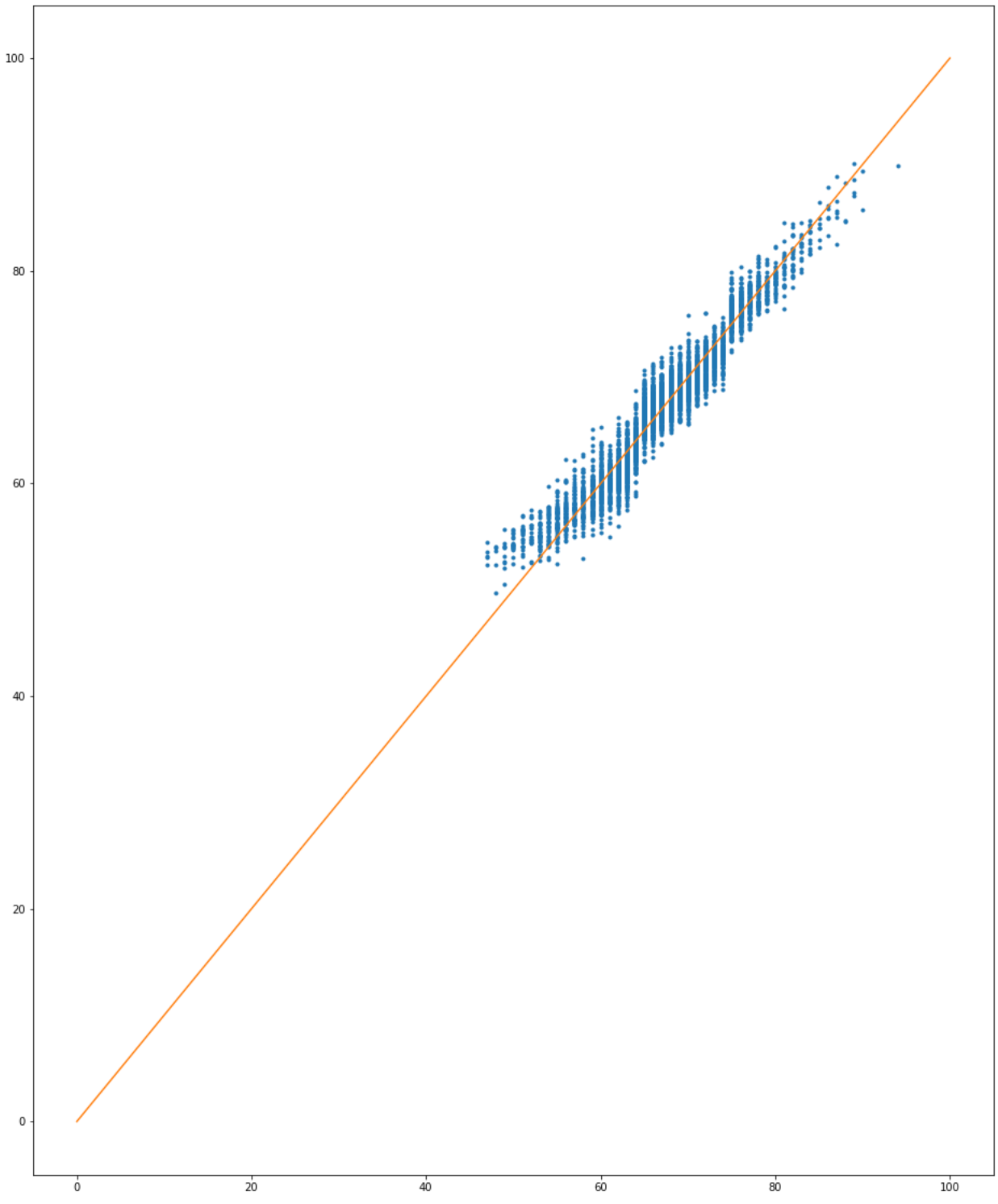
As we can see by its results the linear regression algorithm is simple but robust solution for regression problems. For this specific this project with this results, with a coefficient of determination of 93% using RFE seems a good prediction for an overall of a player to be used in a Fifa Ultimate Team videogame. But this kind of solution, where we want to predict a variable from another variables for sure could be used in several other problems that involves estimate and understand the relationship among variables to

predict the label such as: stock option, customer churn, even for a real soccer players overall!

V. Conclusion

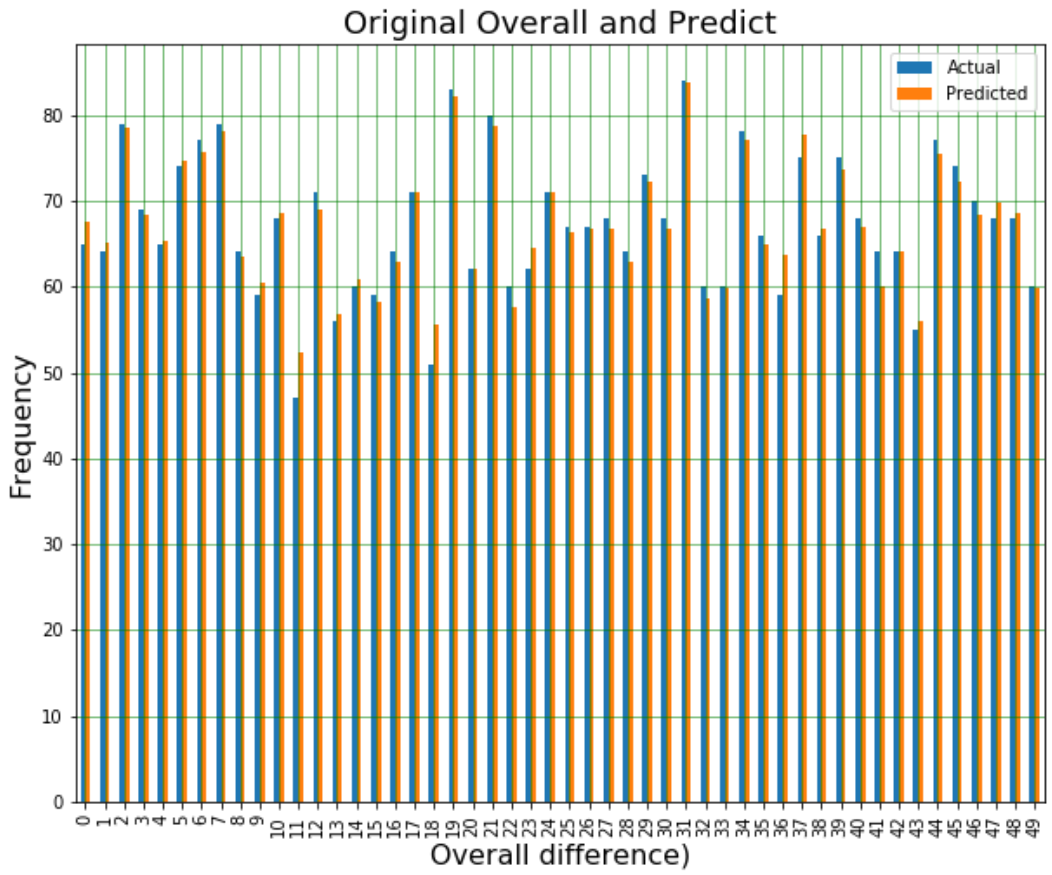
Free-Form Visualization

As the Linear Regression Algorithm has an output, the predicted label, here we can see how its look like after the feature importance RFE technique execution. On this graph clearly that we can see the distribution of the observations (blue dots) over the orange line (the estimate regression line) that has the equation $f(x) = b_0 + b_1 x$, or in other words the predicted values. The gap between the lines and the dots is called "residuals" and we will see it soon below.



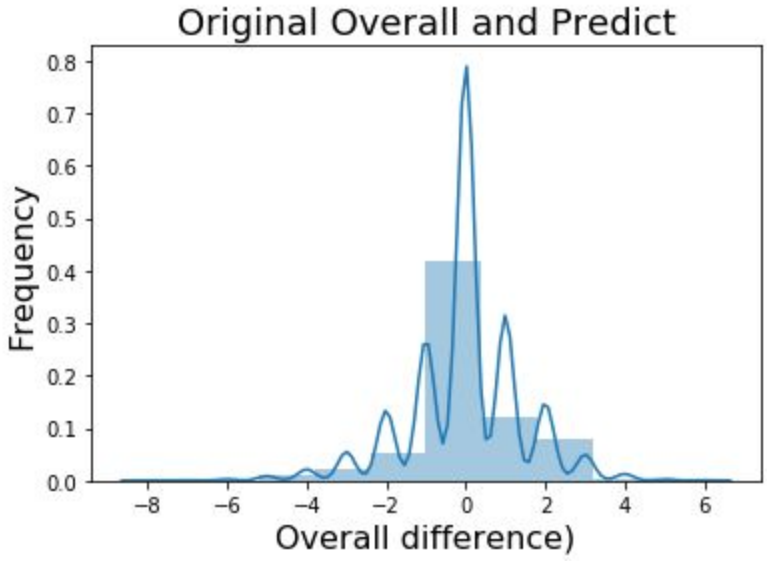
Above, a different view of the same concept, where the values for "actual" are the observations on the data set and "predicted" the results of the algorithm

execution. Both values are really pretty close what confirm the 93% of the coefficient of determination:

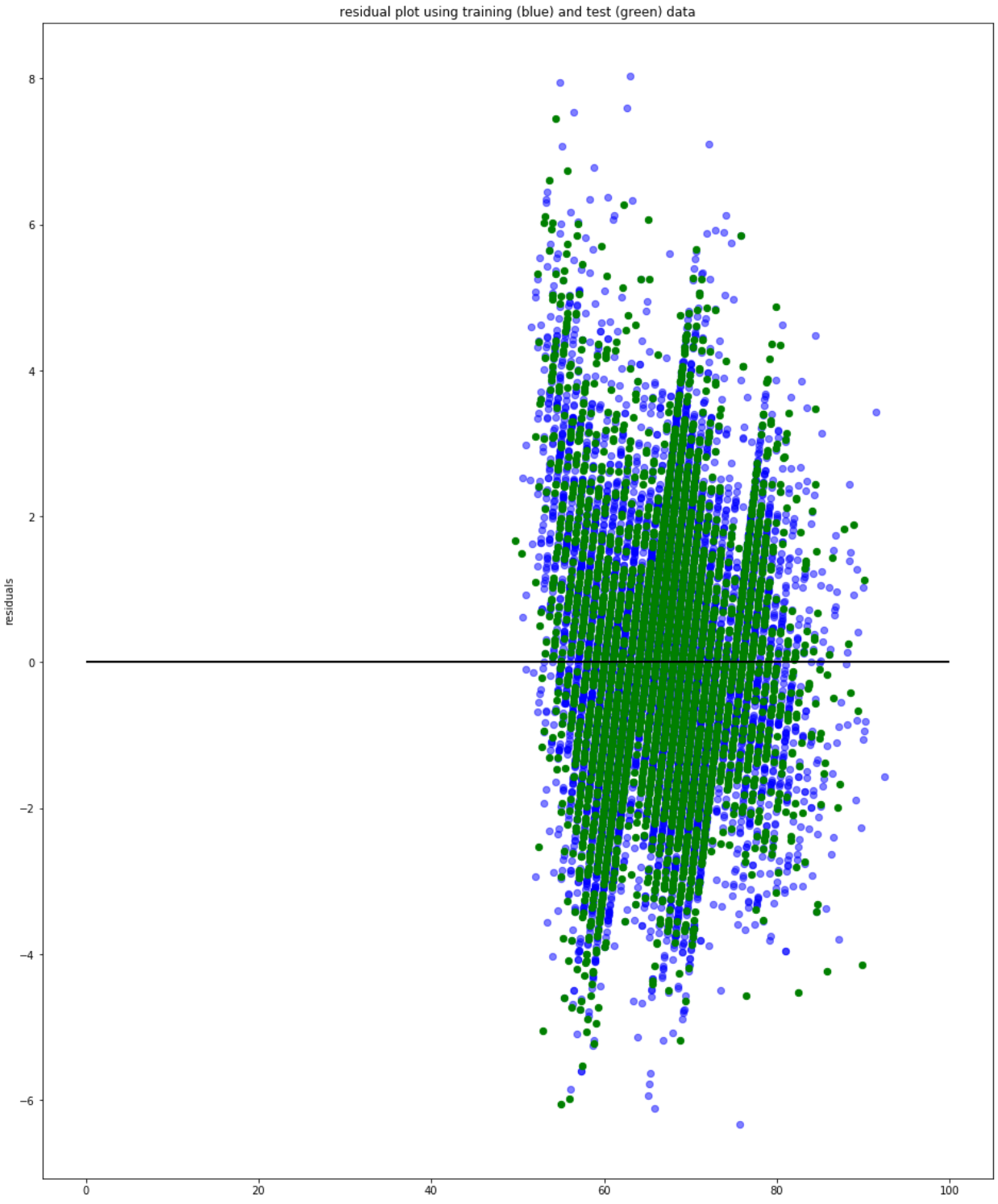


Residuals Visualization

What we can see here is the difference between the original overall feature (observation) against the predicted value generated by our linear regression solution. De um modo geral, is a very good result because the great amount difference lies among -2 to 3 points, ou seja if a player has an overall of 80 the algorithm predicted 78 until 83. Of course there is bad cases like wrong predictions of -8 or 6, but this occurred less on prediction result.



For this graph, I follow the same logic as explained above but here we can see on a different visualization where the blue dots it's about training data and the green dots about testing data. When we had a good prediction quality we can see on the graph a big concentration of dots where indicates a better algorithm prediction.



What we want to see here is a plot with correlations of zeros. But, the outcome of the algorithm shows that still have some correlations between residuals

training and testing. This behavior, it’s normal, because the prediction will be never with 100 percent certainty.

Reflection

Analyzing backwards the project by stages:

1. Data Exploration: interpreting and understanding the data by it’s features and values are really important for modeling the best solution. Data visualization is the best way to find relationship on data.

2. Data Cleaning: dataset understanding in this stage is critical for the next stages. After this understanding we can choose the correct tools to minimize null and duplicate values.

3. Feature engineering: decisions about which features to drop, to create or enhance will be the difference on the model prediction quality. For this project One-hot encoding was used widely and the features created in this project helped highly for the prediction quality.

4. Model execution: Linear Regression is a simple but robust algorithm and for this project fits very well in beating up Naive Predictor. But there is some room to evolution.

I learned with this project that "less is more", strange, but this statement makes a good sense since in my data analysis first steps I created a lot of features, but the execution with all of then brings me a poor algorithm outcome, resulting in a big challenge to identify which features should I needed to drop in order to improve the algorithm output. For this I researched many techniques and found on Lasso and RFE a good choice for my issue of feature importance. Another thing that made a difference was the use of the one hot encoding for some features, otherwise, without these kind of technique the feature should be dropped because its non numeric characteristic.

Improvement

I didn't use all tools for this project. I would try more possibilities in use different regression models on the data set. Another improvement could be made is pick up a different target variable, for instance the value or cost of a player and data sets of different years. Another example could be a clusterization of players. For the step of Feature Engineering, an execution of a PCA algorithm probability could worth it. A good challenge will be use this solution as a new benchmarking!