Data Analytics in Moneyball Theory: the Story of a Winning Team

Zhenghan Zhang[1]

[1]**NYU Shanghai,1555 Century Avenue,Shanghai**

ABSTRACT

Since the beginning of team sports, people have been living with a fantasy of a ”dream team”. More specifically, a team composing of players who can maximize the team’s performance would naturally gain an advantage against other teams. In this paper, I would try to identify what makes a player great in soccer using data analytic tools. I would also try to develop more insight on a managerial level, to identify which players are more ”worthy”.

Keywords: Moneyball, Data, FIFA

INTRODUCTION

The famous ”Moneyball Theory” refers to a revolutionary thought originated from professional baseball. In the 1990s, Billy Beane, the general manager of the Oakland Athletics, wnet against professional scouts and build a team of ”undervalued” talents using analytic tools. Using the statistics, he built a seemingly losing team and turned them into winners.(2) Here, I would like to examine the theory using FIFA 2019 data. Specifically, I would look into the statistics of players, and try to develop visualizations of player metrics and valuation of players.

DATA ACQUISITION AND CLEAN-UP

I used the data collected from the FIFA 2019 official dataset(amanthedorkknight), and load the data using Pandas package. All coding has been done using Jupyter Notebook.

```
import pandas as pd
```

```
import numpy as np

import matplotlib.pyplot as plt


dataset = pd.read_csv("data.csv")

dataset.head()

dataset.shape

dataset.info()
```

The codes above show us that the dataset contains 18207 entries and 89 columns. The question is which dimensions to keep. For our purposes, we assume that a player does not change positions, at least in the near future. Therefore we eliminate all columns regarding the player's performance in different positions.(As is shown in figure 1). Also, we need to convert the height and weight metrics. Height used empirical metrics, and follows a form like ”5’7”, therefore we need to split the object into two integers, and then translate them into centimeters using the formula $CM = 30.48 * Feet + 2.54 * Inch$. For weight, we first truncate the ”lbs” string, convert into integers, and then translate it into kilograms using the formula $KG = 0.45359 * lbs$. Furthermore, I would like to avoid having NA or NaN in the dataset. I omitted these variables after excluding the dataset of irrelevant columns. The following code serves this purpose.

```
#Drop one column from the dataset
#@param: df : dataframe to edit
#      String: the label
#@return: the edited dataframe
def dataDrop(df,String):
   x = df[df.columns.drop(String)]
   return x


#preprocess the data
#change this method if want to change parameters
#@param: df : dataframe to edit
```

```
#@return: the edited dataframe
def eliminateColumns(df,lst):
   for i in lst:
      df = dataDrop(df,i)
   return df


#process the data, and drop NAs
eliminate = ['LS','ST','RS','LW','LF','CF','RF','RW','LAM','CAM','RAM','LM',
'LCM','CM','RCM','RM','LWB','LDM','CDM','RDM','RWB','LB','LCB',
          'CB','RCB','RB','Special','International Reputation',
          'Work Rate','Body Type','Real Face','Joined','Loaned From',
          'Contract Valid Until','Release Clause','Flag','Nationality','Club
              Logo]
dataset_new = eliminateColumns(dataset,eliminate).dropna()
# Clean up the weight column
dataset_new['WeightKG'] =
    dataset_new['Weight'].str.rstrip(to_strip='lbs').astype('float64') *
    0.45359
dataset_new = dataDrop(dataset_new,'Weight')
# Clean up the Height column
temp = dataset_new['Height'].str.split('\'',expand=True).astype('float64')
dataset_new['HeightCM'] = temp[0] * 30.48 + temp[1] * 2.54
dataset_new = dataDrop(dataset_new,'Height')
dataset_new.info()
```

The new dataset contains 17918 entries and 51 columns, some metrics are important intuitively (As we see in figure 2)

LS	16122 non-null	object
ST	16122 non-null	object
RS	16122 non-null	object
LW	16122 non-null	object
LF	16122 non-null	object
CF	16122 non-null	object
RF	16122 non-null	object
RW	16122 non-null	object
LAM	16122 non-null	object
CAM	16122 non-null	object
RAM	16122 non-null	object
LM	16122 non-null	object
LCM	16122 non-null	object
CM	16122 non-null	object
RCM	16122 non-null	object
RM	16122 non-null	object
LWB	16122 non-null	object
LDM	16122 non-null	object
CDM	16122 non-null	object
RDM	16122 non-null	object
RWB	16122 non-null	object
LB	16122 non-null	object
LCB	16122 non-null	object
CB	16122 non-null	object
RCB	16122 non-null	object
RB	16122 non-null	object

Figure 1. Deleted variables

Value	17918 non-null	object
Wage	17918 non-null	object
Preferred Foot	17918 non-null	object
Weak Foot	17918 non-null	float64
Skill Moves	17918 non-null	float64
Position	17918 non-null	object
Jersey Number	17918 non-null	float64
Crossing	17918 non-null	float64
Finishing	17918 non-null	float64
HeadingAccuracy	17918 non-null	float64
ShortPassing	17918 non-null	float64
Volleys	17918 non-null	float64
Dribbling	17918 non-null	float64
Curve	17918 non-null	float64
FKAccuracy	17918 non-null	float64
LongPassing	17918 non-null	float64
BallControl	17918 non-null	float64
Acceleration	17918 non-null	float64
SprintSpeed	17918 non-null	float64
Agility	17918 non-null	float64
Reactions	17918 non-null	float64
Balance	17918 non-null	float64
ShotPower	17918 non-null	float64
Jumping	17918 non-null	float64
Stamina	17918 non-null	float64
Strength	17918 non-null	float64
LongShots	17918 non-null	float64
Aggression	17918 non-null	float64
Interceptions	17918 non-null	float64
Positioning	17918 non-null	float64
Vision	17918 non-null	float64
Penalties	17918 non-null	float64
Composure	17918 non-null	float64
Marking	17918 non-null	float64
StandingTackle	17918 non-null	float64
SlidingTackle	17918 non-null	float64
GKDiving	17918 non-null	float64
GKHandling	17918 non-null	float64
GKKicking	17918 non-null	float64
GKPositioning	17918 non-null	float64
GKReflexes	17918 non-null	float64
WeightKG	17918 non-null	float64
HeightCM	17918 non-null	float64

Figure 2. Important Variables

Then, we would also need to clean up the ”Value” column. This column follows the form of ” €110.5M” or ”€40K”, Therefore, I would first eliminate the Euro signs, then eliminate outliers such as ”€0”, and then use a regex expression to convert it into numerical values.

```
# Clean up the value
temp = dataset_new['Value'].str.lstrip(to_strip='€')
temp_index = dataset_new['Unnamed: 0']
# Use RegEx to convert to numerical values
import re
#Check if the string ends with M or K
for i in temp_index:
```

```
    x = re.search("M$", temp[i])
    y = re.search("K$",temp[i])
    if (x):
        temp[i] = float(temp[i].rstrip('M')) * 1000000
    elif (y):
        temp[i] = float(temp[i].rstrip('K')) * 1000
    else:
        temp[i] = float(temp[i])
dataset_new['Value'] = temp
# Eliminate outliers
outlier = np.where(dataset_new['Value'] <= 1000)
dataset_new.drop(dataset_new.index[outlier],inplace = True)
```

Eventually, it would make more sense if we seperate goal keepers from other players. Goal keepers are usually measured by their GK scores instead of other scores.

```
# Seperate the dataset
dataset_GK = dataset_new.copy()
dataset_General = dataset_new.copy()
temp = np.where(dataset_GK['Position'] != 'GK')
dataset_GK.drop(dataset_GK.index[temp],inplace = True)
temp = np.where(dataset_General['Position'] == 'GK')
dataset_General.drop(dataset_General.index[temp],inplace = True)
```

PERFORMANCE VISUALIZATION FOR INDIVIDUAL PLAYERS

Unfortunately it is unlikely to plot a graph with around 50 dimensions. Therefore, using knowledge from traditional star-scouting, I identified these main aspects of a player: Attacking, Defending, Physical Attributes, Passing, Mobility, Control and Game Intelligence & Mental.(Ertheo)

```
def attacking(data):
    return int(data[['Aggression','Finishing', 'Volleys', 'FKAccuracy',
```

```
                        'ShotPower','LongShots','Penalties',
                      'HeadingAccuracy','Curve']].mean())
def defending(data):
   return int(data[['Marking', 'StandingTackle',
                        'SlidingTackle','Interceptions']].mean())
def ballControl(data):
   return int(data[['Dribbling','BallControl']].mean())
def mentality(data):
   return int(data[['Positioning','Vision','Composure']].mean())
def passing(data):
   return int(data[['Crossing', 'ShortPassing','LongPassing']].mean())
def mobility(data):
   return int(data[['Acceleration', 'SprintSpeed',
                        'Agility','Reactions']].mean())
def physical(data):
   return int(data[['Balance', 'Jumping', 'Stamina','Strength',]].mean())

# adding these categories to the data
General_Polar = dataset_General.copy()
General_Polar['Defending'] = dataset_General.apply(defending, axis = 1)
General_Polar['Control'] = dataset_General.apply(ballControl, axis = 1)
General_Polar['Mentality'] = dataset_General.apply(mentality, axis = 1)
General_Polar['Passing'] = dataset_General.apply(passing, axis = 1)
General_Polar['Mobility'] = dataset_General.apply(mobility, axis = 1)
General_Polar['Physical'] = dataset_General.apply(physical, axis = 1)
General_Polar['Attacking'] = dataset_General.apply(attacking, axis = 1)
General_players =
    General_Polar[['Name','Defending','Control','Mentality','Passing',
            'Mobility','Physical','Overall','Attacking','Age', 'Club']]
```

```
General_players.head()
```

Given these data, we are able to generate a polar graph to visualize the individual players. Taking a few famous players for example, we visualize how these players differ in their performance. This could be use for scouts when they manually scrutinize players to understand their strengths and weaknesses. Considering the length of the code, I would not show this part in the paper. Specifications of the implementation can be found in the attached Jupyter Notebook.

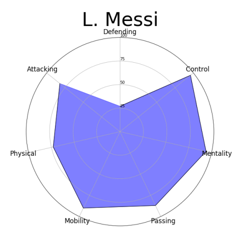

Figure 3. Messi

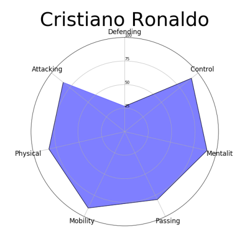

Figure 4. Ronaldo

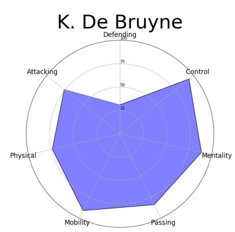

Figure 5. De Bruyne

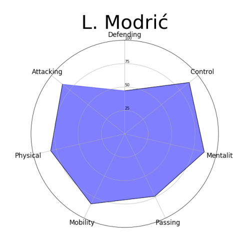

Figure 6. Modric

MANAGERIAL POINT-OF-VIEW

For this part, I would like to focus on the high-level statistics that managers would be interested in: what factors contribute the most to individual ratings and valuations, and what makes a reasonable team.

Plotting the distributions

As we can easily see, the Overall Performance, as expected, demonstrated a normal distribution pattern. This means that most of the players are concentrated in the middle level, with fewer players performing really well or really bad. However, the distribution of Value is skewed to the left, which means that only a few superstars are paid much more than others. This gave us the implication that many players are in fact

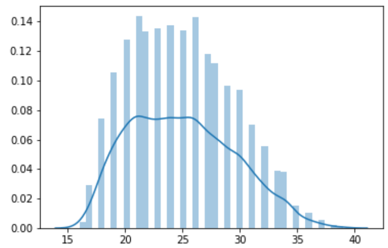

Figure 7. Age Distribution

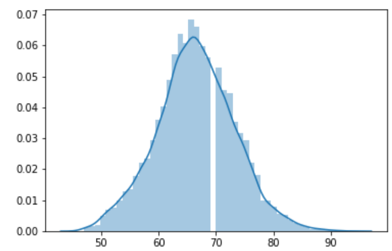

Figure 8. Overall Performance Distribution

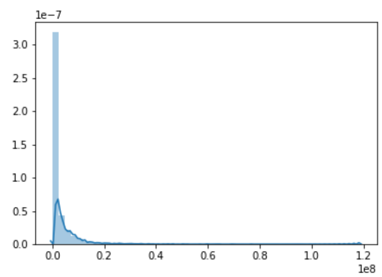

Figure 9. Value Distribution

playing moderately well, but are paid at a relatively low level.

What Relates to a player's performance

I would like to use a simple correlation matrix to identify the correlation of different performance metrics to the overall performance. A player's performance should be measured both by "Overall Performance" and "Potential Performance". I used the following code to accomplish this.

```
#Find the correlation
corr = players[['Age', 'Crossing', 'Finishing', 'HeadingAccuracy',
     'ShortPassing', 'Volleys', 'Dribbling', 'Curve', 'FKAccuracy',
     'LongPassing', 'BallControl', 'Acceleration', 'SprintSpeed', 'Agility',
     'Reactions', 'Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength',
     'LongShots', 'Aggression', 'Interceptions', 'Positioning', 'Vision',
     'Penalties', 'Composure', 'Marking', 'StandingTackle', 'SlidingTackle',
     'WeightKG', 'HeightCM']].corrwith(players['Overall'])
print(corr.sort_values(ascending=False))
```

```
#Find the correlation
corr = players[['Age', 'Crossing', 'Finishing', 'HeadingAccuracy',
     'ShortPassing', 'Volleys', 'Dribbling', 'Curve', 'FKAccuracy',
     'LongPassing', 'BallControl', 'Acceleration', 'SprintSpeed', 'Agility',
     'Reactions', 'Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength',
     'LongShots', 'Aggression', 'Interceptions', 'Positioning', 'Vision',
     'Penalties', 'Composure', 'Marking', 'StandingTackle', 'SlidingTackle',
     'WeightKG', 'HeightCM']].corrwith(players['Potential'])
print(corr.sort_values(ascending=False))
```

The result is illustrated in these graphs.

Reactions	0.848950
Composure	0.803705
ShortPassing	0.724298
BallControl	0.720986
LongPassing	0.585774
ShotPower	0.565120
Vision	0.525972
Dribbling	0.518555
Curve	0.503961
LongShots	0.503115
Crossing	0.497384
HeadingAccuracy	0.468149
Stamina	0.462111
Age	0.457102
FKAccuracy	0.456814
Aggression	0.454796
Volleys	0.452194
Positioning	0.440056
Penalties	0.390919
Finishing	0.373892
Strength	0.342113
Interceptions	0.334736
Marking	0.307288
StandingTackle	0.265473
Agility	0.244509
Jumping	0.228689
SlidingTackle	0.225215
WeightKG	0.185195
SprintSpeed	0.170393
Acceleration	0.151057
HeightCM	0.067842
Balance	0.059761

Figure 10. Correlation with Overall Performance

BallControl	0.522945
Reactions	0.505626
ShortPassing	0.495916
Composure	0.470442
Dribbling	0.417096
LongPassing	0.359565
Vision	0.348613
ShotPower	0.331622
Curve	0.306351
LongShots	0.288316
Positioning	0.274038
Crossing	0.273398
Volleys	0.268384
Finishing	0.253364
FKAccuracy	0.234780
SprintSpeed	0.233360
Penalties	0.229240
Acceleration	0.228676
HeadingAccuracy	0.225146
Agility	0.207838
Stamina	0.206068
Aggression	0.156844
Marking	0.146611
Interceptions	0.131898
StandingTackle	0.121881
Balance	0.116388
SlidingTackle	0.104758
Jumping	0.066395
Strength	0.050525
WeightKG	0.015979
HeightCM	0.012819
Age	-0.266158

Figure 11. Correlation with Potential Performance

The correlation matrix shows that some of the most important features regarding to a player's performance are: Ball Control, Reactions, Short Passing, Composure,Dribbling, Vision and Long passing.

Something worth noticing is that there is a negative correlation between age and potential. While this doesn't imply causal relationship, we can infer that with the increase of age, the potential decreases. Using a regression to confirm:

```
import matplotlib.patches as mpatches
from sklearn.linear_model import LinearRegression
player_age = players[['Age','Potential']]
y = player_age[['Potential']].values.ravel()
X = player_age[player_age.columns.drop('Potential')]
model1 = LinearRegression()
model1.fit(X,y)
y_pred1 = model1.predict(X)
player_age = players[['Age','Overall']]
y = player_age[['Overall']].values.ravel()
X = player_age[player_age.columns.drop('Overall')]
model2 = LinearRegression()
model2.fit(X,y)
y_pred2 = model2.predict(X)
plt.subplot(223)
plt.plot(X, y_pred1, color = 'red',label="Potential")
plt.plot(X, y_pred2, color = 'blue',label = "Overall")
plt.title('Linear Regression')
plt.xlabel('Age')
plt.ylabel('Score')
plt.savefig('Score~Age.png')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
```

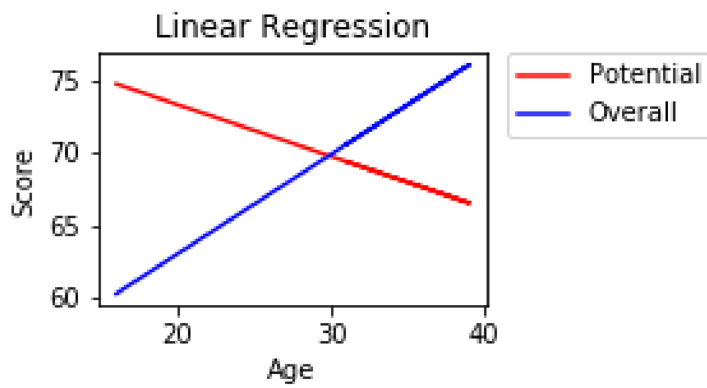

Figure 12. Regression with both Scores and Age

Here we can conclude that the increase of Age is associated with increase in Overall score but decrease in Potential Score. My interpretation is that older players are more experienced, but younger players have more potential which can be realized. The decision is up to the management, whether they want to ”win-now” or ”win-later”. Still, I assume the team would need to compose of a total of considerable ”Overall” scores, therefore it would be a good idea to construct a team with both younger and older players.

Player Valuation: Are they really worth that much?

Since now we have working metrics to measure the performance of the players (Overall Score, Potential Score),we would train a model which can evaluate a player’s expected value. If the player has a lower actual value, he is a considerable deal.

But first, let’s plot the relationship between ”Value” and ”Overall” and ”Potential” (See Figure 13 and 14)

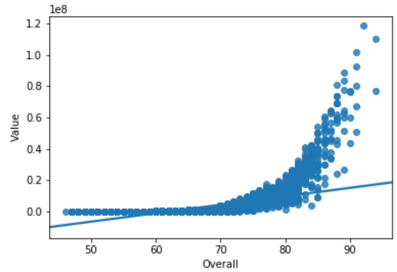

Figure 13. Value Related to Overall

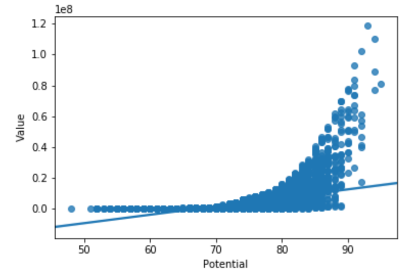

Figure 14. Value Related to Potential

This shows that, in general, the higher the player's overall score is, the more expensive he would be. However, it is also worth noticing that there are several players who have a reletively high performance but not that high salary. From the polt, I think it would fit a ploynomial regression:

```
# process the data into X and y form
player_score = players[['Value','Overall']]
y = player_score[['Value']].values.ravel()
X = player_score[player_score.columns.drop('Value')]
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
param_test = {
      'poly__degree':range(1,3)
   }
estimator = Pipeline([('poly', PolynomialFeatures()),
                ('linear', LinearRegression(fit_intercept=False))])
gsearch = GridSearchCV(estimator , param_grid = param_test,
    cv=10,scoring='neg_mean_squared_error')
gsearch.fit(X,y)
gsearch.best_params_, gsearch.best_score_
print('best score is:',str(gsearch.best_score_))
print('best params are:',str(gsearch.best_params_))
```

With this code, we used 10-fold cross validation to find the hyperparameters: the ideal degree of the polynomial is 2:

```
# Visualising the Polynomial Regression results
y = player_score[['Value']].values.ravel()
X = player_score[player_score.columns.drop('Value')]
model = Pipeline([('poly', PolynomialFeatures(degree = 2)),
                  ('linear', LinearRegression(fit_intercept=False))])
model.fit(X,y)
y_pred = model.predict(X)


plt.plot(X, y_pred, color = 'red')
plt.scatter(X, y, color = 'blue')
plt.title('Polynomial Regression')
plt.xlabel('Overall Score')
plt.ylabel('Value')
plt.savefig('Value~OveallScore.png')
plt.show()
```

Then we do the same for ”Potential Score”. With the results, we get these two figures:

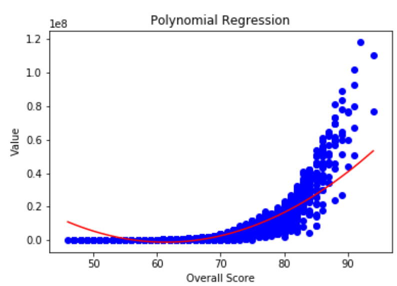

Figure 15. Value Related to Overall

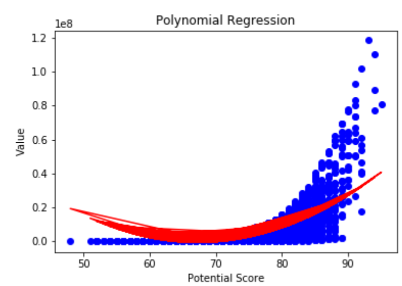

Figure 16. Value Related to Potential

So, my idea here is to identify the players who has less real value than their predicted value. In other words, I need to find the players that are not ”over-priced”. Also, some players are underpriced, but their scores are terrible. Intuitively, although those players are cheap, we do not want to buy these players.

Therefore I set a constraint: Either the player's Overall or the Potential should be over 80.

The codes below would help me do this. Note that a new cross-validation shows the optimal hyperparameter of degree is still 2.

```
player_score = players[['Value','Overall','Name','Potential']]
y = player_score[['Value']].values.ravel()
X = player_score[player_score.columns.drop('Value').drop('Name')]
param_test = {
      'poly__degree':range(1,3)
   }

estimator = Pipeline([('poly', PolynomialFeatures()),
                ('linear', LinearRegression(fit_intercept=False))])
gsearch = GridSearchCV(estimator , param_grid = param_test,
    cv=10,scoring='neg_mean_squared_error')
gsearch.fit(X,y)
gsearch.best_params_, gsearch.best_score_
print('best score is:',str(gsearch.best_score_))
print('best params are:',str(gsearch.best_params_))
```

With this code, we used 10-fold cross validation to find the hyperparameters: the ideal degree of the polynomial is 2:

```
model = Pipeline([('poly', PolynomialFeatures(degree = 2)),
                ('linear', LinearRegression(fit_intercept=False))])
model.fit(X,y)
y_pred = model.predict(X)
A = player_score[['Overall']].values.ravel()
B = player_score[['Potential']].values.ravel()
lst = []
for i in range(y.size):
   if (A[i] >= 80 or B[i] >= 80):
```

```
        if y_pred[i] >= y[i]:
            lst.append(i)
listName = []
for i in lst:
    listName.append(player_score[['Name']].values.ravel()[i])
print(listName)
```

The resulting list is a list of undervalued players:'Dani Sandoval', 'B. Adekanye', 'A. Wilson', 'I. Sauter'. These are our "target players". Unsurprisingly, famous superstars, such as L.Messi and C.Ronaldo are not on this list, because they are much more expensive per performance point.

CONCLUSION

Using quantitative methods and data analytics, we have identified what are the important factors that relates to a player's abilities, which players are undervalued, and came up with a visualization of individual players. Using this data, scouts and managers can find cost-effective team members to choose from.

ACKNOWLEDGMENTS

This paper is used for final project in Business Analtics, Spring 19, taught by Professor Renyu Zhang. The data analytical skills taught in this class is of great benefit to this paper. The exact codes are in the corresponding Jupyter Notebook. Thanks to Qiheng Fang for a brief review of the paper.

REFERENCES

[amanthedorkknight] amanthedorkknight. fifa18-all-player-statistics.

[2] Dimitris Bertsimas, Allison K.O'Hair, W. R. (2016). *The Analytics Edge*. Dynamic Ideas LLC.

[Ertheo] Ertheo. 15 key skills to achieve success in football (& tips to improve).