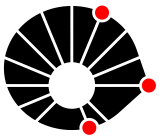

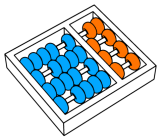

Data Mining Approach to Prediction of Nominee Soccer Players for Ballon d'Or Award

R. T. Shibata *H. Pedrini*

Technical Report - IC-18-08 - Relatório Técnico
July - 2018 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Data Mining Approach to Prediction of Nominee Soccer Players for Ballon d'Or Award

Renato Toshiaki Shibata[1], Helio Pedrini[2]

[1] Instituto de Computação, Universidade Estadual de Campinas (UNICAMP)
Undergraduate Course, Computer Engineering, ra082674@dac.unicamp.br

[2] Instituto de Computação Universidade Estadual de Campinas (UNICAMP)
Campinas-SP, Brazil, 13083-852, helio@ic.unicamp.br

Abstract. This report describes the main activities developed during the Final Undergraduate Project in the Computer Engineering Bachelor of the Institute of Computing at the University of Campinas (UNICAMP). In this work, a data set of professional soccer players' historical in-game statistics from recent years was built through feature engineering methodology and then machine learning algorithms were applied in order to characterize players who were nominated to the annually award called Ballon d'Or. The task was addressed as a classification problem, where the predictor should classify correctly whether or not the current player, according to his annual performance, will be nominated to the Ballon d'Or award at the end of the year. Some data mining and machine learning techniques were evaluated, such as ZeroR, OneR, Iterative Dichotomiser 3 (ID3), Logistic Regression (LR), and Support Vector Machine (SVM). The obtained results with each method are discussed and compared among them to evaluate how accurate they are according to a soccer expert's opinion.

Keywords: data mining; machine learning; classification; feature engineering; ballon d'or; soccer game

1. Introduction

Soccer is a very popular game worldwide, which was invented in England in the XIX century and it is now played regularly by more than 240 million people according to Fédération Internationale de Football Association (FIFA) [1].

Ballon d'Or is an award given by France Football to the player who is considered the best in the previous year. France Football is a French magazine that belongs to the Amaury Group. On 5th July 2010, the president of the Group, Marie-Odile Amaury, signed an agreement with former president of FIFA, Joseph Blatter, to unify Ballon d'Or award and FIFA Best Player of the Year award into FIFA Ballon d'Or award. However, in September 2016, France Football announced that their partnership with FIFA Ballon d'Or award had come to an end and they would revive the Ballon d'Or Award [2].

The Ballon d'Or award consists basically of two processes, however, for this research only the first process is important to us. First, we arbitrarily selected several candidates for the award. The shortlist used to be made by the FIFA's Football Committee and a group of experts from France Football magazine until 2015, where they chose 23 candidates. As of 2016, only France Football decided this shortlist, which increased its size to 30 candidates [2].

The purpose of this study is to verify possible objective criteria for the 30 nominees in the annual shortlist, where only one of these candidates will receive the Ballon d'Or award and then will be recognized as "Best Player of the World" of the most popular sport in the planet.

We assume that the nominee's selection is made by reasonable and unbiased criteria. To clarify all these underlying possibilities, we have decided to investigate objective criteria to identify patterns of the nominee player. In addition, we assume that the choice of Ballon d'Or nominee players made by FIFA (from 2010 to 2015) and by France Football magazine has sufficient positive relationships with the player's in-game statistics and we investigate those using compelling objective criteria. For this objective, we chose to use an approach based on machine learning algorithms.

There are some previous studies on the application of machine learning and data mining techniques in soccer. For example, Wang and Wang [3] used the Apriori data mining algorithm [4] to determine association rules between different types of technical movements in soccer. These technical movements were related to shots converted into goals. Their objective was to provide a scientific decision for coaches to train and guide the soccer players. Another example using similar approach was used by Chai [5] to mine soccer data patterns using serial data to determine which chains of possession between players occurred frequently. The provided model aimed to guide coach to establish an effective mechanism in the soccer match.

Finally, the research whose intentions are closer to ours is by Nunes and Sousa [6]. One of the reasons is because they applied data mining techniques for searching interesting and unexpected association rules. Their data set was based on historical data from European tournaments. In addition, they used classification methods using Weka Software [7] to test

their hypothesis that they would be able to classify matches according to their results based on available historical data.

However, our objective is not focused on predicting the outcome of European soccer matches, as Nunes and Sousa intended. Furthermore, we found that it is not very effective to predict future outcome of matches based on this because we assume that a club's historical data does not have an influential impact on the outcome of the game compared to the current player staff of the club, coach staff and club administrator, among other factors.

2. Methods

I. Reasons for the use of Machine Learning and Data Mining Algorithms

Data Mining is the process of extracting valid, previously unknown, comprehensible, and actionable information from large databases and using it to make crucial business decisions (Connolly, 2004). In other words, we can say that data mining aims to extract knowledge from the data. In Data Mining and Machine Learning there are other types of methods, such as Clustering and Neural Networks. However, for the purpose of this research, we are interested in finding associations between features of the data set and creating a classifier model.

Classification maps data into predefined groups or classes (Dunham, 2002). Thus, classification is the process of finding a model that describes the data classes or concepts. Its purpose is to be able to use this model to predict the class of objects whose class label is unknown.

This mathematical approach fits well in the case of dealing with soccer statistics around the world. As we know, there is a huge amount of data about players that need to be transformed into useful information, so data mining would be adequate for dealing with big data in order to extract from it only what we want to know, that is, typical feature patterns for a Ballon d'Or nominee. In addition, player statistics are sometimes unknown, so we need a mathematical approach appropriate to deal with player's imperfect knowledge.

Moreover, we have previously unknown information about the characteristics of Ballon d'Or nominees and we are assuming that the choice of France Football nominees is reasonable.

Tools for Data Mining are very powerful, however, they require a very skilled specialist who can prepare the data and understand the output. Data Mining brings out the patterns and relationships, but the significance and validity of these patterns must be made by the user. In addition, some Data Mining and Machine Learning methods may yield better results than others, depending on the nature of the data set. That is why comparing the various results obtained through the experience and knowledge of the user about soccer is fundamental.

Among all types of data mining methods, we have chosen only classification algorithms. As seen in the previous definition, the reason for using this kind of method is obvious. In this research, we distinguish two groups of players: those assigned to the Ballon

d’Or and non-Ballon d’Or nominees. These groups are already predefined since we already know who they are in the previous indications of the Ballon d’Or.

II. Feature Engineering

There are few websites that provide detailed player statistics, and even rarer ones throughout the season. We have attempted to collect as many individual recorded statistics as we could get on the Internet scattered accross multiple websites[1]. Many of them did not provide all 29 attributes, so we gathered them together, taking data piece by piece, especially for players like Neymar, who played a key role in the South American tournaments during 2012, which are not well considered by European journalists.

We made a training data set with 181 instances and 30 attributes, one of which is the target attribute, which specifies whether a player is a Ballon d’Or nominee or not. The instances are represented by soccer players and their respective season. We could not have time to get a larger number of players given the work rate to gather all the statistical information about them within a few months. Otherwise, we would not have enough time to apply data mining methods and analyze their results.

These 181 instances were composed of player statistics during the following period: August 13, 2012 through December 25, 2013, and August 22, 2011 through December 25, 2012. These two periods correspond approximately to the 2012 season and to 2013 season. Among the 181 players selected, 22 were nominee players for the Ballon d’Or award in 2013 and 20 of them were nominee players for Ballon d’Or award. So in total, there are 42 nominee player statistics and 139 non-nominee player statistics. We did not choose the total of 23 players in those years, because some of them were goalkeepers.

We did not include the goalkeepers because the relevant statistics for a goalkeeper are very different from all other types of players. It would take another data set only to deal with goalkeepers to get an accurate result.

In order to obtain a balanced data set in terms of player’s position, we got a similar proportion of players who act in the same position among the 42 nominee players and among the 139 nominee players. This is very important in this research because we have a very limited number of instances to work with, and most of those are Center or Wide Forwards. Thus, in this case, there would be a risk that the classification models obtained by Data Mining and Machine Learning methods could only describe a common feature of any average Forward if there were not enough proportion of Forwards among the non-nominee players. There are 20 Forwards, 11 Attacking Midfielders, 7 Defensive Midfielders, 2 Fullbacks and 2 Center-backs among the 42 nominees. The Forwards account for nearly half of all nominees. That is why we collected data from 51 Forwards,

[1] http://espnfc.com/
http://www.ogol.com.br/
http://www.fifa.com/
http://br.soccerway.com/
http://www.whoscored.com/

38 Attacking Midfielders, 23 Defensive Midfielders, 13 Fullbacks and 14 Center-backs among the 139 non-nominees.

We managed to gather more attributes than those 29, which are:

(0) Number of times a player started a game playing in that season;

(1) Number of times a player was on the bench and came into the match in that season;

(2) Total number of goals a player scored in that season;

(3) Total number of passes a player made that lead directly to a goal for his teammate, that is, assists, in that season;

(4) Average number of goals a player scored per match in that season;

(5) Average number of assists a player did per match in that season;

(6) Total number of yellow cards a player received in that season;

(7) Total number of red cards a player received in that season;

(8) Average number of shots a player attempted to the goal either on or off target per game;

(9) Average number of winning a header in a direct contest with an opponent per game;

(10) Average number of time a player dispossess an opponent, whether the tackling player comes away with the ball or not, per game;

(11) Average number of opponent's pass intercepted by a player per game;

(12) Average number of fouls committed by a player per game;

(13) Average number of times a player is the last man to step up to catch an opponent in offside position per game;

(14) Average number of fouls clearances a player made to remove attacking threat on his goal per game;

(15) Average number of times a player was dribbled by an opponent without winning a tackle per game;

(16) Average number of times a player blocked a shot by opponent per game;

(17) Total number of own goals by a player in that season;

(18) Average number of passes a player made to lead to a shot at goal from a team mate per game;

(19) Average number of times a player successfully dribbles past per game;

(20) Average number of time a player is fouled by opposite team per game;

(21) Average number of times a player is caught offside per game;

(22) Average number of times a player is tackled by an opponent without attempting to dribble past them per game;

(23) Average number of times a player loses possession due mistake or poor control per game;

(24) Average number of passes a player attempts (short passes, long balls, through balls, crosses) per game;

(25) Percentage of attempted passes by a player that successfully found a teammate per game;

(26) Average number of attempted and accurate pass from a wide position to a central attacking area by a player per game;

(27) Average number of attempted and accurate pass of 25 yards or more by a player per game;

(28) Average number of an attempted and accurate pass between opposition players in their defensive line to find an onrushing teammate per game;

Some other attributes, such as the number of wins and losses by the team he is playing, have been ignored because we assume that the Ballon d'Or is an award exclusively for individual skills. From some tournament types, we were not able to collect data for all 29 attributes. But at least, from the 5 major European national leagues[2] plus the Brazilian national league, we gathered all the data. Fortunately most of the matches played by a regular player in a season comes from the national league.

In addition, from the UEFA Champions League, which is the most important tournament in Europe, we gathered all the data as well. We have also included data related to the UEFA Europa League since it is the second most important tournament in Europe. From all other tournaments, such as African Cup of Nations, AFC Asian Cup, Copa America, Club World Cup, National Cups of the 5 major European soccer leagues, we were not able to collect data for all these 29 attributes.

In order to get the best possible data set to avoid overfitting, we consider to choose the statistics of the players not just in terms of their position but also in terms of the league he plays. This is because we believe that each league has specific characteristics of game style and an algorithm may have the risk of learning the underlying concept of a given game style for a certain league in detriment of others. This is the reason why the data set has a similar proportion of players acting in each league. This was not in a strictly way as was done to equalize the proportion of players by position, but we tried not to choose the players in a way that a given league becomes much more prevalent than the others in relation to the group of the Ballon D'Or nominees and the non-nominees.

Everything that has been said before in this section is related to the training data. However, a very similar approach was made to create the testing data in order to have consistency with the training data. For example, there was no goalkeeper among the instances in the testing data. Again, we used only statistics of the players in the 5 major

European Soccer Leagues. We only included data from the National Leagues, UEFA Champions League and the UEFA Europa League.

The differences between training data relies on the number of instances. We have chosen to gather 52 instances, where 26 are Ballon d'Or nominees and the other 26 are non-nominees.

We chose each player by position and league he plays, so we had equally divided them in both groups of Ballon d'Or nominees and non-nominees: 11 Forwards, 8 Attacking Midfielders, 3 Defensive Midfielders, 3 Center-backs and 1 Fullback. Thus, we kept the same proportion of positions between the Ballon d'Or nominees and non-nominees, in order to make the test more reliable.

Another important difference with the training set is that the testing set was done with statistics of the players of the 2016/2017 season, that is, much more recent data than the training set, which were from 2011/2012/2013.

III. Data Preparation

We organized the Data Preparation step into 3 stages: data cleaning, data transformation and feature selection.

On data cleaning, that is, preparing the data in a way that reduces noise and manipulates missing values, we had no real trouble since we chose the attributes in relation to the data available on the Internet. Otherwise, there would be many non-nominee players with missing values, since the statistical data for these types of players are difficult find on the Internet.

On data transformation, for some experiments, we decided to use normalization for the training set. Specifically, we use the Z-score normalization, whose transformation of the value of an attribute *a* of a n instance x_i can be described as:

$$x'_i(a) = \frac{x_i(a) - \mu(a)}{\sigma(a)} \tag{1}$$

where:

$x'_i(a)$ is the new value for the instance *i* for attribute *a*,

$x_i(a)$ is the initial raw value for the instance *i* for attribute *a*,

$\mu(a)$ is the mean of the set of values of attribute *a*,

$\sigma(a)$ is the standard deviation of the set of values of attribute *a*.

On feature selection, we experimented with two different functions to select the *k* attributes with the highest score, according to the definition of its function. *k* is an integer that can be arbitrarily chosen between 1 and the number of attributes in that data set. The functions we chose were: Chi-square statistics ($\chi 2$) [8] between each attribute and the target vector and the ANOVA F-value [9] between each feature and the target vector.

IV. Algorithms

We have selected algorithms ranging from the simplest one to the most sophisticated, where all of them possess a very different concept between each one of them. We will briefly explain how each of them works as follows.

1. ZeroR

ZeroR, short for Zero rule, is the simplest classification algorithm since it depends only on the values of the target attribute and ignores all other feature attributes. ZeroR simply predicts the majority class in a data set by building a model based on the most frequent value in the target attribute. In other words, the ZeroR predictor model would be:

'target attribute' = 'most frequent value of the target attribute'

2. OneR

OneR, short for "One Rule", is a simple and accurate classification algorithm that generates a rule for each attribute in the data, and then selects the rule with the lowest total error as a "single rule" associated with only 1 attribute.

OneR is considered to be one of the simplest existing classifiers. Nevertheless, it is always good to try a simple version firstly. It has been shown that OneR produces rules only slightly less accurate than very sophisticated classification algorithms, while producing rules that are simple for humans to interpret.

In short, this algorithm chooses only 1 of all the attributes to create a rule. In other words, it is based on the value of ony one, the best attribute to classify all future data. The pseudocode of this algorithm is presented as follows.

For each attribute A:

For each value V of that attribute, create a rule:

Count how often each class appears

Find the most frequent class, c

Make a rule "if A=V then C=c"

Calculate the error rate of this rule

Pick the attribute whose rules produce the lowest error rate

[2]Bundesliga - German 1st division football league
Barclays Premier League - English 1st division football league
La Liga - Spanish 1st division football league
Serie A - Italian 1st division football league
Ligue 1 - French 1st division football league

3. Decision Tree

CART, short for Classification and Regression Trees, is very similar to C 4.5. C 4.5 is used in classification problems and it is one of the most used algorithm for building Decision Trees [10]. It improves and extends the ID3 algorithm [11] by dealing with both continuous and discrete attributes, missing values and pruning trees after construction. C 4.5 dynamically defines a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. C 4.5 converts the trained trees into sets of if-then rules. These accuracy of each rule is then evaluated to determine the order which they should be applied. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it [12].

The difference between CART and C 4.5 is that CART supports numerical target variables and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node [12].

In the following we present the pseudo code of CART.

Check for base cases.
*For each attribute **a** calculate:*
*Normalized information gain from splitting on attribute **a**.*
Select the best attribute
Create a decision node that splits on best of attribute
Recurs on the sub lists obtained by splitting on best of attribute and add those nodes as children node.

The selection of best attribute in CART refers to selecting an attribute that produces the purest daughter nodes. The purest node possible is a leaf with only one class either "Yes" or "No", so it will not have to be split further and it will favors for smaller tree. In other words, we select an attribute on which objects are partitioned most clearly after objects are partitioned. This brings on the concept of entropy and information gain.

$$H(T) = -\sum p_c log_2 p_c = -p_+ log_2 p_+ - p_- log_2 p_- \quad ...(2)$$

Entropy is a function defined by equation (2). Here p_c is the probability that a certain instance belongs to class *c*. In our case, probability can be defined just as frequency, that is, the proportion of instances which belong to class *c* in the data set *T*.

In the case of our research, where there are only 2 classes possible: “Yes Ballon d’Or” and “No Ballon d’Or”, we could reduce the sum in two parts only. The first term is related to the positive instances of *T* and the second one to the negative instances of *T*.

Entropy measures the impurity of a set of training instances *T*. That is, it represents the expected amount of information that would be needed to specify whether a new instance should be classified in a positive class “Yes Ballon d’Or” or a negative class “No Ballon d’Or”, in the case of our research.

Using this concept of entropy, it is used the concept of information gain, which is defined by the difference of entropy of parent node and average entropy of the children nodes. This is better described in the equation (3).

$$IG(T,a) = H(T) - \sum_{v \in vals(a)} \frac{|\{\mathbf{x} \in T | x_a = v\}|}{|T|} \cdot H(\{\mathbf{x} \in T | x_a = v\}) \quad \text{...(3)}$$

Here *T* denotes a set of training examples, each of the form (**x**,y) = (x_1, x_2, x_3 ,.., x_k ,y) where x_a belongs to *vals(a)* is the value of the ath attribute of the instance **x** and y is the corresponding target value. The information gain of attribute *a* in terms of entropy *H*.

In that way it’s possible to check which of the attributes give us the highest information gain. So that attribute is selected as the best and taken as criteria for splitting. Then the nodes are expanded until all leaves are pure. The pruning is done while expanding the leaves, where the node will split if this split induces a decrease of the impurity greater than or equal to a threshold, arbitrarily chosen by us. In the figure 1 there is the function of entropy for the case of 2 classes.

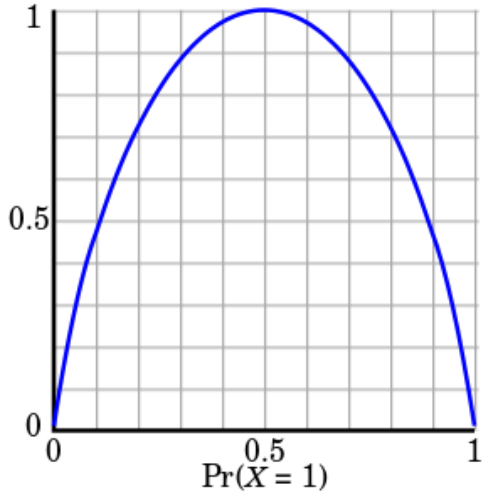

Figure 1: Graphical representation of the entropy function,where the *x*-axis is proportion of instances in the set belonging to the class *X=1*. Notice that this graph illustrates only the case where there are only two classes in the data set.

Assuming there are 2 classes, one class labeled “0” and one class as “1”, it is possible to see that the entropy of a set *S* is maximum when the probability of a class *X*, *Pr(X=1)* is exactly 0.5. This value occurs when the set *S* is equally divided into class “0” and class “1”.

When *S* is completely homogeneous, that is, when *S* represents the purest child nodes possible in a decision tree, then the entropy equals 0. This means that there is no need to split more nodes since it can not have any information gain thereafter.

4. Logistic Regression

It is a modification of the Linear Regression method [13], but instead of producing a continuous numerical value, it generates only two possible values for the target value.

Unlike Linear Regression which simply uses the difference of the actual target value with its linear function to compute cost, Logistic Regression calculates cost in a different way. Another crucial difference is that whereas Linear Regression model function is only a linear combination with the attribute weights, the Logistic Regression model function is a function σ whose parameter is a linear combination with the attribute weights. This can be expressed in Equation (4), where

$$h_\theta(x) = \sigma(\theta^T x) \tag{4}$$

Here, h_θ is our hypothesis function, *x* is a vector of all the attribute values of a given instance and θ^T is the transpose vector of the weights that we need to find to construct a classifier for the hypothesis function.

Logistic Regression uses a sigmoid function σ(t) , which receives any real input and outputs a value between zero and one, as show in Figure 2.

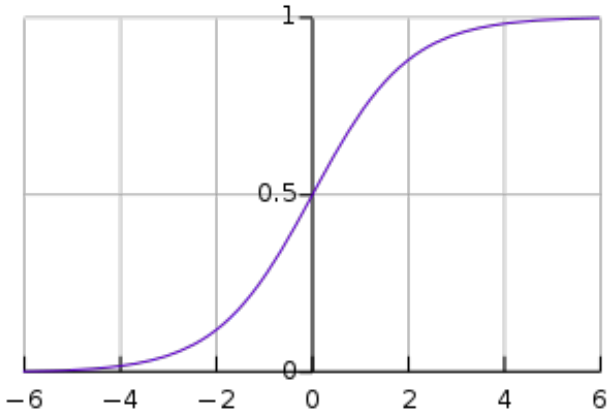

Figure 2: Graphical representation of the sigmoid function. Notice that its *y* intercept is exactly 0.5 and that all the possible outputs are in the interval of [-1,1].

This function is defined in Equation (5).

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{5}$$

The cost function is defined in Equation (6), which measures how well the algorithm learns in each iteration.

$$Cost(h_\theta(x), y)) = \begin{cases} -log(h_\theta(x)), & if \quad y = 1 \\ -log(1 - h_\theta(x)), & if \quad y = 0 \end{cases} \tag{6}$$

Thus, we penalize the learning algorithm by a very large cost as it commits mistakes, because that equation states that:

$$Cost(h_\theta(x), y) = 0 \begin{cases} if \quad y = 1 \quad and \quad h_\theta(x) = 1 \\ or \quad if \quad y = 0 \quad and \quad h_\theta(x) = 0 \end{cases} \tag{7}$$

and

$$Cost(h_\theta(x), y) \to \infty \begin{cases} if \quad y = 0 \quad and \quad h_\theta(x) = 1 \\ or \quad if \quad y = 1 \quad and \quad h_\theta(x) = 0 \end{cases} \tag{8}$$

Then, the algorithm tries to minimize the *Cost J(θ)* defined in Equation (9).

$$J(\theta) = \frac{1}{m} \sum_{j=1}^{m} Cost(h_\theta(x)^j, y^j) \tag{9}$$

After a defined number of iterations, the algorithm stops and delivers the hypothesis function h_θ with the weights θ that have been calculated so far.

5. Support Vector Machines

Support Vector Machines (SVM) is a powerful algorithm for machine learning classification, which aims to design a hyperplane that best classifies all training instances into two classes. That is done by maximizing the margin between two classes. The margin is the distance between the hyperplane and the nearest instance of a class.

The pseudocode of the SVM algorithm is shown as follows:

Define an optimal hyperplane: maximize margin.

Extend the above definition for non-linearly separable problems: have a penalty term for misclassification.

Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space.

The hyperplane is designed equidistantly for its closest instance of both classes. The hyperplane can be defined in Equation (10):

$$g(\boldsymbol{x}) = \boldsymbol{w}.\boldsymbol{x} + b \tag{10}$$

Here, **w** is a vector of weights and *b* is a constant, such as a bias. It will provide values equal to or greater than 1 for all instances of class 1 and will provide values equal to or less than -1 to all instances of class 2. The vectors that define ***w*** are called support vectors.

In the end, an optimization problem is used to maximize the width of the margin *w,* with the constraints defined for the classification of class 1 and class 2, as stated in the previous paragraph.

Therefore, in order to maximize the separability between the two classes, it is necessary to minimize the value of the vector **w.** An ideal SVM analysis should produce a hyperplane that completely separates the vectors (cases) into two non-overlapping classes. However, perfect separation may not be possible. In this situation, the SVM finds the hyperplane that maximizes the margin and minimizes the classification error. For this, it penalizes each instance that falls off the margin with a slack variable ξ_i. This slack variable for a given instance is then added to the constraint equation for that instance.

The new objective adds a new term related to the penalization of all instances in the model. This is described as the product of a constant *C* and the sum of the slack variables of all instances.

3. Proposed Solutions

In Feature Engineering, *LibreOffice Calc was used* to manually construct the training and testing sets. After collecting data from several soccer-related websites, the attributes were filtered and arranged in columns to be included in the data set. This task was done manually, from choosing the player statistics to placing the data sets and formatting the values in the data set table before converting them to *CSV* files. These data sets used in our research were:

(A) For the data without the attribute which counts the number of times a player started playing a match:

- ***CSV2012-2013_ALL_PLAYERS _No_app_no goalkeeper_ No MOM.csv*** as the training set;

- ***2016-2017season.csv*** as the testing set;

(B) For the data with the attribute which counts the number of times a player started playing a match:

- ***CSV2012-2013_ALL_PLAYERS _YES_app_no goalkeeper_ No MOM.csv*** as the training set;

- ***2016-2017season_YESapp.csv*** as the testing set;

In the end, after analyzing the results from both data sets, we decided to discuss in this report only the results from data set B. It was found out that the attribute which counts the number of times a player is much more important as we thought. Initially, we believed that the results were not coherent to our knowledge of soccer, however, later on we reconsidered to have a different perspective about the game. Thus, the data set B brought more relevant and complete results than data set A. We will discuss that later in the section of results.

All the code generated along this work was implemented in Python programming language, version 2.7. We also used the following libraries:
- **pandas** – to manipulate the data set;
- **csv** – to open the CSV files;
- **numpy** – for several purposes, such as manipulatation of arrays and other data structures;
- **scikit-learn** – the most important library in this research, which contains the machine learning algorithms used in our experiments.

The only algorithms that we did not use any scikit-learn implementation to code them were ZeroR and OneR. All other algorithms, such as Decision Trees, Logistic Regression and Support Vector Machines, were implemented using **sklearn.tree** , **sklearn.linear_model** and **sklearn.SVC**, respectively. For all these three algorithms, we

used the following classes, respectively, to run each of these algorithms: *DecisionTreeClassifier, LogisticRegression* and *SVC*.

We set the default parameters to initialize each algorithm. There were exceptions, such as the *DecisionTreeClassifier* class, in which we used the *criterion*=”entropy” parameter, since we intended to use this concept instead of using the Gini Index to become the concept for the criteria for dividing each node. In addition, the *min_impurity_decrease* parameter (that is, the threshold for splitting each node) was set to its default value of 0. This means that it would not prune any tree.

Another exception was the *SVC()* class, where we used the *kernel*=”linear” parameter to retrieve the coefficients of each attribute. This was made to maintain it coherent with the method proposed in Section II, where there was no transformation of the original space resulting from the kernel transformation and, therefore, the coefficients are directly related to the dimension of the data set.

Specifically, in the OneR implementation, we implemented the discretization of the data set to 5, 6, 7, 8, …, 19, 20 categorical values for each attribute.

For OneR only, we implemented a version where we used *k*-fold cross-validation [14]. We applied *k=10*, which is dividing the training set into 10-folds, where 9 of them were used to build the classifier and 1 of them was used as the validation fold in each one of the 10 total iterations. One important aspect to notice is that we used *StratifiedKFold* class from *scikit-learn* library, so the folds were made by preserving the percentage of samples for each class. When using the testing set, among the 10 generated OneR classifiers, only one is chosen, the one that obtained the highest accuracy in the validation set.

For the other algorithms, except ZeroR, since it does not work with the non-target attributes, the data preparation was made with the *scikit-learn* library. Standardization with Z-norm was done with **sklearn.preprocessing**, whereas the Feature Selection was made with **sklearn.feature_selection**.

To normalize using *Z*-score, we used the *StandardScaler()* method. To make the feature selection, the choice of the method depended on the type of experiments we did.

For experiments in which we needed to work with discrete data, we used Chi-square as the function to be computed for the *SelectKBest()* method to select the *k* best attributes. To work with the data without discretizing them, we used *f_classif* , that is, the ANOVA F-value, *as* the function to be computed for the *SelectKBest()* method.

It is important to mention that the *Z*-score normalization may produce negative values in the data set. Since Chi-square function does not accept negative values, it was not possible to use feature selection using *chi2*. We only used *f_classif* for feature selection after *Z*-score normalization.

4. Results

All the results were obtained by running the programs in Python. Some of the results were obtained via Weka Software, but the objective of doing that was just to compare them with its equivalent algorithm in a program coded by us.

We used the same training and testing sets that were defined in Section II for the Weka and Python programs.

I. ZeroR

ZeroR gave very modest results, but it was useful in order to have a baseline for the performance of the other algorithms.

It does not matter whether we applied Z-score normalization or used the *SelectKBest()* during the data preparation because this algorithm uses only the target attribute values. We tested the same algorithm in the Weka software and obtained exactly the same results. Prediction of class value: Ballon d'Or Nominee = "No".

Here are the results for the testing data:

* Correctly classified instances =50%

* Confusion matrix:

m=52	Classified as "Yes"	Classified as "No"
Actual "Yes"	0	26
Actual "No"	0	2

It is obvious that the prediction rule would be to set the target value Ballon d'Or Nominee to "No" value, since most instances in the training set are not nominees. Only a few players are selected each year to be nominated.

Moreover, since the testing set was equally divided between the instances into the two classes, it was evident that the accuracy of ZeroR classifier would be 50%. As it is possible to see from the confusion matrix, ZeroR classified all instances in the testing set as "No". Hence, we did not have true positives among the correctly classified instances. Because of this, the other half of the instances that were misclassified were all false positives.

II. OneR

For all classifiers generated by OneR, we always had to discretize the numerical values to obtain categorical values for the algorithm to work. Some experiments were applied to the *k*-fold cross-validation, for others only Z-score normalization, for others only feature selection , for other two of them together, and for others no kind of preprocessing.

First, we will show OneR results with 10-fold cross-validation. After testing with several number of bins (*b*) for discretization, as mentioned in Section III, we found that the best accuracy for the testing, using the minimum possible value for *b*, occurred when *b*=6.

Its overall mean accuracy for all the 10-fold validations were 89.40% and their accuracy in the testing data was 59.62%.

For instance, for *b*=10 it gave a total mean accuracy of 89.90% for all 10 folds and an accuracy in the testing data equals 55.77%. For *b*=5, it was 82.73% and 51.92%, respectively. For *b*=20, it was 88.72% and 55.77%, respectively.

The set of rules with the best validation accuracy among all iterations for *b* = 6 is shown below:

IF Appearances = 0.0 to 10.0 THEN Ballond'OrNominee = No

IF Appearances = 10.0 to 20.0 THEN Ballond'OrNominee = No

IF Appearances = 20.0 to 30.0 THEN Ballond'OrNominee = No

IF Appearances = 30.0 to 40.0 THEN Ballond'OrNominee = No

IF Appearances = 40.0 to 50.0 THEN Ballond'OrNominee = Yes

IF Appearances = 50.0 to Infinite THEN Ballond'OrNominee = Yes

Incredibly, the accuracy in the validation set of that specific iteration was 100%. The confusion matrix for *b*=6, using 10-fold cross-validation or not using it, was exactly the same. By applying *SelectKBest* with the parameter *f_classif*, the results also remained the same.

* Correctly classified instances =59.62 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	5	21
Actual “No”	0	26

OneR can correctly predict all 26 players who were not candidates for the Ballon d’Or. However, it was difficult to correctly classify the nominee players.

Probably the main reason for that is that the generation of rules based on the values of a single one attribute is not enough to classify a Ballon d’Or nominee, that is, the characterization of the nominee is not só simple to be described.

However, using the *chi2* parameter, *b=6* using 10-fold cross-validation, the total average accuracy for all iterations was slightly lower at 86.69%, but the accuracy for the testing data drastically increased to 67.31%. Its confusion matrix is as follows:

* Correctly classified instances =67.31 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	9	16
Actual “No”	0	26

There were no significant difference with the confusion matrix using *f_classif* or not using any feature selection, since it could correctly guess all 26 non-nominee players, but it was not very effective to detect Ballon d’Or nominees, even though it improved.

No matter how we set the parameter *b*, the attribute chosen to generate the rules has always been the number of times a player starts playing a match (‘Appearances’), attribute (0). Astoundingly, even without using the 10-fold cross-validation or even without using the *SelectKBest* feature selection method, the chosen attribute was always ‘Appearances’, the attribute (0), no matter if *f_classif* or *chi2* is used as the function to calculate the score of each attribute.

Not only that, but the accuracy metrics, the classifying rules and the confusion matrix were exactly the same. In this case, no matter what value for *k* for *SelectKBest* is chosen, the results remain the same. Every variation applied to the algorithm does not seem to have any change in the results, except when using *Z*-score normalization before the feature selection.

It was found that the results of the confusion were very different from the previous results. For instance, using *b*=6, the total average accuracy for all 10-fold validations was slightly lower than 88.78% and their accuracy in the testing data was also lower, 50.00%.

* Correctly classified instances =50 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	26	0
Actual “No”	26	0

Using *Z*-score normalization before applying *SelectKBest* with the *f_classif* parameter, even if the statistical metrics dropped, specially the accuracy testing, the results were promising. After all, this time it could guess correctly all the Ballon d’Or nominees. However, it could not detect any non-nominee.

According to Weka software, we tried to replicate the same kind of experiments, however, it does have some limitations. One was that it could not use the set of rules

generated in the cross-validation stage to test it in the testing data. Then, instead we got only the accuracy of the selected set of rules from all iterations. For this, we chose to use a filter in Preprocessing section of Weka, which is called *Discretize*, to discretize the data. We set its parameters to discretize into 6 bins. The results were almost the same, since their percentage of Correctly Classified Instances was 88.95% and the set of rules was exactly the same as the algorithm developed by us. In this case, we used stratified 10-fold cross-validation and number of bins for discretization equal to 6.

To test a set of rules generated by Weka with the testing data created by us, we did not have to discretize the training data. Unfortunately, Weka has some incompatibilities to deal with categorical values (generated by the discretization of the data into bins) and the numerical values of the testing data. Weka can not interpret a categorical value from the transformed training data and associate it with a range of numerical values that would be used for the values of the training set. This is because it can not split the string name from the categorical value and find out what numeric interval it refers to. On the contrary, in our coded program, we were able to deal with that.

We then used the raw numeric value in the training and provided the testing data and obtained the following results:

Apps:

< 39.5	-> No
>= 39.5	-> Yes

* Correctly Classified Instances = 59.62 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	5	21
Actual “No”	0	26

Here, ‘Apps’ refers to attribute (0), the number of times a player starts playing a match. Working with numerical values, Weka discovers the best numerical value to create only two predicting rules. We set the *minBucketSize* parameter to its default value, 6. That is the minimum number of instances for discretizing the numeric values.

As it is possible to see in the confusion matrix, the results were exactly the same as the algorithm coded by us, when we did not use any normalization or when we used *f_classif* as the function for *SelectKBest.*

Using the Weka *Standardize* preprocessing filter, that is, Z-score normalization before doing the training and testing, the results obtained were exactly the same as that obtained with the algorithm coded by us. That is:

* Correctly classified instances =50 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	26	0
Actual “No”	26	0

In short, OneR brought results that go against the known knowledge about soccer that number of goals is the determining statistical metric for a player to be considered a Ballon d’Or nominee. On the contrary,it shows that the number of appearances of a player starting a match is the most important statistical metric to define whether a player will be nominated for Ballon d’Or.

III. Decision Tree

We did experiments on the program coded by us, where we used *scikit-learn* libraries. In order to compare Weka results with the results of our program, we had to adapt the Weka decision tree algorithm since it is based on C4.5 and not in CART.

Without using any kind of preprocessing method, we had the following results when we ran the program coded by us:

* Correctly Classified Instances = 71.15 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	11	15
Actual “No”	0	26

This was a large increase in accuracy over OneR experiments, even without Z-Score normalization and feature selection. However it is possible to see in the confusion matrix that the model remained with characteristics similar to those generated by OneR. Scikit-learn CART could perfectly predict all true negatives, but failed to perform well to predict the true positives. Only 11 of them were detected, that is, less than half, in the same way that the models generated by OneR behaved. Figure 3 shows the tree generated by *scikit-learn* CART.

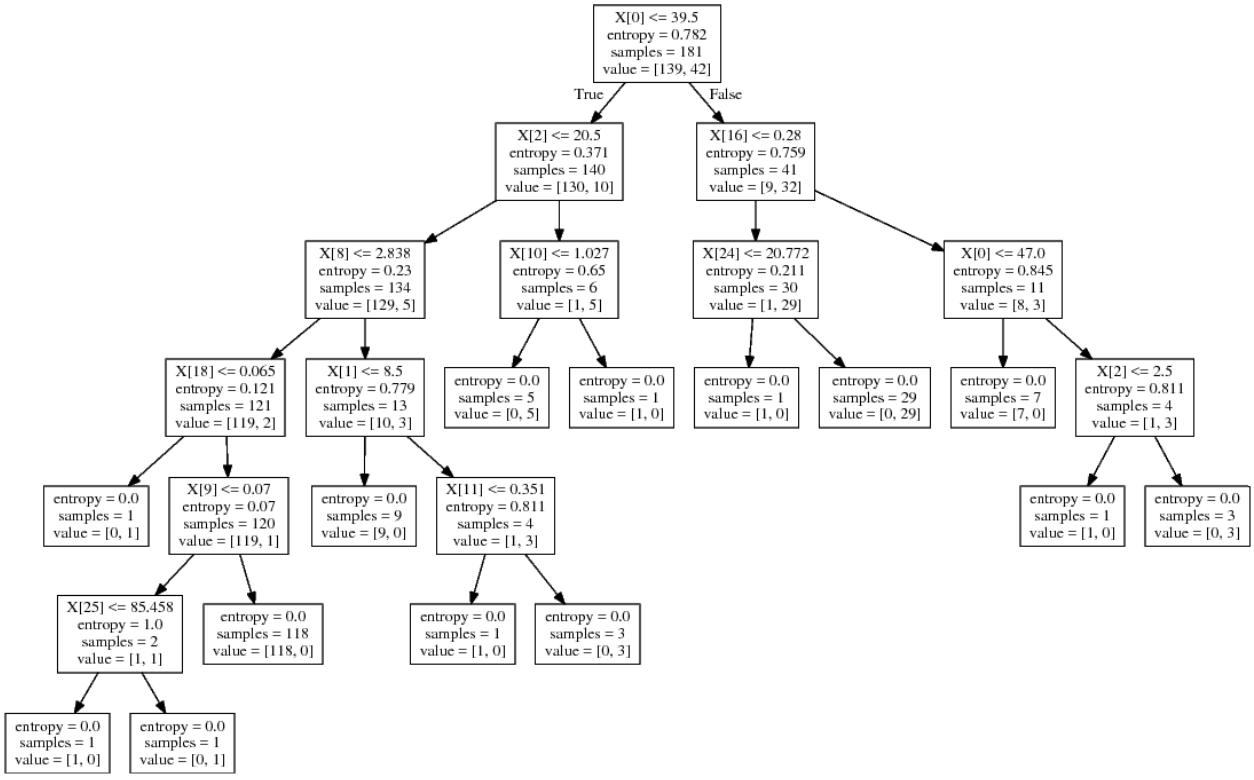

Figure 3: Visual representation of the decision tree generated by the program coded by us in Python programming language using the scikit-learn libraries. Here, no feature selection or *Z*-score normalization was performed before the training.

As it is possible to see in the decision tree of Figure 3, the first attribute to split is the attribute (0), the number of times a player starts playing a match, as it was stated earlier in Section II. It is worth mentioning that CART does no prune with our current algorithm configuration.

The other attributes chosen to split the nodes were from top to bottom:

- attribute (2), number of goals;
- attribute (16), average times he blocked a shot per game;
- attribute (8), average shots in the goal per game;
- attribute (10), average number of tackles per game;
- attribute (24), average number of passes per game;

- attribute (18), average number of keypasses per game;

- attribute (1), number of times the player came in from the bench;

- attribute (9), average aerial duels won per game;

- attribute (11), average number of pass interceptions per game;

- attribute (25), percentage of successful pass per game.

By using only *SelectKBest* as a preprocessing method, we decided to try different values of *k*, but specifically values around 11. This is because, in experiment with no preprocessing method, only 11 attributes were chosen to split the nodes. Therefore, there is no reason to set *k* as high as 25, for example.

First, we will analyze the results from setting the *chi2* (Chi-square) parameter to the *SelectKBest.* For k=5,6,7,8, we had accuracy lower than the previous CART experiment, fluctuating around 65%. For $k \geq 15$, we still had a decrease in accuracy compared to the previous CART experiment, where it fluctuates around 68%. For k=9,10,11,12,13,14, the accuracy was equal to or higher than 70%. The best accuracy in the testing was obtained with k=13, where the results are below:

* Correctly Classified Instances = 80.77 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	18	8
Actual “No”	2	24

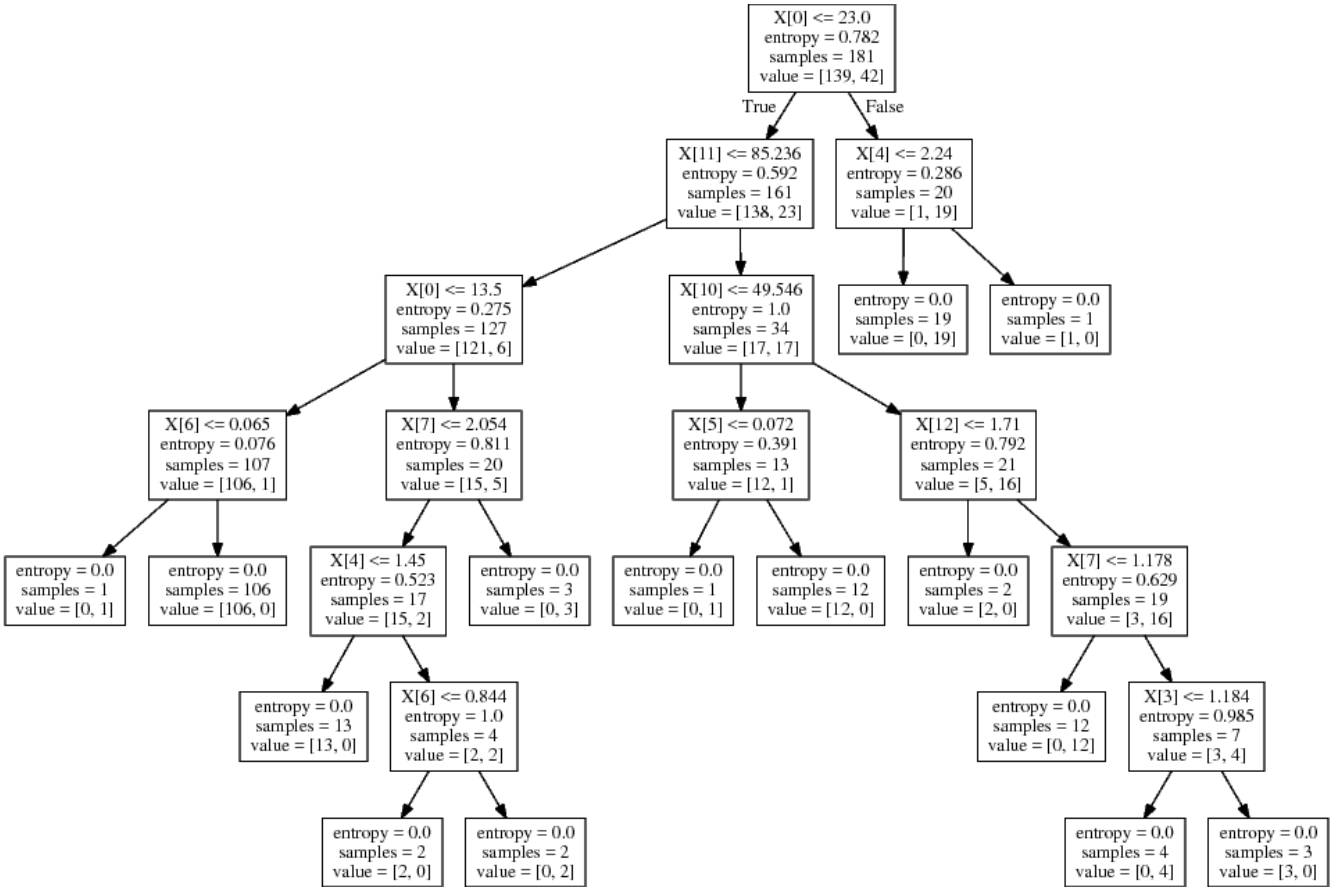

Figure 4: Visual representation of the decision tree generated by the program coded by us in Python, using scikit-learn libraries. Here, only feature selection using chi-squared function to calculate the score of each attribute was used. Here, k=13 for the SelectKBest feature selection function.

By using *f_classif* as parameter for *SelectKBest*, we were able to get worser results than with *chi2*. In fact, even worse than the first CART experiment. The accuracy fluctuated around 61% for any value of *k.* The best result was obtained with *k*=13:

* Correctly Classified Instances = 67.31 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	11	15
Actual “No”	2	24

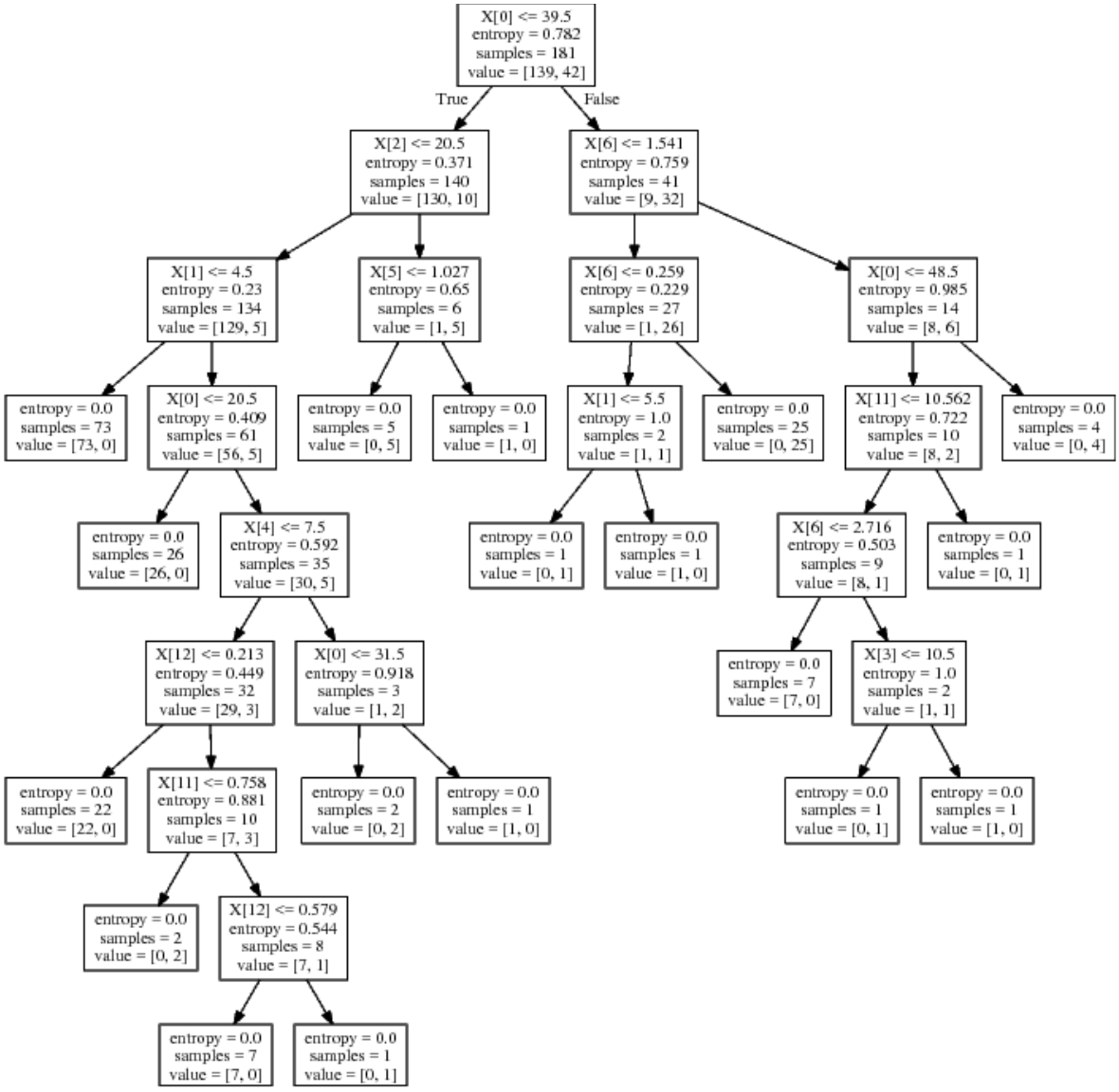

Figure 5: Visual representation of the decision tree generated by the program coded by us in Python, using scikit-learn libraries. Here, only feature selection using ANOVA F-value to calculate the score of each attribute was used. Here, *k=13* for the *SelectKBest* feature selection function.

Finally, we performed the experiment where we applied Z-score normalization before *SelectKBest.* However, the results were very disappointing as most of the values for *k* produced accuracy rates in the testing set of 50.0%. The best result was for *k=12*:

* Correctly Classified Instances = 51.92 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	26	0
Actual “No”	25	1

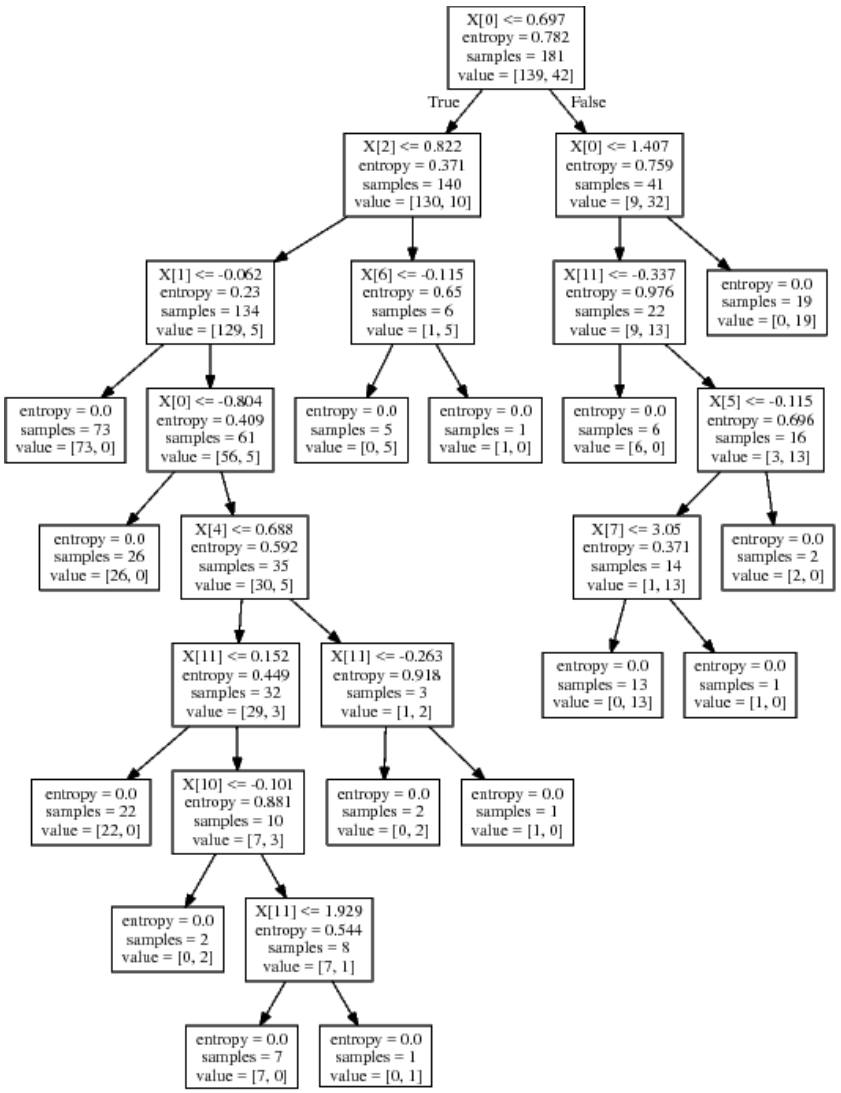

Figure 6: Visual representation of the decision tree generated by the program coded by us in Python, using scikit-learn libraries. Here, *Z*-score normalization was applied in the data set before applying feature selection, which used ANOVA F-value to calculate the score of each attribute was used. Here, *k=12* for the SelectKBest feature selection function.

Again, as in the previous OneR experiment with Z-score normalization, the generated model was good to predict true positives, all 26 instances in this case, but failed poorly in predicting the true negatives, only 1. For the other values of *k*, they were not able to get a single true negative.

At Weka, we chose to use the J48 algorithm to generate the decision trees, because it is based on C4.5 and has some similarities with CART. To make it more similar to CART, we have activated the parameter *Unpruned*, since CART does not prune the trees either.

Then, without using any Weka preprocessing method, we set the default parameters of J48 and obtained the following results:

* Correctly Classified Instances = 71.15 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	11	15
Actual “No”	0	26

```
Goals <= 22
|  Apps <= 41
|  |  Assists <= 8
|  |  |  Red <= 0: No
|  |  |  Red > 0
|  |  |  |  Goals <= 11: No
|  |  |  |  Goals > 11
|  |  |  |  |  Apps <= 30: Yes
|  |  |  |  |  Apps > 30: No
|  |  Assists > 8
|  |  |  SpG <= 2.9
|  |  |  |  TB <= 0.6: No
|  |  |  |  TB > 0.6: Yes
|  |  |  SpG > 2.9: Yes
|  Apps > 41
|  |  OffWon <= 0.641463
|  |  |  BlkShots <= 2.04: Yes
```

```
|   |   |   BlkShots > 2.04: No
|   |   OffWon > 0.641463: No
Goals > 22: Yes
```

Figure 7: Visual representation of the decision tree generated by Weka J48. Here, there was neither feature selection nor normalization.

As it is possible to see from the previous confusion matrix, J48 is good to predict true negatives, such as the decision trees given by our program, and it struggles to properly classify the true positives.

By applying the *Standardize filter* before running J48, we obtained similar results from our program when we applied Z-score normalization in the training data:

* Correctly Classified Instances = 53.85 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	26	0
Actual “No”	24	2

```
Goals <= 0.935882
|   Apps <= 0.812868
|   |   Assists <= 0.446482
|   |   |   Red <= -0.503933: No
|   |   |   Red > -0.503933
|   |   |   |   Trn <= -0.161462: No
|   |   |   |   Trn > -0.161462: Yes
|   |   Assists > 0.446482
|   |   |   SpG <= -0.102306
|   |   |   |   TB <= 2.025302: No
|   |   |   |   TB > 2.025302: Yes
|   |   |   SpG > -0.102306: Yes
|   Apps > 0.812868
|   |   OffWon <= -0.086301
|   |   |   BlkShots <= -0.066309: Yes
|   |   |   BlkShots > -0.066309: No
|   |   OffWon > -0.086301: No
Goals > 0.935882: Yes
```

Figure 8: Visual representation of the decision tree generated by Weka J48. Here, only Z-score normalization was applied in the training data.

One important aspect to notice in both trees generated by Weka is that the attribute which was chosen to be the root of the decision trees was the number of goals, attribute (2).

The other attributes chosen to split the nodes were from the top to the bottom:

-Apps: attribute (0), number of times a player started playing a match;

-Assists: attribute (3), total number of assists;

-Red: attribute (7), number of red cards received;

-Trn: attribute (23), average number of times he lost the ball – turn overs;

-SpG: attribute (8), average number of shots in the goal per game;

-OffWon: attribute (13), average number of times he won an offside per game;

-BlkShots: attribute (16), average times a player blocked a shot per game;

IV. Logistic Regression

First, we will present the classifiers generated by Logistic Regression with no preprocessing. In comparison with the OneR classifier, with no preprocessing, this performed slightly better:

* Correctly Classified Instances = 76.92 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	15	11
Actual “No”	1	25

The weights θ_i of each attribute *i* of the obtained hypothesis function were:

Apps,attribute (0): -0.0896512579183

SubIn,attribute (1): -0.213789836045

Goals,attribute (2): 0.124552431694

Assists,attribute (3): 0.252873708623

AverageGoalspermatch,attribute (4):-0.044132979153

AverageAssistspermatch,attribute (5):-.0495894673716

Yellow,attribute (6): 0.0542204850821

Red,attribute (7): 0.0170118179154

SpG,attribute (8): -0.00115065975868

AW,attribute (9): -0.0222022168281

Tackle,attribute (10): -0.125403971544

Int,attribute (11): -0.233378868554

Fouls,attribute (12): -0.127731974841
OffWon,attribute (13): 0.0518291587652
Clr,attribute (14): -0.14279934855
WasDribbled,attribute (15): -0.177817918935
BlkShots,attribute (16): -0.0472284139999
OG,attribute (17): -0.0379192704386
KeyPasses,attribute (18): -0.246934474823
Dribbles,attribute (19): -0.0558734777214
Fouled,attribute (20): 0.0704385343643
Offs,attribute (21): -0.0190723253703
Disp,attribute (22): 0.00787598410165
Trn,attribute (23): -0.0112693476882
Avg.Passes,attribute (24): 0.0152446752573
PassSuc%,attribute (25): -5.0585247178e-05
Crosses,attribute (26): -0.203654178301
LB,attribute (27): 0.257716874307
TB,attribute (28): 0.0408887908759

This was the first experiment among all the evaluated machine learning algorithms, without using any kind of preprocessing, where it performed relatively well in predicting true positives and true negatives. It was able to correctly classify 25 out of 26 true negatives and more than half of the true positives, 15.

Regarding the attribute weights, it is possible to see that the attribute with the highest weight value θ_i was attribute (27) LB, the average number of accurate long passes. Then, that is closed followed by attribute (3) Assists and by attribute (2) Goals. In 4th place, we have the attribute (20) Fouled, the average number of times a player is fouled per match. In 5th place, we have the attribute (6) Yellow, the total number of yellow cards a player has received in a season.

Then, we decided to use only the *SelectKBest* feature selection method to see if performance would improve even further. Unfortunately, the accuracy in the testing set for several values of *k* decreased to about 70% for the *f_classif* parameter.

For higher values of *k*, such as $k \geq 12$, the accuracy was better, around the same as the first Logistic Regression experiment, 76%. The best result found was for $k=13$ using *f_classif* as parameter for *SelectKBest:*

* Correctly Classified Instances = 78.85 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	15	11
Actual “No”	0	26

Here, we see that there has been a slight improvement in performance, since now it was able to perfectly predict all true negatives. However, the performance to detect the true positives remained the same, classifying more than half of the nominees correctly.

The selected attributes and their respective weights θ_i of the obtained hypothesis function were:

Apps,attribute (0): 0.0111982316675
SubIn,attribute (1): -0.0929594156751
Goals,attribute (2): 0.0996933242685
Assists,attribute (3): 0.192223983619
AverageAssistspermatch,attribute(5):0.70660587160
Yellow,attribute (6): 0.0869674209114
SpG,attribute (8): -0.00039233514483
AW,attribute (9): -0.0014341502202
Fouls,attribute (12): -0.00149380563039
Offs,attribute (21): 0.0093501369622
Disp,attribute (22): 0.00105222885208
LB,attribute (27): 0.00114216070071
TB,attribute (28): 0.00984643565113

It is possible to see that the attribute with the highest value of weight θ_i was attribute (3), Assists. It is then followed by attribute (2), Goals . In 3rd place, we have the attribute (0), Apps, the total number of times a player started playing a match. In 4th place, we have the attribute (28), TB, the average number of accurate ball passes.

By changing the parameter to *chi2*, the classifiers performed slightly worse than with the *f_classif* parameter. Another detail that we noticed was that, for very small *k*, such as *k*=5,6,7,8, the accuracy dropped to about 57% and the attribute with the highest weight was attribute (27), the average number of accurate long ball passes per match. For values greater than k, the accuracy increased to about 67% to 73% and the attribute with higher weight always remained as the attribute (2), total number of goals.

From all the previous experiments with the program coded by us, it was the first time that the attribute (2), Goals, was considered the most relevant attribute for a classifier

model. That is a great achievement for a classifier, since in soccer, the most important aspect of the game is scoring goals. The results for *k*=16 are shown as follows, where the accuracy achieved its maximum:

* Correctly Classified Instances = 78.85 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	16	10
Actual “No”	1	25

The best result using Feature Selection with parameter *chi2* is very similar to the best result using Feature Selection with *f_classif*. The selected attributes and their respective weights θi from the obtained hypothesis function are shown below:

Apps,attribute (0): 0.0506707059288

Goals,attribute (1): 0.29390276458

Assists,attribute (2): 0.159004344493

SpG,attribute (8): -0.241836968065

Tackle,attribute (10): -0.772517293095

Int,attribute (11): -0.197015352283

Fouls,attribute (12): -0.469273095401

Clr,attribute (14): 0.00162924326583

BlkShots,attribute (16): -0.039735344662

KeyPasses,attribute (18): -0.822165422067

Dribbles,attribute (19): -7.23320313241e-05

Fouled,attribute (20): -0.373782462666

Trn,attribute (23): -0.651245331231

Avg.Passes,attribute (24): 0.146103286381

PassSuc%,attribute (25): -0.100363583165

LB,attribute (27): 0.0412252310124

By applying Z-score normalization before the feature selection, we have noticed that the performance of Logistic Regression dropped even more. For most values of *k*, the accuracy remained 50% and stabilized around it. The results for *k*=10 are:

* Correctly Classified Instances = 51.92 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	26	0
Actual “No”	25	1

It is possible to see that the performance to detect true positives for k=10 has greatly increased, performing flawlessly. However, the classifier no longer detected the true negatives, detecting only 1 of them. For other smaller values of k, the classifier could not even detect a single true negative.

The selected attributes and their respective weights θ_i from the obtained hypothesis function are shown below:

Apps,attribute (0): 1.45382002462

Goals,attribute (2): 1.32574403033

Assists,attribute (3): 0.752794826885

Yellow,attribute (6): 0.51169220902

SpG,attribute (8): -0.0202936708261

AW,attribute (9): 0.160291411787

Fouls,attribute (12): 0.190964883888

Offs,attribute (21): 0.35479468468

LB,attribute (27): 0.497315968041

TB,attribute (28): -0.0813653864442

It is possible to see that the attribute with the highest value of weight θ_i was attribute (0), Apps. It is then followed by attribute (2), Goals. In 3rd place, we have the attribute (3), Assists. In 4th place, we have the attribute (27), LB, the average number of accurate long passes. In 5th and with a value very close to the 4th comes the attribute (21), Offs, the average number of times a player is caught offside. Using Z-score normalization, it is clear to see that the weight of the selected attributes is only higher in comparison with the weights obtained in previous Logistic Regression experiments.

V. Support Vector Machines

Although we were expecting a better accuracy with SVM when compared to all the previous experiments using the other algorithms, the results were not so better than the previous ones.

The results when not applying any sort of preprocessing method were:

* Correctly Classified Instances = 71.15 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	11	15
Actual “No”	0	26

Apps,attribute (0): 0.0691666460651
SubIn,attribute (1): 0.0372047861213
Goals,attribute (2): 0.0746694439984
Assists,attribute (3): 0.130409558566
AverageGoalspermatch,attribute(4):-0.0257449362893
AverageAssistspermatch,attribute(5):-0.150991421621
Yellow,attribute (6): -0.0016515270002
Red,attribute (7): -0.0389464619581
SpG,attribute (8): -0.000560025090302
AW,attribute (9): 0.0931713095751
Tackle,attribute (10): -0.261224217197
Int,attribute (11): -0.892310084742
Fouls,attribute (12): -0.0345770328897
OffWon,attribute (13): 0.0800112926712
Clr,attribute (14): 0.0401774928635
WasDribbled,attribute (15): -0.662987773041
BlkShots,attribute (16): 0.380578923743
OG,attribute (17): -0.285385088073
KeyPasses,attribute (18): -0.544282754791
Dribbles,attribute (19): -0.158342815635
Fouled,attribute (20): 0.340399629064
Offs,attribute (21): -0.816133377578
Disp,attribute (22): -0.0857702135888
Trn,attribute (23): -0.0747773595098
Avg.Passes,attribute (24): 0.0421985638021
PassSuc%,attribute (25): 1.57686866942e-06
Crosses,attribute (26): -0.319002176927
LB,attribute (27): 0.168812078952
TB,attribute (28): 0.52579820497

According to the values of the 5 largest coefficients, the most relevant attributes are in this order: attribute (28), attribute (16), attribute (20), attribute (27) and attribute (3).

The accuracy in the testing set was reasonable, but it was no better than Logistic Regression and was equal to the Decision Tree algorithm. The classifier has similar characteristic to the previous ones, such as the Decision Trees: good at predicting true negatives, but not good at predicting true positives.

Applying the feature selection with *f_classif*, we noticed that the performance improved slightly, but incredibly remained high even for low values of k as $k \leq 9$, where the accuracy was around 71% to 73%.

The best results that we obtained, for k=12 using *f_classif* as parameter for the method *SelectKBest*, were:

* Correctly Classified Instances = 75.00 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	13	13
Actual “No”	0	26

Apps,attribute (0): 0.0305908098896

SubIn,attribute (1): -0.0268005083012

Goals,attribute (2): 0.0615295918123

Assists,attribute (3): 0.189554157237

Yellow,attribute (6): 0.162572454409

Tackle,attribute (10): -0.879016612217

Fouls,attribute (12): -0.306479133393

Dribbles,attribute(19):0.00166183872254

Trn,attribute (23): -0.239214509481

Crosses,attribute (26): -0.99210984003

LB,attribute (27): 0.286536702973

TB,attribute (28): 0.229336448984

There has been an improvement over the first SVM experiment, as the classifier can now predict with 50% of reliability to distinguish true positives from false negatives. The

highest coefficient was from the attribute (27), LB, the average number of accurate long ball passes per match and attribute (28), TB, the average number of accurate ball passes per match. The attribute (2), Goals, was only the 5th largest coefficient among the selected attributes.

When exchanging the parameter to *chi2*, the results remained similar to the SVM experiment with *f_classif*, because for most of values of *k* the accuracy fluctuated very little around 71% to 73%. The results for *k*=19 are shown below, the best result we got with the lowest possible *k* using *SelectKBest* with *chi2*:

* Correctly Classified Instances = 76.92 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	14	12
Actual “No”	0	26

Apps,attribute (0): 0.067671232728

Goals,attribute (1): 0.156197251022

Assists,attribute (2): 0.135797222985

SpG,attribute (8): -0.000611496143071

Tackle,attribute (10): -0.157172057037

Int,attribute (11): -0.389386470346

Fouls,attribute (20): -0.609677357671

OffWon,attribute (13): 0.388556695879

Clr,attribute (14): 0.00922404729827

WasDribbled,attribute (15): -1.00494778692

BlkShots,attribute (16): 0.110227469587

KeyPasses,attribute (18): -0.69535878156

Dribbles,attribute (19): 0.00265564936444

Fouled,attribute (20): -0.112983509619

Disp,attribute (22): -0.0264271147589

Trn,attribute (23): -0.282688310721

Avg.Passes,attribute (24): 0.0917683211418

PassSuc%,attribute (25): -0.034247014047

LB,attribute (27): 0.00276411919278

It is interesting that by applying feature selection with the function *chi2* as the score of an attribute, the best accuracy in the testing set was achieved with a very high number of selected attributes. Compared to all other classifiers shown in this report so far, the optimal values of *k* were around 12 and 13. However, this compensated a bit because the accuracy increased slightly compared to previous SVM experiments.

A strange behavior occurred with the SVM classifiers generated when the Z-score normalization was applied before the feature selection. For all results, no matter what value was set for *k*, the accuracy and the confusion matrix remained exactly the same. Even when only Z-score normalization and no feature selection were applied, the results remained the same:

* Correctly Classified Instances = 50.00 %

m=52	Classified as “Yes”	Classified as “No”
Actual “Yes”	26	0
Actual “No”	26	0

Apps,attribute (0): 1.56815577613

SubIn,attribute (1): 0.143726455708

Goals,attribute (2): 1.02114616628

Assists,attribute (3): 0.389867650195

AverageGoalspermatch,attribute(4): 0.0158012120793

AverageAssistspermatch,attribute(5): -0.0156668974297

Yellow,attribute (6): 0.230149845737

Red,attribute (7): 0.0474276358559

SpG,attribute (8): -0.207892012549

AW,attribute (9): -0.902180621659

Tackle,attribute (10): -0.0110217393071

Int,attribute (11): -0.0109137540541

Fouls,attribute (12): -0.044968350705

OffWon,attribute (13): -0.0104289342007

Clr,attribute (14): -0.0383385789957

WasDribbled,attribute (15): -0.129640822321

BlkShots,attribute (16): 0.00332567581064

OG,attribute (17): -0.536709058974

KeyPasses,attribute (18): -0.000339580609229

Dribbles,attribute (19): -0.0335981914283

Fouled,attribute (20): -0.0278680223573

Offs,attribute (21): -0.506999763484

Disp,attribute (22): -0.043148488996

Trn,attribute (23): -0.0742089254777

Avg.Passes,attribute (24): 0.0437469568713

PassSuc%,attribute (25): 0.00703605950738

Crosses,attribute (26): -0.162623385914

LB,attribute (27): 0.241753576979

TB,attribute (28): 0.417234197656

For some reason, the highest coefficient in all cases has always been the attribute (0), the number of times a player starts playing a match.

5. Conclusion

In this research, we obtained several classifiers using five Machine Learning methods, namely ZeroR, OneR, CART (Decision Tree), Logistic Regression and Support Vector Machines to gain insights into players who are nominated for the Ballon d'Or award each year.

We obtained common-sense rules and surprisingly unexpected rules. This was interesting because it involved some discussion in preconceived concepts and ideas regarding modern soccer as the prominence of a player on a team being a regular starting XI in order to increase his chances of being nominated Ballon d'Or, for example, according to the results of OneR and Decision Trees, where attribute (0) was the chosen attribute of OneR and the root node for Decision Tree CART.

The same occurred for SVM with Z-score normalization, where the attribute (0) had the highest coefficient associated with all values of *k* for *SelectKBest*. But this experiment

and all other experiments using Z-score normalization are not reliable, since their accuracy barely exceeds 50% for all algorithms.

A strange pattern in the results we have seen using Z-score normalization is that no matter which algorithm do you use, the accuracy in testing set was always low around 50% and it could not detect true negatives, but could detect very well true positives.

Another pattern in the results that we noticed in all algorithms used in this research, except ZeroR, is that they tend to be very consistent and often perfect for detecting true negatives. However, they are not good at detecting true positives, most of the times they failed to detect 50% of true positives in the testing set.

References

[1] FIFA. FIFA Survey: approximately 250 million footballers worldwide, 2000. Available from: http://www.fifa.com/aboutfifa/organisation/news/newsid=88048/index.html [Retrieved 2018-30-06].

[2] 2016 Ballon D'Or. Available from: https://en.wikipedia.org/wiki/2016_Ballon_d%27Or [Retrieved 2018-30-06].

[3] B. Wang and L. Wang, *Research of Association Rules in Analyzing Technique of Football Match*, presented at Second International Conference on Power Electronics and Intelligent Transportation Systems, 178-180, 2009.

[4] R. Agrawal and R. Srikant, *Fast Algorithms For Mining Association Rules*, presented at 20th VLDB Conference,487-499, 1995.

[5] B. Chai, *Time Series Data Mining Implemented on Football Match*, Applied Mechanics and Materials, vol. 26-28, pp. 98-103, 2010.

[6] S. Nunes and M. Sousa, *Applying Data Mining Techniques to Football Data from European Championships*, presented at Conferência de Metodologias de Investigação Científica (CoMIC'06), 4-16, 2006.

[7] Weka 3 - *Data Mining with Open Source Machine Learning Software in Java*. Available from: http://www.cs.waikato.ac.nz/ml/weka/ [Retrieved 2014-11-08].

[8] C. E. Weatherburn, *A First Course in Mathematical Statistics*, Cambridge, pp. 164-182, 1968.

[9] D. Rumsey, *Statistics II for Dummies*, Wiley Publishing, pp. 151-218, 2015.

[10] T. Mitchell, *Machine Learning*, McGraw-Hill, pp. 52-78, 1997.

[11] J. R. Quinlan, *Induction of Decision Trees*, pp 81-106,1986.

[12] Decision Trees. Available from: http://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart [Retrieved 2018-30-06].

[13] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, pp. 137-147, 2007.

[14] T. Hastie, R. Tibshirani and J. Friedman, *Elements of Statistical Learning*, Springer, Second edition, pp. 241-245, 2009.