# Machine Learning for Predicting Soccer Matches

Daniel Medina Sada, *M.S. Student, Texas Tech University*

**Abstract**—In this paper, we attempt to create a system to predict the outcome of the soccer matches for a home team. For our data, we use a dataset that contains over 25,000+ matches, with teams along with the attributes describing the type of attack and defense of the team. The teams are from different European leagues. The dataset also contains data of over 10,000+ players and their attributes and the betting odds for each game from many different betting sites. Although the dataset does have all this information, for simplicity and brevity, we only focus on the following data: home team goal difference, away team goal difference, games won by home team, games won by away team, games won against the away team, games lost against the away team, and the overall rating of each individual player form both home and away team. We explore different approaches using different classifiers and techniques to achieve the highiest accuracy possible. In all of our approaches, we trained 4 different classifiers. We use Support Vector Classifier, Nave Bayes, K Nearest Neighbors and Logistic Regression and we use 88.5% of the data for training and 11.5% for testing. The first approach considers all of our matches to train the classifiers. The second approach, we split the matches by league and train the classifiers to predict the outcome of the matches depending on the league its played on. The third approach, we use all of the matches to train the different classifiers. For this approach we train a Bernoulli Nave Bayes classifier and use the result of all of the classifiers to compare their results and choosing the majority or K Nearest Neighbor if there is no majority.

**Index Terms**—Machine Learning, Prediction, Classifiers, Nave Bayes, K Nearest Neighbor, Logistic Regresion, Support Vector Classifier, Voting Classifier.

## 1 Introduction

The prediction of soccer matches has been an interest to many people. However, there hasnt been a system that is able to get a high precision. Achieving the goal of correctly predicting soccer matches could be of huge interest to many companies running betting sites or casinos. Although achieving the goal, of creating a system with high precision, is a great challenge due to the unpredictablility characteristic of soccer. If the same game was to be executed twice, it could easily have two different outcomes in the number of goals scores. This means that by only trying to predict whether a game will be won, lost or tied we can achieve a higher accuracy than predicting the score, and could make such a system plausible for achieving a higher accuracy. This is what we will attempt in this paper, to create a system that predicts if the home team will win, lose or draw a game.

May 9th, 2017

## 2 Explaining the Dataset

### 2.1 About

For this system, we downloaded a dataset from kaggle, by Hugo Mathien [1]. The dataset has information of over 25,000+ matches and over 10,000+ players. The matches of the dataset are from 11 different European countries with their top leagues from the 2008-2009 season to 2015-2016 season and are always in reference to the home team. The dataset contains the following match data for each team: goal difference, games won, games won against opposing team, the starting players, players who scored, players booked and betting odds for that game, from different sources. The data for the teams contains information about line-ups, type of attack, type of defense, type of build play and type of chance creation. The player data contains the overall rating and the rating for each of the attriubtes used in the videogame FIFA, from EA Sports. The information for the teams, as well as for the players was obtained from the videogame FIFA.

### 2.2 Features

In this paper we only focus in a certain set of features from our dataset rather than using all of it. The features that we take into account are: Home team goal difference, away team goal difference, games won home team, games won away team, games against away team won, games against away team lost, overall rating of individual players of home team and overall rating of individual players of away team. All of the data is from the beginning of the last season up to the match that is being predicted. We do not use the betting odds provided by the dataset since they do not influence how a team performs in a game. Even though the betting odds provides more information about a match, we want to analyze only data that directly influences the result of a match. We also discard many of the team attributes, like formation, type of attack, type of defense, etc., since they are labels in the dataset rather than a number and there for cannot be quantifiable in their current state.

• *Daniel Medina Sada is a Masters Student undergoing his studies of Software Engineering in the department of Computer Science at Texas Tech University.*
*E-mail: daniel.medina-sada@ttu.edu*

*Manuscript received May 9th, 2017*

### 2.3 Training Set

For our training set, we use the features mentioned above (Section 2.2). It was decided use 88.5% of our matches for training in order to use a high number of games for training. Due to some matches not having complete data, our matches got reduced to 21,364 matches.

### 2.4 Testing Set

For our testing set, we use the rest of our data, which is the remaining 11.5%.

## 3 Tools

In order to conduct our experiments, we used Python 3 to create our system. Also, we took advantage of the Scikit Learn [2] library for python to use their classifier classes. Also the pandas [3] library for python was used to import the data into a DataFrame and handle all the data from our dataset.

## 4 Classifiers

In order to attempt to get the highest accuracy possible, with out features, serveral classifiers were trained and tested to get their accuracy and be able to select the most precise classifier. Scikit learn allows us to simplify the training of the classifiers. First the data is put into a DataFrame and we extract all the features into a Dataframe x, which doesnt have the classification of that match, that is win, draw or defeat. The classification is stored into an array y such that the first element in the array y contains the result of the match in the first element of the DataFrame x. The variables x and y will then be used to train our classifiers.

### 4.1 Calculating the Accuracy

To calculate the accuracy of classifiers, we create a function called get_accuracy (Listing 1) that takes in as parameters the name of the classifier, the instance of the classifier, the test size and the test set. We then user our classifier and predict the outcome and compare it to the result in the test set. If the outcome and the result are the same we add 1 to a variable x and after checking all the test set we divide x by test set size.

Listing 1. This function is used to calculate the Accuracy of the classifiers

```
def get_accuracy(name, clf, test_size, compare):
    pred = clf.predict(pred.tail(test_size))
    x = 0
    for i in range(0,test_size):
if pred[i] ==
      str(compare['label'][len(compare)
      -test_size+i]):
            x = x+1
    accuracy = x/test_size
    print(name + ":", accuracy)
```

### 4.2 SVM

First, we use scikit learns SVM library to instantiate the SVM classifier. We use scikit learns default settings for the SVM and use the function fit(X,y) to train it, by feeding the function with our x and y variables.

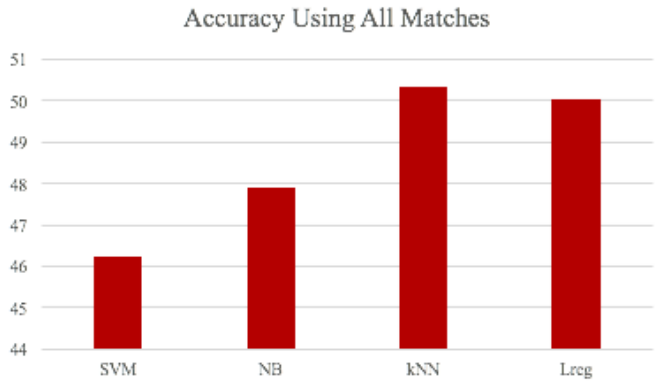


Fig. 1. Display for the results of the accuracy of SVM, Nave Bayes, K Nearest Neighbor and Logistic Regression, respectively. The results are measure in percentage.

### 4.3 Naive Bayes

The second classifier used is a Nave Bayes Classifier. It is also instantiatied with scikit learns default settings and trained with the same fit(X,y) function and x and y variables.

### 4.4 K Nearest Neighbors

The third classifier is k nearest neighbor. For this classifier we train for k = 5, k = 7, k = 13, k = 21 and k = 23 to choose the k that gives us the highest accuracy. After testing for all of the ks, the lowest k nearest neighbor classifier was k = 5. At k = 21 the kNN classifier peaked and in k = 23 dropped. So we focused on the k = 21 classifier, since it was the most reliable.

### 4.5 Logistic Regression

Finally, we used the Logistic Regression classifier. Just like the SVM and the Nave bayes, we instantiated this classifier with the default settings and trained it with the fit(X,y) function and x and y variables.

## 5 First Approach

My first approach into creating a predicting soccer match system was to obtain the freatures from the dataset and use all of the matches to train and test my classifiers. The four classifiers, SVM, Nave Bayes, k Nearest Neighbor and linear regression were all trained with the 88.5% of the matches in the dataset, that were not missing any any information. 18,916 matches were used for this training and the 2,448, 11.5%, other matches were used to test the classifiiers and get the accuracy. For this approach, the results for the classifiers were 46.25% for SVM, 47.88% for the Nave Bayes classifier, 50.32% using k Nearest Neighbor and 50.04% using Logistic Regression ( Figure 1). With this approach, we are able to get a max accuracy of 50.32% when classifying matches from any of the top leagues in Europe although the precision of deciding win for every match in this approach is 46.25%. Although this doesnt seem like a very efficient system, 50.32% is a good percentage for the first approach due to the uncertainty of matches in soccer.

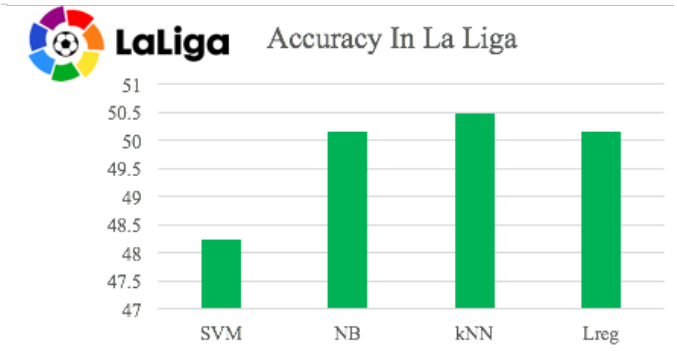


Fig. 2. Display for the results of the accuracy of SVM, Nave Bayes, K Nearest Neighbor and Logistic Regression, respectively in the prediction of matches within the Spanish league, La Liga Santander. The results are measured in percentage.

## 6 The Second Approach

In this second approach, we try to increase the accuracy of the classification with the hypothesis that different leagues have different playing styles and different league configurations and many other things that change between them. Instead of mixing all the matches and leagues together, lets try and predict the matches for each individual league and evaluate the results.

### 6.1 La Liga Santander

In this second approach, we first take a look at the Spanish leage, La Liga Santander. From our dataset, we extract all the matches from 2008 to 2016 that belong to this league and we use them to, again, train our SVM classifier, Nave Bayes classifier, k Nearest Neighbor classifier and the Logistic Regression classifier. Out of our dataset, we got 2,707 matches which 2,396 (88.5%) are used to train the classifiers and 311 (11.5%) matches are used for testing. The results we achieved were the following: SMV scored 48.23%, Nave Bayes scored 50.16%, k Nearest Neighbor scored 50.48% and lastly Logistic Regression scored 50.16% ( Figure 2)With this approach, in this league we were able to increase the accuracy of all of the classifiers. However, our max score was 50.48%, which is a small improvement form our past score of 50.32%.

### 6.2 Barclays Premier League

We now analyze the English league, Barclays Premier League.Same as above, we extract all of the matches from our dataset that belong to this league.The matches totaled 2,962 from 2008 to 2016. Again, we use our 88.5% split for training and 11.5% for testing, which are 2,621 matches used for training and 341 matches used for testing. We run our get_accuracy function with out testing set and we get the following results for our classifiers. SVM achieved a 41.64% accuracy. Nave Bayes achieved an accuracy of 43.98%. K Nearest Neighbor scored an accuracy of 41.48%. Lastly, Logistic Regression guessed correctly 41.35% of the time ( Figure 3). Our results for predicting this league really dropped from our last result from La Liga. However, this was expected. The Barclays Premier League is the most unpredictable league. It is one of its known characteritics.

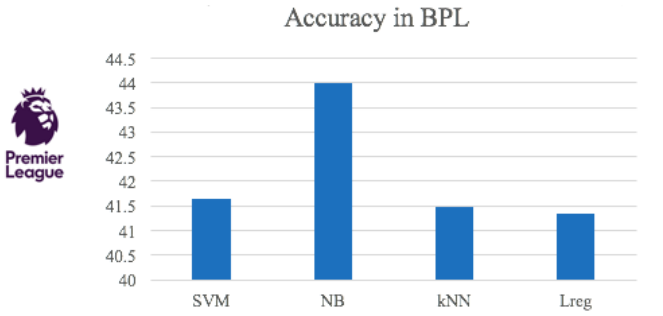


Fig. 3. Display for the results of the accuracy of SVM, Nave Bayes, K Nearest Neighbor and Logistic Regression, respectively in the prediction of matches within the English league, Barclays Premier League. The results are measured in percentage.

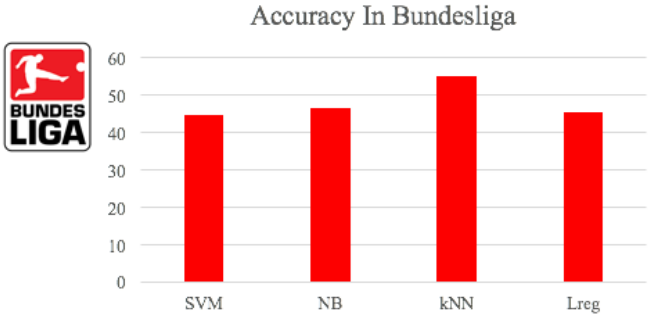


Fig. 4. Display for the results of the accuracy of SVM, Nave Bayes, K Nearest Neighbor and Logistic Regression, respectively in the prediction of matches within the German league, Bundesliga. The results are measured in percentage.

As for example we have the Team Leicester City, who finished the season of 2014/2015 in 14th place out of 20 teams. The next season, 2015/2016 they ended in 1st place, becoming the champions of that season. Today, in the season of 2016/2017 they are back in 9th place, thus showing the unpredictability of this league, therefore, making it very hard to predict it.

### 6.3 Bundesliga

Next, we take a look at the German league, Bundesliga. Form the dataset we extract a total of 2,286 games. To this games we do our 88.5% and 11.5% split which is 2,102 matches for training and 274 matches for testing, respectively. One again, we use our get_accuracy function on the classifiers: SVM, Nave Bayes, k Nearest Neighbors and Logistic Regression. For SVM, we achieved an accuracy of 44.89% predicting matches. Using Nave Bayes, we raise that accuracy to 46.71%. With k Nearest Neighbors we get our max accuracy of 54.98%. Finally, with Logistic Regression we score a 45.25% in accuracy ( Figure 4). By analyzing this league individually, we achieved a great accuracy of 54.98% with the k Nearest Negihbors classifier, which means this league is more easily predictable and being able to build our desired system should be more viable if we only focus in this league.

### 6.4 Serie A

The last league we analyze individually is the Italian league, Serie A. From our dataset we are able to extract 2,747

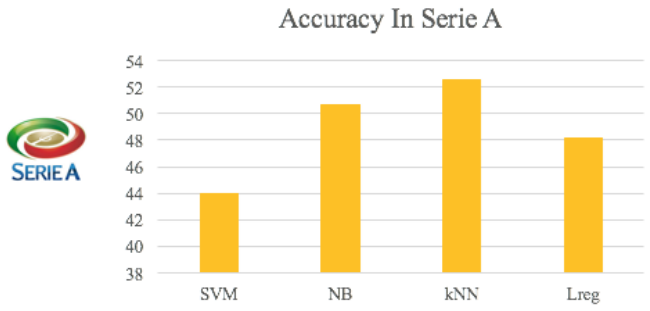


Fig. 5. Display for the results of the accuracy of SVM, Nave Bayes, K Nearest Neighbor and Logistic Regression, respectively in the prediction of matches within the Italian league, Serie A. The results are measured in percentage.

matches of the league that we can use. Once more, we train our classifiers with 88.5% of the data for training and 11.5% of data for testing, which equilize to 2,431 matches and 316 matches, respectively. Training our classifiers, we get the following results: SVM predicts 43.98% of the matches accurately, Nave Bayes predicts 50.63%, k Nearest Neighbor predicts 52.53% and lastly Logistic Regression scores 48.10% for correct predictions ( Figure 5). With this league we are able to obtain a max accuracy of 52.53% of accuracy. Again we can see that this league is more predictable than the English Premier League and the spanish league, La Liga Santander, therefore increasing the viability of our desired system.

## 7 Third Approach

In this third approach, we try to further improve our accuracy of predicting all soccer matches by using the results of all of our classifiers.

### 7.1 Voting

The voting algorithm takes in as parameters, a list with the instances of the classifiers and the input to the classifiers. Then, we compare the results of all of our classifiers, with our algorithm and we select the result that is a majority. However, by having only four classifiers we are prone to having our classifiers tie in the decision making, so we add a new classifier. We train a Bernoulli Nave Bayes classifier, form our scikit learn library and again train it using the fit(X,y) function with our x and y variables. Now that we have 5 classifiers, the probability of having a tie in the decision of our prediction should be significantly lower. However, since it can still happen, we decide that when this happens we just choose the result from K Nearest Neighbors classifier since it turned to be the most accurate all of the time. Again we use 88.5%, 18,916 matches, of all of our data to train the classifiers and we use 11.5%, 2,448 matches, to test our voting classifier. After using our get_accuracy function to test our voting classifier we en up with an accuracy of 37.5% ( Figure 6). This meaning that this approach is not good and should not be used as the means to predict a soccer match.


Fig. 6. Display for the results of the accuracy of SVM, Nave Bayes, Bernoulli Nave Bayes, K Nearest Neighbor and Logistic Regression and our voting classifier, respectively in the prediction of any match. The results are measured in percentage.

Listing 2. Voting Classifier Algorithm

```
def Voting(clfs, inp):
    preds = []
    decisions = []
    vote = []

    for clf in clfs:
        pred = clf.predict(inp)
        # print(pred)
        preds.append(pred)

    for i in range(len(preds[0])):
        new = []
        for clf in preds:
            new.append(clf[i])
        decisions.append(new)

    for d in decisions:
        if(d.count('Win') > d.count('Draw') and
           d.count('Win') > d.count('Defeat')):
            vote.append('Win')

        elif (d.count('Draw') > d.count('Win') and
              d.count('Draw') > d.count('Defeat')):
            vote.append('Draw')

        elif (d.count('Defeat') > d.count('Draw')
              and d.count('Defeat') >
              d.count('Win')):
            vote.append('Defeat')

        else:
            vote.append(d[2])

        print("kNN Outcome:", d[3])

    return vote
```

## References

[1] Hugo Mathien. (2016). European Soccer Database (2). Retrieved from https://www.kaggle.com/hugomathien/soccer.

[2] Scikitlearn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 28252830, 2011.
[3] NTLK, Bird, Steven, Edward Loper and Ewan Klein (2009),Natural Language Processing with Python. OReilly Media Inc.