Soccer Result Prediction using Machine Learning

Parul
Indian Institute of Technology Ropar
Ropar, Punjab
2016csb1048@iitrpr.ac.in

Prerna Garg
Indian Institute of Technology Ropar
Ropar, Punjab
2016csb1050@iitrpr.ac.in

Komal
Indian Institute of Technology Ropar
Ropar, Punjab
2016csb1124@iitrpr.ac.in

ABSTRACT

Prediction of sport results, stock prices etc. are one of the fastest expanding areas necessitating good predictive accuracy. The prediction of sport tasks depends on a large number of factors like the team attributes, geographical properties of the scheduled game, player attributes etc. This project aims to harness various classification techniques in machine learning that can serve the prediction purpose for soccer. Classification has been done using Artificial Neural Networks, Support Vector Machines and Logistic Regression (built from scratch) and the relative performance of these models has been compared. Appropriate means of evaluation have been used and the models have been tested on three different data sets with varying properties. The best prediction results were obtained using Artificial neural network with test accuracy of 70%.

KEYWORDS

*sport prediction, machine learning, neural networks, soccer

ACM Reference Format:
Parul, Prerna Garg, and Komal. 2018. Soccer Result Prediction using Machine Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Football is one of the most popular sports across the world and betting in this game is a 40 billion dollar industry. Although the number of possible outcomes is 3 (Home Win, Draw, Away Win) but the result depends on a large number of features such as the the last team meetings, rivalries, offensive and defensive skills, characteristic abilities of teams, weather, stadium location, positioning strategies of the team players, and even the psychological impact of fans in the stadium. Therefore appropriate feature selection is one of the most crucial prerequisites for building a "good" classifier. In the following sections we present a detailed description of data pre-processing and hyper parameter tuning along with a comparative study of various models.

2 RELATED WORK

In this paper[1] the authors have described comparison between various approaches for predicting the results of football matches using Bayesian Networks, KNN, ANN, decision trees etc. They predict the outcome of Champions League matches. They have included time series data where they take the current form of teams on the basis of their last 6 games together. This paper approaches the problem with similar classifiers. But the dataset used by these authors is very small, they have considered only less than 100 instances. It is not possible to inculcate the dynamicity of a worldwide famous sport in only 100 instances. Our first dataset has quarter to million instances (>7,80,000) and approximately 100 features.
In this paper [2] the authors have tried various ML techniques and compared the results. Our work is also build on the similar line.

3 METHODOLOGY

3.1 Pre-Processing of Datasets

We have used three different datasets to test our models. The first one is European soccer database that consists of eight different tables(Team attributes, Players, Matches, Leagues and so on..) linked through foreign keys. We de-normalized the database and converted it into a single .CSV file to serve as input to our models. A major part of pre-processing was feature selection. There were more than 200 features in the original dataset out of which we manually selected 76 features that were relevant to predicting the result of a particular match. After denormalization the dataset has >7,80,000 instances. The second database has 6000 instances of English Premiere League from the year 2001 to 2018. The third database is International football results from 1872 to 2018 which contains 40,000 instances. Some of the important selected features of these three datasets are:

(1) Team Attributes like dribbling speed, build up speed, team aggression etc.
(2) Match Attributes like season, location of stadium , tournament type etc.
(3) Game attributes like red cards, yellow cards, fouls committed etc.

- **Class Imbalance Problem**: In all the three data sets the number of observations belonging to one class is significantly lower than those belonging to the other class (70:30 ratio). To solve this issue we have used down sampling of data where we choose a subset of the majority class such that the sizes of both the classes are equal. Thus the data sets share 50:50 ratio of both positive and negative class.
- **Indicator features**: The features of our data, for example country, league id etc. have 10-15 possible values, which don't have any ordered relation; hence, if any feature has k different possible values, then rather than representing them as 1 to k, we make k different indicator features, each of which is binary and is 1 if the original feature has value corresponding to the indicator feature.

Conference'17, July 2017, Washington, DC, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

3.2 Model Learning

3.2.1 ***Support Vector Machine (SVM)***. : Support Vector Machine (SVM) is a discriminative classifier which learns a separating hyperplane and this hyperplane is then used to categorise new examples. The optimal hyperplane is the one which maximises the margin between two different classes such that each class lies on either side. Here, we have labelled data which is fed to SVM model and the model learns the optimal hyperplane by updating its weights through gradient descent. Support vectors are data points that are closest to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. The stopping criteria is set using number of iterations. The loss function that helps maximize the margin is hinge loss:

$$C(y_i, f(x_i)) = \begin{cases} 0 & if\ y_i * f(x_i) \geq 1 \\ 1 - y_i * f(x_i) & otherwise \end{cases} \tag{1}$$

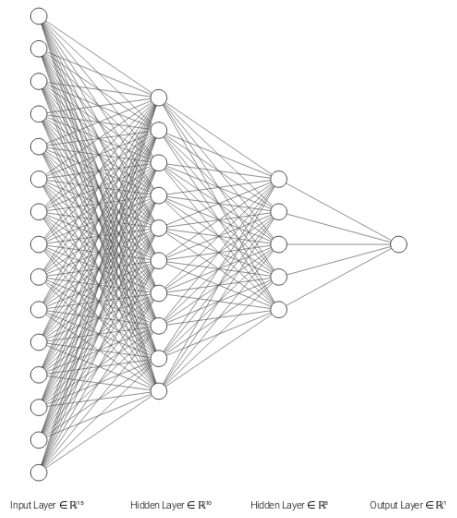
After adding the regularization parameter, the cost function looks as below.

$$\lambda\ ||W||^2 + \sum_{i=1}^{N} \big(1 - y_i * f(x_i)\big)_+ \tag{2}$$

where λ is the regularization parameter. The weights are updated for each training instance as given by the equation:

$$W_{t+1} = \begin{cases} W_t - \alpha * \big(2\ \lambda\ W_t\big) & if\ y_i * f(x_i) \geq 1 \\ W_t - \alpha * \big(2\ \lambda\ W_t - Y_i * X_i\big) & otherwise \end{cases} \tag{3}$$

3.2.2 ***Artificial Neural Network (ANN)***. : Artificial Neural Network is a machine learning framework that processes complex data inputs to learn to perform certain tasks. It is not programmed with any task-specific rules. However it involves large amounts of experimentation and parameter tuning to arrive at the best results. An ANN is a collection of neurons in the form of several layers. For our task we are using some variations of the following model :



Input Layer $\in \mathbb{R}^{15}$
Hidden Layer $\in \mathbb{R}^{10}$
Hidden Layer $\in \mathbb{R}^{5}$
Output Layer $\in \mathbb{R}^{1}$

Figure 1: Abstract ANN architecture

3.2.3 ***Logistic Regression (LR)***. : Logistic Regression is a classification model which is used in case the output is binary. Logistic regression is better than Linear regression in such cases due to the application of sigmoid function to the product of parameters learnt with the input, which reduces the probability of any example being near the threshold. The output is not the value of the class but the probability that the example will fall in that class. The parameters can be learnt either through Gradient descent or Newton-Raphson methods. Here, the labelled data has been fed to the model and the randomly initialised parameters have been learnt through gradient descent. The loss function used in this method is called the Cross-entropy loss function or the Log-loss function and is given as follows:

$$-\sum_{i=1}^{N} \Big(y_i \log f(x_i) + \big(1 - y_i\big) \log \big(1 - f(x_i)\big)\Big) \tag{4}$$

where N is the total number of examples, y_i is the correct output and $f(x_i)$ is the predicted output. The weight update equation for gradient descent is given by:

$$W^{t+1} = W^t - \alpha \sum_{i=1}^{N} \big(f^t(X_i) - y_i\big) X_i \tag{5}$$

where α is the learning rate and W is the vector of parameters to be learnt at t^{th} iteration.

4 EXPERIMENTAL SETTINGS

4.1 Datasets

We have used three different datasets while experimenting our models and their description is given in 3.1 :

(1) https://www.kaggle.com/hugomathien/soccer
Train and Test Split = 1:1
(2) http://www.football-data.co.uk/matches.php
Train and Test Split = 7:3
(3) https://www.kaggle.com/martj42/international-football-results-from-1872-to-2017
Train and Test Split = 7:3

4.2 Output Format

We have considered home team win for the matches whose outcome is a draw while training our models in all the datasets. Also,the output is 1 if the home team wins but if the away team wins, it is 0 in case of logistic regression and -1 for the other 2 models.

4.3 Hyper-parameter Tuning

For all of the above datasets, we tried three models namely SVM, ANN and LR. Different experiments with varying parameters are shown below :

(1) Dataset 1
- **ANN** : We tried different number of hidden layers and their respective sizes and came to a conclusion that 2 hidden layers with 20 to 60 is the most reasonable size for the neural network. The results with hidden layers 1 or more

than 2 were not giving good results. Some of the experiments are shown in the following table. We achieved the best accuracy on learning rate = 0.001 and 2 hidden layers with 60 and 30 nodes each.

For the following experiments common parameters are:
Number of iterations = 1000
Mini batch size = 128

SNo.	alpha	Model	Train Acc	Test Acc
1	**0.001**	**125->60->30->1**	**75.07**	**69.63**
2	0.005	125->60->30->1	75.85	67.93
3	0.01	125->60->30->1	75.9	66.97
4	0.01	125->40->20->1	72.32	66.75
5	0.01	125->20->10->1	68.49	65.58

Here, Model = 125 -> 60 -> 30 -> 1 implies 125 nodes in input layer, 60 nodes in hidden layer 1, 30 nodes in hidden layer 2 and output layer of size 1 and alpha is the learning rate.
Following graph was observed for training and test accuracies for ANN on this dataset.

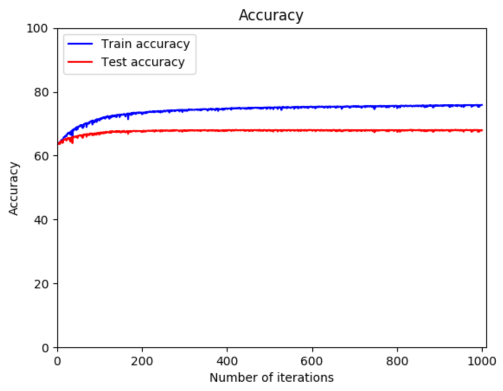


Figure 2: Variation of accuracy with number of iterations

- **SVM** : For SVM, the best accuracy was observed for learning rate (alpha) = 10^{-6} and regularization parameter (lambda) = 10^{-6}. Train Accuracy was approximately 63.413% and test accuracy was approximately 63.34%. SVM performed better than LR for this case but not more than ANN.

- **LR** : For LR, the best results were obtained for a learning rate of 10^{-7} with regularisation parameter, $\lambda = 10^{-5}$. The Training accuracy came out to be 57.35% while the Test accuracy was 57.39% in this case. The performance of LR was the worst in this dataset among the 3 models.

(2) Dataset 2
- **ANN** : In this dataset the accuracy does not change in first few iterations (60-100) and suddenly begins to increase. This shows that the initial iterations correspond to saddle point. The optimal number of iterations is 200, beyond this the model tends to get highly biased towards the training data and test accuracies begin to reduce. The final accuracies observed are as follows:

SNo.	alpha	Model	Train Acc	Test Acc
1	**0.001**	**856->40->10->1**	**79.0**	**63.28**
2	0.001	856->100->40->1	75.23	62.72
3	0.001	856->200->40->1	75.5	62.67
4	0.001	856->300->50->1	74.65	62.38
5	adam	856->100->1	73.76	62.68
6	adam	856->200->1	71.91	61.64
7	adam	856->100->40->10->1	53.9	53.9
8	adam	856->200->50->5->1	53.8	53.8

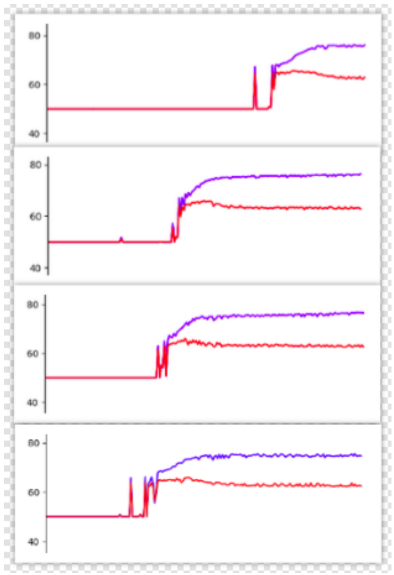


Figure 3: Variation of saddle point with increasing number of hidden layer neurons

- **SVM** : Various values of parameters like alpha and lambda were tried and their output is shown below :
Number of iterations : 500

SNo.	alpha	lambda	No.of SV	Train Acc	Test Acc
1	**0.00001**	**0.00001**	**2213**	**76.26**	**64.68**
2	0.0001	0.01	2255	76.33	64.2
3	0.001	0.01	1698	66.68	60.02

Here, No. of SV denotes number of support vectors after 500 iterations. The initial number of support vectors on first iteration were 3960. The above values show that alpha is more sensitive than lambda as changing alpha from 0.0001 to 0.001 drops the test accuracy from 64.2 to 60.02 while changing lambda value from 0.00001 to 0.01 doesn't make much difference. Moreover, 500 iterations are not required as model gives the optimal hyperplane in first 100 iterations.

The parameters for the following graph are alpha = 10^{-5} and lambda = 10^{-5}.

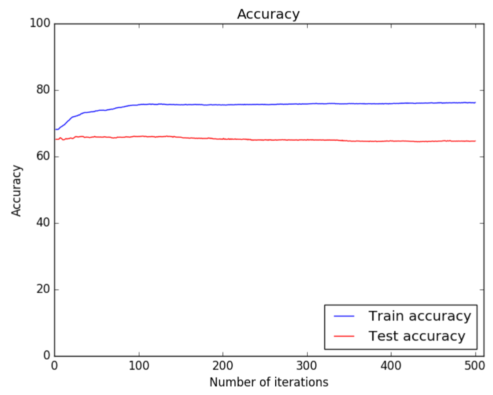


Figure 4: Variation of Accuracy of SVM with iterations

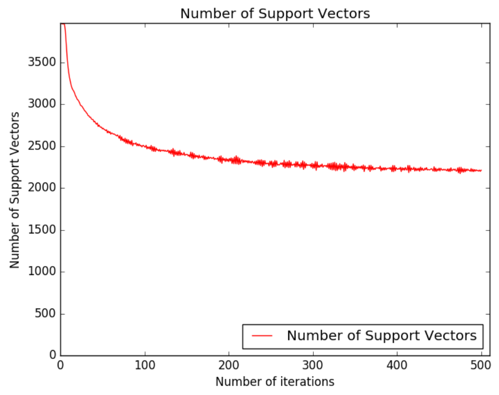


Figure 5: Variation of Number of support vectors with iterations

- **LR** : Principal Component Analysis (PCA) was applied with varying number of components and the results have been shown in the table given below :
  Number of iterations : 1500
  Learning rate, α = 10^{-5}
  Regularization parameter, λ = 10^{-4}

SNo.	No. of PCA comp.	Train Acc	Test Acc
1	100	58.19	57.23
2	500	63.65	57.96
3	Without PCA	67.3	60.42

Although, the results obtained are not as good as those obtained with SVM.

(3) Dataset 3

- **ANN** : For neural network this dataset gave a very high accuracy on the training dataset but relatively low on the test data set. We tried different models for various different hyperparameters and the average train accuracy was 93% and average test accuracy was 63%. The best accuracy was achieved on 200 neurons in first layer and 10 neurons in the second layer. This is a clear case of overfitting. We applied dropout optimisation to this model but the results did not improve. The accuracies fell down to 50% for both train and test set.

- **SVM** : Most values of alpha and lambda showed two kinds of behaviour : either the accuracy fluctuates too much or they converge very slowly. Various experimentation gave optimal value of alpha = 10^{-4} and lambda = 10^{-4} and train and test accuracies of 75.40% and 65.36% respectively.

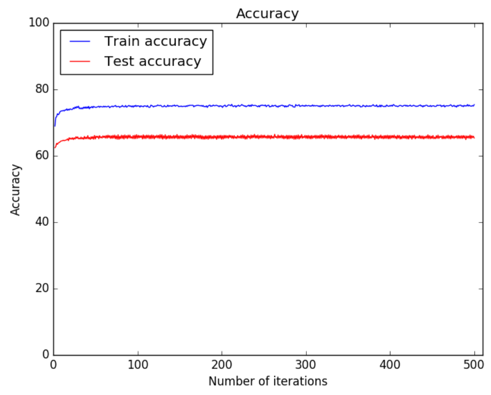


Figure 6: Variation of Accuracy of SVM with iterations

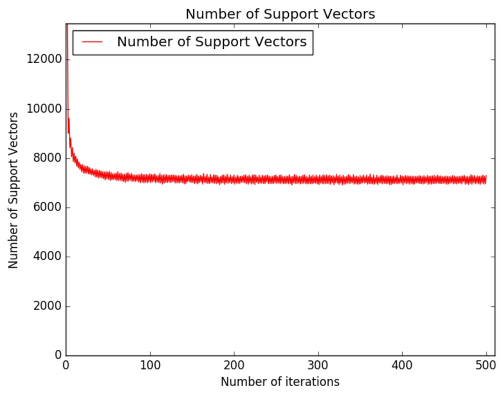


Figure 7: Variation of Number of support vectors with iterations

- **LR** : For logistic regression the best accuracy was observed for learning rate (alpha) = 10^{-5} and regularization parameter (lambda) = 10^{-4}. Train Accuracy was approximately 70.26% and test accuracy was approximately 62.9%.

Since the number of features for this dataset were approximately 2500 we tried applying Principal Component Analysis (PCA) but the results obtained were not good because just like dropout, PCA also reduces the amount of information from the datasets, but in our datasets there are no redundant features due to which reducing the number of equivalent features results in decrease in performance.

4.4 Evaluation Metric

- Dataset 1 : Following is the confusion matrix for ANN performed on dataset 1 with parameters as alpha = 0.001, model architecture as 856 -> 40 -> 10 -> 1 :

	positive	negative	total
positive	77226	34212	111438
negative	23298	58140	81438
total	100524	92352	

- Dataset 2 : For the second dataset, best results were obtained using SVM with parametric values of $\alpha = 0.00001$, $\lambda = 0.00001$ and its confusion matrix is shown below :

	positive	negative	total
positive	665	486	1151
negative	397	952	1349
total	1062	1438	

- Dataset 3 : For the third dataset, best accuracy was achieved using SVM with values of $\alpha = 0.001$, $\lambda = 0.001$ and its confusion matrix is shown below :

	positive	negative	total
positive	2764	1724	4488
negative	1742	2746	4488
total	4506	4470	

5 RESULTS AND DISCUSSION

The referred paper had achieved an average accuracy of 60% and the best accuracy of 68.8% for a neural network model. We have achieved average accuracy of 63% and the performance of our neural networks on unseen data for our first dataset has been the best with 69.3% accuracy. This is because we had included more relevant features and our large dataset enabled a better learning for the input distribution.

For the second dataset with 6000 instances SVM gave the best performance with train accuracy 76.26% and test accuracy 64.68%. Neural networks also performed at par with these values with 62.8% being the highest test accuracy achieved in the experiments.

For the third dataset, SVM gave the best results on unseen data with an average accuracy of 65.36%. ANN also gave good results on this dataset with around 63% test accuracy, but the performance worsened on applying dropout; which shows that the features were actually not correlated, i.e. there wasn't any redundancy among the considered features.

Clearly, since the datasets are not linearly separable, the overall performance is the best for ANN which is capable of learning more complex functions; as compared to SVM and LR where the output function is linear in the parameter space.

6 SUMMARY AND FUTURE WORK

Predicting the sport result is a challenging task due to the large number of diverse features involved in the analysis. In our project, we have established that Artificial Neural Networks give the best average accuracy for this classification problem. It performed at par with other classifiers for all three datasets.
Time plays a considerable role in determining the result of an upcoming game. The matches that the teams played some years ago should have less effect on the prediction analysis than the ones played recently. This time series behaviour is a room for further improvement in our models primarily in the weights given to each instance.

REFERENCES

[1] Josip Hucaljuk and Alen Rakipovic. 2011. Predicting football scores using machine learning techniques. *2011 Proceedings of the 34th International Convention MIPRO* (2011), 1623–1627.
[2] Enn Ong and A Flitman. 1997. Using neural networks to predict binary outcomes. In *Intelligent Processing Systems, 1997. ICIPS'97. 1997 IEEE International Conference on*, Vol. 1. IEEE, 427–431.