Data driven approach to analyzing football matches and predicting full time result

A review report submitted for the course

Machine Learning

CSE4020

Embedded Project

WINTER SEMESTER 2017-18

Submitted by:

Name	**Reg. No.**
Inamdar Chaitanya Ravindra	15BCE0163

SCOPE, VIT VELLORE,

TAMIL NADU- 632 014

ABSTRACT

It is quite evident, that football is one of the most watched sports in the world. The craze with which people show their support for their club, and the national pride that players take on representing their country, means that this is a very good market for Data Analysis and prediction. Moreover, after such inroads into technological advancements, the amount of data that can be processed and categorized into factors affecting the outcome, has Huge volumes of data that can be mined and results can be predicted.Football data of a particular league itself, can be worth quite a few gbs of data.Now that we have this immense volume of data with us, we need to mine it to a useful outcome.

Thus, we will be extracting the different features into which the data can be predicted, and how we can channel this prediction into some useful outcome using Machine Learning.Off late, this technique, of analysing past game results, training the computer about their outcomes, and then testing the model on present games , has burgeoned into a very useful technique, and in this project we will be showing the effects of some supervised learning models on some historic football data.

KEYWORDS

Xg boost, Logistics, Support Vector Classification, English Premier League, Data Analysis, Prediction of FT Result

INTRODUCTION

Football or Soccer for our American friends is more than just a game. It is played by **250 million** players in over **200 countries** making it the most popular sport. Each of these countries has a domestic league of their own in which teams compete for being labelled the best football team of that country. Being the staunch football fan I am, I decided to investigate the most popular domestic league of the world ((English Premier League) for factors that can influence the outcome of any match. This was my first proper data analysis project and I'm not necessarily aiming to use the results obtained to make a model or a prediction system. This project just aims to satisfy my obsession with football as well as get my hands dirty working as a Data Analyst.

Our basic aim in this project is to train our model on the basis of naïve bayes and SVM algorithms on a pre provided data set on last year's football matches and using the various features included with the dataset, to try and predict the outcome of testing data. On the basis of precision and recall we are going to compare the outcome of the model.

This project aims to discover implicit, previously unknown and potentially useful information or knowledge from data and also predict and evaluate the result and outcome from it . These data can be ranged from traditional shopper basket market data to the emerging social network data.

These have led to the research problems of social networks as well as sports data mining and prediction. Many existing sports data mining and prediction research projects focus on scheduling of games , visualization of games or players ,sport advising as well as identification and extraction of interesting moments or players from sports game video.

While these research projects are interesting and practical, it is also interesting to predict the outcomes of games (e.g., predict the outcomes of basketball games or soccer games. In general, making predictions is not an easy problem. Traditionally, some human domain experts make predictions on the game results based on their experience, instinct, and/or gut feeling.For example, pre-game analysis of televised sporting events often includes expert predictions. These predictions vary in accuracy as they are typically based on subjective claims and anecdotal evidence. Common sports statistic—strength of schedule (SOS) , which measures the opponents' record (OR) and was originally defined as a quotient of the sum of all the opponents' wins divided by the sum of opponents' games played—is used to track the quality of a team's opponents.

We have used three prediction models namely (i)Logistics (ii) SVC (Support Vector Classification) (iii) Xg boost to predict the result of the upcoming matches of English Premier League based on the data collected from every game played between 2015-2017 seasons.

LITERATURE SURVEY

The world of competitive gaming has a distinct advantage over typical competitive sports, in that it is much easier to record each individual statistic [1]. Analyzing the statistics of each hero has been done in the past. However, the target game has constantly shifted due the quick rise and fall of many popular games. Let us take an online game called League of Legends§ [2-6] as an example. In it, players have a similar selection of predefined champions. However, there are over 130 champions in League of Legends (cf. only 24 heroes currently in Overwatch). This means that there is far too much information to handle each hero manually. Existing website Champion.gg** is constantly analyzing how each champion is affected by each update in the game. It calculates the win rate in the current patch, and records statistics about changes (e.g., raise or fall) in the win rate. It also covers other aspects of the game, such as order that skills are used, trinkets chosen to improve performance, and the most common skill “trees” (i.e., a tiered system of skill sets earned by players). All these aspects of the game play an important role in how to use each hero efficiently. In addition, there are a few recent studies on analyzing data from online game such as League of Legends. For instance, Ferrari [2] studied the generative and conventional play on the general multiplayer online battle arena and the specific one on League of Legends. Kou and Gu [3] tried to understand temporary teams in League of Legends when playing with strangers. Lee and Ramier [4] investigated the impact of game features on champion usage in League of Legends. Kica et al. [5] analyzed the impact of patching on League of Legends. Kim et al. [6] used collective intelligence to make a strong team of players by predicting team performance in League of Legends. Fortunately for Overwatch, despite only being a few months old, the website Master Overwatch †† has accomplished similar results as the website Champion.gg. Specifically, the website Master Overwatch displays statistics which have risen/fallen over the past days. Despite seeing the statistics change (e.g., win rate, amount of kills), this does not necessarily mean that they are correlated. Perhaps the reason a hero is able to achieve a higher win rate is the result of another hero who synergizes well, and allows the hero more opportunities to achieve kills. Another possibility is that the time required to obtain a heroes ultimate ability has decreased, allowing them to achieve more kills as a result. Either way, while the website Master Overwatch displays the win rate and many other statistics, there is no proper correlation between them to determine win rate.

WORKING MODEL

In this project we use machine learning model to predict the results of the upcoming matches of the English Premier League.In the process of the predicting the full time match results we use the machine learning. In order to predict the upcoming results we retrieve the data for the football games in the past few years. The machine learning model will take the data and find some relationship between the result and the other attributes.

The data is not organised so, we clean the data According to our analytical needs.Since these features are heterogeneous by nature, we need to firstly standardise the individual values into a common executable one. So we will try to convert each of these features from a categorical type to a numerical type.So that is they can be analysed easily.

After the data has been cleaned of noise and properly pre processed, we will Split the given data into Training sets and Testing Sets. We use 85% of the data for training the model and use the remaining data to test the prediction results.

Splitting the Data into Training and Testing sets with probably 12 features and 1 target variable-:Full Time Result (Home Away or Draw)Then we train model using supervised learning and predict the outcome for test data.

Using the existing data we train the data and create the models using the 3 different machine learning algorithms. We use Logistic Regression, SVC (Support Vector Classification) and Xg boost to train the data and create the machine learning models.

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It’s an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$1 / (1 + e^{-value})$$

Where e is the base of the natural logarithms (Euler’s number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

$$y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

The Support vector Classification (SVC) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVC training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVC model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVCs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

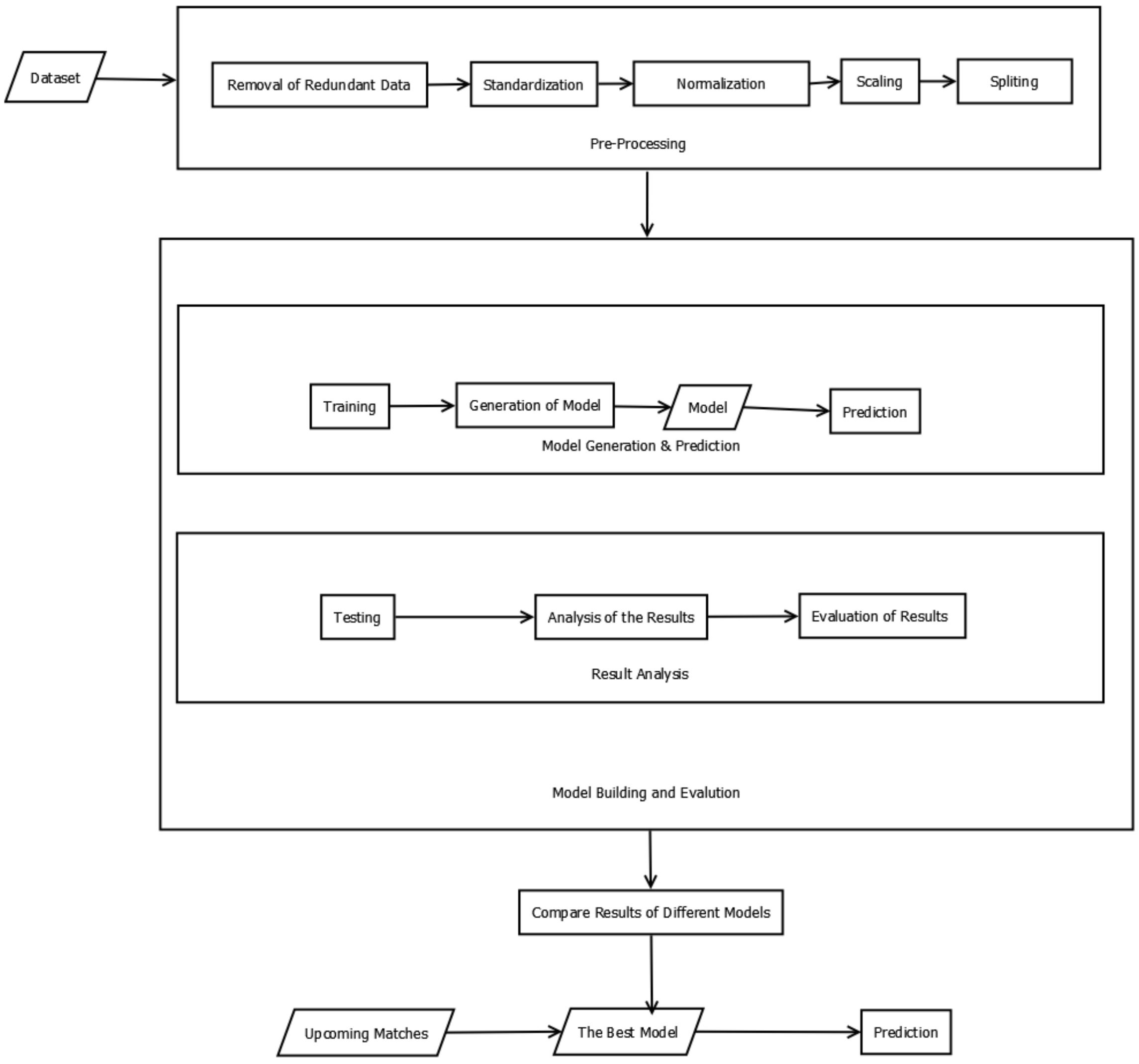
XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. It belongs to a broader collection of tools under the umbrella of the Distributed Machine Learning Community or DMLC who are also the creators of the popular mxnet deep learning library.

The prediction results of the each models are tested against the existing data. The we use this trained model over the remaining testing data and analyse which algorithm is giving best outcome by means of precision and recall and compare the success of The algorithms.

Based upon the Precision and recall each model will be evaluated.The evaluation will decide will model is best predicting the results with minimum error. This model will be termed as best model since it give the best prediction results with minimum error.

After this process we,are hoping to use different classifiers like logical regression to try and optimise our goal.The Best model then will be optimised Utilising different classifiers available.The optimization will reduce the time taken to predict the results and increase the utility of the model.The Model can be optimised Utilising different classifiers available to achieve the outcome with maximum optimisability. The optimised model will be used to predict the results of the upcoming matches.



Implementation – Code

```
# -*- coding: utf-8 -*-

#Importing all required packages

from time import time

import pandas as pd #Used for importing dataset
from pandas.plotting import scatter_matrix #Visualise the dataset
from IPython.display import display #Display the output
import xgboost as xgb #Implement xgboost
from sklearn.linear_model import LogisticRegression #Implement
LogisticRegression
from sklearn.svm import SVC #Implement SVC
from sklearn.metrics import f1_score #Finding out the f1 score
from sklearn.preprocessing import scale #Scaling the dataset
from sklearn.model_selection import GridSearchCV #for Grid view of the data
from sklearn.metrics import make_scorer #for performance metric or loss
function
from sklearn.model_selection import train_test_split
#Splitting dataset for training and testing

#data preprocessing
data = pd.read_csv('final_dataset.csv')

# Remove first 3 matchweeks
data = data[data.MW > 3]
```

```
# Removing useless attributes

data.drop(['Unnamed: 0','HomeTeam', 'AwayTeam', 'Date', 'MW', 'HTFormPtsStr', 'ATFormPtsStr', 'FTHG', 'FTAG',

                        'HTGS', 'ATGS', 'HTGC', 'ATGC','HomeTeamLP', 'AwayTeamLP','DiffPts','HTFormPts','ATFormPts',

'HM4','HM5','AM4','AM5','HTLossStreak5','ATLossStreak5','HTWinStreak5','ATWinStreak5',

            'HTWinStreak3','HTLossStreak3','ATWinStreak3','ATLossStreak3'],1, inplace=True)


# Preview data.

display(data.head())


# Calculate number of matches.

n_matches = data.shape[0]


# Calculate number of features.

n_features = data.shape[1] - 1


# Calculate matches won by home team.

n_homewins = len(data[data.FTR == 'H'])


# Calculate win rate for home team.

win_rate = (float(n_homewins) / (n_matches)) * 100


# Print the results.
```

```
print ("Total number of matches: {}".format(n_matches))

print ("Number of features: {}".format(n_features))

print ("Number of matches won by home team: {}".format(n_homewins))

print ("Win rate of home team: {:.2f}%".format(win_rate))


# Visualising the dataset

scatter_matrix(data[['HTGD','ATGD','HTP','ATP','DiffFormPts','DiffLP']],
figsize=(10,10))


# Saperating dependant and independant variables

X_all = data.drop(['FTR'],1)

y_all = data['FTR']


# Standardising the data.

cols = [['HTGD','ATGD','HTP','ATP','DiffLP']]


for col in cols:

   X_all[col] = scale(X_all[col])


X_all.HM1 = X_all.HM1.astype('str')

X_all.HM2 = X_all.HM2.astype('str')

X_all.HM3 = X_all.HM3.astype('str')

X_all.AM1 = X_all.AM1.astype('str')

X_all.AM2 = X_all.AM2.astype('str')

X_all.AM3 = X_all.AM3.astype('str')
```

```
def preprocess_features(X):
    ''' Preprocesses the football data and converts catagorical variables into
dummy variables. '''

    # Initialize new output DataFrame
    output = pd.DataFrame(index = X.index)

    # Investigate each feature column for the data
    for col, col_data in X.iteritems():

        # If data type is categorical, convert to dummy variables
        if col_data.dtype == object:
            col_data = pd.get_dummies(col_data, prefix = col)

        # Collect the revised columns
        output = output.join(col_data)

    return output

# preprocessing the independant variable
X_all = preprocess_features(X_all)
print ("Processed feature columns ({} total
features):\n{}".format(len(X_all.columns), list(X_all.columns)))

# Displaying the feature values
print ("\nFeature values:")
display(X_all.head())
```

```
# Shuffle and split the dataset into training and testing set.
X_train, X_test, y_train, y_test = train_test_split(X_all, y_all,
                                                    test_size = 50,
                                                    random_state = 2,
                                                    stratify = y_all)

def train_classifier(clf, X_train, y_train):
    ''' Fits a classifier to the training data. '''

    # Start the clock, train the classifier, then stop the clock
    start = time()
    clf.fit(X_train, y_train)
    end = time()

    # Print the results
    print ("Trained model in {:.4f} seconds".format(end - start))

def predict_labels(clf, features, target):
    ''' Makes predictions using a fit classifier based on F1 score. '''

    # Start the clock, make predictions, then stop the clock
    start = time()
    y_pred = clf.predict(features)

    end = time()
    # Print and return results
    print ("Made predictions in {:.4f} seconds.".format(end - start))


       return (f1_score(target, y_pred, pos_label='H'), sum(target == y_pred) / float(len(y_pred)))



def train_predict(clf, X_train, y_train, X_test, y_test):
    ''' Train and predict using a classifer based on F1 score. '''

    # Indicate the classifier and the training set size
               print ("Training a {} using a training set size of {}. . .".format(clf.__class__.__name__, len(X_train)))

    # Train the classifier
    train_classifier(clf, X_train, y_train)

    # Print the results of prediction for both training and testing
    f1, acc = predict_labels(clf, X_train, y_train)
    print (f1, acc)
     print ("F1 score and accuracy score for training set: {:.4f} , {:.4f}.".format(f1 , acc))

    f1, acc = predict_labels(clf, X_test, y_test)
    print ("F1 score and accuracy score for test set: {:.4f} , {:.4f}.".format(f1 , acc))

#Implementing LogisticRegression
```

```
clf_A = LogisticRegression(random_state = 42)
clf_B = SVC(random_state = 912, kernel='rbf')
clf_C = xgb.XGBClassifier(seed = 82)

# Reporting prediction values training and testing dataset
train_predict(clf_A, X_train, y_train, X_test, y_test)
print ('')
train_predict(clf_B, X_train, y_train, X_test, y_test)
print ('')
train_predict(clf_C, X_train, y_train, X_test, y_test)
print ('')

# TODO: Create the parameters list you wish to tune
parameters = { 'learning_rate' : [0.1],
            'n_estimators' : [40],
            'max_depth': [3],
            'min_child_weight': [3],
            'gamma':[0.4],
            'subsample' : [0.8],
            'colsample_bytree' : [0.8],
            'scale_pos_weight' : [1],
            'reg_alpha':[1e-5]
           }

# TODO: Initialize the classifier
clf = xgb.XGBClassifier(seed=2)
```

```
# TODO: Make an f1 scoring function using 'make_scorer'
f1_scorer = make_scorer(f1_score,pos_label='H')

# TODO: Perform grid search on the classifier using the f1_scorer as the scoring method
grid_obj = GridSearchCV(clf,
                        scoring=f1_scorer,
                        param_grid=parameters,
                        cv=5)

# TODO: Fit the grid search object to the training data and find the optimal parameters
grid_obj = grid_obj.fit(X_train,y_train)

# Get the estimator
clf = grid_obj.best_estimator_
print (clf)

# Report the final F1 score for training and testing after parameter tuning
f1, acc = predict_labels(clf, X_train, y_train)
print ("F1 score and accuracy score for training set: {:.4f} , {:.4f}.".format(f1 , acc))

f1, acc = predict_labels(clf, X_test, y_test)
print ("F1 score and accuracy score for test set: {:.4f} , {:.4f}.".format(f1 , acc))
```

```
# TODO: Create the parameters list you wish to tune
parameters = { 'learning_rate' : [0.1],
            'n_estimators' : [40],
            'max_depth': [3],
            'min_child_weight': [3],
            'gamma':[0.4],
            'subsample' : [0.8],
            'colsample_bytree' : [0.8],
            'scale_pos_weight' : [1],
            'reg_alpha':[1e-5]
          }

# TODO: Initialize the classifier
clf = xgb.XGBClassifier(seed=2)

# TODO: Make an f1 scoring function using 'make_scorer'
f1_scorer = make_scorer(f1_score,pos_label='H')

# TODO: Perform grid search on the classifier using the f1_scorer as the scoring method
grid_obj = GridSearchCV(clf,
                  scoring=f1_scorer,
                  param_grid=parameters,
                  cv=5)

# TODO: Fit the grid search object to the training data and find the optimal parameters
```

```
grid_obj = grid_obj.fit(X_train,y_train)


# Get the estimator

clf = grid_obj.best_estimator_

print (clf)


# Report the final F1 score for training after parameter tuning

f1, acc = predict_labels(clf, X_train, y_train)

print ("F1 score and accuracy score for training set: {:.4f} , {:.4f}.".format(f1 ,
acc))


# Report the final F1 score for testing after parameter tuning

f1, acc = predict_labels(clf, X_test, y_test)

print ("F1 score and accuracy score for test set: {:.4f} , {:.4f}.".format(f1 , acc))


print(clf.predict(X_test))
```

Results and Discussions

Google Drive Link of Explanation and Demo

https://drive.google.com/file/d/1E2rPhTB_WNFTtlJcu9QdUzf71TMH50Tr/view

DATA SET

The data set was obtained through http://www.football-data.co.uk/data.php. The data sets have been attached as separate files in the repository.For this particular project, we decided to download data for the last 15 seasons of EPL starting from 2000-01 season to 2014-15 season.We have taken match results of different matches played between premier league teams from last 16 years(seasons).We total have 6079 match results on the basis of which we are going to make our machine learning model

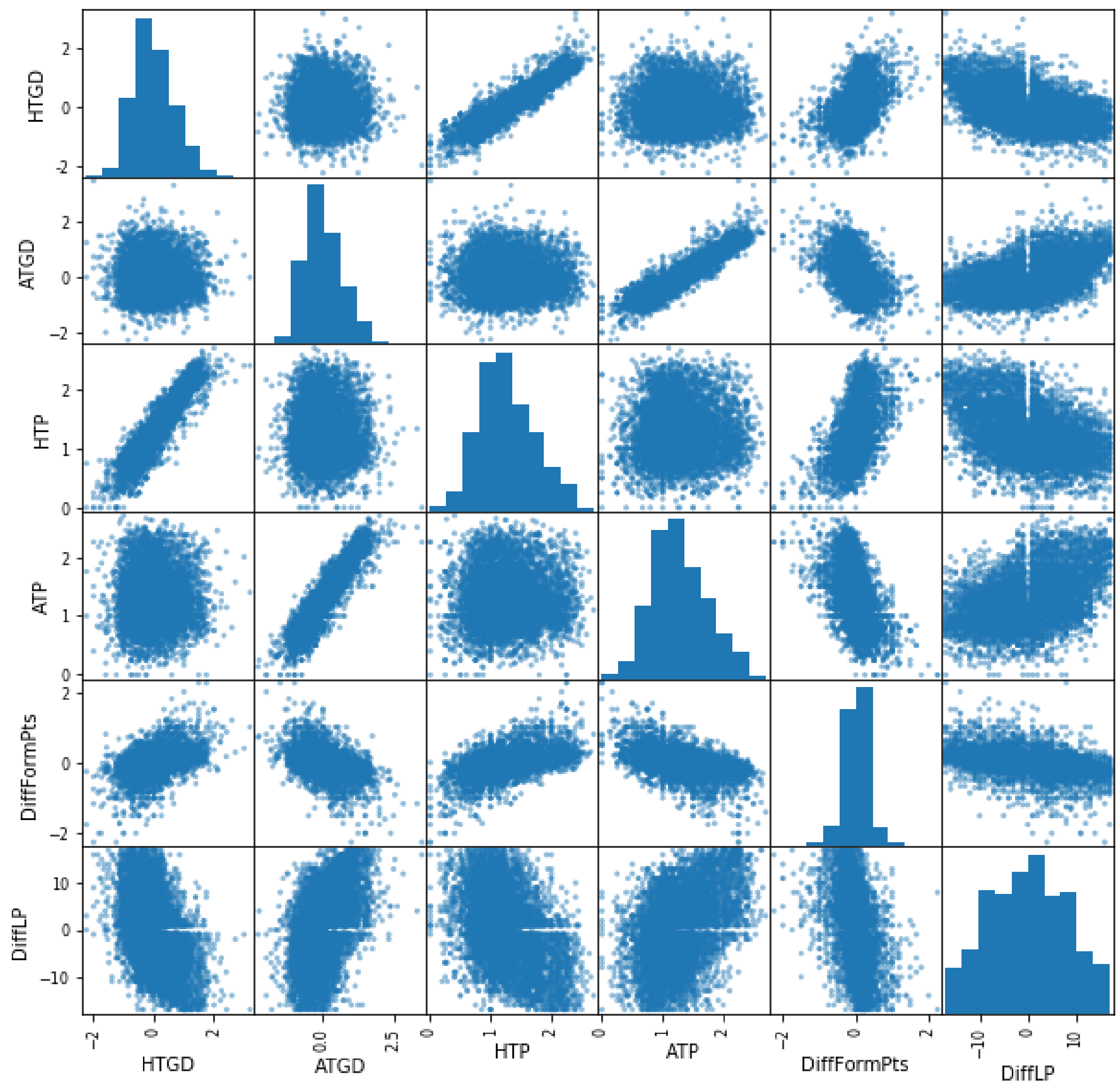
The dataset we are going to use in this project is about statistics of matches played between two teams which are further differentiated as home team and away team. Hometeam and Away Team will contain names of the respective teams.All the attributes other than Home Team,Away Team,Date are based on the performance of teams during matches. There are some key attributes in the dataset like FTHG (Full Time Home Team Goals),FTAG(Full Time Away Team Goals).In a football match home team has advantage as the home team gets to play on their home stadium. Because of this for the home team chances of getting a win increases.Each team gets to play on their home ground against each other. So both team get same advantage in the respective matches.

There are three types of results in the match which are win ,lose or draw. On the basis of winning, losing and a draw. teams get the points or drop the points.We have also taken the point difference between two teams. We have also taken streaks of matches played between home and away team. We have also maintained the data about scored and conceded goals by teams and differentiated these goals into home and away goals. In the these dataset we also have home team and away team goal differences.On the basis of all these attributes we will be predicting the main result of the match.FTR is the main Attribute which will give the final result about the match.

Results

```
FTR    HTP    ATP  HM1  HM2  HM3  AM1  AM2  AM3   HTGD   ATGD   DiffFormPts
DiffLP

30   H  1.25  1.00   D   D   W   D   W   L  0.50  0.25        0.25   -16.0

31  NH  0.75  0.25   L   L   W   D   L   L -0.50 -0.75        0.50    -2.0

32   H  1.00  1.00   L   D   W   D   W   L  0.00  0.25        0.00    -3.0

33  NH  0.75  0.50   L   L   W   D   L   D -0.25 -0.25        0.25     3.0

34  NH  1.00  1.50   D   L   W   W   W   L  0.00  0.75       -0.50     3.0

Total number of matches: 5600

Number of features: 12

Number of matches won by home team: 2603

Win rate of home team: 46.48%

Processed feature columns (24 total features):

['HTP', 'ATP', 'HM1_D', 'HM1_L', 'HM1_W', 'HM2_D', 'HM2_L', 'HM2_W',
'HM3_D', 'HM3_L', 'HM3_W', 'AM1_D', 'AM1_L', 'AM1_W', 'AM2_D', 'AM2_L',
'AM2_W', 'AM3_D', 'AM3_L', 'AM3_W', 'HTGD', 'ATGD', 'DiffFormPts', 'DiffLP']


Feature values:

         HTP       ATP  HM1_D  HM1_L  HM1_W  HM2_D  HM2_L  HM2_W  HM3_D
\

30 -0.043829 -0.611968     1     0     0     1     0     0     0

31 -1.120644 -2.238746     0     1     0     0     1     0     0

32 -0.582236 -0.611968     0     1     0     1     0     0     0

33 -1.120644 -1.696487     0     1     0     0     1     0     0

34 -0.582236  0.472551     1     0     0     0     1     0     0


   HM3_L   ...    AM2_D  AM2_L  AM2_W  AM3_D  AM3_L  AM3_W     HTGD  \
30     0    ...       0     0     1     0     1     0  0.753719
31     0    ...       0     1     0     0     1     0 -0.737082
32     0    ...       0     0     1     0     1     0  0.008318
33     0    ...       0     1     0     1     0     0 -0.364382
34     0    ...       0     0     1     0     1     0  0.008318

       ATGD  DiffFormPts    DiffLP
30  0.355995         0.25 -1.989216
31 -1.138834         0.50 -0.248963
32  0.355995         0.00 -0.373267
33 -0.391419         0.25  0.372556
34  1.103409        -0.50  0.372556

[5 rows x 24 columns]
Training a LogisticRegression using a training set size of 5550. . .
Trained model in 0.0220 seconds
Made predictions in 0.0010 seconds.
0.621561035256 0.6654054054054054
F1 score and accuracy score for training set: 0.6216 , 0.6654.
Made predictions in 0.0000 seconds.
F1 score and accuracy score for test set: 0.6957 , 0.7200.

Training a SVC using a training set size of 5550. . .
Trained model in 1.7302 seconds
Made predictions in 1.1318 seconds.
0.620453572957 0.6803603603603604
```

```
F1 score and accuracy score for training set: 0.6205 , 0.6804.

Made predictions in 0.0100 seconds.

F1 score and accuracy score for test set: 0.6818 , 0.7200.


Training a XGBClassifier using a training set size of 5550. . .

Trained model in 0.5634 seconds

Made predictions in 0.0220 seconds.

0.652147113211 0.694954954954955

F1 score and accuracy score for training set: 0.6521 , 0.6950.

Made predictions in 0.0010 seconds.

F1 score and accuracy score for test set: 0.7451 , 0.7400.


XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,

       colsample_bytree=0.8, gamma=0.4, learning_rate=0.1,

       max_delta_step=0, max_depth=3, min_child_weight=3, missing=None,

       n_estimators=40, n_jobs=1, nthread=None,

       objective='binary:logistic', random_state=0, reg_alpha=1e-05,

       reg_lambda=1, scale_pos_weight=1, seed=2, silent=True,

       subsample=0.8)

Made predictions in 0.0120 seconds.

F1 score and accuracy score for training set: 0.6365 , 0.6827.

Made predictions in 0.0010 seconds.

F1 score and accuracy score for test set: 0.7826 , 0.8000.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,

       colsample_bytree=0.8, gamma=0.4, learning_rate=0.1,

       max_delta_step=0, max_depth=3, min_child_weight=3, missing=None,
       n_estimators=40, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=1e-05,
       reg_lambda=1, scale_pos_weight=1, seed=2, silent=True,
       subsample=0.8)
Made predictions in 0.0120 seconds.
F1 score and accuracy score for training set: 0.6365 , 0.6827.
Made predictions in 0.0010 seconds.
F1 score and accuracy score for test set: 0.7826 , 0.8000.
['H' 'H' 'H' 'NH' 'NH' 'H' 'NH' 'H' 'H' 'H' 'NH' 'NH' 'H' 'NH' 'H' 'H' 'H'
 'H' 'H' 'NH' 'NH' 'H' 'NH' 'NH' 'NH' 'NH' 'H' 'NH' 'NH' 'H' 'NH' 'H' 'NH'
 'H' 'NH' 'H' 'H' 'H' 'NH' 'NH' 'NH' 'NH' 'NH' 'NH' 'NH' 'NH' 'NH' 'H' 'NH'
 'H']
```



HTGD
ATGD
HTP
ATP
DiffFormPts
DiffLP

Discussions

Logistic regression

Time taken by logistic regression to predict results is much lesser than SVC and XGboost but results are poor as compared to XGBOOST but better than SVC.

SVC

SVC Is giving results which are not at all significant and precise. And time taken to predict results is more. So we are not going to consider SVC in our final results because of worst results which we got.

XGboost

There is no parameter training in XGboost. The time taken for predicting results is slightly more than that of linear regression.But we are getting better results than linear regression and SVC.

XGboost is giving us better results in less time so we used XGboost classifier. Because of parameter tuning we used we got results in very less time and also accurate got increased

On the basis of Scattered plots we conclude that

1.home team points are directly proportional to home team goals

2.Difference in points is linearly increasing with away team points.

3.Away team points and difference in points is not at all giving us anything.

4. Away team points and home team points has no relation.

5. Home team goal difference and away team goals is linearly proportional

Finally we can conclude that XGboost is giving best results in terms of precision and also time taken to predict results.

REFERENCES

1. Budhia BP, Cuzzocrea A, Leung CK. Vertical frequent pattern mining from uncertain data. In: Proceedings of the KES 2012. IOS Press;
2012, p. 1273-1282.

2. Choros K. Temporal aggregation of video shots in TV sports news for detection and categorization of player scenes. In: Proceedings of the
ICCCI 2013. Springer; 2013, p. 487-497.

3. Chowdhury NK, Leung CK. Improved travel time prediction algorithms for intelligent transportation systems. In: Proceedings of the KES
2011, Part II. Springer; 2011, p. 355-365.

4. Christodoulou A. Anthony Christodoulou - football algorithms. http://anthonychristodoulou.com/

5. Cuzzocrea A, Leung CK, MacKinnon RK. Mining constrained frequent itemsets from distributed uncertain data. Future Generation
Computer Systems 2014; 37:117-126.

6. Delen D, Cogdell D, Kasap N.
A comparative analysis of data mining methods in predicting NCAA bowl outcomes. International Journal
of Forecasting 2012; 28(2):543-552.

7. Elo A. The rating of chessplayers, past and present. New York, NY: Arco Pub.; 1978.

8. Elo A. The rating of chessplayers: past and present. Bronx, NY: Ishi Press; 2008.

9. Johnson G. Baseball committee recommends changes to RPI. NCAA News; Aug. 03, 2011.

10. Leung CK, Jiang F. Frequent pattern mining from time-fading streams of uncertain data. In: Proceedings of the DaWaK 2011. Springer;
2011, p. 252-264.

11. Leung CK, Medina IJM, Tanbeer SK. Analyzing social networks to mine important friends. In: Xu G, Li L, editors. Social media mining
and social network analysis: emerging research. IGI Global; 2013, p. 90-104.

12. Leung CK, Tanbeer SK, Cameron JJ. Interactive discovery of influential friends from social networks. Social Network Analysis and Mining
2014; 4(1): art. 154.

13. Lu WL, Ting JA, Little JJ, Murphy KP. Learning to track and identify players from broadcast sports videos. IEEE Transactions on Pattern

Analysis and Machine Intelligence 2013; 35(7):1704-1716.

14. Mandel S. College football pickoff: 2013-14 bowl season. CNN Sports Illustrated; Dec. 20, 2013.

15. Matsumotoa T, Kojirib T. Baseball coaching ability development system based on externalization of decision process. In: Proceedings of
the KES 2013. Elsevier; 2013, p. 653-661.

16. Mentzelopoulos M, Psarrou A, Angelopoulou A, Rodríguez JG. Active foreground region extraction and tracking for sports video
annotation. Neural Processing Letters 2013; 37(1):33-46.

17. Messelodi S, Modena CM. Scene text recognition and tracking to identify athletes in sport videos. Multimedia Tools and Applications 2013;
63(2):521-545.

18. Miller SJ. A derivation of the Pythagorean won-loss formula in baseball. Chance Magazine 2007; 20(1):40-48.

19. Min B, Kim J, Choe C, Eom H, McKay RI. A compound framework for sports results prediction: a football case study. Knowledge-Based
Systems 2008; 21(7):551-562.

20. Mosqueira-Rey E, Prado-Gesto D, Fernández-Leal Á, Moret-Bonillo V. Prosaico: characterisation of objectives within the scope of an
intelligent system for sport advising. In: Proceedings of the KES 2012. IOS Press, p. 238-247.

21. Mueller F, Khot RA, Chatham AD, Pijnappel S, Toprak C, Marshall J. HCI with sports. In: Proceedings of the ACM CHI 2013, Extended
Abstracts. p. 2509-2512.

22. Odachowski K, Grekow J. Predicting the final result of sporting events based on changes in bookmaker odds. In: Proceedings of the KES
2012. IOS Press, p. 278-287.

23. Pedersen B. College football bowl picks 2013-14: predictions for every game. Bleacher Report; Dec. 08, 2013.

24. Schatz A, Benoit A, Connelly B, Farrar D, Fremeau B, Gower T, Hinton M, McCown R, McIntyre B, Stuart C, Tanier M, Tuccitto D,
Verhei V, Weintraub R. Football outsiders almanac 2013: the essential guide to the 2013 NFL and college football seasons. Seattle, WA