Prediction of football match results based on Python parallel machine learning and classification of clubs based on Spark machine learning

Student IDs : 53545289, 40101063
Student Names : WANG HAOYING, Malo Ferriol
Student Emails : haoyiwang3-c@my.cityu.edu.hk, mferriol2-c@my.cityu.edu.hk

Title:

Prediction of football match results based on Python parallel machine learning and classification of clubs based on Spark machine learning

Abstract:

Football has been paid considerable attention due to its dual play of the highest economic value and the most influential. How to establish a prediction model for the outcome of football games and solve the prediction problem with scientific methods has become an interesting topic for experts and scholars. Various types of football matches have outstanding similarities in some aspects. In theory, they can find their rules from a large number of football matches and find a way to improve the level of winning and losing contests. However, due to the huge amount of data collected in a competition, the traditional theoretical model cannot fully weigh the importance of these variables, and the prediction models and results are often greatly limited. In this study, three models, namely Logistic Regression (LR), Radius Neighbors Classifier (RNC), and Xtreme Gradient Boosting Classifier (XGBC), have been applied in prediction for the results of the Premier League's 2009-2018 competition and one model, K-means has been used to associate clubs on the Spark framework. Additionally, parallel computing is implemented to observe changes in the running speed of the models. Parallel computing is already implemented in Spark, so we compare the use of pipelining in the performance and deploy on different machine: personal computer, VM and AWS cluster. The training time of three models is different, among which LR is the longest, RNC is the second, and XGBC is the shortest. Furthermore, with the increase in the number of CPU cores, the training time of the three algorithms shows two different trends. Because when the amount of computation required is small, as the number of CPU cores participating in the operation increases, the overhead of initializing and releasing each thread reduces the computational efficiency. For more complex models, increasing the number of cores involved in the operation can improve the efficiency of the operation.

1 Introduction

With the advancement of the times and the popularity of the Internet, the industrialization of football has developed rapidly. Football has been paid considerable attention due to its dual play of the highest economic value and the most influential. According to relevant statistics, the football industry, which is known as the “17th largest economy in the world”, accounts for 43% of the total industrial output value of the sports industry, reaching US$500 billion, surpassing many developed countries. Moreover, the region's GDP is undisputedly the world's number one sport, far beyond the rest of sports such as basketball, golf, baseball, and F1 racing. According to FIFA statistics [1], as of July 2016, more than 200 million athletes

from 1.5 million teams from all over the world have performed many football matches, and other people involved in football-related work have also reached 30 million. According to statistics, people engaged in football and related work account for about 3% of the global population. With the rapid development of football, the prediction of the results of football matches has become a hot research direction.

2 Motivation

Nowadays, various algorithms are developing rapidly. How to establish a prediction model for the outcome of football games and solve the prediction problem with scientific methods has become an exciting topic for experts and scholars. Various types of football matches have outstanding similarities in some aspects. In theory, they can find their rules from a large number of football matches and find a way to improve the level of winning and losing contests. The usual forecasting method is that the relevant experts combine the historical battle records of the two sides and the current status of the two teams to make predictions. This method of prediction relies too much on historical engagement records, and the extent to which the current state of the team affects the outcome of the game to be played cannot be quantitatively described.

3 Objective

The traditional method of predicting football matches is mostly based on the establishment of an applicable theoretical model. However, due to the enormous amount of data collected in a competition, the traditional theoretical model cannot thoroughly weigh the importance of these variables, and the prediction models and results are often significantly limited. Therefore, this paper selects three models, including Logistic Regression, Radius Neighbors Classifier, and Xtreme Gradient Boosting Classifier, to predict the results of the competition and compare the differences. Besides, the degree of change in the training time of the model is observed by implementing parallel computing.The reason why we do clustering and aggregation is that the result is helpful of comparing the quality of two clubs.

4 Data Sources

The Premier League holds a season from August to May of the following year, implementing a double-cycle system of home and away. There are 20 teams in each season for 38 rounds (19 home games, 19 away games), and each round has a total of 10 games, so there are 380 games in each season. The data set was collected from http://football-data.co.uk/data.php [1].

5 Methodology

First, the data is preprocessed, and then the model is established and paralleled, as shown in Figure 1.

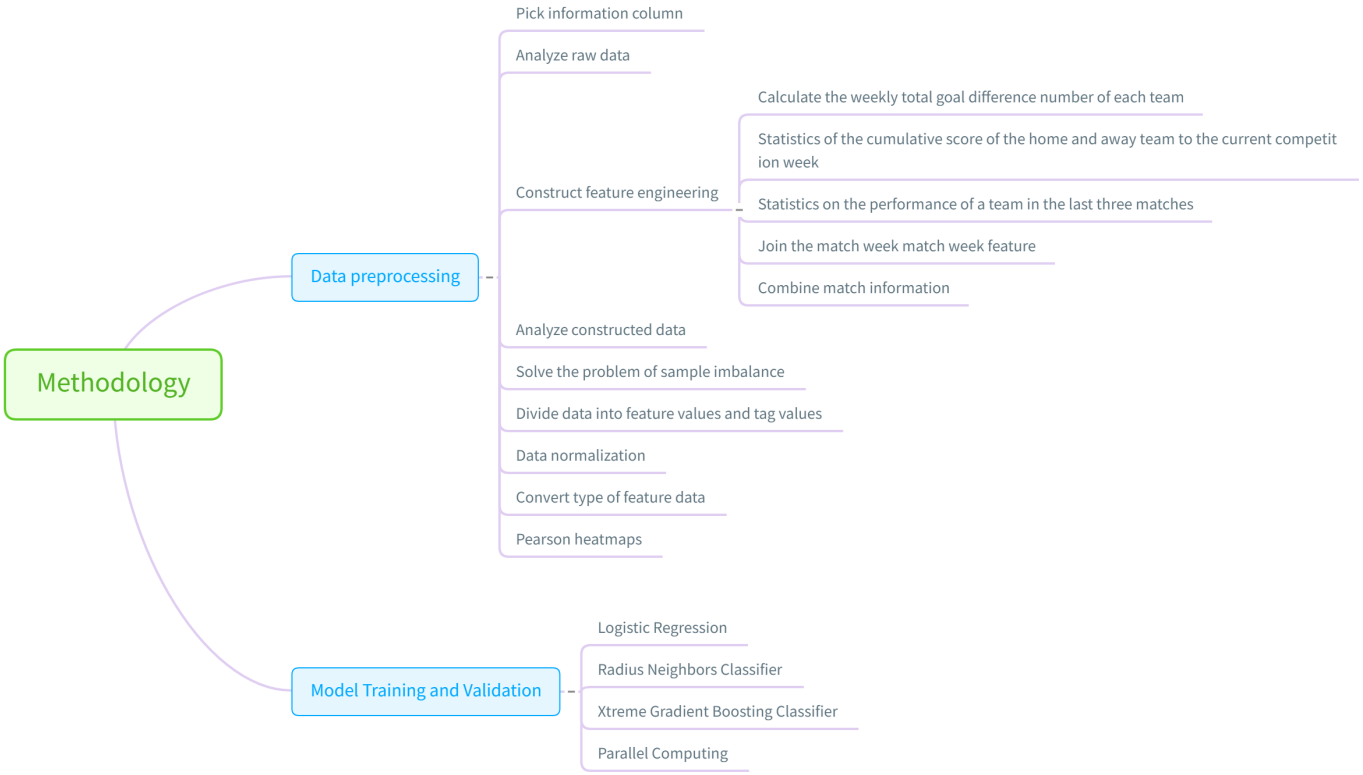

Figure 1 Method frame

5.1 Data preprocessing

5.1.1 Data preprocessing

Considering that there are 19 teams in the Premier League and each team needs to play 20 games, the data with less than 380 matches is deleted (so the 2014 data is removed). Select five columns of data as the original feature data, including Home team, Away team, Full-Time Home team Goals (FTHG), Full-Time Away team Goals (FTAG), Full-Time Result (FTR, H = Home Win, D = Draw, A = Away Win).

First of all, predict that all home teams will win and predict that all away games will win. Compare the results: count the accuracy of all home teams and all away teams.

Secondly, we carry out the construction of feature engineering. Feature engineering refers to the process of transforming raw data into training data of a model. Its purpose is to obtain better training data features and get a better training model. Feature engineering can improve the performance of the model, and sometimes even a good model can achieve excellent results. Feature engineering plays a critical role in machine learning. It is generally considered to include three major parts: feature construction, feature extraction, and feature selection. Because this game is a season of the year, there is a sequence; we can count the number of goal difference teams in the home and away team throughout the season before the game. Then for each week of each season, the difference between the number of goals scored by each team and the number of lost goals is counted, which the number of goal difference is. In this process, we need to perform some operations to obtain the structural features: (1) Calculate the weekly total goal difference number of each team; (2) Statistics of the cumulative score of the home and away team to the current competition

week; (3) Statistics on the performance of a team in the last three matches; (4) Join the match week feature; (5) Combine match information.

After constructing the features, it is found that the proportion of home wins is close to 50%, so the labeling ratio is not balanced for this three-category problem. We simplified it into a two-category problem, that is, whether the home team will win, which is also a way to solve the problem of uneven label ratio. Also, the data is divided into feature values and tag values. Then, the feature HTP of all matches is normalized. Furthermore, transform the feature data type, and convert the discrete type features into dummy coding features.

Finally, a Pearson heatmap is generated to observe the correlation between the features. Analysis of the correlation between the two features is usually expressed in terms of Pearson correlation coefficients. This article is visualized with the help of a heatmap tool. The principle is consistent with the Pearson correlation coefficient, which is the quotient of the covariance and standard deviation of the two variables. The formula is below:

$$\operatorname{cov}(x,y)=\frac{\sum_{n}^{i=1}(x_i-\bar{x})(y_i-\bar{y})}{n-1} \tag{1}$$

Where: $\bar{x}$ is the expectation of variable x; $\bar{y}$ is the expectation of variable y.

$$\rho_{x,y}=\frac{\operatorname{cov}(x,y)}{\sigma_x\sigma_y}=\frac{E[(x-\mu_x)(y-\mu_y)]}{\sigma_x\sigma_y} \tag{2}$$

In which, σ_x is the standard deviation of the variable x; σ_y is the standard deviation of the variable y; $\rho_{x,y}$ is the Pearson correlation coefficient.

The Pearson correlation coefficient is obtained by dividing the covariance by the standard deviation of the two variables. The Pearson correlation coefficient ranges from -1 to 1. When the correlation coefficient of the feature is close to 0, the correlation between the features is very weak. When the correlation coefficient is close to -1, the negative correlation between the two features is robust. Conversely, when there is a strong positive correlation between the features, the correlation coefficient between the features is close to 1. Equations (1) and (2) represent the covariance and Pearson correlation coefficients, respectively. The heatmap not only shows the size of the correlation coefficient but also uses the gradient of the color to show the strength of the coefficient visually.

5.1.2 Data preprocessing for club clustering classification

We would like to classify the clubs in three category. The clubs are divided in three categories club fighting for the title and place in European football, the clubs in mid-table and the club that only come and go at the top level. There is more club that are consistent club in the league but are not good enough to fight for the league title.

We use k-means because it is unsupervised. We need to transform the data from the match to data about the club. We decided to select the following features from the last 10 season:

Count(date) = number of match played in the league

Sum(ftr) = number of points won (defeat= 0, draw =1, win=3)

Sum(goal_scored) = number of goals scored by the team

Sum(goal_conceded) = number of goals conceded by the team

Sum(shot) = number of shot

Sum(shot_conceded) = number of shot conceded

Sum(shot_on_target) = number of shot on target

Sum(shot_conceded_on_target) = number of shot conceded on target

team	count(date)	sum(ftr)	sum(Goal_Scored)	sum(Goal_Conceded)	sum(Shot)	sum(Shot_conceded)	sum(Shot_On_Target)	sum(Shot_conceded_On_Target)
Tottenham	380	710	663	414	6005	4201	2697	1793
Brighton	76	76	69	114	748	1137	226	356
Bolton	114	121	140	200	1405	1649	796	925
Sunderland	304	314	328	456	3349	4578	1430	2029
Hull	152	136	142	259	1586	2241	568	859
Arsenal	380	719	729	429	5616	4023	2600	1695

To obtain this result we perform the following step:

1. Define the class fixture

We define a Scala class.

```
case class Fixture(team: String,date: String,ftr: Int, Goal_Scored: Int, Goal_Conceded: Int,
Shot: Int, Shot_conceded: Int, Shot_On_Target: Int, Shot_conceded_On_Target: Int)
```

2. Assign a number for win loose draw

We define a feature.

```
def assignHomeResult(ftr: String):Int = ftr match {
            case "A" => 0
            case "D" => 1
            case "H" => 3
        }
```

3. Group info by team

Group the value of each fixture

4. Select feature

Use VectorSlicer to select only the

5. Normalization

We then perform normalization. We choose to apply normalization because the range of the variable are very different.

5.2 Model establishment

The data set is randomly divided into a training set and a test set, and the corresponding label after the division is returned. When a trial is required, the existence of a random number seed (in fact, the number of the corresponding random number) can guarantee a set of the same random number. If 1 is filled in each

time, the random array obtained under the same parameters is the same. Nevertheless, filling in 0 or not filling, each time will be different. The generation of random numbers depends on the seed, and the relationship between them satisfies the following two rules: 1) different seeds will produce different random numbers; 2) when seeds are the same, even if the instances are different, the same random number will be generated.

5.2.1 Logistic Regression

The Logistic regression (LR) is a log-linear regression model that determines the responsiveness of the dependent variable by fitting the values of the independent variables. The value of the independent variable can be continuous, non-continuous, or binary. The Logistic regression model can be expressed as:

$$p = E(\mathrm{Y}) = \frac{\exp(\alpha + \sum_{i=1}^{k} \beta_i X_i)}{1 + \exp(\alpha + \sum_{i=1}^{k} \beta_i X_i)} \tag{3}$$

Among them, the response variable can be divided into a binary response variable and multiple response variables, and the value of the binary response variable usually consists of 0 and 1.

$E(\mathrm{Y})$: Expected probability when the response variable y = 0. For binary logistic regression, 0 is the target response variable. α : Constant coefficient of logistic regression. β_i : Variable regression coefficient of logistic regression. X_i is a series of independent variables, and p is a probability.

5.2.2 Radius Neighbors Classifier

The Radius Neighbors Classifier (RNC) is an extension of the K-Nearest Neighbor (KNN) algorithm [3]. In the feature space, if most of the most similar (i.e., the closest in the feature space) k samples of a sample belong to a specific category, the sample also belongs to this category. The selected neighbors are all objects that have been correctly classified in the RNC. In the categorization decision, the method determines the category to which the sample to be classified belongs according to only the category of the nearest one or several samples. Although the RNC is, in principle, dependent on the limit theorem, it is only relevant to only tiny amounts of adjacent samples in class decision making. Since the RNC mainly relies on the surrounding limited samples, rather than relying on the method of discriminating the domain to determine the category, the RNC is more suitable for the crossover or overlapping of the sample to be divided than the other methods. Unlike KNN, the RNC defines the nearest maximum distance; that is, we only search for all nearest neighbors within a defined distance. It avoids the problem that the rare samples will consider the far-distance samples when the k-nearest neighbors are found because of the small number of samples, which reduces the prediction accuracy.

5.2.3 Xtreme Gradient Boosting Classifier

The Xtreme Gradient Boosting Classifier (XGBC) algorithm is an efficient implementation of the Gradient Boosting algorithm, which is widely respected by the industry for its excellent performance and efficiency in application practice. XGBC, similar to the Gradient Boosting Decision Tree (GBDT), is an integrated learning method based on the CART tree, which is composed of multiple weak classifiers through

the boosting framework and negative gradient as the learning strategy [4]. The decision tree in the algorithm has a sequential association. It means that the current prediction is based on the prediction error of the previous round, and the model is constructed by using each round of prediction error iteration to improve the accuracy of the prediction.

Suppose $(x_i, y_i), i = 1, \boxed{?}, n$ is a modeled sample, $\hat{y}_i^{(t)}$ is the prediction result of the model after the *t*-th iteration, and $f_t(x_i)$ is the prediction result of the *t*-th decision tree. The solution form of $\hat{y}_i^{(t)}$ is:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{4}$$

Since the prediction result of the t-1 round is fixed at the t-th iteration, the prediction function $f_t(x_i)$ needs to be considered when setting the model objective function, and the following objective function is minimized when solving the model parameters:

$$S^{(t)}(\beta) = L(\beta) + D(f_t) + C \tag{5}$$

Where,

$$L(\beta) = \sum_{i=1}^{n} 1(y_i, y_i^{(t-1)} + f_t(x_i)) \tag{6}$$

$$D(f_t) = \gamma T + 0.5\lambda \sum_{j=1}^{T} \omega_j^2 \tag{7}$$

In equation (5), $L(\beta)$ is the loss function of the degree of fit of the measurement model, $D(f_t)$ is the regularization term of the complexity of the measurement model, and C is a constant term. In equation (6), $1(\cdot)$ is the loss function of the prediction accuracy of the measurement sample. In equation (7), T is the number of leaf nodes in the decision tree, ω_j is the prediction result corresponding to the leaf nodes, and γ and λ are the corresponding adjustment coefficients. The loss function is extended to the quadratic term by Taylor, and the greedy algorithm can solve the parameters of the model.

5.2.4 Parallel Computing

Parallel Computing refers to the process of using multiple computing resources to solve computational problems at the same time and is an effective means to improve the computing speed and processing power of computer systems [5]. Its basic idea is to use multiple processors to solve the same problem collaboratively, that is, the problem to be solved is decomposed into several parts, each part is calculated in parallel by an independent processor. A parallel computing system can be either a specially designed supercomputer with multiple processors or a cluster of independent computers interconnected in some way. The parallel computing cluster processes the data, and the processed result is returned to the user.

5.2.5 Pipeline

A machine learning (ML) pipeline is a complete workflow combining multiple machine learning algorithms together. There could be many steps required to process and learn from data, requiring a sequence of algorithms. Pipelines define the stages and ordering of a machine learning process. In MLlib, stages of a pipeline are represented by a specific sequence of PipelineStages, where a Transformer and an

Estimator each perform tasks.

5.2.6 HDFS

HDFS is the primary distributed storage used by Hadoop applications. A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data. The HDFS Architecture Guide describes HDFS in detail. This user guide primarily deals with the interaction of users and administrators with HDFS clusters. The HDFS architecture diagram depicts basic interactions among NameNode, the DataNodes, and the clients. Clients contact NameNode for file metadata or file modifications and perform actual file I/O directly with the DataNodes.The virtual machine provided by the courses includes Hadoop and HDFS. We run Hadoop then put the data in HDFS and launch the spark-shell with the code importing data from HDFS.

5.2.7 Spark Stand-alone

Spark Standalone cluster (aka Spark deploy cluster or standalone cluster) is Spark's own built-in cluster environment. Since Spark Standalone is available in the default distribution of Apache Spark it is the easiest way to run your Spark applications in a clustered environment in many cases.

Standalone Master (often written standalone Master) is the resource manager for the Spark Standalone cluster.

Standalone Worker (aka standalone slave) is the worker in the Spark Standalone cluster.

5.2.8 EMR Cluster AWS

Amazon Elastic MapReduce (EMR) is an Amazon Web Services (AWS) tool for big data processing and analysis. Amazon EMR offers the expandable low-configuration service as an easier alternative to running in-house cluster computing. It is based on Apache Hadoop, a Java-based programming framework that supports the processing of large data sets in a distributed computing environment. MapReduce is a software framework that allows developers to write programs that process massive amounts of unstructured data in parallel across a distributed cluster of processors or stand-alone computers. Amazon EMR processes big data across a Hadoop cluster of virtual servers on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). The elastic in EMR's name refers to its dynamic resizing ability, which allows it to ramp up or reduce resource use depending on the demand at any given time.

5.2.9 K-means

The most common algorithm uses an iterative refinement technique. Due to its ubiquity, it is often called "the *k*-means algorithm"; it is also referred to as Lloyd's algorithm, particularly in the computer science community. It is sometimes also referred to as "naive *k*-means", because there exist much faster alternatives.

Given an initial set of *k* means $m_1^{(1)},\ldots,m_k^{(1)}$ (see below), the algorithm proceeds by alternating between two steps:

Assignment step: Assign each observation to the cluster whose mean has the least squared Euclidean distance, this is intuitively the "nearest" mean.

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\},$$

where each is assigned to exactly one , even if it could be assigned to two or more of them.

Update step: Calculate the new means (centroids) of the observations in the new clusters.

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

The algorithm has converged when the assignments no longer change.

6 Results and Analysis

6.1.1 Data preprocessing for prediction match result

First, the accuracy of all home teams and all away teams will be counted. Figure 2 shows that the probability of a home win is higher than the probability of failure and draw.

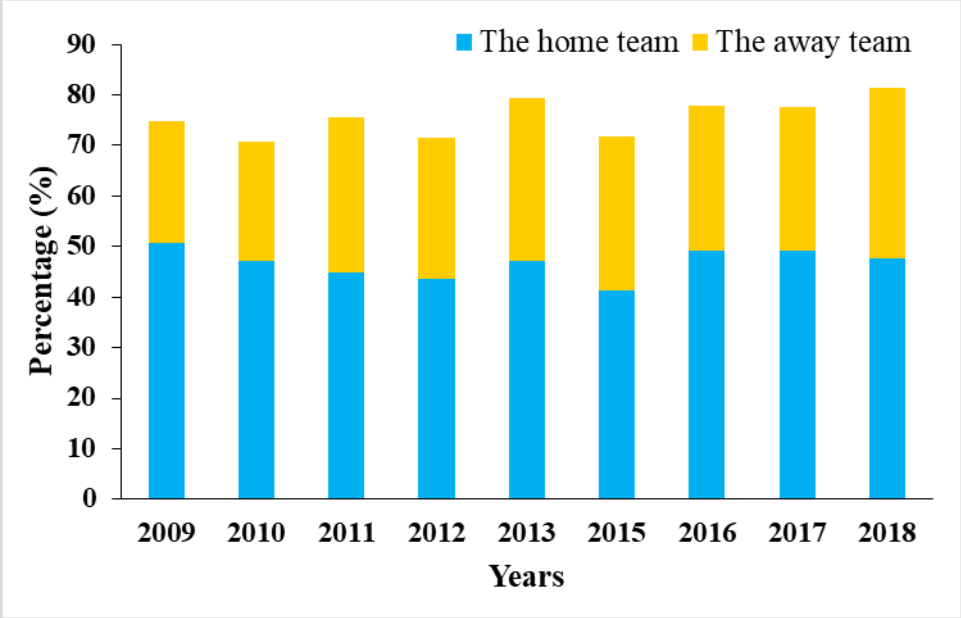

Figure 2 Winning rate of home and away team in 2009-2018

Next is the processing of the construction features:

(1) Calculate the weekly total goal difference number of each team

The processed data characteristics can be reflected by observing a certain number of data in a particular year, for example, the last five data of 2009-2010. Take 376 rows of data as an example (as shown in Table 1), the Swansea and Liverpool teams had a -8 and 8 goals, respectively, by the end of this game.

Table 1 Part of parameter characteristics in the last five data of 2009-2010

Line	Home Team	Away Team	FTHG	FTAG	FTR	HTGD	ATGD	HTP	ATP
375	Sunderland	Man United	0	1	A	0	55	45	86
376	Swansea	Liverpool	1	0	H	-8	8	44	52
377	Tottenham	Fulham	2	0	H	23	-1	66	52

378	West Brom	Arsenal	2	3	A	-6	24	47	67
379	Wigan	Wolves	3	2	H	-21	-41	40	25

Note: HTGD – Home Team Goals Difference; ATGD – Away Team Goals Difference; HTP – Home Team Points; ATP – Away Team Points.

(2) Statistics of the cumulative score of the home and away team to the current match week

Count the cumulative score of the home and away team throughout the season as of the current week. The scoring rules for the game are 3 points for the victory, 1 point for the tie, and 0 points for the loss. We score based on the results of the game before this week. As shown in Table 1, we deal with HTP (the cumulative points of the home team this season as of this week), ATP (the accumulated points of the away team this season as of this week). In the 376th example, Swansea and Liverpool scored 44 and 52 points, respectively.

(3) Statistics on the performance of a team in the last three matches

The characteristics of the previous structure reflect the historical performance of a team this season, and now observe the performance of the team in the three games in 2009-2010. HM1 represents the winning or losing of the last game of the home team, and AM1 represents the winning or losing of the last game of the away team. Similarly, HM2 and AM2 are the results of the former before the last match. As shown in Table 2:

Table 2 Performance of three matches in the last five data of 2009-2010

Line	Home Team	Away Team	HM1	AM1	HM2	AM2	HM3	AM3	MW
375	Sunderland	Man United	L	W	D	L	D	D	38
376	Swansea	Liverpool	L	W	D	L	D	W	38
377	Tottenham	Fulham	D	W	W	W	W	L	38
378	West Brom	Arsenal	D	D	D	D	W	D	38
379	Wigan	Wolves	W	D	W	D	L	L	38

(4) Join the match week feature

Then join the information of the match week, which week of the match in this season. The results after feature construction are shown in MW (Match Week) in Table 2.

(5) Combine match information

Combine the data set match information into one table. Then the above-calculated data includes the score and the goal difference, divided by the number of weeks to get the weekly average value. As shown in Table 3, you can see that the number of rows in the last row of the dataset is 3419, and the first row is 0 rows, which is a total of 3420 data. A total of 9 years of data is counted, with 380 data per year, so the size of the data set after the statistics is accurate.

Table 3 The last 5 data of the data set after constructing the feature

Line	FTR	HTGD	ATGD	HTP	ATP	HM1	AM1	HM2	AM2	HM3	AM3
3415	H	1.710526	0.078947	2.473684	1.5	W	W	W	W	W	W
3416	NH	0.342105	-0.97368	1.736842	0.815789	D	L	D	L	L	L
3417	NH	-0.52632	-1.42105	1	0.394737	L	D	D	L	D	L
3418	NH	0.736842	0.210526	1.842105	1.394737	L	W	L	D	W	W
3419	NH	-0.10526	-0.15789	1.315789	1.289474	L	W	L	W	D	D

Note: H - Home team win; NH – No Home team.

Next, analyze whether the construction data is realistic. The calculated win rate of the home team in the valid data is 46.76%, which is very close to the result of the original data analysis in Figure 1 (average 46.78%), indicating that the structural features are sufficient and close to reality.

Finally, a correlation graph of some features (shown in Figure 3) is generated to see the correlation between the features. It can be seen that the correlations between HTP and HTGD, ATP, and ATGD are robust, indicating the existence of multiple collinearities. It is because of the higher the average score of the week at home, the higher the average number of goals per game at home. If these variables that give almost the same information are taken into account, causing multiple collinearities. Therefore, the two features of HTP and ATP are removed and the two features of HTGD and ATGD are retained. Pearson heatmaps are ideal for detecting this situation, and they are an essential tool in feature engineering. At the same time, we can also find that HM3 and AM3 have less impact on the results of the current match, so consider retaining these features. Considering the correlations between HTP and HTGD, ATP and ATGD are over 90%, so the features HTP and ATP are deleted. Extracting the 10 features most relevant to FTR (as shown in Figure 4), it can be seen that the most relevant feature is HTGD, indicating that the higher the average number of goals per game at home, the higher the probability of winning.

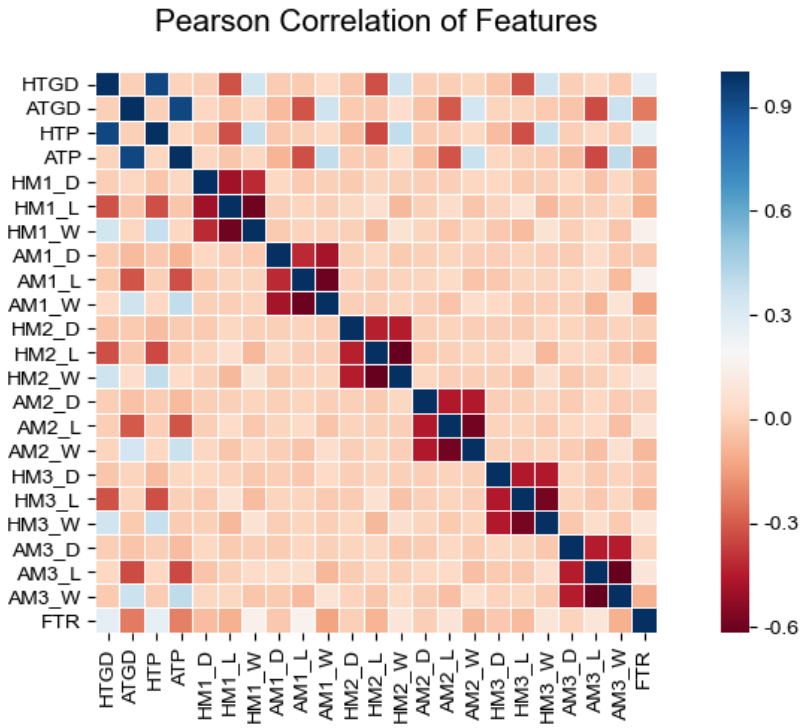

Figure 3 Pearson correlation coefficient heat map of game data variables

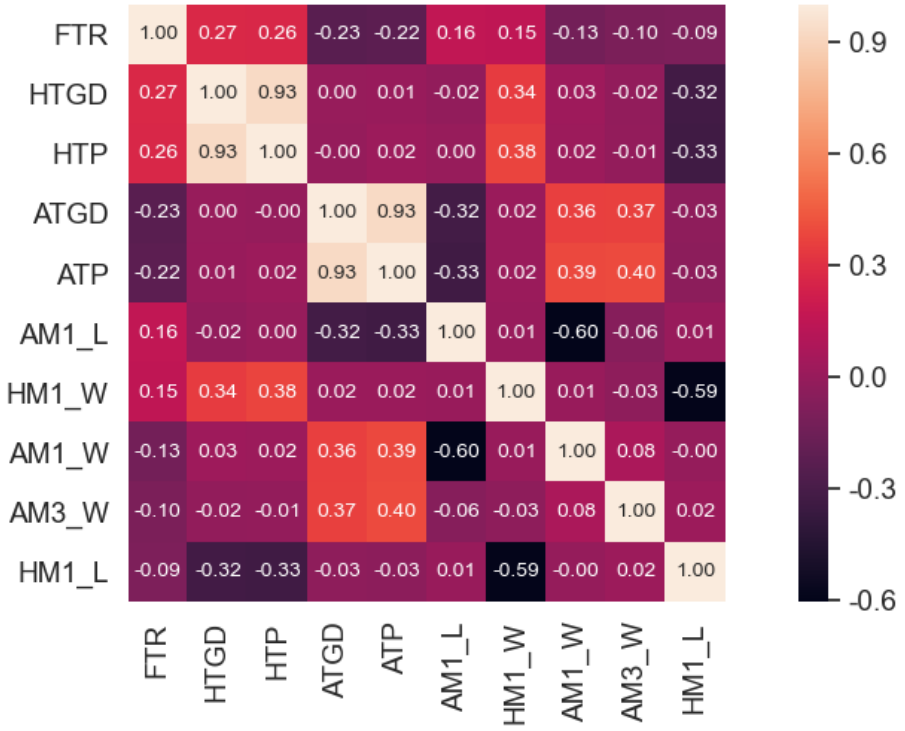

Figure 4 Pearson correlation coefficient heat map of game data variables (relate to FTR)

6.2 Model Training and Validation

6.2.1 Running results of three models

The LR, RNC, and XGBC models were established, and 70% and 30% of the data were randomly selected as training data and test data, respectively. The F1 score and the accuracy of the model were chosen as comparative indicators to evaluate the three models. F1 Score is an indicator used in statistics to measure the accuracy of the two-class model, taking into account the accuracy and recall rate of the classification model [6, 7]. The results in Table 4 can be obtained. For the F1 score in the train data set, the XGBC score is the highest, and the RNC score is the lowest. For the F1 score in the test data set, the LR score is the highest, and the RNC is the lowest. For Accuracy in the train data set, XGBC is the best, the other two models have similar results but are lower than XGBC. For Accuracy in the test data set, the results of the three models are similar, LR is slightly higher.

Table 4 Predicted F1 scores and accuracy of the three models

Parameter	LR	RNC	XGBC
F1 score in the train data set	0.6060	0.5945	0.6802
F1 score in the test data set	0.6375	0.6084	0.6288
Accuracy in the train data set	0.6503	0.6535	0.7143
Accuracy in the test data set	0.6751	0.6635	0.6614

6.2.2 Parallel Computing

Box-plot is a kind of statistical chart used to display a set of data dispersion data. It is mainly used to reflect the characteristics of the original data distribution and to compare the distribution characteristics of multiple sets of data. Therefore, box plots are used to compare the results of the parallel computing of the three models. Figure 5 can visually show that the training time of different algorithms is different, with LR time being the most prolonged, RNC second, and XGBC being the shortest. Furthermore, with the increase in the number of CPU cores, the training time of the three models shows two different trends. At LR and RNC (as shown in Figures 4a and b), as the number of CPU cores increases, so does the training time. It is because when the amount of computation required is small, as the number of CPU cores participating in the operation increases, the overhead of initializing and releasing each thread reduces the computational efficiency. For more complex algorithms (as shown in Figure 5c), increasing the number of cores involved in the operation can improve the efficiency of the operation.

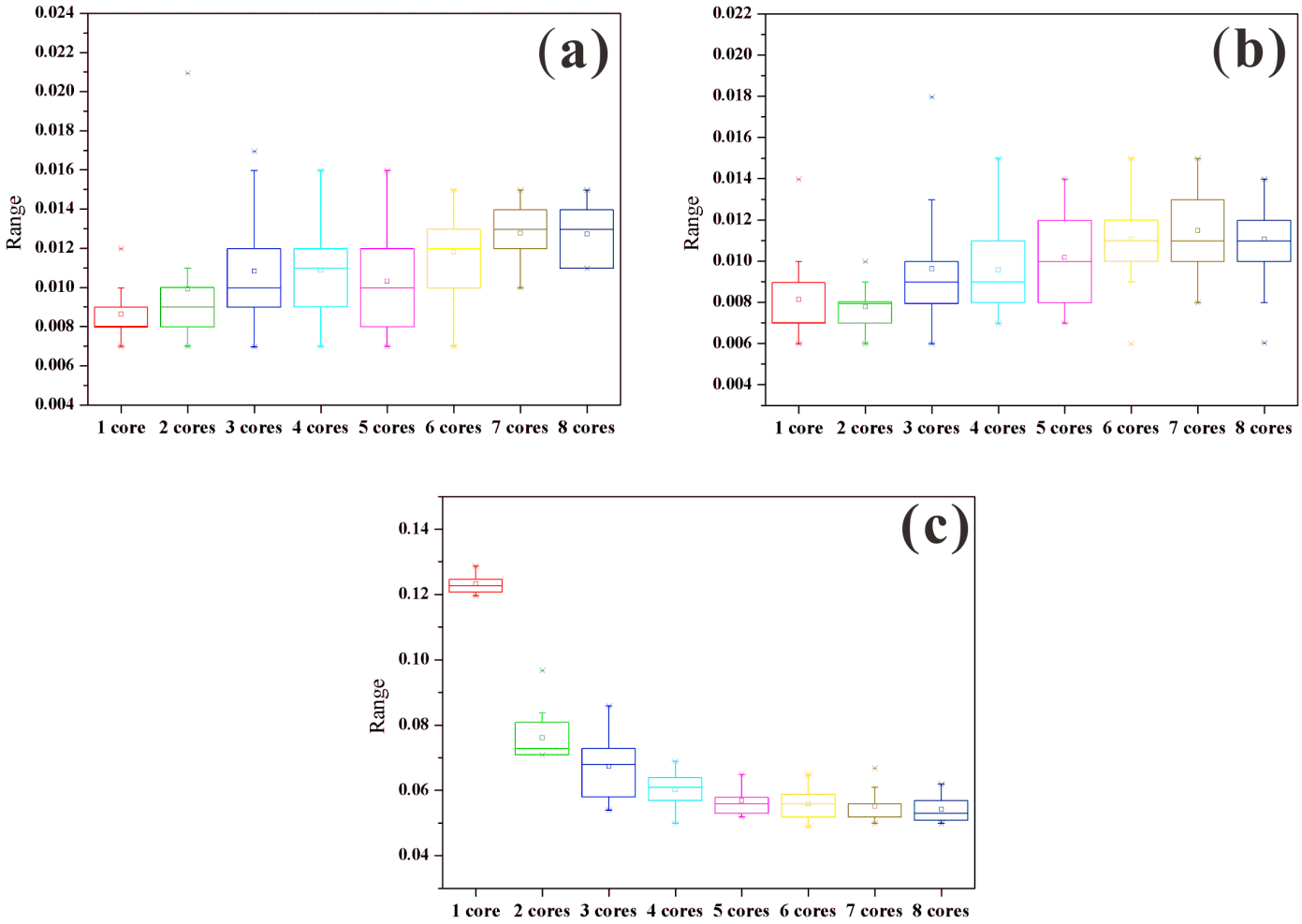

Figure 5 Results of training time for different CPU cores under parallel computing conditions. (a) Result of Logistic Regression; (b) result of Radius Neighbors Classifier; (c) result of XGB Classifier.

6.2.3 Stand-Alone and AWS EMR Cluster comparison

Table 5 Comparison of Run-time of the same program on different set-up

mode	Stand-alone		VM hdfs		Cluster	
Usage	Pipeline	No Pipeline	Pipeline	No Pipeline	Pipeline	No Pipeline
1-core	19,80s	17.85s	73,29s	92,49s		
2-core	21.16s	17.28s	70,98s	102,41s		
3-core	20,83s	19.51s				
6-core					29.34s	32s

6.2.3.1 Stand-Alone on machine without HDFS

The test is performed on a machine with 16GB of RAM and 4 cores. We launch a Spark standalone cluster on it with the "spark-shell" command. Using the option "--executor-cores" we are able to define the number

of core used by the process.

We observe that the best result are obtain when the program doesn't use pipeline and with one or two core. Our interpretation is that the sample of data is too small. Spark was design for Big Data. The program doesn't require a lot of computation so by decreasing the level of parallelism it increased the overall performance. It is because when the amount of computation required is small, as the number of CPU cores participating in the operation increases, the overhead of initializing and releasing each thread reduces the computational efficiency.

6.2.3.2 Stand-Alone on VM with HDFS

We use the VM from the university with Hadoop. We launch Hadoop and put the data into HDFS. Then we launch Spark-shell which launch a stand alone cluster. We define the data input from HDFS. Then we launch our program. Because it is on a virtual machine, the performance has been affected. We can compare and see that it was more than twice longer than any other test.

We cannot conclude that the use of HDFS with our program decrease the performance. However because we use a different file system, it is interesting to compare the performance of the pipeline. With the use of HDFS we can see that pipeline improves drastically the performance of the program by 30% when we launch the cluster with 2-core (the maximum number of core on the VM).

6.2.3.3 AWS EMR Cluster with AWS S3

We use the EMR service on AWS cloud. The service creates and manage a cluster with Hadoop configuration. The cluster is composed of one master-node and two core-node

Each node is an instance with following characteristic :

m5.xlarge : 4 vCPU, 16 GiO of memory, instance storage EBS :64 Gio

The use of a cluster dose not improve the performance compared to Standalone cluster on a laptop. This could be for several reason. Our interpretation is that the task did not require a high level of computational power, so the overhead complexity due to the management of a cluster and allocation of task is not compensated by better performance during the execution of jobs.

We notice that the use of pipeline has improved the performance on a cluster with AWS S3 file system as it does on the VM with HDFS, our interpretation is that the use of pipeline is beneficial when combing with a distributed file system. Since Spark is designed to be deployed on a cluster, it is recommended to use data pipeline with Spark.

7 Discussion

The accuracy of the currently used model is not very high and can be further improved. If the time is sufficient, the model can be improved in the future by trying the following:

1. Get more data or use more features;
2. Cross-validate the data set;
3. Model fusion techniques can be used;
4. The kicking technical information and health information of the participating players can be taken into

account;

5. Since only F1 scores and accuracy are currently considered, a more comprehensive model assessment mechanism can be tried.

6. The results of the actual football match are also affected and restricted by other factors, such as the weather conditions during the game, the historical confrontation between the two sides of the game, the injury situation of the players, and other uncertain factors, resulting in an unpopular result.

8 Conclusion

In this study, three models, namely Logistic Regression (LR), Radius Neighbors Classifier (RNC), and Xtreme Gradient Boosting Classifier (XGBC), have been applied in prediction for the results of the Premier League's 2009-2018 competition. Additionally, parallel computing is implemented to observe changes in the running speed of the models. The training time of three models is different, among which LR is the longest, RNC is the second, and XGBC is the shortest. Moreover, with the increase in the number of CPU cores, the training time of the three algorithms shows two different trends. Because when the amount of computation required is small, as the number of CPU cores participating in the operation increases, the overhead of initializing and releasing each thread reduces the computational efficiency. For more complex models, increasing the number of cores involved in the operation can improve the efficiency of the operation.

9 Individual Contribution Statement

9.1 WANG HAOYING

Contribution:

1. Data pre-processing (analysis of raw data, construct feature engineering, analysis of constructed data, etc.)
2. Establish the Logistic Regression model.
3. Build the Radius Neighbors Classifier model.
4. Establish the Xtreme Gradient Boosting Classifier model.
5. Performing Parallel computing.
6. Analyzing the data.
7. Drawing.
8. Writing the report part related to Python parallel machine learning and integrating the final report.
9. Making slides related to Python parallel machine learning part.

Reflection:

This project involves predicting football big data based on Python and implementing parallel operations is very challenging for my current level of knowledge. Great difficulties have been encountered in the selection process of the models due to the variety of models, hence how to select the models that is suitable for the data is a key point. Additionally, I realized that a project involving big data requires not only the use of traditional methods, but also innovation and improvement. But I do not have enough time, knowledge and skills to perfect this work for the current progress. The deficiencies have been added in the **Discussion** in the article. When doing parallel computing, I have improved my ability from the blank to the smooth

completion. I appreciate the growth brought by this project.

9.2 Malo Ferriol

Contribution:

1. Suggested the topic
2. Designed and Implemented the K-means clustering solution in Scala
3. Presentation slides
4. Deploying Spark
5. Writing the pipeline in Scala
6. Deploying EMR cluster on AWS cloud
7. Creating Spark app to be deployed on Cluster (issue with Scala compatibility)
8. Deploying spark app on VM

Reflection:

The project was very interesting for me. I mostly focus my attention on the tool Spark. I had no experience related to Scala nor Spark. However previously during my career I had the opportunity to use AWS cloud. It was a great experience to combine previous experience with my newly learned skills. During the design of the code I had the opportunity to develop several version of the same program. I had to adapt for cloud computing by incorporating my code into a Spark app and also integrate the use of pipeline into it.

This project gave a new perspective on programming and how to leverage the tools available to improve the performance. I also learned that we should always test the performance and understand how the program works because the result are not always what you would expect.

10 Reference

[1] Data from http://football-data.co.uk/data.php

[2] Pham BT, Tien Bui D, Prakash I. Landslide Susceptibility Assessment Using Bagging Ensemble Based Alternating Decision Trees, Logistic Regression, and J48 Decision Trees Methods: A Comparative Study. Geotechnical and Geological Engineering 2017;35(6):2597-611.

[3] Verma AK, Pal S, Kumar S. Comparison of skin disease prediction by feature selection using ensemble data mining techniques. Informatics in Medicine Unlocked, 2019:16:100202.

[4] Chen T, Guestrin C. Xgboost: A Scalable Tree Boosting System[C]. The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016: 785-794.

[5] Mendizabal A, Márquez-Neila P, Cotin S. Simulation of hyperelastic materials in real-time using deep learning. Med Image Anal, 2020: 59:101569.

[6] Fujino A, Isozaki H, Suzuki J. Multi-label text categorization with model combination based on F1-score maximization. In: Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP), Hyderabad, India: ACL, 2008: 823-828.

[7] Dembcznski K, Wageman W, Cheng W, et al., An exact algorithm for f-measure maximization. In: Proceeding of 25th Annual Conference on Neural Information Processing Systems (NIPS), Granada, Spain: NIPS Foundation, 2011: 223-230.

--- END ---