Jacobs University

Semester project

Football Prediction Challenge

Rahul Bhat

supervised by
Dr. Prof. Adalbert F.X. Wilhelm

January 31, 2017

Contents

List of Figures

List of Tables

Abstract

Data Science is gaining traction in every sector and using "Big Data" and "Analytics" allows to find correlations, discover unexpected patterns and predict future outcomes which can be used to forecast future probabilities. Different techniques can be used, including data mining, statistical modeling and machine learning which helps the analysts to forecast the outcome from the given set of data. In this project, the football matches of the last 8 years of the Italian League are provided and using predictive analytical techniques, the final results of football matches are predicted.

Before applying any mining techniques, the first thing is to do data-preprocessing and preparation, which helps the analysts to make a good choice of predictive analytical technique to apply. So after exploring the data, we decided to use four different predictive analytical techniques to predict the final result of the football matches, that are Basic Tree Model Technique, Random Forest, Neural Networks and Xgboost. Cross-Validation method is also used in Basic Tree Model and Xgboost to improve the model. The four techniques which are used in this project has different methods of usage; these different methods were used on the provided data set first and the best method in each technique was noted and used for comparison with other techniques for the same data set.

Keywords: Big Data, Data Science, Forecast, predictive analytics

A Introduction

Demand for ”Big Data” expertise started growing over the last few years and so as the demands for the new tools and the techniques that transform this huge amount of data into useful knowledge. The data volume is growing at a rapid rate in every industry or organization and this is due to explosive growth of cloud computing, social media, online videos and more. There is a huge amount of information present in the databases, which is potentially important, but has not yet been discovered. This tremendous growth in data and databases has spawned a pressing need for new techniques and tools that can intelligently and automatically transform data into useful information and knowledge.

"Data mining is the most important research domain in the 21st century" [30]. Data mining is sorting through data and to extract regularities to identify patterns and establish relationships. In data mining, different parameters are used, that are the association, sequence or path analysis, classification, clustering, and forecasting. Forecasting in data mining is to collect the data, discover patterns in data and statistical model is formulated which provides the ability to predict the future. This area of data mining is also known as predictive analytics. There are different approaches and techniques that can broadly be grouped into regression techniques and machine learning techniques. [30].

The predictive analytical techniques are widely used in many felds for example, Financial Data Analysis, Retail Industry, Telecommunication Industry, Biological Data Analysis, Intrusion Detection and many more. Predictive analytics is mostly used in business because future planning is very important and the estimation of the future values of business variables are needed. From sales, large amount of data is collected and that data can be used for many beneficial purposes such as identifying what type of products customers are buying more (also according to the age groups or in what season what products people want to buy), there are many answers like this which helps the industry to improve its standard and this also helps to improve their quality of customer service and satisfaction [15].

A.1 Research Background: Origin of Predictive Analytics

"Necessity is the mother of invention" [22]. According to "Yihao Li" [22] in Ancient times people used to search useful information from data by hand, but as the volume of the data started growing rapidly and it became impossible to search for some useful information manually (by hand). At this time computers were slow with little capacity. So, to handle this large amount of data which was growing at a fast rate, new automatic techniques and approaches were required. This has This has led to the creation of computer algorithms and new technologies in computer science [22].

Earlier there were only two methods to identify patterns and these two methods were the only key tools which were helpful in extracting useful information from data: Baye's theorem (1700s) and regression analysis (1800s). As with the time, data started growing, the speed of computers had become faster with more storage capabilities. 1936 was the dawn of computer age, that time it became possible to collect and process large amount of data and in 1950's more technologies emerged such as neural networks, clustering and many more algorithms which helped the analysts to make a better decision [22].

During 1980s advanced data analysis emerged and more advanced data analysis methods sprang up which improved database and information industry. This was the reason that the huge databases could be stored in different kinds of database repositories and this led to the new technology called data warehouse (database repository). Data cleaning, data preparation, online analytical processing, all these methods come under data warehouse technology [15]. Data cleaning and data preparation are very important methods which have to be done before applying the data-mining techniques which helps the analysts to make a better decision.

During the 1990s the amount of data started growing so rapidly and the reason behind this was the fast-growing Internet speed, easy access to the Internet and advances in mobile devices that include digital video, photography, audio, and advanced email and text features and this data is generated by us, that is, 70% of the data is user-generated and the third reason is, Organizations and the Companies which started generating data at a fast speed. The digital universe started growing at a fast speed, in 2009, it grew to 60% that is almost 800,000 petabytes, In 2010, it was estimated 1.2 million and, in the end, it was in petabytes and, In 2020, it is predicted that the digital world will grow 44 times as big as it was in 2009 [6]. There was a time when data was used to count in kilobytes, megabytes, gigabytes, but nowadays data is estimated in zettabytes [18]. "In 2005, the World Wide Web had 11.5 billion indexable documents. By 2007, that number grew to 25 billion, and by 2012 it doubled again, topping out at more than 56 billion" [28].

As a result, companies started collecting data in huge repositories. These database repositories had some valuable information and to gather and to extract that valuable information, companies or industries needed some tools and techniques from which they could extract that beneficial information for their companies future. Data mining was the field which provides these techniques. Consequently, retail companies and the financial communities started making decisions based on the database history. More and more companies started using data mining techniques to analyze the data to predict their customer demands, fluctuations in interest rates, stock prices and many more valuable information related to their companies [13].

Data mining and knowledge discovery in databases (KDD) handle the advanced techniques and tools for handling a large amount of data. These techniques are used to extract valuable information from the database which is very beneficial for an organization to find the patterns to make their business better [13]. To understand the concept of data mining and knowledge discovery in databases, we have briefly explained the steps that are involved in data mining and knowledge discovery in databases (KDD).

A.1.1 Knowledge Discovery

The list of steps involved in the knowledge discovery process is Data Cleaning, Data Integration, Data Selection, Data Transformation, Data Mining and Pattern Evaluation [9].

- Data Cleaning - Data cleaning is removing the inconsistent data and noise.
- Data Integration - Multiple data sources are combined in this step.
- Data Selection - In this step, the relevant data for the analysis task are retrieved from the database.
- Data Transformation - Data is transformed into appropriate structure for mining by performing summary or aggregation operations.
- Data Mining - Different methodologies are applied in order to extract data patterns.

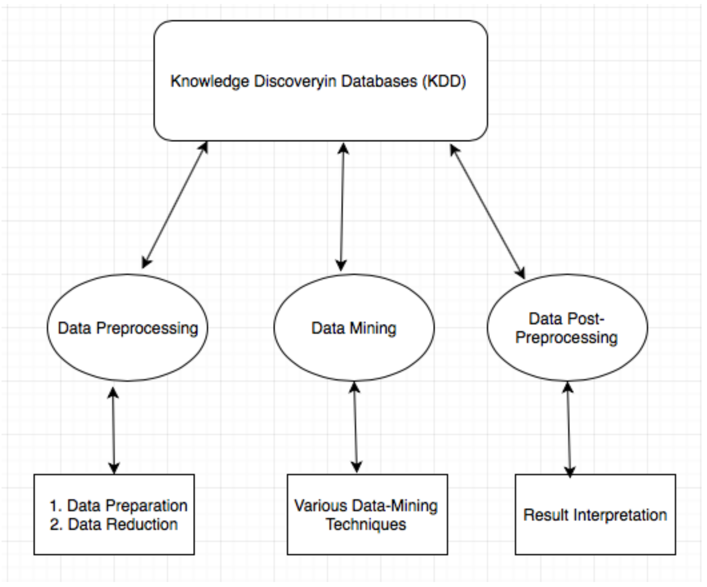


Figure 1: Fig taken from [12] The three stages of Knowledge Discovery in Database (KDD).

- Pattern Evaluation - Data patterns are evaluated in this step.
- Knowledge Presentation - In this step, knowledge is represented.

A.1.2 Data Mining

There are two categories of functions involved in Data Mining, the first is Descriptive and the second is Classification and Prediction [9].

(i) **Descriptive Function**

The descriptive function deals with the characterize properties of data in the database(target data set). The list of descriptive functions is shown below.

(a) Class/Concept Description
(b) Mining of Frequent Patterns
(c) Mining of Associations
(d) Mining of Correlations
(e) Mining of Clusters

(a) Class/Concept Description

Class/Concept is the simplest kind of descriptive data mining and data can be associated with classes or concepts. Let's take an example, if there is a showroom of Nike/Adidas, there would be items for sales such as shoes, t-shirts, caps and much more, these are classes of items and the concepts of customers include frequent buyers or regular buyers. Such description of a class or a concept is called class/concept description. There are two ways in which we can describe this, one is Data Characterization and Data Discrimination [15].

- Data Characterization - Data Characterization provides a concise and succinct summarization of the given dataset and this class under study is called as Target Class.
- Data Discrimination - It provides discriminations comparing two or more collections of data.

(b) Mining of Frequent Patterns

The patterns which occur frequently in transactional data. There are many kinds of Frequent patterns which are listed below.

- Frequent Item Set - The items that frequently appear together, for example, corn flakes and milk.
- Frequent Subsequence - A sequence of patterns that occur frequently or an item which customers tend to purchase in the pattern such as mobile phone followed by screen guard for that phone and then memory card.
- Frequent Substructure - Substructure refers to different forms of the structure such as graphs, trees or lattices, which may be combined with item-sets or subsequences.

(c) Mining of Association

Association in data mining is to find the frequent patterns. Association is the probability of the co-occurrence among sets of items in the relational database. Association analysis is mostly used for the market basket or transaction data analysis to identify patterns that are frequently purchased together. For example, an association rule is generated in a retail shop that shows that 60% of time bread is sold with eggs and only 30% of times milk are sold with bread.

(d) Mining of Correlations

It is a kind of additional analysis performed to discover interesting or unusual patterns, it indicates the strength and direction between two item sets to analyze if they have positive, negative or no effect on each other.

(e) Mining of Clusters

A group of similar kind of objects is called cluster and in cluster analysis, a group of objects is formed that are very identical or similar to each other.

(ii) **Classification and Prediction**

(a) Classification - Classification is the process by which we can describe and distinguish classes or concepts through a set of models or we can say that Classification predicts categorical class labels which are unknown. The main objective is to find a derived model, which is based on the analysis of training dataset.

(b) Prediction - Prediction models are continuous-valued functions, it is used to predict numerical data values which are missing or unavailable. Regression Analysis is generally used for prediction.

In figure 2 , the steps of Data Mining are shown in detail.

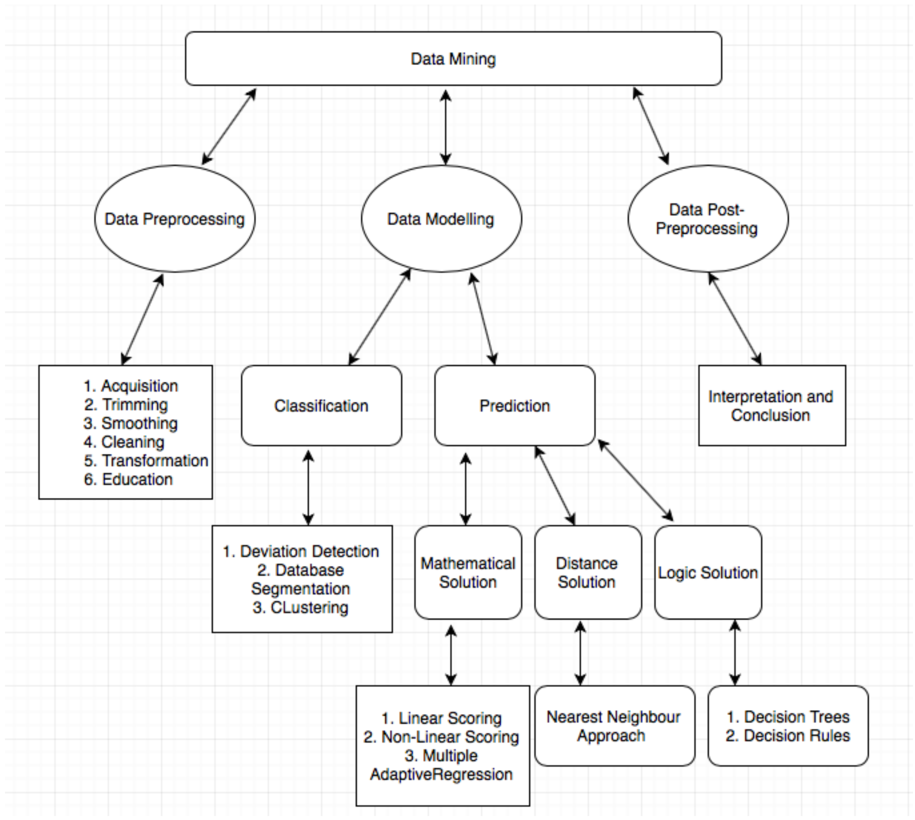


Figure 2: Figure taken from [12] Data-Mining steps

A.2 Trends and Developments

Data mining is an emerging discipline, but still, there are many issues and challenges in terms of technologies and methodologies. One of the reasons is the tons of data which are generating every day and just because of this reason, it exceeds the processing capacity of the databases. The development of efficient and effective methods are currently being researched [16]. Nowadays data mining techniques are used in each and every field such as data-mining techniques are adopted by many banks, retailers and also used by many credit firms and because of this data mining has grown into a billion dollar business [17].

One of the most recent trends is developing data-ming and machine learning in which researchers are trying to discover common sense knowledge by using big data (large tons of data). In this research, they are trying to find out how to retain knowledge and results from the past and then that knowledge can be used to help in future learning and problem solving.

Datasets which are used to predict the result are often inaccurate or we can say incomplete because most of the datasets which are generated are based on surveys. In surveys, the completion rate is very low, and most of the people do not even care about it, they just fill it for the sake of completing it. Not all the available data is useful, and historical data may not generate good rules or prediction models. Most of the times surveys are essential, biased in one direction only. So, there is a need of a data-mining tool which supports multiple outcomes and switch to multiple techniques.

A.3 Statement of the Problem

There are many online betting sites and online bookmakers that allow betting money virtually in every sport. These bookmakers operate based on standard principles and they establish the odds for sporting events. The purpose of this paper is to predict the final result of the football matches of the Italian League based on the odds of various bookmakers. In data mining, it depends on the analyst to make a choice of the technique to analyze a data. So, In this project, we are free to use any predictive analytical technique to predict the result, but the tool which we have to use in this project to predict the final results using predictive analytics is "R".

In this project, we have used four predictive analytical techniques : Basic Tree Model Technique, Random Forest, Neural Networks and Xgboost. In this project, we have discussed some of the data preprocessing techniques that will help to reveal the nature of the data sets. We have also done some experiments to compare the techniques and methods by changing the parameters and have recorded the final results. Hence, this project would help the learners and experts as well by helping them to choose the best method in case of predictive analytics.

A.4 Structure of the Report

In this report, there are five chapters, that are A,B,C,D and E. The chapters are described briefly as follows:

Chapter A - The first chapter provides an introduction to the thesis, It discussed and explained the meaning of data-mining. It also covers the evolution of data-mining, problem statement (objective), trends and development.

Chapter B - It includes a literature review on predictive analytics, predictive analytic process and the limitations of predictive analytics.

Chapter C - In this chapter, we have mentioned some important data-reprocessing techniques such as data filtering and smoothing.

Chapter D - In this chapter, we have described the methodology wich are employed in this thesis and an introduction of the data sets used in this project. In this chapter, we have explained the code and also shown the observations by changing some of the parameters and which we found out during the implementation.

Chapter E - In this chapter, we have given the summary of the results. Chapter fifth includes the advantages of the techniques which are used and the conclusions about the report. In this section, we have also discussed the future possibilities of the research.

B Literature Review

In recent times, ”Big Data” has been the talk in the analytic sector. The questions which one should ask is what is ”Big Data”? Why is it so important? Where does ”Big Data” come from? What’s the relevance of ”Big Data”? This chapter gives the idea of Big Data and how predictive analytics is associated wth Big Data. This chapter also explains the various predictive data-mining techniques used to accomplish the goals and the methodology.

B.1 Predictive Analytical Techniques

”Big Data” is an act of gathering and storing a large amount of data, but it is not only collecting and storing the data or arranging it in order, the important thing is after gathering and storing what benefits, an Organization will get from that data which would be fruitful. This large amount of data which has been collected from the different sources is then analyzed which gives the answers like smart decision making, cost reductions, time reductions, new product development and optimized offerings. This large amount of data is analyzed which would be helpful in determining the cause of failure, issues, and defects, what are the behaviors that affect the Organization and much more [28].

Predictive analytics is a branch of data mining or one can say that it is a subset of data science which is used to make predictions about unknown future events. Predictive analytics is used to analyze data which helps to predict trends and behavior patterns. Predictive Analytics encompasses many techniques from statistics, data mining, machine learning and artificial intelligence to analyze the data. Predictive Analytics anticipates the future using past events. [8].

Consider the power of predictive analytics:
Below we have given some examples, how Predictive Analytics would be helpful in some areas [8].

- Predictive analytics can help the scientists (research groups) to improve the ability to classify and cure cancer or brain tumors.
- Predictive Analytics would be very helpful in a travel industry by increasing the revenue by helping them to predict what the customers want and then adapt the offer accordingly. For example bringing together a flight, extra baggage, a hotel, while also slightly adapting the price dynamically.
- Predictive Analytics can help the University to predict how many students will enroll for a particular course by using the admission history.

Predictive analytics play an important role in many areas such as detecting frauds, managing risk, targeting and retaining and even cyber-attacks on the systems. Some of the applications of data mining are shown below.

B.2 The Predictive Analytic Process

Most of the experts agree that Predictive Analytics is an artistic work and it requires great skill. Many frameworks have been proposed in the literature for the data-mining model building but

According to Wayne's survey (Based on 167 respondents who have implemented predictive analytics) [8] most of the groups (52%) have their own methodology, 22% says that there is no particular framework and 29% adapts vendor methodology or process. Some of the methodologies have been proposed which are based on some basic industrial engineering frameworks or business improvement concepts and these frameworks proposed to serve as blueprints for how to organize the process of data collection, analysis,results from dissemination and implementing and monitoring for improvements. One of them was proposed in 1996, called the Cross Industry Standard Process for Data Mining (CRISP-DM) According to Wayne's survey [8] only 15% groups are using this methodology.

CRISP-DM

This is the Cross-Industrial Standard Process for data mining proposed by a European consortium of companies in the mid-1990s. CRISP is a robust and well-proven methodology which provides a structured approach to planning a data mining project based on business and data understanding, then data preparation and modeling, and then on to evaluation and deployment [4].

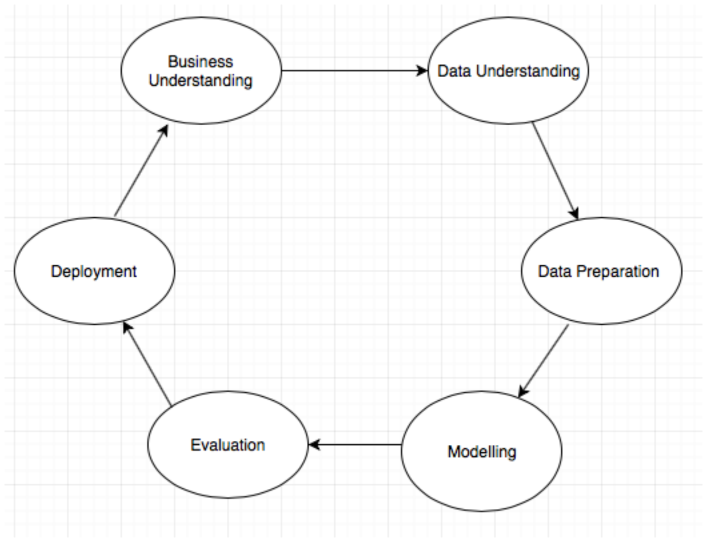


Figure 3: Figure is taken from [4]: Process diagram showing the relationship between the different phases of CRISP-DM

Regardless of all methodologies, the most common processes for creating predictive models incorporate the following steps:

- Project Definition: In this step, Business objective and desired outcomes are translated into predictive analytic objectives and tasks.
- Data Collection: In this step, the data is prepared from multiple sources for analysis.
- Data Preparation or Data Analysis: In this step, select, extract, clean and transform the data to create models.

- Model Building: Test and validate the models which are created and then evaluate whether they will meet project metrics and goals.
- Deployment: Apply model results in everyday decision-making process to get results, reports, and output by automating the decision based on the modeling
- Model Management: In this step, the models are managed improve performance and monitored to ensure that it is providing the expected result.

B.3 Limitations

Predictive analytics is growing rapidly in every sector and Predictive Analytics is not without flaws because data is created by human, So it's apt to have some limits on its usability. Some of the limitations of Predictive Analytics are mentioned below [29].

- The data which is used for prediction could be incomplete which could limit its usability.
- Sometimes data is taken from a survey and we should keep in mind that people don't always provide accurate information
- The data which is collected from different sources can vary in quality and format.
- There are missing values in the data which could create a problem.
- Always check for extreme values (outliers) and decide on whether to include them in the analysis or not.
- The model which is used may be successful but customer behavior changes with time and therefore a model must be updated.
- The training data must be representative of the test data otherwise predictive analytics cannot be successful.

C Data Preparation

Data preparation is cleaning and transforming the data. The data which is in a raw form need to be cleaned (cleaning the data from errors) and analyzed. The data must be preprocessed and prepared for analysis and after preprocessing the data is transformed into a format which is readable by an analytical tool. There are many techniques for data preparation because predictive data-mining techniques behave differently depending on the preprocessing and transformational methods [8]. The techniques for data preparation that can be used to achieve different data-mining goals, which are explained below in detail [12].

C.0.1 Data Filtering and Smoothing

Data preprocessing is a technique in which the data is filtered to get rid of noise and outliers. There is a process called smoothing, in which filtering is used to pass some data values and some hold back depending on the modeler's restrictions [33]. There are several filtering methods, the techniques which are important and relevant to this project are explained below in detail [12].

(a) Moving Average Filtering

The moving average filter is used for general-purpose filtering commonly used for smoothing an array of sampled data. It is used for both high and low frequencies. This filtering technique reduces the variance of the series [12] but it has a drawback that it forces all the sample points in the window averaged to have equal weightings. The smoothness of the output increases with the increase in the filter length, but it made the data blunt which shows that the time domain response of this filter is very good but the frequency response is poor [24].

(b) Median Filtering

The median filtering is a non-linear digital filtering method which is widely used in digital image processing and for time-series data sets in order to remove outliers or bad data points. The median filter is also a sliding-window spatial filter, in which it moves through the image pixel by pixel and replaces the center value in the window with the median value of neighboring pixels. It is used in signal enhancement for the smoothing of signals, the suppression of impulse noise, and the preserving of edges but it has some side-effects, for example, it blurs the changes in the image such as line features, sharp edges and other image details all corresponding to high spatial frequencies [31].
Specifically, the median filter replaces a pixel by the median, instead of the average, of all pixels in a neighborhood w

$$\text{y[m,n]} = \text{median } \{ \text{ x [i , j], (i , j) } \varepsilon \; w \; \}$$

where w represents a neighborhood defined by the user, centered around location $[m, n]$ in the image.

Consider a 5x5 window sliding (either horizontal or vertical) of pixels. Assume the twenty five pixels currently inside the windows are:

23	25	26	30	40
22	24	26	27	35
18	20	50	25	34
19	15	19	23	33
11	16	10	20	30

Table 1: Median Filtering Example

Table 3.1 In the above example, the middle pixel in the window with value 50 is an isolated out-of-range and is, therefore, likely to be noisy. We have to replace this value with the median which can be found by sorting the values (in either ascending or descending order) and then taking the median. So we are taking all the neighboring values around the pixel 50 in ascending order and that are 15, 19, 20, 23, 24, 25, 26, 27, 35. The original pixel value 50 is replaced with 24 because 24 is the median.

(c) Normalization/Standardization

Normalization is a scaling technique/mapping technique or data preprocessing stage [32].There are two important goals of data normalization, first to eliminate or reduce data redundancy by organizing the columns (attributes) and tables (relations) of a relational database and the second to ensure data dependencies make sense by storing only the related data in a table. It is a method to expose information content within the data set by changing the instance values in specific and clearly defined ways. The measured values can be scaled to a range from -1 to +1. This method includes both the decimal and standard deviation normalization technique. When normalization is performed the value magnitudes and scaled to appreciably low values and its is important for many neural networks and k-Nearest Neighbourhood algorithms. [21] [34]. According to [21] there are two most common methods for this scope and that are min-max normalization and z-score normalization.

- min-max normalization

  $$v' = \left(\frac{v - min_A}{max_A - min_A}\right) * (new\_max_A - new\_min_A) + new\_min_A)$$

  Where

  v' contains Min-Max Normalized data one
  A is the range of original data

  The min-max normalization is a technique which keeps relationship among original data. There's only one thing we have to keep in mind while using this technique and that is the value of unseen data point shouldn't exceed [minA, maxA] interval.

- z-score normalization

  $$v' = \left(\frac{v - \mu_A}{\sigma_A}\right)$$

  where
  v is the old feature value and v' the new one.

  $$\mu_A$$

  and

  $$\sigma_A$$

  are the mean and standard deviation of attribute A.

The above formulae can be used to normalize the unstructured data by using the concepts like mean and standard deviation. If

$$\mu_A$$

and

$$\sigma_A$$

are not known they can be estimated from the sample.

where,

$$\sigma_A = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(v - \mu_A)^2}$$

and

$$\mu_A = \frac{1}{n}\sum_{i=1}^{n}(v)$$

(d) Fixing missing and empty values

Missing or empty values is a very common problem in data preparation process. There are several methods to process missing data in datasets and the problems caused by it. A problem often occurs in a variable, whose values are missing or has not been entered into the data set. For example, in surveys, people generally don't want to mention their income and they leave that field blank and sometimes people don't have any information available and they leave that field blank. Sometimes may be accidently data lost while collecting information or data from multiple sources . These missing values should be fixed before data mining process proceeds because most tools for data-mining find it difficult to digest such values. Whereas there are some data-mining tools which ignore these missing and empty values, while some tools automatically determine some suitable values as a substitute for the missing values. A problem occurs when the tools automatically fix values or proceeds without fixing these missing values because in that case modeler is not in control of the operation and that the possibility of introducing bias in the data is high [7] [33].

There are some better ways to tackle the missing or empty values in a dataset but the two main important approaches are : (a) The first approach is to replace the missing values by the mean of the existing data. This approach does not change or disturb anything while applying the approach (b) The second approach is better than the first one, in this approach the missing values are replaced by the standard deviation which is closest to the actual data [12].

(e) Categorical Data

Most of the times data-mining models are done using quantitative variables. Quantitative variables are the variables that are measured on a numerical scale or number line. Sometimes dataset contains quantitative variables (categorical or indicator variables). Categorical data is a data which belong to a definable category for example gender (male or female) and the values of this type are not ordered. In that case, the categorical data is converted to numerical data by assigning a numeric value (or code) for each label. There is a naive way of making this translation and which is commonly used in the data-mining process. In this approach, the numbers are assigned to the nominals to create a number list. In qualitative variables there are different levels, suppose yes or no, high or low, or sometimes may be more than two levels for examples high/low/medium [12] [27].

D Data Introduction and Methodology

Before proceeding to the methodology and the data set introduction, we have also provided the basic information about the tool (R) we have used.

D.0.1 R

R is developed at Bell Laboratories (formerly AT&T, now Lucent Technologies). R is a language, which is used for graphics and statistical computing like linear and nonlinear modeling, classical statistical- cal tests, time-series analysis, classification, clustering and much more. R can be easily extended and there are about eight packages in R and much more are available through CRAN family of Internet sites cover- ing a very wide range of modern statistics. R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, One of the R's strength is the ease to use mathematical symbols and formulae. R is available as free software for both Windows and Mac-OS [11].

D.0.2 Data Introduction

In this project, one dataset is used to predict the final result of the football matches. The data set is obtained from https://inclass.kaggle.com/c/football-data-challenge/data, in which the variables are described in detail. For the purpose of this work, R was used for all of the analyses. To gain a superficial knowledge and to see the nature of the datasets, preliminary analyses were done on the data sets before applying any technique. Preliminary analyses are very important in real-life analyses because it helps the analyst to gain knowledge about the data and making a choice of the technique to be used in the regression work.

The data set which is obtained from https://inclass.kaggle.com/c/football-data-challenge is split non-randomly into two data sets, train data, and test data. The training data (training dataset) is used for supervised learning, in which the input data is given with the correct or expected output for every data row and the test data (testing dataset) is on which the model is applied.

train.csv - the training set : 610 obs. of 22 variables
test.csv - the testing set : 1520 obs. of 23 variables

Format of the File

The format of the file is "CSV" stands for "Common Separated Values" which is often used to exchange and convert data between various spreadsheet programs. There are cells inside such data file, which is separated by a special character, which usually is a comma and other characters can also be used as well. The first row of this data file contains the column names instead of actual data. [35].

Data Fields

- ID = An anonymous ID unique to a given match
- Date = Date of the match
- FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win). This is the target variable
- HomeTeam = Home Team
- AwayTeam = Away Team
- B365H = Bet365 home win odds
- B365D = Bet365 draw odds
- B365A = Bet365 away win odds
- BWH = Bet&Win home win odds
- BWD = Bet&Win draw odds
- BWA = Bet&Win away win odds
- IWH = Interwetten home win odds
- IWD = Interwetten draw odds
- IWA = Interwetten away win odds
- LBH = Ladbrokes home win odds
- LBD = Ladbrokes draw odds
- LBA = Ladbrokes away win odds
- VCH = VC Bet home win odds
- VCD = VC Bet draw odds
- VCA = VC Bet away win odds
- WHH = William Hill home win odds
- WHD = William Hill draw odds
- WHA = William Hill away win odds

Importing Data

Before performing any function, the first step is to import data in R, which is fairly simple. After downloading the data, there is a function called "read.csv" through which the data can be read.

```
train = read.csv("/Users/Bhat/Downloads/train.csv",header = TRUE,sep=",")
test = read.csv("/Users/Bhat/Downloads/test.csv",header = TRUE,sep=",")
```

In the above code, it shows that the data is imported into R where testing data is stored into test and training data is stored into the train.

In the second step, we have converted our dataset values into numeric with the command (as.numeric). Lets us understand why we have used this command and change the dataset to numeric before proceeding any methods on that.
Imagine that the our dataset is stored in train.csv, then the R command:

train = read.csv("/Users/Bhat/Downloads/train.csv",header = TRUE,sep=",")

will store the data in the R variable "train" as a data frame. If we will change our dataset to a matrix, then it would be of no use because all our data are numeric. The R command for matrix :

train = as.matrix(train)

Here the problem comes, if one of our columns contained non-numeric data then all the data need to be converted to characters. For example, the value 5 would now be stored as "5" and if we perform any arithmetic operation would result in an error. This error could be avoided by using a (as.numeric) command, this command will convert 2 back to the numeric value. More generally, if the entries in the column five were stored as "1" (married) or "2" (single) we will sum the values with the command : **(sum(as.numeric(test[,5])**. It is also possible to have the column "5" stored as numeric with other column stored as characters with the data frame.

D.1 Basic Tree Model

The first technique which is used to predict the final result is using decision tree model. In this approach, all the teams are ignored and only the odds which are given are considered and then the basic tree model is used to get the final result. A decision tree model is one of the most common techniques in data mining. This technique is very popular because to understand the resulting model is very easy and the algorithms use the recursive partition approach. Let us first understand what is basic tree model and what is CART.

Tree-Based Models

Recursive partitioning is a primary tool in data mining which helps to explore the structure of a dataset. Tree-based methods are simple and very useful for interpretation. Tree-based learning algorithms are one of the best and mostly used supervised learning methods while developing it helps to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome which is also called as CART (classification and regression tree) [38].

Classification Trees vs Regression Trees

The terminal nodes (leaves) lies at the bottom of the decision tree. So it means that decision trees are typically drawn upside down such that the leaves are at the bottom and the roots are at the top.

Regression Trees and Classification Trees are almost similar to each other. The primary differences and similarity between them are discussed below.

- The primary objective is to choose the right tree and it is the dependent variable that determines the type of decision tree needed. When the dependent variable is continuous then Regression Trees are used and when the dependent variable is categorical then Classification Trees are used.
- Regression Tree - In training data set the terminal nodes to obtain some value and that value is the mean response of observation falling in that region. So if there is an unseen data falls in that region, we make its prediction with mean value.
- Classification Tree - In the training dataset, the value obtained by the terminal node is the mode of observations falling in that region. Thus, if an unseen data observation falls in that region, we make its prediction with mode value.
- Classification Tree and Regression Tree divides the independent variables (predictor space) into distinct and non-overlapping regions and these regions are nothing but the high dimensional boxes.
- Classification Tree and Regression Tree follow a top-down greedy approach from a root node which is also called as a recursive binary splitting. In this approach, the data is partitioned into subsets that contain instances with similar values (homogenous). It is a greedy approach because the algorithm only looks for best variable available and only cares about the current split but not about future splits which will lead to a better tree.
- The splitting process continues in both the cases until it reaches the value which is defined by the user.
- The splitting process results in fully grown trees. The fully grown trees sometimes overfit data which leads to poor accuracy on unseen data.
- The technique called pruning is used to tackle overfitting and this technique is performed in order to remove anomalies in the training dataset due to noise or outliers. The pruned trees are smaller and less complex.

D.1.1 Methodology

Every methodology or technique has its own packages, before applying the technique we need to download the package and then we can start working on that technique. In this method, we have used a library (tree), which is a primary R package used for classification and regression trees. This package can be used to prune the trees and used for general plotting functions and the misclassification (total loss) as well. By using complete classification tree, we will fit the data using the variables and then we will try to prune the tree to make it smaller using cross-validation technique. First, we will try to fit the tree model using training data and then we will use that model on test data to predict the final FTR.

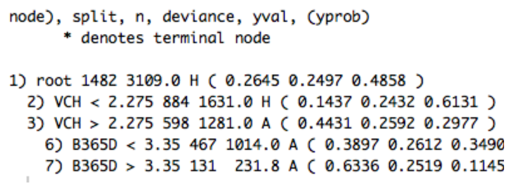
train.tree=tree(FTR ∼. -ID-Date-HomeTeam-AwayTeam,data = train)

The main idea behind the classification tree is to first start with all variables in one group and then find some characteristic variable that best separates the groups. So in our case, we are going to predict FTR and the rest of the variables (except FTR) are the predictors, the predictors will help us to predict the final result and the formula to fit the data using the variables are shown above. In the above formula, we have excluded these four variables (ID, Date, HomeTeam, AwayTeam) and we will use the rest of the variables to predict the FTR.

We have visually tried to inspect the data in fig. 4, which is shown below. Fig 4 (a) is a legend for reading the information in a textual description of the decision tree. This textual description is actually giving us the idea, how the tree structure has been built.

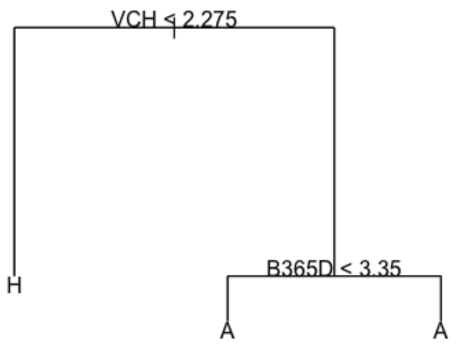
The legend indicates that: numbers which are shown below are the nodes, followed by a split, then "n" is the number of entities at that node, "deviance" shows the probability distribution of the classes, "yval" is the default classification for the node and then "yrob" is the distribution of classes in that node, "*" denotes a terminal node of the tree (a leaf node). The distribution is ordered by levels of the class and the order is the same for all nodes.

Fig. 4(b) is the graphical representation of the tree. Fig. 4(b) shows that if the probability of betting odd "VCH" is less than 2.275 then the result will be "H" (Home Team) and if the probability of "VCH" is greater than 2.275 then it will split again on another betting odd "B365D" which gives us the final result "A" (Away).



```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 1482 3109.0 H ( 0.2645 0.2497 0.4858 )
  2) VCH < 2.275 884 1631.0 H ( 0.1437 0.2432 0.6131 )
  3) VCH > 2.275 598 1281.0 A ( 0.4431 0.2592 0.2977 )
    6) B365D < 3.35 467 1014.0 A ( 0.3897 0.2612 0.3490
    7) B365D > 3.35 131  231.8 A ( 0.6336 0.2519 0.1145
```

(a)



(b)

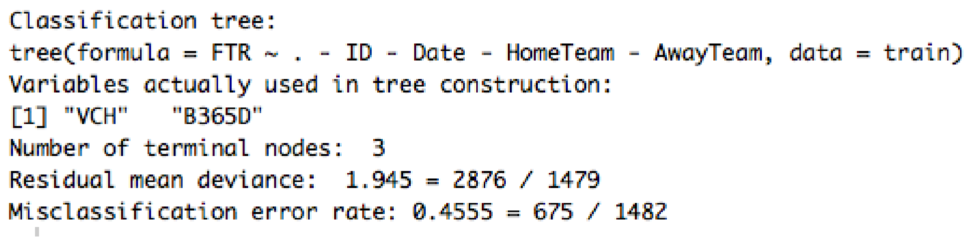
Figure 4: A classification tree showing at each internal node the betting odds and at each terminal node the final result FTR

Before using our model for test data, we will first see the summary of the classification tree.

The summary() function below shows:

- The formula which we have used to fit the model usign the variables.
- The variable are used in tree construction.
- The number of terminal nodes.
- The residual mean deviance (along with residual deviance).
- The misclassification error rate.

Summary(train.tree)



```
Classification tree:
tree(formula = FTR ~ . - ID - Date - HomeTeam - AwayTeam, data = train)
Variables actually used in tree construction:
[1] "VCH"   "B365D"
Number of terminal nodes:  3
Residual mean deviance:  1.945 = 2876 / 1479
Misclassification error rate: 0.4555 = 675 / 1482
```

Figure 5: Classification Tree

From the above summary, we learn that two variables are used to predict FTR (VCH, B365D) and the number of terminal nodes are three. 45% of the predicted values are misclassified in this model (i.e the training error rate is 45%) and that the residual deviance is 1.945. Here deviance means the mean squared error. Deviance reported in the output of the summary for classification tree is given by :

$$-2 \sum_{i=m} \sum_{i=k} n_{mk} \, log \, \hat{p}_{mk},$$

where,

$n_{mk} - is$ the number of observations in the m^{th} terminal node which belongs to the k^{th} class

and residual mean deviance is simply the deviance divided by $n - |t_0|$ which is in our case is (1482 - 2 = 1479)

A number of leaves control the flexibility of the tree since that's how many cells they partition things into. The fitting function that is how many trees will grow depends on the control settings, tree fitting function has a number of controls settings. There are a certain number of points in each node and the error can be reduced by adding the nodes by at least a certain amount. The default for the latter, min.dev is 0:01. Where we have written the formulae, we can add min.dev manually. Let's make the deviation 0.001 and see what happens next.

Now our model is ready and we will check the model using test data to predict the final result (FTR) using the "Predict" function, which will give us the probability for each class. The final result (FTR) will be saved as "submission1".

D.1.2 Cross-Validation to Improve the Model

We got the final result using tree-based models, but now we will cross check our result using cross-validation method. Cross-validation method is used in order to determine the optimal level of tree complexity and pruning technique is used to improve the test error rate. After improving our model, we teste that improved model on our testing dataset to get the final result. In this method, we will prune the tree to prevent overfitting. We will use the function prune.tree() which allows us to choose the number of leaves we want for the tree, and then it will return the best tree with that size. To understand the concept of cross-validation, it is explained below.

Cross-Validation Technique

Cross-validation technique is used to improve our previous model. The purpose of cross-validation is to qualify the model. Cross-validation gives us the better estimate of the performance of our trained model when used on different data and this process can be repeated using different parameters until we are satisfied with the performance. Then it helps us to train the model with the best parameters on the whole data [20].

Cross-validation procedure is based on the optimal proportion between the complexity of the tree and misclassification error. Misclassification error can be decreased with the increase in the size of the tree. Misclassification error becomes zero if the tree size is maximum. Complex decision trees poorly perform on independent data. If the data is independent then the performance of the decision tree is called the true predictive power of the tree. The main aim to use the cross-validation technique is to find the optimal proportion between the tree complexity and misclassification error which can be achieved through the cost-complexity function [37] shown below.

$$R_\alpha(T) = R(T) + \alpha\tilde{T} \ \rightarrow \underset{T}{min}$$

where,

R(T) is the misclassification error of the tree (T)

$\alpha(T)$is the complexity measure which depends on"$\tilde{T}$ "which is the total sum of terminal nodes in the tree.

"α" is a parameter which can be found through the sequence of in-sample testing.

This parameter can be found when a learning sample is used to build a tree and the other part of the data is taken as a testing sample. The process is repeated several times for randomly selected learning and testing samples [37].

Although, we don't need to adjust the parameters everytime while using cross-validation technique because this process is very time-consuming since the sequence of the trees is already constructed. There is one more reason for that which is, testing and learning samples are chosen randomly and the final result would be different every time [37].

Methodology

Now using cross-validation, we will try to improve accuracy by "pruning" the tree, it means we will cut off some of the terminal nodes of the tree. We can also check which tree has the lowest error rate. Since the tree was grown to full depth, we will now use 10-fold cross validation which is by default "K=10" i.e. the data is divided into 10 parts (using cv.tree() function).

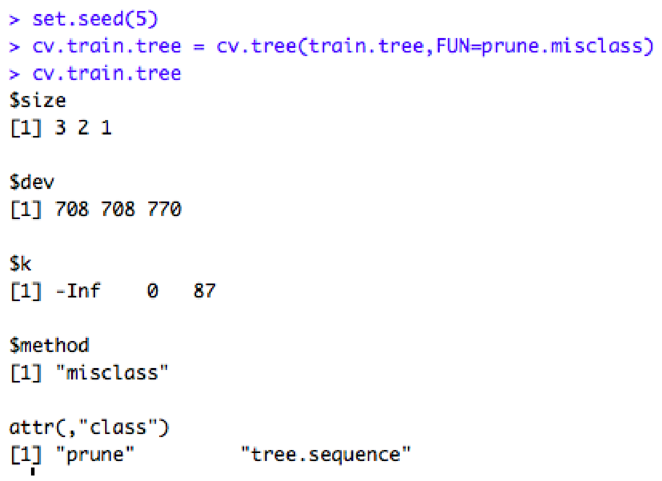
set.seed(5)

The first line is "set.seed", which we have to set every time to get a reproducible random result. It is basically used to optimize a function that involves randomly generated numbers. If we do not fix the seed, the optimization algorithm might fail due to drawing different random number.

cv.train.tree = cv.tree(train.tree,FUN=prune.misclass)

In the second line we have used a cv.tree () function which helps us to determine the optimal level of tree complexity, it reports the number of terminal nodes of each tree considered as well as the corresponding error rate and the value of the cost-complexity parameter. Along with that we have used the function (FUN = prune.misclass). Since we are doing classification, we have to mention the function as prune.misclass. This function (prune.misclass) tells us that we want classification error rate to guide the cross-validation and pruning process rather than the default for the function cv.tree(), which is deviance.

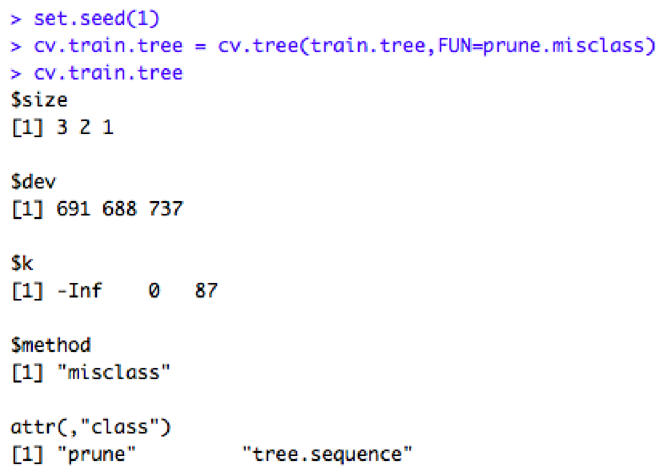
summary(cv.train.tree)



```
> set.seed(5)
> cv.train.tree = cv.tree(train.tree,FUN=prune.misclass)
> cv.train.tree
$size
[1] 3 2 1

$dev
[1] 708 708 770

$k
[1] -Inf    0   87

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Figure 6: Summary using set.seed(5)

In Fig. 6 the function cv.tree() reports the "size" of the terminal node for each tree considered, the corresponding error rate "dev"and the value of the cross complexity parameter "k" . "dev" is the cross-validation error rate and it is shown above that the tree has three terminal nodes which results from the lowest cross-validation error rate of 708. If we change the value of seed then the error rate fluctuates.



```
> set.seed(1)
> cv.train.tree = cv.tree(train.tree,FUN=prune.misclass)
> cv.train.tree
$size
[1] 3 2 1

$dev
[1] 691 688 737

$k
[1] -Inf    0   87

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Figure 7: Summary using set.seed(1)

In Fig. 7, we changed the seed value to "1" and the cross-validation error rate went down to 688 the lowest. We have plotted the error rate as a function of both size and k (with seed value set to 1).

plot(cv.train.tree$size,cv.train.tree$dev,type="b")
plot(cv.train.tree$k,cv.train.tree$dev,type="b")

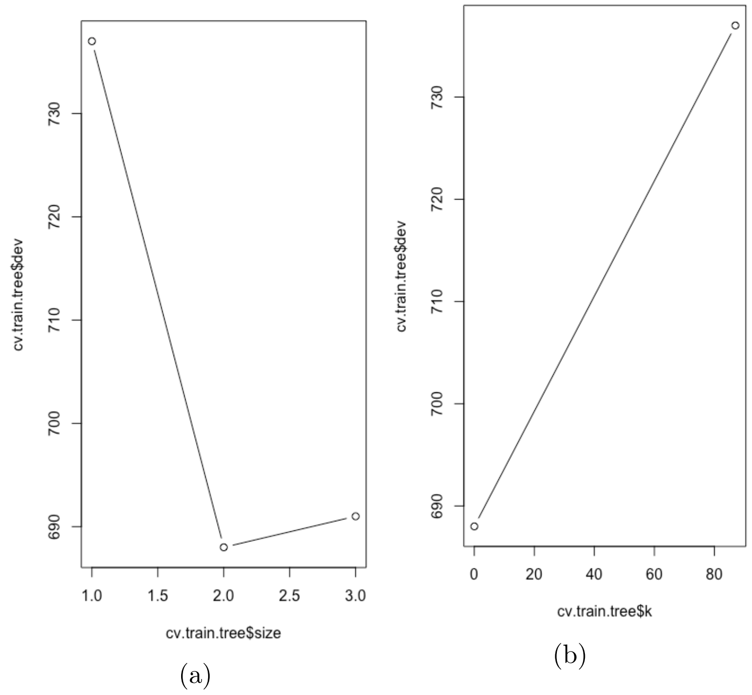


Figure 8: The error rate as a function of both size and k

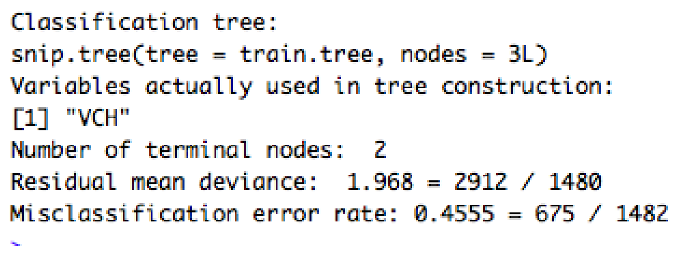
In the above figures, Fig. 8(a) shows the lowest error rate corresponds to the most complex tree with 2 leaves (in our case) if we wanted to prune the tree, we have to select the best size to 2 (lower error rate) using the prune.tree() function. The Fig. 8 (b) shows the tree has three terminal nodes which result from the lowest cross-validation error rate of 688 and highest to 737.

We used the plots and after careful observation, we got to know that, In our case, the best size is "2". Now we will use the prune.misclass() function to prune the tree and after that once again we will use the predict() function to check the performance of the pruned tree on the test data.

prune.train = prune.misclass(train.tree,best=2)
tree.predict = predict(prune.train,test,type="class")

Now let's again see the summary of the classification tree, to check after using cross-validation technique, is there any difference in the residual mean deviance and misclassification error rate than the previous classification tree or not.

summary(prune.train)



```
Classification tree:
snip.tree(tree = train.tree, nodes = 3L)
Variables actually used in tree construction:
[1] "VCH"
Number of terminal nodes:  2
Residual mean deviance:  1.968 = 2912 / 1480
Misclassification error rate: 0.4555 = 675 / 1482
```

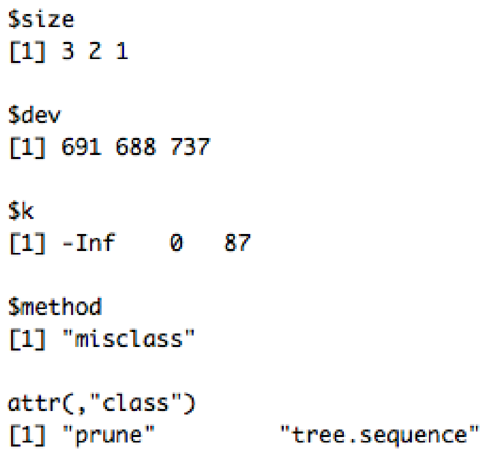
Figure 9: Classification Tree

We can see that the classification tree for the cross-validation is not same as the classification tree before using cross-validation technique (Fig. 5). Some changes happened after using cross-validation technique. Previously before using cross validation technique the number of terminal nodes was three and now it is two and the variables used in tree construction were two (VCH and B365D) and now it is only one (VCH). Misclassification rate is same for both the classification tree but there is a slight change in the residual mean deviance. Previously it was 1.945 and after using cross-validation it changed to 1.968. So in our case, we can say that the cross-validation technique doesn't improve our model.

D.1.3 Observations

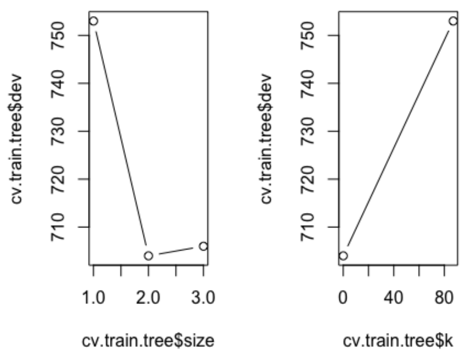
We did some experiments to observe some changes in the classification tree. We changed the values of the seeds and we found out the change in the error rate, shown below.

Without Seed



```
$size
[1] 3 2 1

$dev
[1] 691 688 737

$k
[1] -Inf    0   87

$method
[1] "misclass"

attr(,"class")
[1] "prune"          "tree.sequence"
```

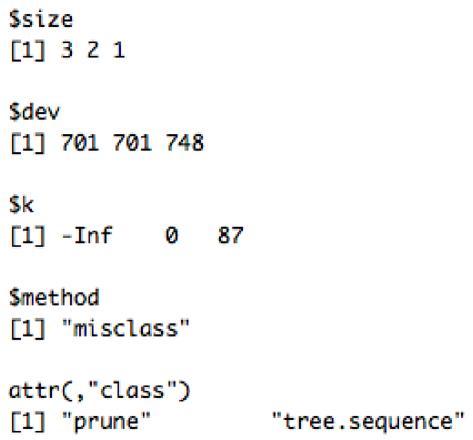
(a) Summary without seed



(b) graph of the summary (without seed)

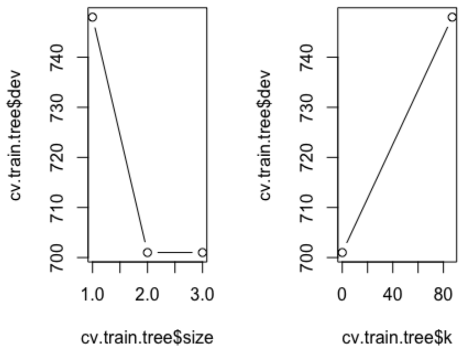
Figure 10: Error in the classification tree with leaves without seed

Seed value set to 10



```
$size
[1] 3 2 1

$dev
[1] 701 701 748

$k
[1] -Inf    0   87

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

(a) Summary with seed value 10



(b) graph of the summary with seed value 10

Figure 11: Error in the classification tree with seed value 10

As we can see in the above fugures Fig. 10 and Fig. 11, the difference between the two. How the error rate got changed with setting the seed value to 10 and the best to select for pruning is now 2 and 3. We can randomly select the seed value and can check the difference.

D.1.4 Packages

library(Tree)

The package which is required for tree-based models is "Tree".

For cross-validation no extra packages are required.

D.2 Random Forest

Random forest is a very popular method for predictive analytics. Random forest is an ensemble of decision trees. It is a type of “ensemble learning” technique for classification, regression and for some other tasks as well which extends on decision trees. Ensemble learning is a procedure which gives a prediction value, in this approach a team of predictive models is constructed to solve the given prediction task [23].

Random Forest grows many classification trees using a subset of the available input variables and their values. Each tree in the forest gives a classification by putting an input vector down step by step to each of the trees in the forest and at every step, a new object is formed from the input vector. It is like tree votes for that class and the forest chooses the one with the maximum number of votes [23].

D.2.1 Properties of Random Forests

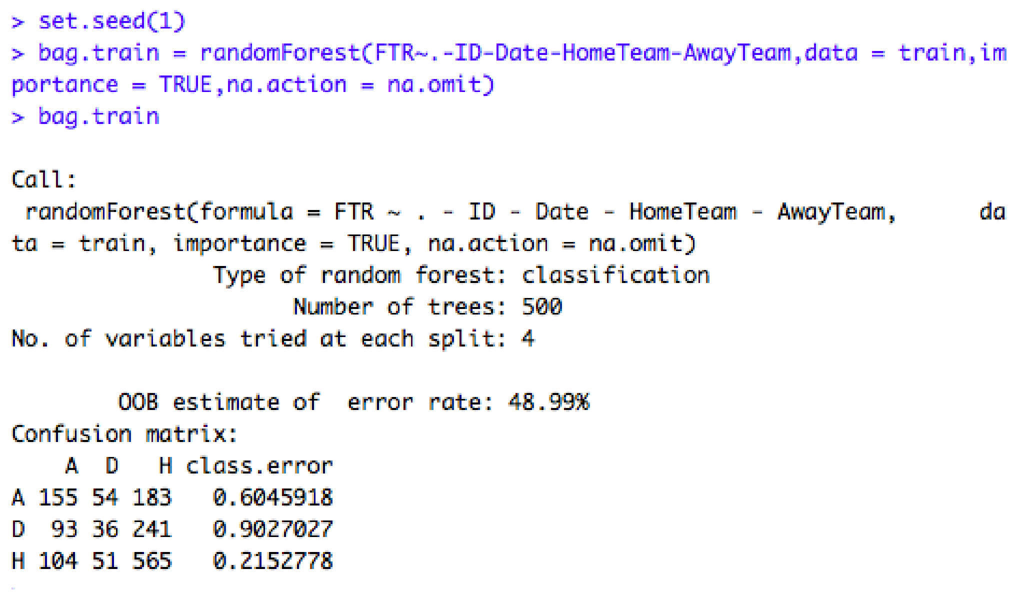
The properties of Random Forest are listed below.

- Random forest can run multiple trees in parallel, therefore it is easy to train large number of data.
- Random forest works very well on a wide variety of data.
- Random forest is also very effective in eliminating noise in the model input data.
- Random forest can also handle large number of input variables without variable deletion.
- The forests which are generated can be saved in future for some other data.

D.2.2 Methodology

In this section we are going to apply random forest approach, In this approach, we will compare the performance of random forest to that of the single tree we fit in the previous "basic tree model" technique. Random forests are like a body of trees that are built using a variant which is specially called "bagging". The important thing in any technique is to construct a model and for that random forests only uses a proportion of the predictors.

Before implementing random forest technique, we need a package called "randomForest" which contains a randomForest () function, along with that bagging can also be performed using this function because bagging is a special case of random forest. Random forest is performed as follows :



```
> set.seed(1)
> bag.train = randomForest(FTR~.-ID-Date-HomeTeam-AwayTeam,data = train,im
portance = TRUE,na.action = na.omit)
> bag.train

Call:
 randomForest(formula = FTR ~ . - ID - Date - HomeTeam - AwayTeam,      da
ta = train, importance = TRUE, na.action = na.omit)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 48.99%
Confusion matrix:
    A  D   H class.error
A 155 54 183   0.6045918
D  93 36 241   0.9027027
H 104 51 565   0.2152778
```

Figure 12: OOB estimate error and confusion matrix

First, a formula is used to fit the data for randomForest () function. Before proceeding to the next step, we have to first get rid of the empty values (N.A) and we have to fix them. For missing and empty values we have used a function (na.action = na.omit), which will omit the missing values, otherwise, warnings and errors will be present.

After using the formula, we got the summary which is shown in the above figure. The summary contains a briefing of the variables and the trained model. It shows the type of random forest "classification", a number of tree "500" and the number of the variables tried at each split "4". The default number of splits at each node is the square-root of the total number of predictors for classification trees, and the number of predictors divided by three for regression trees. The OOB (out-of-bag) error is nothing but the complementary to the accuracy. The confusion matrix gives us the summary of how many cases were guessed right from our model. The confusion matrix is based on actual response variable and predicted value. In our case, the OOB error is 48.99%, which is equivalent to saying that the accuracy of our model is 51.01%. We can also manually select the number of trees by (ntree=100) and how many predictors we want to consider for the each split of the tree by (m=13) in the formula.

Plot(bag.train)

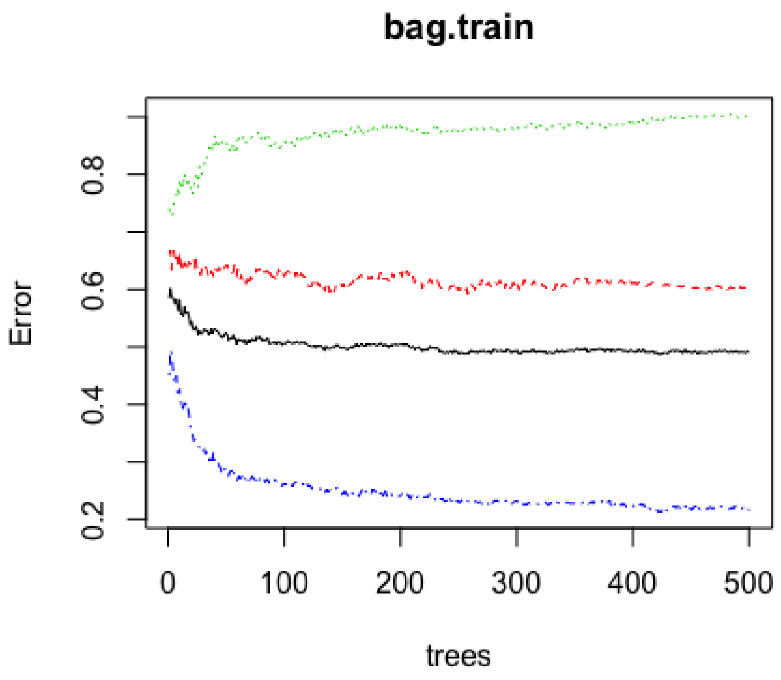


Figure 13: Error Rate vs Number of Trees

Though the initial errors were higher, as the number of trees increased, the errors slowly dropped somewhat over all. The black line is the OOB "out of bag" error rate for each tree and the rest of the three lines are our outcomes, red is for "A", green is for "D" and blue is for "H". When there are few trees the error rate is high, but as more trees are added the error rate decreases and eventually flatten out.

There are three tuning parameters which are important when building random forests, that are node size, the number of trees, and the number of predictors sampled at each split. If we tune

Tree Size	Error Rate
100	50.54%
200	50%
300	49.46%
400	49.26%
500	48.99%
600	48.53%
1000	48.25%

Table 2: OOB Estimate Error vs Tree Size

these parameters carefully, it can prevent extended computations with little gain in error reduction. For example, if we reduce the size of the tree manually, the error rate will increase and vice versa.

We changed the size of the tree and we tested the error rate.

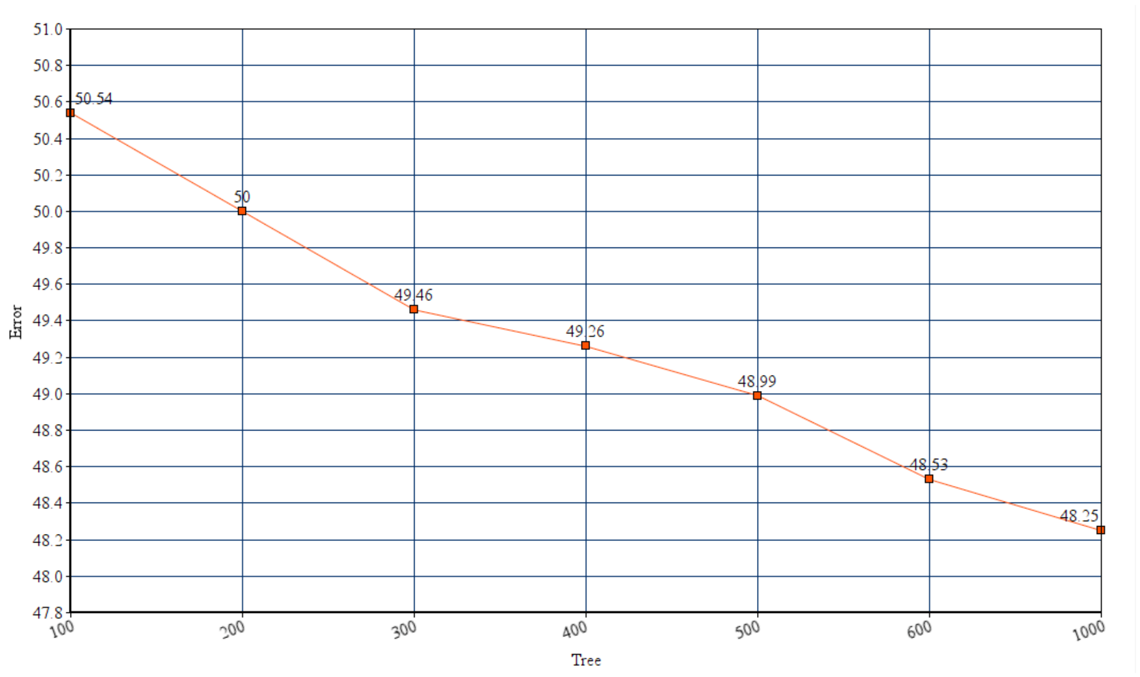
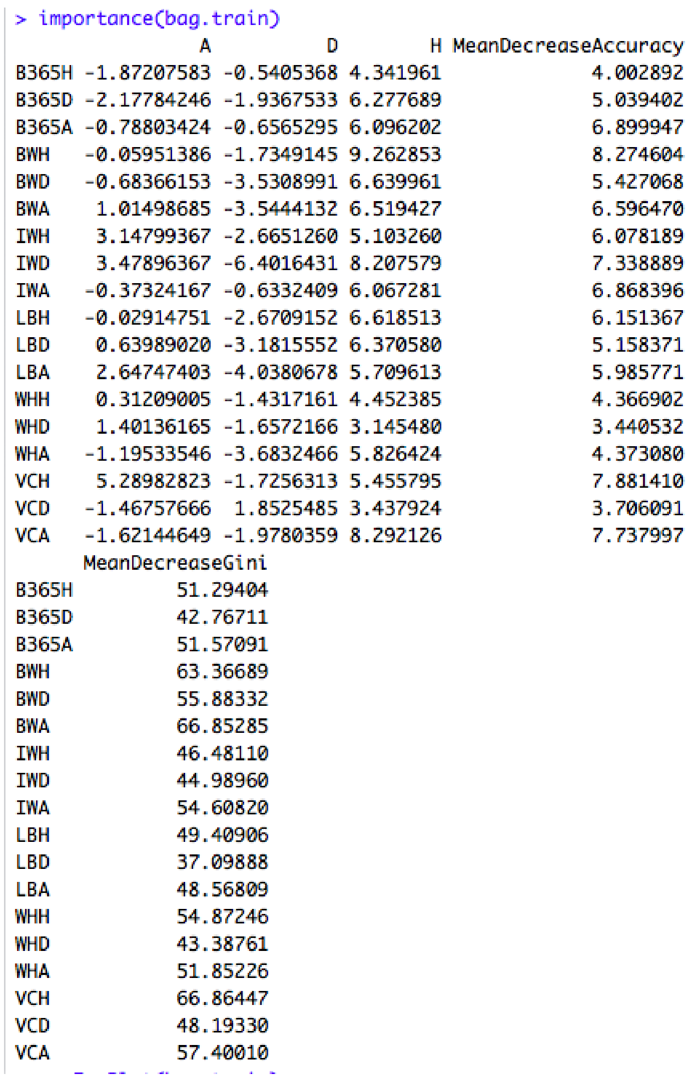


Figure 14: Graph of OOB Estimate Error vs Tree Size

We can see that as the tree size increases the error rate decreases, but it is decreasing very slowly, you can see there was only a 0.93% decrease in the OOB error rate when going from 600 to 300 trees. If there is small data set then the trade-off between computation time and OOB error is so minimal that we can easily run the random forest with 10,000 trees, but as the tree size increases the process will slow down and take a lot of time to execute. If there are larger datasets it may be worth building the forest up slowly and checking the above plot to see when the error begins to flatten out.

Random forests have many advantages but the one major advantage is that they can provide a measure of relative importance by ranking predictors based on how much they influence the response. Importance can be obtained by using the importance () function and we can also plot the importance using the varImpPlot () function.



```
> importance(bag.train)
                A          D        H MeanDecreaseAccuracy
B365H -1.87207583 -0.5405368 4.341961             4.002892
B365D -2.17784246 -1.9367533 6.277689             5.039402
B365A -0.78803424 -0.6565295 6.096202             6.899947
BWH   -0.05951386 -1.7349145 9.262853             8.274604
BWD   -0.68366153 -3.5308991 6.639961             5.427068
BWA    1.01498685 -3.5444132 6.519427             6.596470
IWH    3.14799367 -2.6651260 5.103260             6.078189
IWD    3.47896367 -6.4016431 8.207579             7.338889
IWA   -0.37324167 -0.6332409 6.067281             6.868396
LBH   -0.02914751 -2.6709152 6.618513             6.151367
LBD    0.63989020 -3.1815552 6.370580             5.158371
LBA    2.64747403 -4.0380678 5.709613             5.985771
WHH    0.31209005 -1.4317161 4.452385             4.366902
WHD    1.40136165 -1.6572166 3.145480             3.440532
WHA   -1.19533546 -3.6832466 5.826424             4.373080
VCH    5.28982823 -1.7256313 5.455795             7.881410
VCD   -1.46757666  1.8525485 3.437924             3.706091
VCA   -1.62144649 -1.9780359 8.292126             7.737997
      MeanDecreaseGini
B365H         51.29404
B365D         42.76711
B365A         51.57091
BWH           63.36689
BWD           55.88332
BWA           66.85285
IWH           46.48110
IWD           44.98960
IWA           54.60820
LBH           49.40906
LBD           37.09888
LBA           48.56809
WHH           54.87246
WHD           43.38761
WHA           51.85226
VCH           66.86447
VCD           48.19330
VCA           57.40010
```

Figure 15: Variable Importance Table

As we can see on the table, it reports the mean decrease in the Gini Index (Fig. 15) and mean decrease in the Accuracy Index (Fig. 15). Mean decrease gini measures the impurity for the categorical data. In that case it is used to obtain the error rate by passing the OOB sample from all the predictor for each tree. Unpermuted OOB error rate is then subtracted from the error rate on the permuted OOB data and it takes the average across all trees. Now after that we get to know which variable has a stong relationship with the response model by observing the values, highe value means strong relationship and if the value is very low, then it means it has the wekest relationship with the model [1]. We could use gini index to describe the overall explanatory power of the variables. It shows us that, the variables in our data all equally as important or does one have much greater explanatory value. The mean decrease accuracy is caused by a variable which is determined during the out of bag error calculation phase. [2].

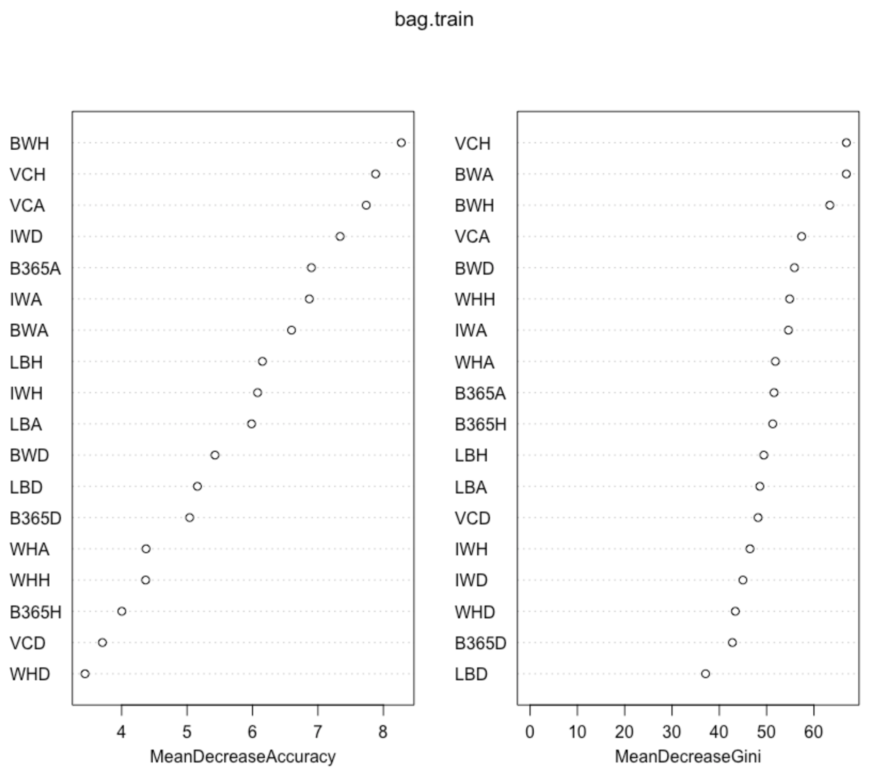
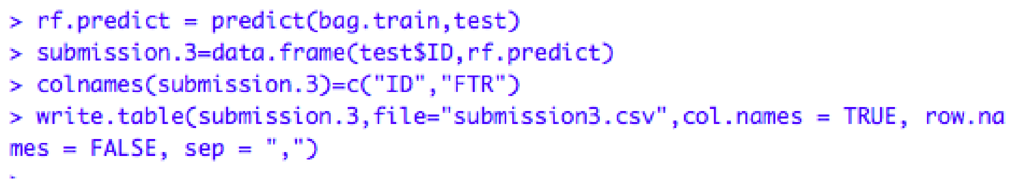


Figure 16: Variable Importance Plot

The variable importance plot shows above (Fig. 16) is an output of the random forest algorithm. Every variable is shown in the above plot for our data and the importance of each variable. On the Y-axis, it shows each variable and on the X-axis, it shows how important that variable is (Importance).The variables are ordered from top-to-bottom according to theri importance, which is given by the position of the dot on the X-axis. The main advantage of this plot is that it shows us which variable we could use for the data analysis and this tool also helps us in reducing the number of variables for other data analysis technique [25].

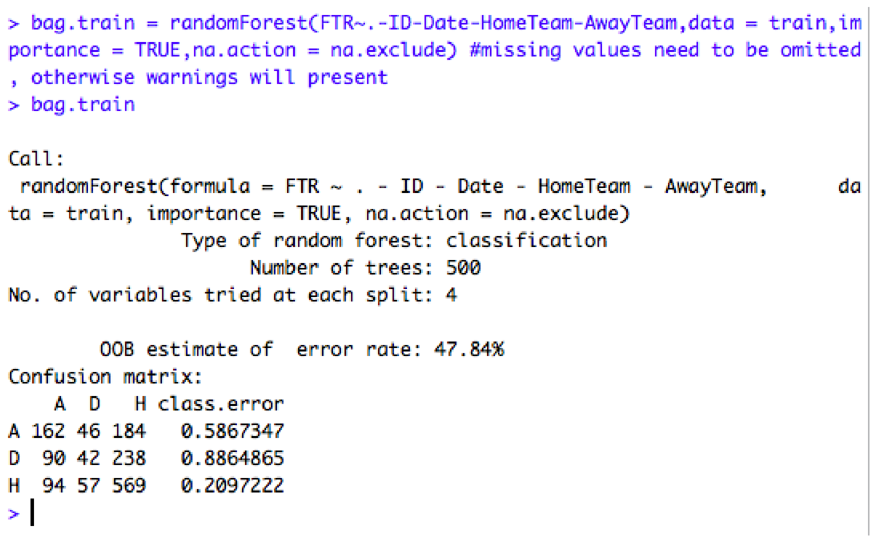
The results have been saved and after learning, our machine can make predictions from these results. We will use generic predict () function to our testing dataset. R will then take the new data frame, process the variables according to the random forest formula and give out a result for each row of the new data frame.



```
> rf.predict = predict(bag.train,test)
> submission.3=data.frame(test$ID,rf.predict)
> colnames(submission.3)=c("ID","FTR")
> write.table(submission.3,file="submission3.csv",col.names = TRUE, row.na
mes = FALSE, sep = ",")
```

D.2.3 Observations

We noticed one thing while writing the formula for the random forest where we have fixed the missing values using (na.action=na.omit). We tried to change the omit function and replaced it with (na.action = na.exclude) and we found out that the OOB estimate error rate went down to 47.84%.



```
> bag.train = randomForest(FTR~.-ID-Date-HomeTeam-AwayTeam,data = train,im
portance = TRUE,na.action = na.exclude) #missing values need to be omitted
, otherwise warnings will present
> bag.train

Call:
 randomForest(formula = FTR ~ . - ID - Date - HomeTeam - AwayTeam,      da
ta = train, importance = TRUE, na.action = na.exclude)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 47.84%
Confusion matrix:
    A  D   H class.error
A 162 46 184   0.5867347
D  90 42 238   0.8864865
H  94 57 569   0.2097222
> 
```

Figure 17: Revised OOB estimate error rate and confusion matrix

Previously it was 48.99% and when we replaced it with the function (na.action=na.exclude), the error dropped to 47.84%.

Historically, "na.omit" was the only and recommended method for omitting rows with missing values through the "na.action" argument, but then "na.exclude" came with the extra functionality. With "na.omit" the missing values will be removed from the fitting process and in this case (na.exclude) also the missing values will be removed but included as NAs in the residuals and fitted values [26].

So, with "na.exclude" we can increase the size of the tree and can check the error rate, which would be low as compared to the previously used (na.omit) and by using "na.exclude" the result would be better which is shown below.

D.2.4 Packages

- library(caret)

Tree Size	Error Rate
100	50.54%
200	50%
300	49.46%
400	49.26%
500	48.99%
600	48.53%
1000	48.25%

Table 3: Number of Trees vs Error Rate Table Using "na.omit"

Tree Size	Error Rate
100	50.13%
200	49.12%
300	49.9%
400	48.38%
500	47.84%
600	48.2%
1000	47.96%

Table 4: Number of Trees vs Error Rate Table Using "na.exclude"

- library(ggplot2)
- library(randomForest)

These are the list of packages which are used for "Random Forest".

- caret - Caret is used for for training and plotting classification and regression models.
- ggplot2 - This package is used for plotting and ggplot2 package is based on the grammar of graphics. The main advantage of this packages is, it tries to take the good parts of base and lattice graphics and none of the bad parts.

D.3 Neural Networks

- Neural Networks is machine learning technique.
- Neural networks are non-linear statistical data modeling tools.
- Neural networks are the artificial systems which are sophisticated, perhaps intelligent.
- Neural networks perform the computations same as the human brain routinely performs, and thereby possibly enhance understanding of the human brain.
- Neural networks are very good in recognizing patterns and good at fitting non-linear functions.

Neural networks need training sessions in which it adapts itself based on examples. After completion of the training sessions, the neural networks is able to relate the problem data to the solutions and then it offers a viable solution to a brand new problem. Neural networks can also generalize and handle incomplete data. Their ability to learn by example makes them very flexible and powerful. There is no need to devise an algorithm in order to perform a specific task [36].

Neural networks are tools that have application in many areas and that is why neural networks are used in the aerospace, automotive, banking, defense, electronics, entertainment, financial, insurance, manufacturing, oil and gas, robotics, telecommunications, and transportation industries [14].

D.3.1 Properties of Neural Networks

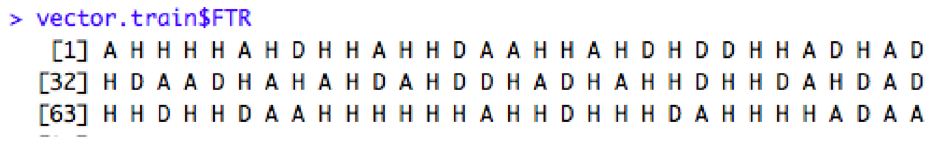
Below are some of the properties of neural networks.

- Neural networks are asynchronous, parallel and computation is collective.
- Memory is distributed and internalized.
- Neural networks are fault tolerant and redundant.
- Neural networks can be divided into dynamic as well as static neural networks, dynamic means that the network is permanently adapting the functionality or we can say that it learns during the operation. The static neural networks adapt their properties in the so called learning or training process.

D.3.2 Methodology

To start with neural networks, we need two packages that are, nnet and neuralnet. Before fitting a neural network, we need to do some preparations which is must before applying neural networks. So first we have to do data pre-processing in which we will find and remove the rows with missing data and for that, we can use a function called "complete.cases ()". This function returns a logical vector that specifies TRUE for complete case, it means that after observing the data, there are no missing values (NAs) in that row and FALSE for incomplete cases, observations that contain at least one missing value (NA). The second step which is very important is to normalize the dataset. To do normalization or not, it depends on the dataset but avoiding normalization may lead to useless results or to a very difficult training process.

If we look at the train$FTR, where FTR is the full-time result which is our target, it is in the form of vector values (A,D, H) shown below.



```
> vector.train$FTR
  [1] A H H H H A H D H H A H H D A A H H A H D H D D H H A D H A D
 [32] H D A A D H A H A H D A H D D H A D H A H H D H H D A H D A D
 [63] H H D H H D A A H H H H H H A H H D H H H D A H H H H A D A A
```

We have to change this because the "nnet" package which we are using accepts only the target variable of the classification (i.e FTR) in a particular format. To convert this, we need a class.ind () function which generates a class indicator function from a given factor. How to use class.ind () function is shown below.

train$FTR = class.ind(train$FTR)

Now we are ready to fit a neural network and for that, we need to first write the formula and then pass it as an argument in the fitting function.

x = as.formula(paste("A + D + H ",paste(colname[!colname %in% c("ID","Date","HomeTeam","AwayTeam","FTR","A","D","H")],collapse = " + ")))

This is our formula and this is what goes to the neural network function.

x is the name of the variable in which this formula will get stored and then we will put "x" in the neural networks function. Second, **as.formula ()** is forcing the variable type to a "formula" type, it has the general form of Response ∼ "Var1"+"Var2", which means use the variable 1 and variable 2 to predict the response value. **paste** is a function which is used to concatenates the pieces of string. **paste(n[n %in% "pred_con"], collapse = " + ")**, where **n** is the names of the columns in the training dataset which will give us each column name with a "+" sign between them. Finally, we make it of type formula instead of string and now we have to wrap it all together with as.formula () function and put it in the neuralnet () function below.

nnet = neuralnet(x, data=scaled.train, hidden=c(10), linear.output = FALSE, threshold=0.01,stepmax=1e6)

The neuralnet () function is used for training a neural network which tells us how many hidden layers and neurons we want according to our needed complexity. Neural networks are comprised of an input, hidden and output nodes. As such, there is not a particular method of determining the number of nodes in the hidden layer, however, there are some rules of thumb.The default value is one hidden layer with one hidden neuron.

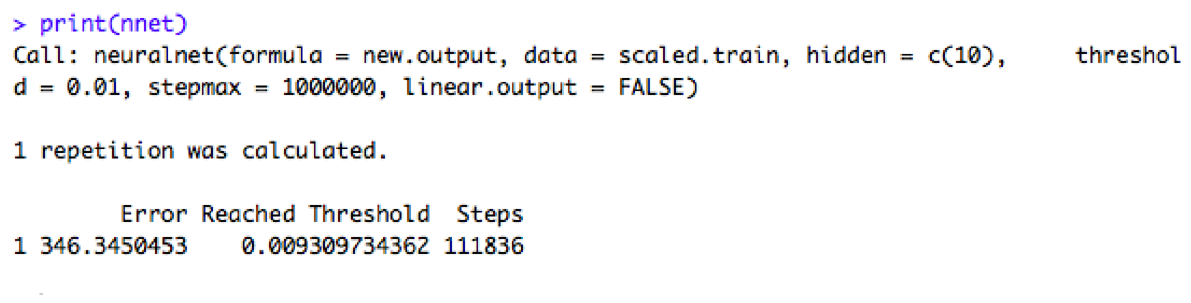
Use of Hidden layer

- Each hidden layer is used to produce an output by applying a function to the previous layer.
- The hidden layer transforms the inputs into something valuable which can be used by the output layer.
- The output layer transforms the hidden layer activations into any scale we want our output to be on.

In our case, we have used "ten" hidden layers, because though we have eighteen input variables, we have increased the number of internal nodes here as an experiment to see if it yields a better result or not.

The argument "linear.output" is used to specify whether we want to do regression or classification. linear.output=TRUE is used for regression and linear.output=FALSE for classification. The output which we will get wouldn't be linear, so we have to use a threshold value. The "threshold" parameter is used to determine when the neuralnet algorithm should stop and stepmax is used to determine the level of computational resources available to the algorithm.

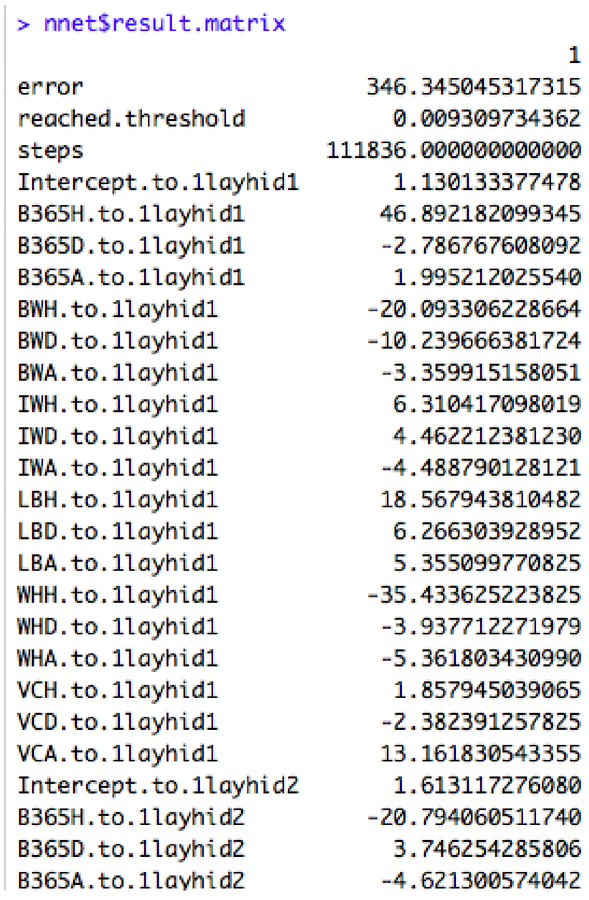
A neural network with, for example, ten hidden neurons is trained by the following statement :



```
> print(nnet)
Call: neuralnet(formula = new.output, data = scaled.train, hidden = c(10),     threshol
d = 0.01, stepmax = 1000000, linear.output = FALSE)

1 repetition was calculated.

        Error Reached Threshold  Steps
1 346.3450453    0.009309734362 111836
```

As we can see there is an error, reached threshold, and steps. This means that the model starting iterating and was trying to find the best solution till it reached a point from where the error of the model was not reducing by more than the given threshold value, which is 0.01.

We have saved the basic information about the training process and the trained neural network in "nnet" with ten hidden nodes. This information is very useful and can be used to reproduce the results as for instance the starting weights. We can also produce the summary of the main results by using : **nnet$result.matrix**.



```
> nnet$result.matrix
                                          1
error                      346.345045317315
reached.threshold            0.009309734362
steps                   111836.000000000000
Intercept.to.1layhid1        1.130133377478
B365H.to.1layhid1           46.892182099345
B365D.to.1layhid1           -2.786767608092
B365A.to.1layhid1            1.995212025540
BWH.to.1layhid1            -20.093306228664
BWD.to.1layhid1            -10.239666381724
BWA.to.1layhid1             -3.359915158051
IWH.to.1layhid1              6.310417098019
IWD.to.1layhid1              4.462212381230
IWA.to.1layhid1             -4.488790128121
LBH.to.1layhid1             18.567943810482
LBD.to.1layhid1              6.266303928952
LBA.to.1layhid1              5.355099770825
WHH.to.1layhid1            -35.433625223825
WHD.to.1layhid1             -3.937712271979
WHA.to.1layhid1             -5.361803430990
VCH.to.1layhid1              1.857945039065
VCD.to.1layhid1             -2.382391257825
VCA.to.1layhid1             13.161830543355
Intercept.to.1layhid2        1.613117276080
B365H.to.1layhid2          -20.794060511740
B365D.to.1layhid2            3.746254285806
B365A.to.1layhid2           -4.621300574042
```

The summary of the final result shows that the training process needed "111836.0" steps until all absolute partial derivatives of the error function were smaller than 0.01 (the default threshold) and the intercepts of the first hidden layer are 1.130 (Intercept.to.1layhid1) and 1.613 (intercept.to.1layhid2).

The final results of the training process can be visualized simply plotted by:

plot(nnet)

The resulting plot is shown in Figure 18.

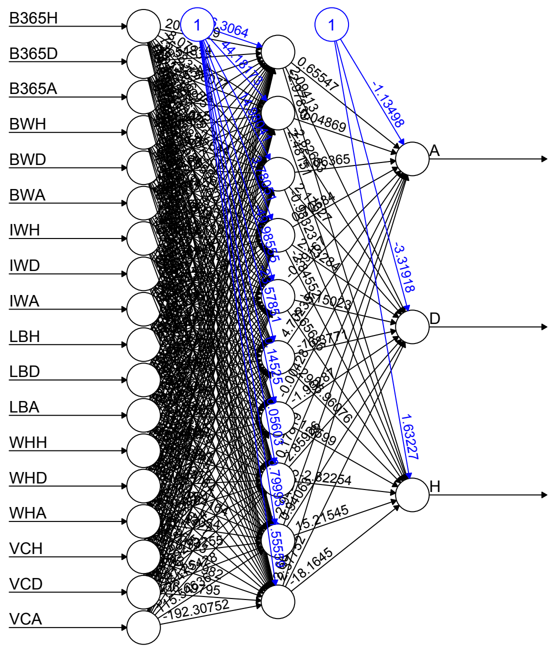


Figure 18: Plot of the trained neural network including trained synaptic weights and basic information about the training process.

Fig. 18 is the graphical representation of the trained neural network, i.e. the network topology, which shows that the dataset contains eighteen input variables and three output variables. The black lines in the plot show the connections between each layer and the weights on each connection while the blue lines show the bias term added in each step. The bias is nothing but the intercept of a linear model. Now our model is ready to be used.

Once the neural network is trained, we are ready to test it on our testing data. The compute function is applied for computing the outputs which will give us the final result.

D.3.3 Observations

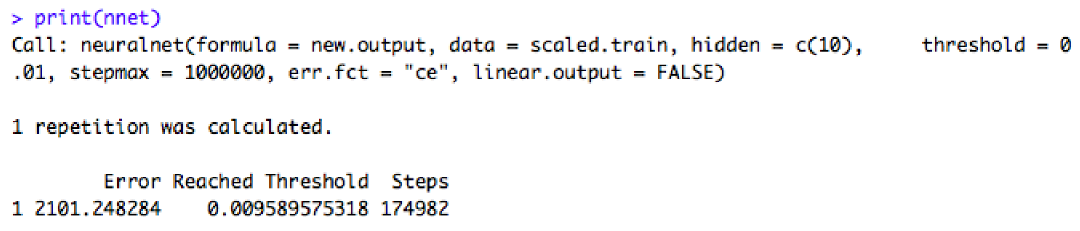
We tried to edit the neural network function by using the "algorithm" to see if it improves the model or not. "Algorithm" is a string that contains the algorithm type to calculate the neural network. There are different type of algorithms :’backprop’, ’rprop+’, ’rprop-’, ’sag’, or ’slr’. ’backprop’ refers to backpropagation, ’rprop+’ and ’rprop-’ refer to the resilient backpropagation with and without weight backtracking, while 'sag' and 'slr' induce the usage of the modified globally convergent algorithm (grprop) [10]. We tried some of these algorithms to check the result if by adding these algorithms the model would improve or not with (10 Hidden layers are used). The results are shown below.

Algorithms	Error Rate	Steps
rprop+	341.38	172221
rprop-	343.76	202319

Table 5: Changing Algorithms

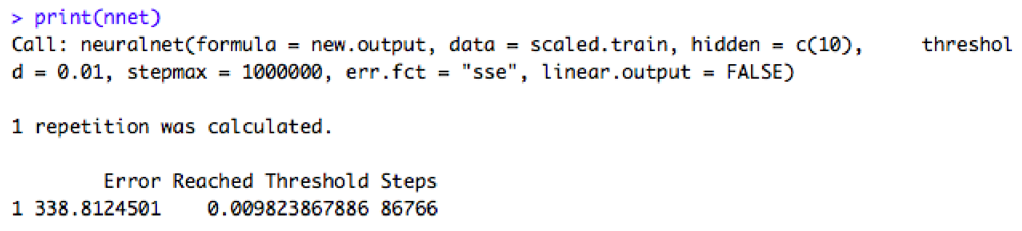
We also tried to change the neural network function by changing the number of hidden layers and we edit the neural network function by adding (err.fct="ce") which is a differentiable function that is used for the calculation of the error, where "ce" is the cross-entropy and we can also use "sse" which stand for the sum of squared errors [10]. After adding err.fct="ce" in the neural network function with the same number of hidden layers i.e equal to 10, the error rate increased. But when we err.fct="sse" instead of "ce" the error rate and the number of steps went down, which is shown below.

Neural network function with err.fct="ce and err.fct="sse"



```
> print(nnet)
Call: neuralnet(formula = new.output, data = scaled.train, hidden = c(10),     threshold = 0
.01, stepmax = 1000000, err.fct = "ce", linear.output = FALSE)

1 repetition was calculated.

        Error Reached Threshold  Steps
1 2101.248284    0.009589575318 174982
```

Figure 19: Neural network function with err.fct="ce"

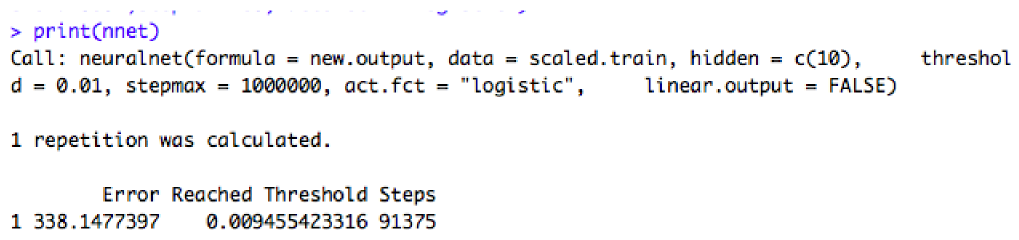


```
> print(nnet)
Call: neuralnet(formula = new.output, data = scaled.train, hidden = c(10),     threshol
d = 0.01, stepmax = 1000000, err.fct = "sse", linear.output = FALSE)

1 repetition was calculated.

        Error Reached Threshold Steps
1 338.8124501    0.009823867886 86766
```

Figure 20: Neural network function with err.fct="sse"

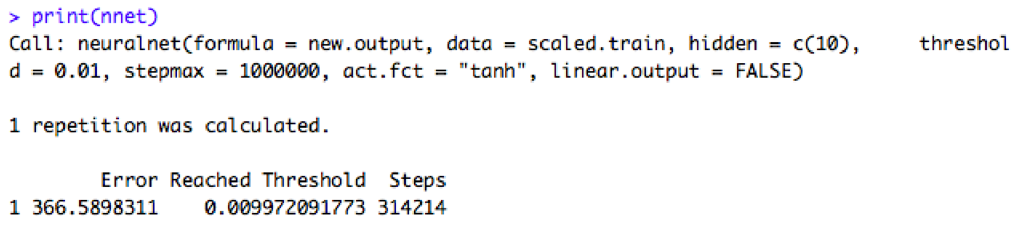
As we can see, there is a lot of difference in the error rate as well as in a number of steps. Using err.fct="sse", the error rate went down to 338.81 and the number of steps to 86766. Without using err.fct="sse" with 10 hidden layers, the error rate was 346.345 and the number of steps was 111836.0.

Along with that we also tried one more function which is used for smoothing the result of the cross product of the covariate or neurons and the weights, that function is "act.fct". There are two strings which we can be used with this function : 'logistic' for the logistic function and 'tanh' is for tangent hyperbolic [10]. The results are shown below.



```
> print(nnet)
Call: neuralnet(formula = new.output, data = scaled.train, hidden = c(10),     threshol
d = 0.01, stepmax = 1000000, act.fct = "logistic",     linear.output = FALSE)

1 repetition was calculated.

        Error Reached Threshold Steps
1 338.1477397     0.009455423316 91375
```

Figure 21: Neural network function with act.fct="logistic"



```
> print(nnet)
Call: neuralnet(formula = new.output, data = scaled.train, hidden = c(10),     threshol
d = 0.01, stepmax = 1000000, act.fct = "tanh", linear.output = FALSE)

1 repetition was calculated.

        Error Reached Threshold  Steps
1 366.5898311     0.009972091773 314214
```

Figure 22: Neural network function with act.fct="tanh"

After using these two functions, there is the difference in error. With act.fct="logistic", the error is 338.147 and when we used act.fct="tanh", the error raised to 366.589. So, we can change these functions for different datasets, record the results and chose the best among them.

Changing Hidden Layers

We changed the number of hidden layers and saved the information in a table given below. We noticed after reading the table that as the number of hidden layers increasing the error rate is decreasing and decreasing in error rate increasing the number of steps. But if we increase the number of hidden layers the calculation time also increases. The results are shown below.

Hidden Layers	Error Rate	Steps
5	389.83	86104
8	369.69	31242
10	346.34	111836
12	317.98	264012
15	293.33	428579

Table 6: Changing Hidden Layers

D.3.4 Packages

- library(nnet)
- library(neuralnet)

The packages which are required for neural networks are "nnet" and "neuralnet". Package "nnet" is used to generate a class indicator function from a given factor. The "neuralnet" package is used to visualize the generated model and shows the found weights.

D.4 Xgboost

Xgboost is short for eXtreme Gradient Boosting package. Xgboost is a library which is designed and optimized for boosting tree algorithms. Extreme Gradient Boosting (xgboost) is same as gradient boosting framework but Xgboost is more efficient, flexible and portable. It is the package which is used to solve data science problems which include both linear model solver and tree learning algorithms. [5].

D.4.1 Features

- Xgboost has several features but the most important feature of this package is, it can automatically do parallel computation on a single machine which could be more than 10 times faster than existing gradient boosting packages. So that's why xgboost is able to utilize the more computational power and get a more accurate prediction.
- Xgboost supports various objective functions, including regression, classification, and ranking.
- Xgboost can take several types of input data, for example, dense matrix, sparse matrix, data file (local data files) and xgb.DMatrix, which is nothing but xgboost's own class and it speeds up Xgboost as well.
- Xgboost can also do cross validation and can be used to find important variables.
- Xgboost has better performance on large number of different datasets.

D.4.2 Parameters

There are various parameters used in Xgboost model : general parameters, booster parameters, and task parameters [19].

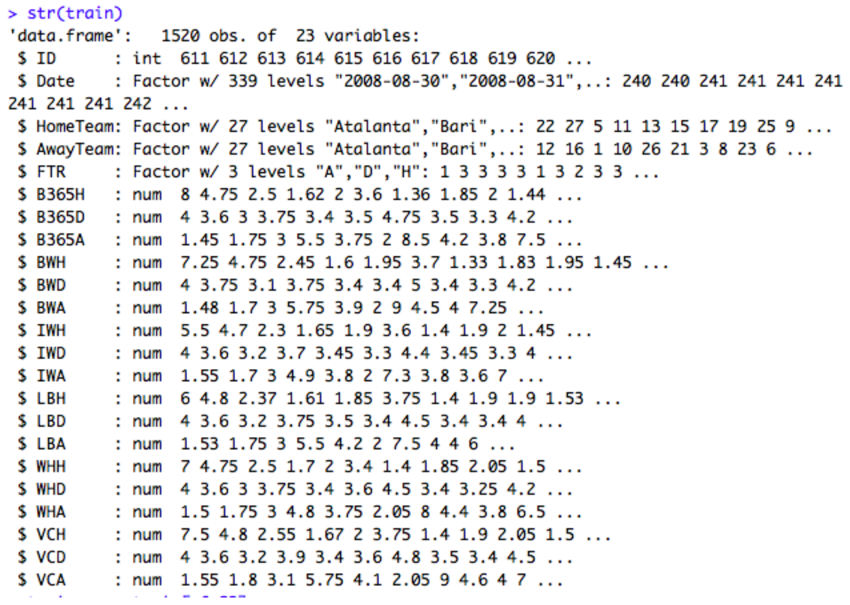
- General parameters are used to guide the overall functioning, it refers to which booster is used for boosting and the commonly used are the tree or linear model.
- Booster parameters depends on the booster which has been chosen.
- Task parameters are used to guide the optimization performed and task parameters that decide on the learning scenario, for example, different tasks may use different parameters.

These parameters are used to improve the model and for that parameter tuning is must.

D.4.3 Methodology

The first thing we have to do before building the model and tuning is data pre-processing because Xgboost only works with numeric vectors and doesn't accept data frames. So, first we will inspect our data by using:

str(train)



```
> str(train)
'data.frame':   1520 obs. of  23 variables:
 $ ID      : int  611 612 613 614 615 616 617 618 619 620 ...
 $ Date    : Factor w/ 339 levels "2008-08-30","2008-08-31",..: 240 240 241 241 241 241
241 241 241 242 ...
 $ HomeTeam: Factor w/ 27 levels "Atalanta","Bari",..: 22 27 5 11 13 15 17 19 25 9 ...
 $ AwayTeam: Factor w/ 27 levels "Atalanta","Bari",..: 12 16 1 10 26 21 3 8 23 6 ...
 $ FTR     : Factor w/ 3 levels "A","D","H": 1 3 3 3 3 1 3 2 3 3 ...
 $ B365H   : num  8 4.75 2.5 1.62 2 3.6 1.36 1.85 2 1.44 ...
 $ B365D   : num  4 3.6 3 3.75 3.4 3.5 4.75 3.5 3.3 4.2 ...
 $ B365A   : num  1.45 1.75 3 5.5 3.75 2 8.5 4.2 3.8 7.5 ...
 $ BWH     : num  7.25 4.75 2.45 1.6 1.95 3.7 1.33 1.83 1.95 1.45 ...
 $ BWD     : num  4 3.75 3.1 3.75 3.4 3.4 5 3.4 3.3 4.2 ...
 $ BWA     : num  1.48 1.7 3 5.75 3.9 2 9 4.5 4 7.25 ...
 $ IWH     : num  5.5 4.7 2.3 1.65 1.9 3.6 1.4 1.9 2 1.45 ...
 $ IWD     : num  4 3.6 3.2 3.7 3.45 3.3 4.4 3.45 3.3 4 ...
 $ IWA     : num  1.55 1.7 3 4.9 3.8 2 7.3 3.8 3.6 7 ...
 $ LBH     : num  6 4.8 2.37 1.61 1.85 3.75 1.4 1.9 1.9 1.53 ...
 $ LBD     : num  4 3.6 3.2 3.75 3.5 3.4 4.5 3.4 3.4 4 ...
 $ LBA     : num  1.53 1.75 3 5.5 4.2 2 7.5 4 4 6 ...
 $ WHH     : num  7 4.75 2.5 1.7 2 3.4 1.4 1.85 2.05 1.5 ...
 $ WHD     : num  4 3.6 3 3.75 3.4 3.6 4.5 3.4 3.25 4.2 ...
 $ WHA     : num  1.5 1.75 3 4.8 3.75 2.05 8 4.4 3.8 6.5 ...
 $ VCH     : num  7.5 4.8 2.55 1.67 2 3.75 1.4 1.9 2.05 1.5 ...
 $ VCD     : num  4 3.6 3.2 3.9 3.4 3.6 4.8 3.5 3.4 4.5 ...
 $ VCA     : num  1.55 1.8 3.1 5.75 4.1 2.05 9 4.6 4 7 ...
```

Xgboost allows dense and sparse matrix as an input and as we can see our data, it's a data frame. Therefore, the first thing we need to do is to convert that data frame into a matrix form and all other forms of data into numeric vectors. After converting the data we will build the model and use the parameters.

FTR = as.matrix(as.numeric(train$FTR)-1) train.xg = as.matrix(train.xg) mode(train.xg) = 'numeric'

In the above code first we converted our train dataset and full-time result column (FTR) to numeric mode (because FTR contains the result as A,D, and H) and then we have bound them together as the columns of a matrix and in the case if FTR, we removed the "ID" column because each column in a dataset represents a feature measured by an integer and we know that the first column (ID) doesn't contain any useful information. To let the algorithm focus on real stuff, we will delete it and make that "NULL". Factors and ordered factors are replaced by their internal codes. We did the same thing to our test data as well.

After preparing of training and test dataset, it's time to run cross-validation to choose parameters. We have to set the parameters before running Xgboost, these parameters are already explained above. Next step is to fit the model with the arbitrary parameters specified below.

(a) **Learning Task Parameters**

(i) **objective = "multi:softprob" or "multi:softmax"**
This parameter is used for multiclass classification. We can also use "softmax". The difference between them is It's output which separates them. In softmax, we get the class with the maximum probability as output. In softprob, same as "softmax" but output a vector of ndata * nclass which can be further reshaped to ndata, nclass matrix. So, the result contains the predicted probability of each data point belonging to each class. We can also say that in "softprob" we get a matrix with a probability value of each class we are trying to predict. If we are using "softmax" parameter we also need to set num_class(number of classes) [19].

(ii) **eval_metric**
It is an evaluation metrics used for validation data. In this case, a default metric is assigned according to the objective. The objective could be rmse for regression, error for classification, mean average precision for ranking [19].

(b) **Parameters for Tree Booster**

(i) **max_depth**

It is the maximum depth of a tree and its default value is "6". Increasing the value of maximum depth makes the model more complex. The range is $[1, \infty]$ [19].

(ii) **eta**

The default value of eta is "0.3". It is a step size shrinkage used in an update to prevents overfitting. We can directly get the weights of new features after each boosting step and eta actually shrinks the feature weights to make the boosting process more conservative.The range is [0,1] [19].

(iii) **gamma**

The default value of gamma is 0. Gamma is a minimum loss reduction required to make a further partition on a leaf node of the tree. The larger the value of gamma, the more conservative the algorithm will be. The range is $[0, \infty]$ [19].

(iv) **subsample**

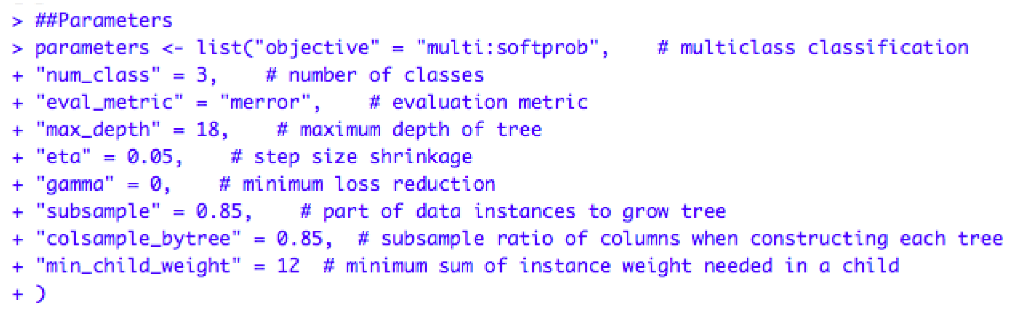
The default value of subsample is "1". The subsample is the ratio of the training instance. If we set the value of subsample to 0.5, it means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting. The range is [0,1] [19].

(v) **colsample_bytree**

The default value of colsample_bytree is "1". Colsample_bytree is a subsample ratio of columns when constructing each tree. The range is [0,1] [19].

(vi) **min_child_weight**

It is a minimum sum of instance weight needed in a child. If the partitioning results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning but in linear regression, this simply corresponds to a minimum number of instances needed to be in each node. Larger the value of min_child_weight, the more conservative the algorithm will be. The range is $[0, \infty]$ [19].



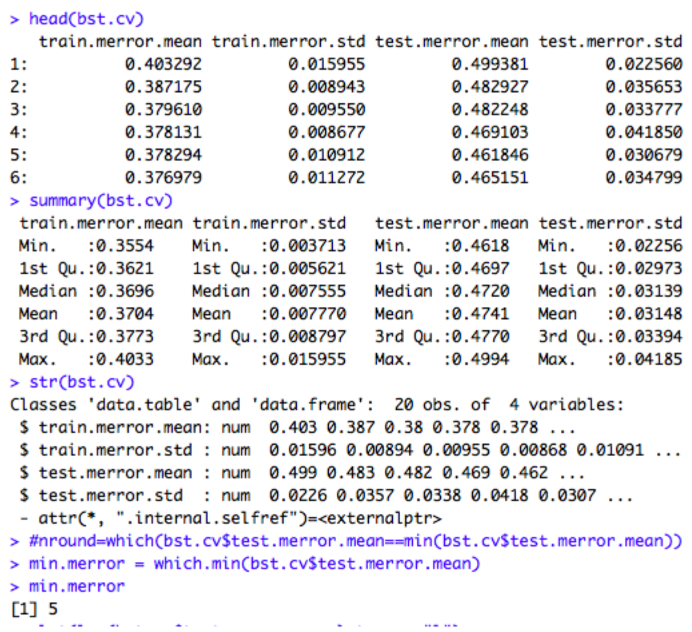
```
> ##Parameters
> parameters <- list("objective" = "multi:softprob",     # multiclass classification
+ "num_class" = 3,     # number of classes
+ "eval_metric" = "merror",     # evaluation metric
+ "max_depth" = 18,     # maximum depth of tree
+ "eta" = 0.05,     # step size shrinkage
+ "gamma" = 0,     # minimum loss reduction
+ "subsample" = 0.85,     # part of data instances to grow tree
+ "colsample_bytree" = 0.85,  # subsample ratio of columns when constructing each tree
+ "min_child_weight" = 12  # minimum sum of instance weight needed in a child
+ )
```

How we have taken the parameters and their values are shown above. The main idea behind Xgboost is, it divides the training data into parts and then Xgboost will retain the first part and use it as the test data. After this, it will reintegrate the first part to the training dataset and retain the second part and it goes on.

Now after choosing the parameters the next step is cross-validation. In cross-validation we have to set the seed value first and after that we have to choose the number of decision trees in the final model (nrounds) and in how many parts we want to divide the train data into for the cross-validation (nfold). We can edit the values for nfold and nrounds to get better results. Now the next step is to run the xgb.cv () function which is used for cross-validation with the command shown below.

bst.cv = xgb.cv(param=param, data=train.xg, label=FTR, nfold=cv.nfold, nrounds=cv.nround, missing = NaN)

After running this, xgb.cv returns a data.table object containing the cross validation results. This is helpful for choosing the correct number of iterations. It will give us the iteration number which has minimum "merror" value.



```
> head(bst.cv)
   train.merror.mean train.merror.std test.merror.mean test.merror.std
1:          0.403292         0.015955         0.499381        0.022560
2:          0.387175         0.008943         0.482927        0.035653
3:          0.379610         0.009550         0.482248        0.033777
4:          0.378131         0.008677         0.469103        0.041850
5:          0.378294         0.010912         0.461846        0.030679
6:          0.376979         0.011272         0.465151        0.034799
> summary(bst.cv)
 train.merror.mean train.merror.std   test.merror.mean test.merror.std
 Min.   :0.3554    Min.   :0.003713   Min.   :0.4618   Min.   :0.02256
 1st Qu.:0.3621    1st Qu.:0.005621   1st Qu.:0.4697   1st Qu.:0.02973
 Median :0.3696    Median :0.007555   Median :0.4720   Median :0.03139
 Mean   :0.3704    Mean   :0.007770   Mean   :0.4741   Mean   :0.03148
 3rd Qu.:0.3773    3rd Qu.:0.008797   3rd Qu.:0.4770   3rd Qu.:0.03394
 Max.   :0.4033    Max.   :0.015955   Max.   :0.4994   Max.   :0.04185
> str(bst.cv)
Classes 'data.table' and 'data.frame':  20 obs. of  4 variables:
 $ train.merror.mean: num  0.403 0.387 0.38 0.378 0.378 ...
 $ train.merror.std : num  0.01596 0.00894 0.00955 0.00868 0.01091 ...
 $ test.merror.mean : num  0.499 0.483 0.482 0.469 0.462 ...
 $ test.merror.std  : num  0.0226 0.0357 0.0338 0.0418 0.0307 ...
 - attr(*, ".internal.selfref")=<externalptr>
> #nround=which(bst.cv$test.merror.mean==min(bst.cv$test.merror.mean))
> min.merror = which.min(bst.cv$test.merror.mean)
> min.merror
[1] 5
```

As we can see the train.merror is decreasing. Each line shows how well the model explains our data (lower is better). It gave us the result (min.merror=5), which means that the minimum error is at 5th iteration. The main point is to keep an eye on the test.merror which is more important than tran.merror. We have plotted a graph of test.merror.mean for 20 rounds to check if the values are decreasing or not.

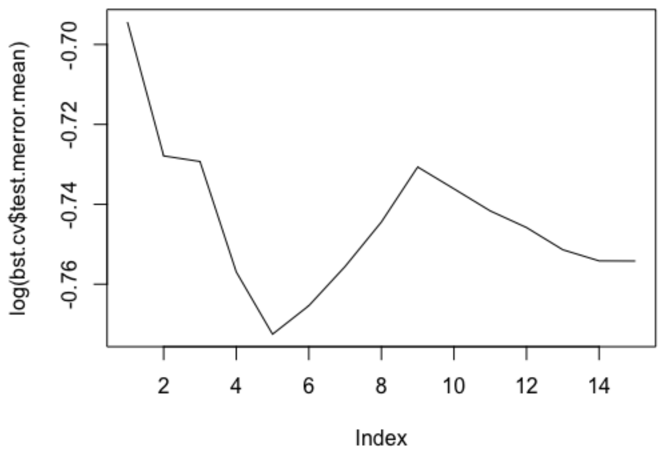
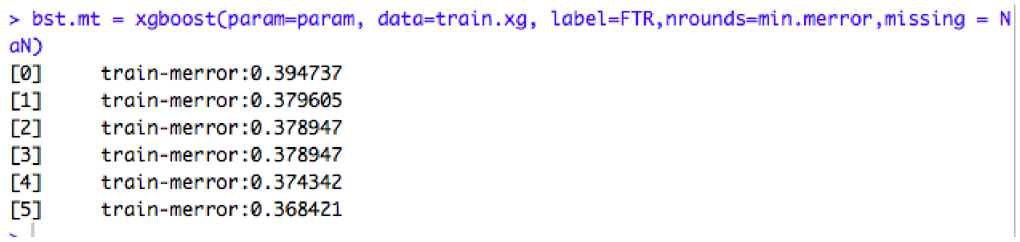


Figure 23: Graph of test.merror.mean vs nrounds

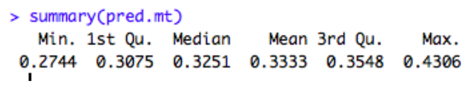
In Fig. 23 we can see that initially merror was decreasing and it decreased to a minimum till the 5th round, then it suddenly raised till 9th round and again started decreasing. The result shows that minimum decrease is at "5th" round. If we will increase the rounds it might decrease more.

After this the next step is to train our model using xgboost () function, the command is same as cross-validation, the only difference is instead of "nrounds=20", we will use the best iteration nround of cross-validation, that is "min.error", which we got earlier in cross-validation.



```
> bst.mt = xgboost(param=param, data=train.xg, label=FTR,nrounds=min.merror,missing = N
aN)
[0]	train-merror:0.394737
[1]	train-merror:0.379605
[2]	train-merror:0.378947
[3]	train-merror:0.378947
[4]	train-merror:0.374342
[5]	train-merror:0.368421
```

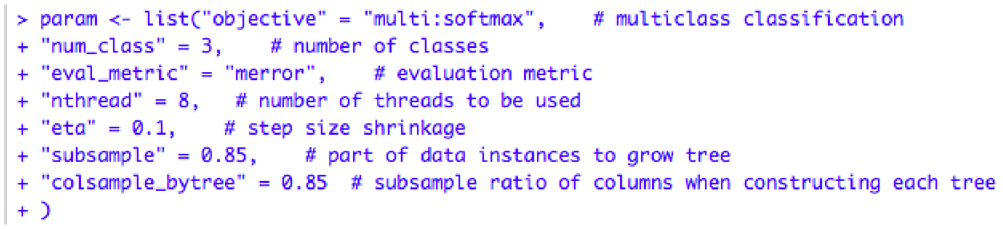
As we can see the train-merror is decreasing when we trained our model. After training the model, the next step is to predict the final result using the predict () function in which we will use our test data. The result is shown below.



```
> summary(pred.mt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2744  0.3075  0.3251  0.3333  0.3548  0.4306 
```

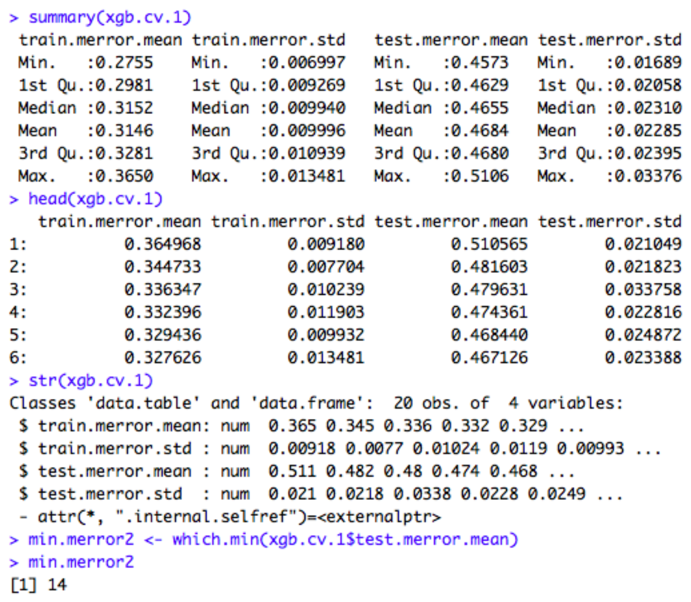
D.4.4 Observations

Now we tried to change some parameters to check for the better result. First, we changed our learning task parameter "objective" = "multi:softprob" which we used earlier to "multi:softmax".



```
> param <- list("objective" = "multi:softmax",    # multiclass classification
+ "num_class" = 3,    # number of classes
+ "eval_metric" = "merror",    # evaluation metric
+ "nthread" = 8,    # number of threads to be used
+ "eta" = 0.1,    # step size shrinkage
+ "subsample" = 0.85,    # part of data instances to grow tree
+ "colsample_bytree" = 0.85  # subsample ratio of columns when constructing each tree
+ )
```

All the parameters are same except "objective" and we did the same thing as we did earlier and we plotted a graph for test.merror with nrounds=20. The result is shown below.



```
> summary(xgb.cv.1)
 train.merror.mean train.merror.std   test.merror.mean test.merror.std
 Min.   :0.2755    Min.   :0.006997   Min.   :0.4573   Min.   :0.01689
 1st Qu.:0.2981    1st Qu.:0.009269   1st Qu.:0.4629   1st Qu.:0.02058
 Median :0.3152    Median :0.009940   Median :0.4655   Median :0.02310
 Mean   :0.3146    Mean   :0.009996   Mean   :0.4684   Mean   :0.02285
 3rd Qu.:0.3281    3rd Qu.:0.010939   3rd Qu.:0.4680   3rd Qu.:0.02395
 Max.   :0.3650    Max.   :0.013481   Max.   :0.5106   Max.   :0.03376
> head(xgb.cv.1)
   train.merror.mean train.merror.std test.merror.mean test.merror.std
1:          0.364968         0.009180         0.510565        0.021049
2:          0.344733         0.007704         0.481603        0.021823
3:          0.336347         0.010239         0.479631        0.033758
4:          0.332396         0.011903         0.474361        0.022816
5:          0.329436         0.009932         0.468440        0.024872
6:          0.327626         0.013481         0.467126        0.023388
> str(xgb.cv.1)
Classes 'data.table' and 'data.frame':  20 obs. of  4 variables:
 $ train.merror.mean: num  0.365 0.345 0.336 0.332 0.329 ...
 $ train.merror.std : num  0.00918 0.0077 0.01024 0.0119 0.00993 ...
 $ test.merror.mean : num  0.511 0.482 0.48 0.474 0.468 ...
 $ test.merror.std  : num  0.021 0.0218 0.0338 0.0228 0.0249 ...
 - attr(*, ".internal.selfref")=<externalptr>
> min.merror2 <- which.min(xgb.cv.1$test.merror.mean)
> min.merror2
[1] 14
```

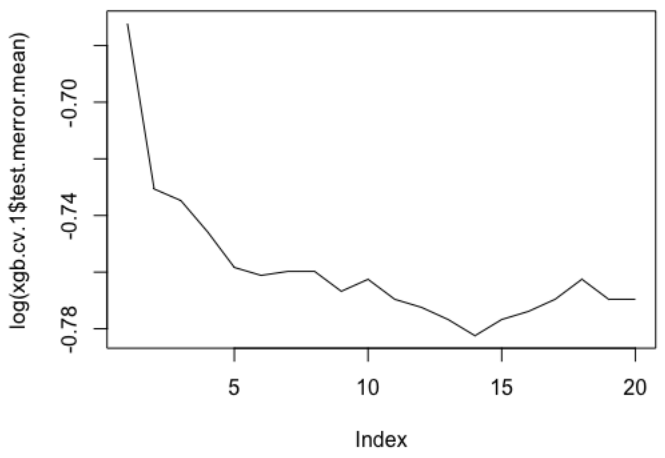
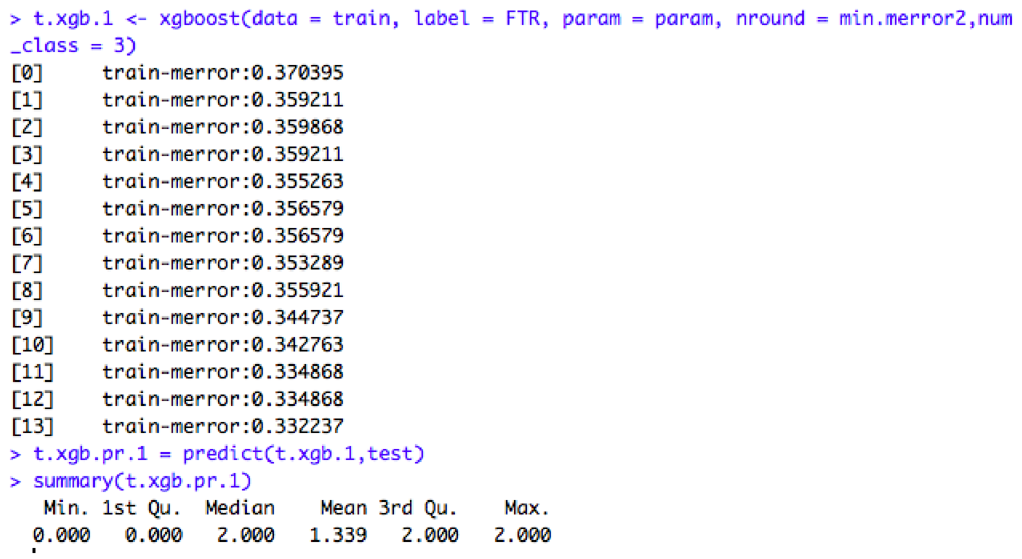


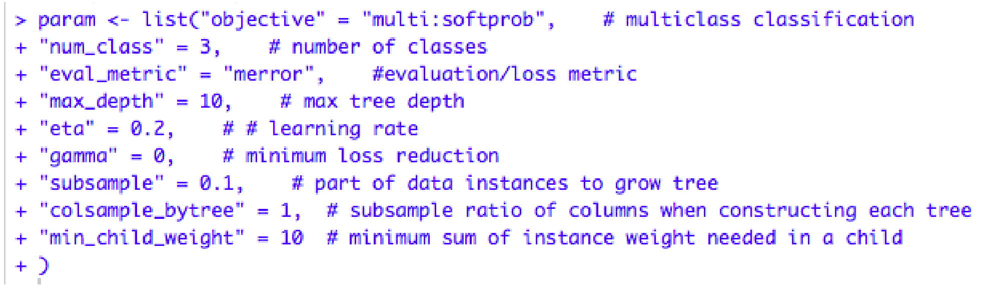
Figure 24: Graph of test.merror.mean

In the graph above, the values were decreasing initially and the minimum decrease is at 14th round and after the 14th round, it increased a bit and at 19th round and 20th round it is constant. After training the model and using a predict () function on the test data, The errors are minimum and the results are better minimum than the previous one (using multi:softprob). The final results are shown below.


```
> t.xgb.1 <- xgboost(data = train, label = FTR, param = param, nround = min.merror2,num
_class = 3)
[0]	train-merror:0.370395
[1]	train-merror:0.359211
[2]	train-merror:0.359868
[3]	train-merror:0.359211
[4]	train-merror:0.355263
[5]	train-merror:0.356579
[6]	train-merror:0.356579
[7]	train-merror:0.353289
[8]	train-merror:0.355921
[9]	train-merror:0.344737
[10]	train-merror:0.342763
[11]	train-merror:0.334868
[12]	train-merror:0.334868
[13]	train-merror:0.332237
> t.xgb.pr.1 = predict(t.xgb.1,test)
> summary(t.xgb.pr.1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.000   2.000   1.339   2.000   2.000 
```

We changed some other parameters and found some worst as well as good results also. Few of the results are shown below with their parameter values and their plots for nrounds=20.

Parameter Change One


```
> param <- list("objective" = "multi:softprob",    # multiclass classification
+ "num_class" = 3,    # number of classes
+ "eval_metric" = "merror",    #evaluation/loss metric
+ "max_depth" = 10,    # max tree depth
+ "eta" = 0.2,    # # learning rate
+ "gamma" = 0,    # minimum loss reduction
+ "subsample" = 0.1,    # part of data instances to grow tree
+ "colsample_bytree" = 1,  # subsample ratio of columns when constructing each tree
+ "min_child_weight" = 10  # minimum sum of instance weight needed in a child
+ )
```

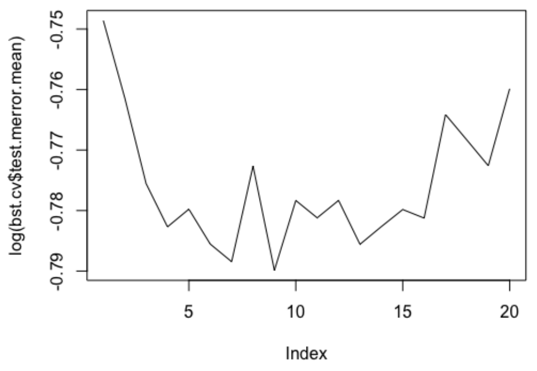


Figure 25: Graph of test.merror.mean

As we can see from the graph Fig. 25 that changing these parameters doesn't provide good results. The graph is not stable, the min value is at 9th round, then after that, the values spontaneously increased in the end.

Parameter Change Two

```
> param <- list("objective" = "multi:softprob",    # multiclass classification
+ "num_class" = 5,    # number of classes
+ "eval_metric" = "merror",    #evaluation/loss metric
+ "max_depth" = 6,    # max tree depth
+ "eta" = 0.01,    # # learning rate
+ "gamma" = 0,    # minimum loss reduction
+ "subsample" = 0.7,    # part of data instances to grow tree
+ "colsample_bytree" = 0.7,  # subsample ratio of columns when constructing each tree
+ "min_child_weight" = 30  # minimum sum of instance weight needed in a child
+ )
```

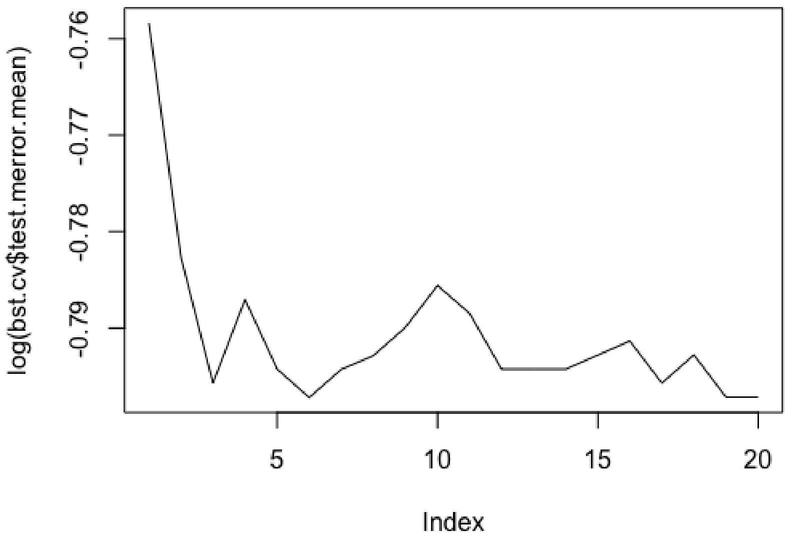


Figure 26: Graph of test.merror.mean

In Fig. 26 Changing the parameters changed the results, this graph is better and more stable than the previous one. We can tune the parameters by changing the values and then save the results for better prediction. It is time-consuming but it helps us to find the better results. We can also try these parameter values on the multi:softmax, which will give us better results than multi:softprob.

D.4.5 Packages

- library(caret)
- library(corrplot)

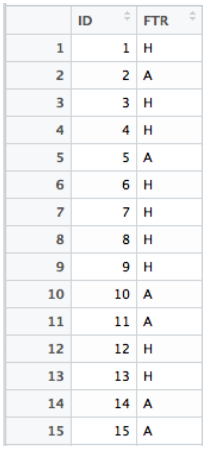
- library(Rtsne)
- library(xgboost)
- library(knitr)
- library(ggplot2)

Above are the list of packages which we have used for "xgboost".

- caret - Caret is used for for training and plotting classification and regression models.
- corrplot - This package is used to display graphs of a corelation matrix and it is good at choosing colors, labels, layouts, etc.
- Rtsne - Rtsne package is used for constructing a low dimensional embedding of high-dimensional data, distances and similarities.
- knitr - This package has more flexible design and new features which are used in caching and finer control of graphics.
- ggplot2 - This package is used for plotting. This package is also used for sophisticated multidimensional conditioning system and a consistent interface to map data to aesthetic attributes.

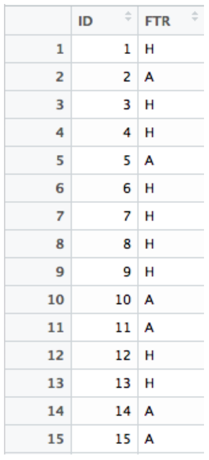
E General Results and Conclusion

In this section, we are taking the final result of first fifteen matches out of 610 which we have predicted using four different techniques. From the sample results, we can see the performance of how these four techniques predicted the final result.



	ID	FTR
1	1	H
2	2	A
3	3	H
4	4	H
5	5	A
6	6	H
7	7	H
8	8	H
9	9	H
10	10	A
11	11	A
12	12	H
13	13	H
14	14	A
15	15	A

(a) Final result using tree-base model



	ID	FTR
1	1	H
2	2	A
3	3	H
4	4	H
5	5	A
6	6	H
7	7	H
8	8	H
9	9	H
10	10	A
11	11	A
12	12	H
13	13	H
14	14	A
15	15	A

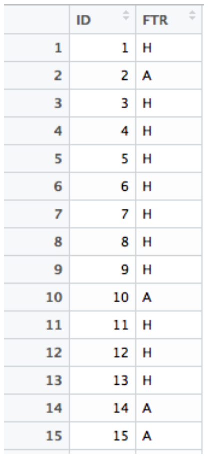
(b) Cross-Validation Technique on Tree-based model

Figure 27: Final Results of the First 15 Matches Using Tree-Based Model Technique

Fig. 27 is the sample final results of the first fifteen football matches which are shown above. Fig. 27 (a) is the final result using tree-based model and cross-validation technique on the tree-based model. As we can see the result is same after using cross-validation technique (Fig. 27 (b)) because, in our case, cross-validation techniques don't improve our model, there was no change in the misclassification error.

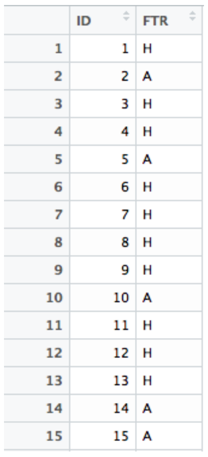
Fig. 28 (a) is the final result using the random forest, as we can see there are some minor changes in the result. ID "5th" and "11th" is changed after using random forest technique, but the rest of the result is same.

Fig. 28 (b) is the final result using neural networks, if we compare it with the tree-based model, the result is same, the only difference is ID "11th" is changed and if we compare it with random forests then ID "5th" is changed after using neural networks and the rest is same.



	ID	FTR
1	1	H
2	2	A
3	3	H
4	4	H
5	5	H
6	6	H
7	7	H
8	8	H
9	9	H
10	10	A
11	11	H
12	12	H
13	13	H
14	14	A
15	15	A

(a) Final Result Using Random Forest Technique

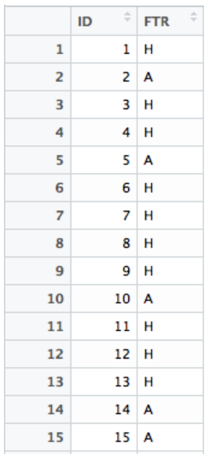


	ID	FTR
1	1	H
2	2	A
3	3	H
4	4	H
5	5	A
6	6	H
7	7	H
8	8	H
9	9	H
10	10	A
11	11	H
12	12	H
13	13	H
14	14	A
15	15	A

(b) Final Result Using Neural Networks Technique

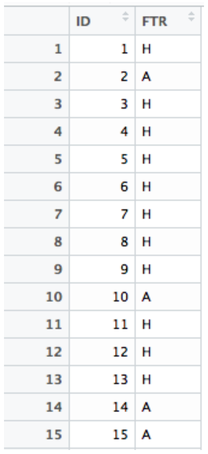
Figure 28: Final Results of the First 15 Matches Using Random Forest and Neural Networks Technique

Fig. 29 is very interesting, shown below. Fig 29 (a) is the final result using Xgboost with multi:softprob and Fig. 29(b) is the final result using multi:softmax. The interesting thing in between these two is, the result of Fig 29 (a) is exactly the same as the result of neural networks Fig. 28 (b) and the result of Fig. 29 (b) is exactly the same as the result of random forest Fig. 28 (a). The result is changing for the ID "5th". So, we can say that there are fifty-fifty chances for both "H" home team and "A" away team to win that match.



	ID	FTR
1	1	H
2	2	A
3	3	H
4	4	H
5	5	A
6	6	H
7	7	H
8	8	H
9	9	H
10	10	A
11	11	H
12	12	H
13	13	H
14	14	A
15	15	A

(a) Final Result Using XGboost Technique with multi:softprob



	ID	FTR
1	1	H
2	2	A
3	3	H
4	4	H
5	5	H
6	6	H
7	7	H
8	8	H
9	9	H
10	10	A
11	11	H
12	12	H
13	13	H
14	14	A
15	15	A

(b) Final Result Using XGboost Technique with multi:softmax

Figure 29: Final Results of the First 15 Matches Using Xgboost Technique

E.1 Conclusion

Predictive analytics is the technique to determine patterns and predict future outcomes and trends from the existing datasets. In predictive analytics, the number of advanced techniques are used, which helps us to make future forecasts. We used four different techniques to predict the final result of the football matches.

Tree-based model is simple yet relatively accurate. The misclassification error we got after using tree-based model was "0.45" (i.e, 45%), then we used the cross-validation technique to improve the model in the tree-based model, but the misclassification rate remains the same as before. We changed the seed values randomly and checked for the better results, the error went down by changing the seeds, but there was not much improvement in the model. So in our case, we can say that the cross-validation technique doesn't improve our model.

Random forests are among the most popular machine learning methods and this technique has good accuracy, robustness, and ease of use. The best thing in random forests is, they provide two straightforward methods for feature selection, that are mean decrease impurity and mean decrease accuracy. Between these two (Mean Decrease Accuracy and Mean Decrease Gini), I would say, the mean decrease accuracy is more important in a case of variable selection. After selecting the variables, mean accuracy gini can give us the information about the relationship between the variables selected. The other important point is that when we fixed the missing values using "na.action=na.omit" we got 48.99% OOB estimate error and when we changed from "na.action=na.omit" to "na.action=na.exclude" the OOB estimate error went down to 47.84%. In our case, It doesn't change the error much but it might be very helpful in other cases. So, after using random forests we also learned that, If we reduce the size of the tree manually, the error rate will decrease but as the size of the tree increases computation time also creases.

Neural networks have a kind of universality. The neural network can be used anywhere, no matter what function we want to compute, we know that there is a neural network which can do the job. Neural network technique was comparatively tough and complicated than other techniques. We tried to change the neural network functions and we observed a lot of things. As we increased the number of hidden layers, the error decreased. But increasing the hidden layers also increases the computation time. We changed the algorithms which were very helpful. Adding (err.fct="ce") which is a differentiable function that is used for the calculation of the error, increased the error to 2101 but when we add err.fct="sse" instead of "ce" the error rate and the number of steps went down. The drawback of a neural network is there are fewer options for visualizing neural networks (less plotting methods).

Xgboost is highly efficient, flexible and portable. The most important feature of Xgboost is it can automatically do parallel computation on a single machine. If there is a large number of data then Xgboost will give the better performance but we have to keep on changing the parameters to check for the better result which is very time-consuming. We changed many parameters and record the best results among them. There are two learning task parameters for multiclass classification: "multi:softprob" or "multi:softmax". We used both of them and changed the parameters as well to check for the better result. In our case, the result of "multi:softmax" was better than "multi:softprob".

So, among these four techniques, as we can see from the final results (sample 15 outcomes) which we have shown above, in this section, that the final result from using these four techniques are almost same. So in our case, we cannot say which one is better and which one is least. All the four techniques worked well in our case, When we changed the parameters, the error rate fluctuates but even after that the final result was same, there was no change.

Data visualization is the fastest and most useful way to learn and understand more about the data, which can yield better results. So more plots and graphs can be constructed to understand the data in depth. Some other techniques can also be used to predict the final result and other parameters like goals, weather, injuries can be added to improve the accuracy.

E.2 Issues and Recommendation for Future Work

"The future of data mining lies in predictive analytics," declares Forrester Research analyst Lou Agosta [3]. There are many future important challenges in data-mining and analytics that arise from the nature of data, which is large, diverse, and evolving.

Area of predictive analytics which is the most important and fertile ground for the researchers is the area of data acquisition and storage. This area has become more challenging because several tasks need real time and high accuracy to accomplish multiple functions. While, collecting data and information from multiple sources through data acquisition is comparatively simple, because of the hardware and software equipment available in the market. The main issue raises when it comes to data storage, mainly due to storage devices, software algorithms and communication channels. There are two main approaches, that are compression and sampling. In compression we don't have to lose anything but it may take more time and in sampling, we choose the data that is more representative, in that case, we might loose some information but in both the cases, we will get more space in the end.

The ability to correctly acquire, clean and store data sets for data mining is very important and it is one of the main tasks. While gathering the data for the analysis, the important thing to keep in mind is that the data shouldn't be too old. To improve these issues, several commercial software packages are produced but most are uniquely made to solve particular types of problem. Most of the data in the field are unstructured, while current data-mining tools operate on structured data. So, there is a need of mining tools that can switch to multiple techniques and support multiple outcomes.

There are many companies which generate and collects the data based on the analysis mostly in the form of surveys. In surveys, the completion rate is very low, and most of the people do not even care about it, they just fill it for the sake of completing it. Not all the available data is useful, and historical data may not generate good rules or prediction models. Most of the times surveys are essential, biased in one direction only. Data is generating at such a high speed. So, the organizations can find more practical methods to make sense of what they had by employing statisticians to create better measures of performance, having a good understanding of the process, selected tools, techniques and better decision-making criteria.

After data gathering, the main task in data mining and analytics is visualization, how to visualize the results. As the data is so big and dense, it creates problems and difficulties in finding the user-friendly visualizations. To get the better visualization, new techniques and frameworks needed which would help in getting better results. Only when these requirements are met then the data analysts can confidently appreciate what the information implies.

References

[1] Selecting good features – part iii: random forests. Dec 01, 2014.

[2] R random forests variable importance. Dec 15, 2015.

[3] Lou Agosta. The future of data mining-predictive analytics. *DM REVIEW*, 14:16–20, 2004.

[4] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. 2000.

[5] Tianqi Chen and Tong He. xgboost: extreme gradient boosting. *R package version 0.4-2*, 2015.

[6] Terence Craig and Mary E Ludloff. *Privacy and big data.* " O'Reilly Media, Inc.", 2011.

[7] Bhavik Doshi. Handling missing values in data mining. *Data Cleaning and Preparation Term Paper.*

[8] Wayne W Eckerson. Predictive analytics. extending the value of your data warehousing investment. tdwi best practices report. q1 2007.

[9] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[10] Stefan Fritsch, Frauke Guenther, and Maintainer Frauke Guenther. Package 'neuralnet'.

[11] Christopher Gandrud. *Reproducible Research with R and R Studio.* CRC Press, 2013.

[12] Godswill's. Data-mining thesis. 2006.

[13] GK Gupta. *Introduction to data mining with case studies.* PHI Learning Pvt. Ltd., 2014.

[14] Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural network design*, volume 20. PWS publishing company Boston, 1996.

[15] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[16] Jeffrey Hsu. Data mining trends and developments: The key data mining technologies and applications for the 21st century. In *The Proceedings of the 19th Annual Conference for Information Systems Educators (ISECON 2002), ISSN*, pages 1542–7382, 2002.

[17] Johan Huysmans, Bart Baesens, David Martens, K Denys, and Jan Vanthienen. New trends in data mining. *Tijdschrift voor economie en Management*, 50(4):697, 2005.

[18] Vilas Jadhav. Data mining of big data: The survey and review.

[19] AARSHAY JAIN. Complete guide to parameter tuning in xgboost. MARCH 1, 2016.

[20] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[21] SB Kotsiantis, D Kanellopoulos, and PE Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[22] Yihao Li. Data mining: Concepts, background and methods of integrating uncertainty in data mining.

[23] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[24] Huan Liu and Hiroshi Motoda. *Instance selection and construction for data mining*, volume 608. Springer Science & Business Media, 2013.

[25] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.

[26] Björn-Helge Mevik, Ron Wehrens, et al. The pls package: principal component and partial least squares regression in r. *Journal of Statistical software*, 18(2):1–24, 2007.

[27] Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John Wiley & Sons, 2010.

[28] Daniel Nunan and MariaLaura Di Domenico. Market research and the ethics of big data. *International Journal of Market Research*, 55(4):2–13, 2013.

[29] Charles Nyce and API CPCU. Predictive analytics white paper. *American Institute for CPCU. Insurance Institute of America*, pages 9–10, 2007.

[30] Neelamadhab Padhy and Rasmita Panigrahi. Data mining: A prediction technique for the workers in the pr department of orissa (block and panchayat). *arXiv preprint arXiv:1211.5724*, 2012.

[31] Dr G Padmavathi, Dr P Subashini, Mr M Muthu Kumar, and Suresh Kumar Thakur. Performance analysis of non linear filtering algorithms for underwater images. *arXiv preprint arXiv:0912.1005*, 2009.

[32] S Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[33] Dorian Pyle. *Data preparation for data mining*, volume 1. Morgan Kaufmann, 1999.

[34] sebastian raschka. About feature scaling and normalization – and the effect of standardization for machine learning algorithms. Jul 11, 2014.

[35] Yakov Shafranovich. Common format and mime type for comma-separated values (csv) files. 2005.

[36] Yashpal Singh and Alok Singh Chauhan. Neural networks in data mining. *Journal of Theoretical and Applied Information Technology*, 5(6):36–42, 2009.

[37] Roman Timofeev. *Classification and regression trees (CART) theory and applications*. PhD thesis, Humboldt University, Berlin, 2004.

[38] Leland Wilkinson. Classification and regression trees. *Systat*, 11:35–56, 2004.