HarvardX: PH125.9x
Data Science: Capstone - Soccer Predictions

Sanver Gozen

May 21, 2021

Contents

1 Introduction

This paper is the second part of the "**HarvardX: Data Science" Professional Course's Capstone Project**"; Soccer Prediction.

Soccer is one of the most popular sports on the planet. In many countries, soccer games are large and well organized sports events with significant economic implications. In this study, I predict the full-time results

Figure 1: Turkish League Soccer Prediction Project

(**FTR**) of soccer matches in the "**Turkish Super Lig**", using data science and machine learning based on results and scores collected over 10 years.

Soccer clubs and organizations have taken the form of a profitable business. including sports bets. Here, predictions of game outcomes are of considerable interest.

Although the result of a football match depends on various variables, it is mainly determined over the offensive and defensive strengths of the teams. The outcomes of sports matches can be difficult to predict, with surprises often popping up. Despite these uncertainties, I will train the algorithm and generate predicted results or scores using by Poisson distribution and other models. Poisson distribution is a discrete probability distribution in the probability theory and statistics science branches. It aims to express the probability of occurrence of certain events in a fixed time unit interval.

$$P(x,\mu) = \frac{(e^{-\mu})(\mu^x)}{x!}$$

- e, the base of the natural logarithm.
- x, The number of occurrences of an event whose probability is given with function.
- $x!$, Factorial for x.
- μ, The expected value of occurrence of an event in a given fixed interval, a positive real number.

Also, I will compare the following models with each other in order to determine which is the best model for the data:

- Conditional Inference Trees
- Naive Bayes",”Support Vector
- Quadratic Discriminant Analysis
- Generalized Linear Model
- Neural Networks

1.1 Libraries

The following libraries are utilized:

```r
# Load Libraries
library(dplyr) # Provides a set of tools for efficiently manipulating datasets.
library(tidyverse) # An opinionated collection of R packages.
library(Metrics) # For Machine Learning, and predictions.
library(caret) # Classification And REgression Training.
library(kableExtra) # For better visualization of the tables.
library(nnet) # For compute Neural Networks model
library(MASS) # Support Functions and datasets
library(clusterSim) # Searching for Optimal Clustering Procedure for a Data Set
library(party) # A Laboratory for Recursive Partytioning
library(e1071) # Misc Functions of the Department of Statistics
library(neuralnet) # Training of Neural Networks
library(GGally) # Allows to build a great scatterplot matrix.
library(skellam) # Densities and Sampling for the Skellam Distribution
library(scales) # Scale Functions for Visualization
```

1.2 Dataset

I use the following code to generate the datasets. It will download the dataset according to a given time frame, which spans the years between 2010-2020:

```r
# Download the data
# Set the time frame
x <- c(2019:2010)
alldata <- data.frame()
# Download each season by the time frame
for (val in x) {
  url1 = "https://sports-statistics.com/database/soccer-data/turkey-futbol-ligi-1-"
  url2 = val
  url3 = "-to-"
  url4 = val + 1
  url5 = ".csv"
  url6 = "turkey-futbol-ligi-"
  href <- paste0(url1,url2,url3,url4,url5)
  href2 <- paste0(url6,url2,url5)
  download.file(href,href2)
  data <- read.csv(href2)
  data$HTHG <- as.integer(data$HTHG)
  # Remove empty rows from each file and clean the data
  final_data <- na.omit(data)
  if("PSCH" %in% names(data)) {final_data <- final_data %>% mutate(PSCH = as.character(PSCH))}
  # Merge the new data to the old one
  alldata <- bind_rows(alldata,final_data)
}
```

Let's clean the NA values from the dataset if it is exists, and remove unnecessary parts of the data.

```r
# Clean the data
alldata <- alldata %>% select_if(~ !any(is.na(.)))
# Remove unnecessary variables from environment
rm(data,final_data,href,href2,url1,url2,url3,url4,url5,url6,val,x)
```

Now, we have downloaded our data set. Next, we need to check the data.

2 Analysis

2.1 Data Summary

2.1.1 First Look

I now analyze the dataset, and visualize the data for better understanding:

Check the data info:

```r
# Check the data
dim(alldata)
```

```
## [1] 2184   25
```

There are 2184 match results (rows) and 25 different type of variables for each match (columns).

```r
# Let's see what is inside
head(alldata)%>%
  kable() %>% kable_styling(font_size = 10, position = "center",
                            latex_options = c("scale_down","HOLD_position"))
```

Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	B365H	B365D	B365A	BWH	BWD	BWA	IWH	IWD	IWA	WHH	WHD	WHA	VCH	VCD	VCA
T1	16/08/2019	Denizlispor	Galatasaray	2	0	H	0	0	D	4.50	3.75	1.75	4.25	3.8	1.75	4.15	3.70	1.75	4.00	3.7	1.83	4.40	3.8	1.75
T1	17/08/2019	Genclerbirligi	Rizespor	0	1	A	0	0	D	2.10	3.30	3.50	2.10	3.3	3.40	2.00	3.25	3.55	2.05	3.4	3.50	2.05	3.4	3.60
T1	17/08/2019	Kayserispor	Alanyaspor	0	1	A	0	0	D	2.40	3.20	3.00	2.35	3.2	3.00	2.35	3.20	2.90	2.35	3.4	2.88	2.40	3.3	2.90
T1	17/08/2019	Sivasspor	Besiktas	3	0	H	1	0	H	2.75	3.50	2.40	2.80	3.4	2.40	2.70	3.40	2.40	2.75	3.5	2.38	2.75	3.5	2.45
T1	18/08/2019	Konyaspor	Ankaragucu	0	0	D	0	0	D	1.72	3.50	5.00	1.75	3.5	4.75	1.73	3.45	4.70	1.73	3.5	4.80	1.73	3.6	5.00
T1	18/08/2019	Goztep	Antalyaspor	0	1	A	0	1	A	1.85	3.60	4.00	1.85	3.6	3.90	1.80	3.60	3.95	1.83	3.6	4.00	1.83	3.7	4.10

Summary of the first 10 headers of the data:

```r
# Summarize the data
summary(alldata[1:10])%>%
  kable() %>% kable_styling(font_size = 10, position = "center",
                            latex_options = c("scale_down","HOLD_position"))
```

	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR
	Length:2184	Length:2184	Length:2184	Length:2184	Min. :0.000	Min. :0.000	Length:2184	Min. :0.000	Min. :0.0000	Length:2184
	Class :character	Class :character	Class :character	Class :character	1st Qu.:1.000	1st Qu.:0.000	Class :character	1st Qu.:0.000	1st Qu.:0.0000	Class :character
	Mode :character	Mode :character	Mode :character	Mode :character	Median :1.000	Median :1.000	Mode :character	Median :0.000	Median :0.0000	Mode :character
	NA	NA	NA	NA	Mean :1.539	Mean :1.192	NA	Mean :0.668	Mean :0.5302	NA
	NA	NA	NA	NA	3rd Qu.:2.000	3rd Qu.:2.000	NA	3rd Qu.:1.000	3rd Qu.:1.0000	NA
	NA	NA	NA	NA	Max. :7.000	Max. :7.000	NA	Max. :5.000	Max. :6.0000	NA

The meaning of the headers is stated below:

```r
## Explain the Heads
# Div = League Division
# Date = Match Date (dd/mm/yy)
# HomeTeam = Home Team
# AwayTeam = Away Team
# FTHG and HG = Full Time Home Team Goals
# FTAG and AG = Full Time Away Team Goals
# FTR and Res = Full Time Result (H=Home Win, D=Draw, A=Away Win)
# HTHG = Half Time Home Team Goals
# HTAG = Half Time Away Team Goals
# HTR = Half Time Result (H=Home Win, D=Draw, A=Away Win)
# 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
# WHH = William Hill home win odds
# WHD = William Hill draw odds
# WHA = William Hill away win odds
# VCH = VC Bet home win odds
# VCD = VC Bet draw odds
# VCA = VC Bet away win odds
```

I check if there are any NA values in our dataset:

```
# Check NA values
anyNA(alldata)
```

```
## [1] FALSE
```

The dataset does not contain any **NA** values, which is of advantage.

2.2 Data Analysis

First the teams are selected and specified:

```
# See and set the Unique Home-Away Teams
alldata %>% summarise(Teams = n_distinct(HomeTeam))
```

```
##   Teams
## 1    32
```

```
alldata %>% summarise(Teams = n_distinct(AwayTeam))
```

```
##   Teams
## 1    33
```

The dataset distinguished between home and away teams. I unify the teams by removing the distinction:

```
# Find out which team is the different one
first <- c(alldata$HomeTeam )
second <- c(alldata$AwayTeam)
setdiff(second,first)
```

```
## [1] "Balikesirspor"
```

```
# Remove the different team
alldata <- alldata %>% filter(!str_detect(AwayTeam, 'Balikesirspor'))
```

Which leads to an array of unique team names:

```
# Set the Unique Team
unique_teams <- alldata %>% group_by(HomeTeam) %>% head(20)
unique_teams
```

```
## # A tibble: 20 x 25
## # Groups:   HomeTeam [18]
##    Div   Date  HomeTeam AwayTeam  FTHG  FTAG FTR    HTHG  HTAG HTR   B365H B365D
##    <chr> <chr> <chr>    <chr>    <int> <int> <chr> <int> <int> <chr> <dbl> <dbl>
##  1 T1    16/0~ Denizli~ Galatas~     2     0 H         0     0 D       4.5   3.75
##  2 T1    17/0~ Gencler~ Rizespor     0     1 A         0     0 D       2.1   3.3
##  3 T1    17/0~ Kayseri~ Alanyas~     0     1 A         0     0 D       2.4   3.2
##   4 T1    17/0~ Sivassp~ Besiktas     3      0 H          1      0 H       2.75  3.5
##   5 T1    18/0~ Konyasp~ Ankarag~     0      0 D          0      0 D       1.72  3.5
##   6 T1    18/0~ Goztep   Antalya~     0      1 A          0      1 A       1.85  3.6
##   7 T1    18/0~ Kasimpa~ Trabzon~     1      1 D          1      1 D       3.3   3.5
##   8 T1    18/0~ Yeni Ma~ Buyukse~     3      0 H          0      0 D       3.5   3.25
##   9 T1    19/0~ Fenerba~ Gaziant~     5      0 H          3      0 H       1.44  4.2
##  10 T1    23/0~ Besiktas Goztep       3      0 H          1      0 H       1.44  4.2
##  11 T1    24/0~ Ankarag~ Kayseri~     1      1 D          0      0 D       2.37  3.2
##  12 T1    24/0~ Alanyas~ Kasimpa~     4      1 H          1      1 D       1.66  4.33
##  13 T1    24/0~ Buyukse~ Fenerba~     1      2 A          1      0 H       2.45  3.2
##  14 T1    25/0~ Rizespor Sivassp~     2      1 H          0      0 D       2.05  3.3
##  15 T1    25/0~ Trabzon~ Yeni Ma~     2      1 H          1      0 H       1.65  3.75
##  16 T1    25/0~ Antalya~ Denizli~     0      2 A          0      1 A       2.62  3.2
##  17 T1    25/0~ Galatas~ Konyasp~     1      1 D          0      0 D       1.4   4.75
##  18 T1    26/0~ Gaziant~ Gencler~     4      1 H          2      0 H       2.37  3.25
##  19 T1    30/0~ Kasimpa~ Ankarag~     0      1 A          0      1 A       1.7   3.8
##  20 T1    30/0~ Kayseri~ Galatas~     2      3 A          1      0 H       4.75  3.6
## # ... with 13 more variables: B365A <dbl>, BWH <dbl>, BWD <dbl>, BWA <dbl>,
## #   IWH <dbl>, IWD <dbl>, IWA <dbl>, WHH <dbl>, WHD <dbl>, WHA <dbl>,
## #   VCH <dbl>, VCD <dbl>, VCA <dbl>
```

The following section identifies duplicates and combines them:

```
# String replace to avoid duplicate teams
alldata$HomeTeam <- lapply(alldata$HomeTeam, gsub, pattern = "Gaziantepspor",
                           replacement = "Gaziantep", fixed = TRUE)
alldata$HomeTeam <- unlist(alldata$HomeTeam)
alldata$AwayTeam <- lapply(alldata$AwayTeam, gsub, pattern = "Gaziantepspor",
                           replacement = "Gaziantep", fixed = TRUE)
alldata$AwayTeam <- unlist(alldata$AwayTeam)
```

This leads to unique team identifiers:

```
# Now we have exact unique teams
unique_teams <- alldata %>% group_by(HomeTeam) %>% summarise()
alldata %>% summarise(Teams = n_distinct(HomeTeam))
```

```
##   Teams
## 1    31
```

Now, we have our real unique teams as we can see from above.

2.2.1 Home Team Effect

The data analysis continues with a count of Full Time Result's(**FTR**):

```
# Home, Away and Draw count
table(alldata$FTR)
```

```
## 
##   A   D   H 
## 621 568 994
```

It becomes apparent that the "**Home**" team winning count exceeds the "**Draw**" or "**Away**" counts. As expected, the home team has an advantage over the away team.

```r
# Summarize results
summary(alldata$FTR)
```

```
##    Length     Class      Mode 
##      2183 character character 
```

The FTR data appears as type "character". In the following, the match data is analyzed by season. Thus, the time/date data are cleared or modified.

```r
# Check the date's class
class(alldata$Date)
```

```
## [1] "character"
```

Date data is of type "**character**" and needs conversion to the date. But, first, I create a "**somedata**" variable to assign **alldata**, such that I can modify it without losing any data from **alldata**.

```r
# Create "somedata" to not lose or mess any data from "alldata" data set.
somedata <- alldata
```

Now, I create **newdate** column in **somedata** for clear or modify date data:

```r
# Change Date format
somedata$newdate <- strptime(as.character(alldata$Date), "%d/%m/%Y")
```

I then clean and replace bad data with valid characters or dates:

```r
# Replace bad data with the valid ones
somedata$newdate <- lapply(somedata$newdate, gsub, pattern = "/00", replacement = "20", fixed = TRUE)
somedata$newdate <- lapply(somedata$newdate, gsub, pattern = "/2019", replacement = "19", fixed = TRUE)
# Create a for loop for fix the dates
url1 = "00"
url2 <- c(19:10)
url3 = "20"
for (val1 in url2) {
  d <- paste0(url1,val1)
  f <- paste0(url3,val1)
    somedata$newdate <- lapply(somedata$newdate, gsub, pattern = d, replacement = f, fixed = TRUE)
  }
# Unlist the dates column
somedata$newdate <- unlist(somedata$newdate)
```

I check the date class:

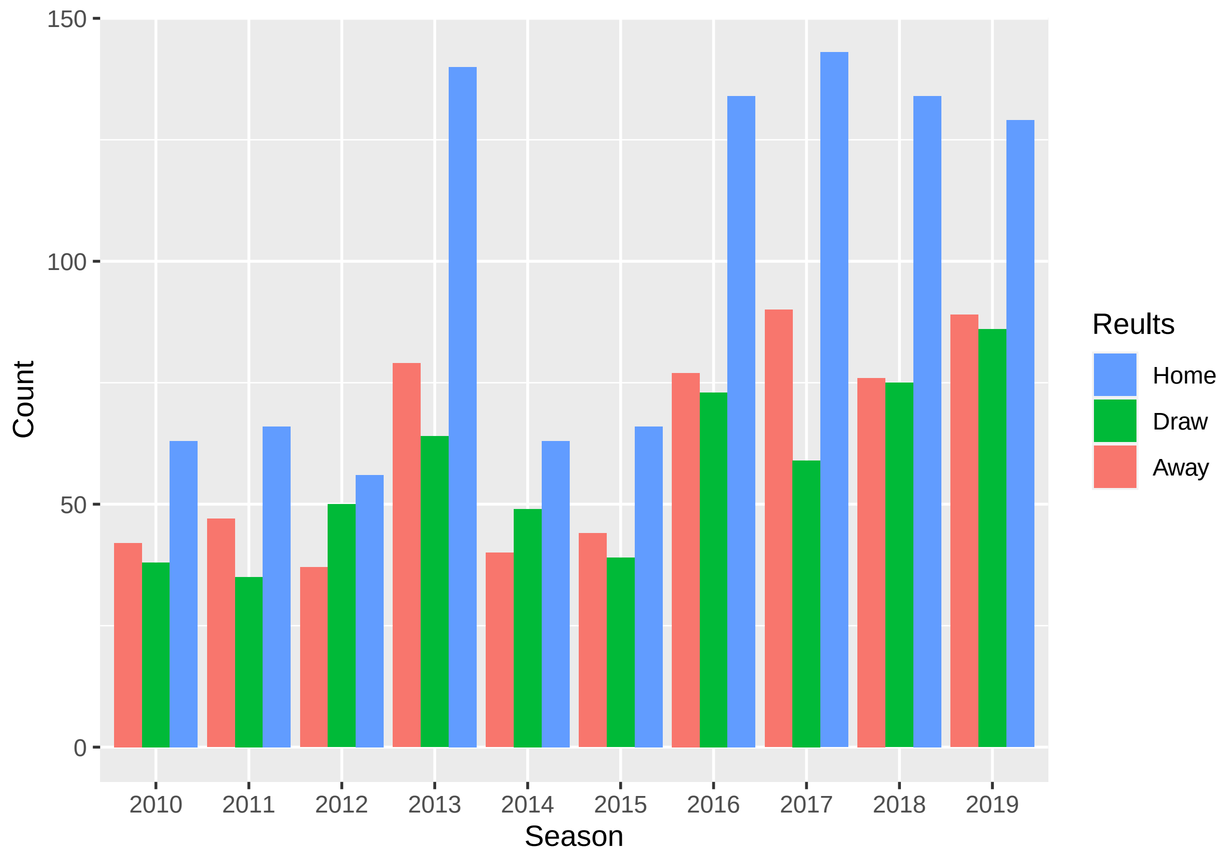
```r
# Check the class
class(somedata$newdate)
```

```
## [1] "character"
```

```
# Change the class of the date data
somedata$newdate <- as.Date(somedata$newdate)
```

I am now proceeding to analyzing the seasonal data:

```
# Plot Full Time Result by Season
season_graph <- somedata %>%
  mutate(month = format(newdate, "%m"), year = format(newdate, "%Y")) %>%
  group_by(month, year) %>% group_by(newdate) %>% ggplot(aes(x = year, fill = FTR)) +
  geom_bar(position = position_dodge()) +
  ylab("Count") +
  xlab("Season")+ scale_fill_discrete(name = "Reults", labels = c("Away", "Draw", "Home"),guide = guide
# Plot the data
season_graph
```



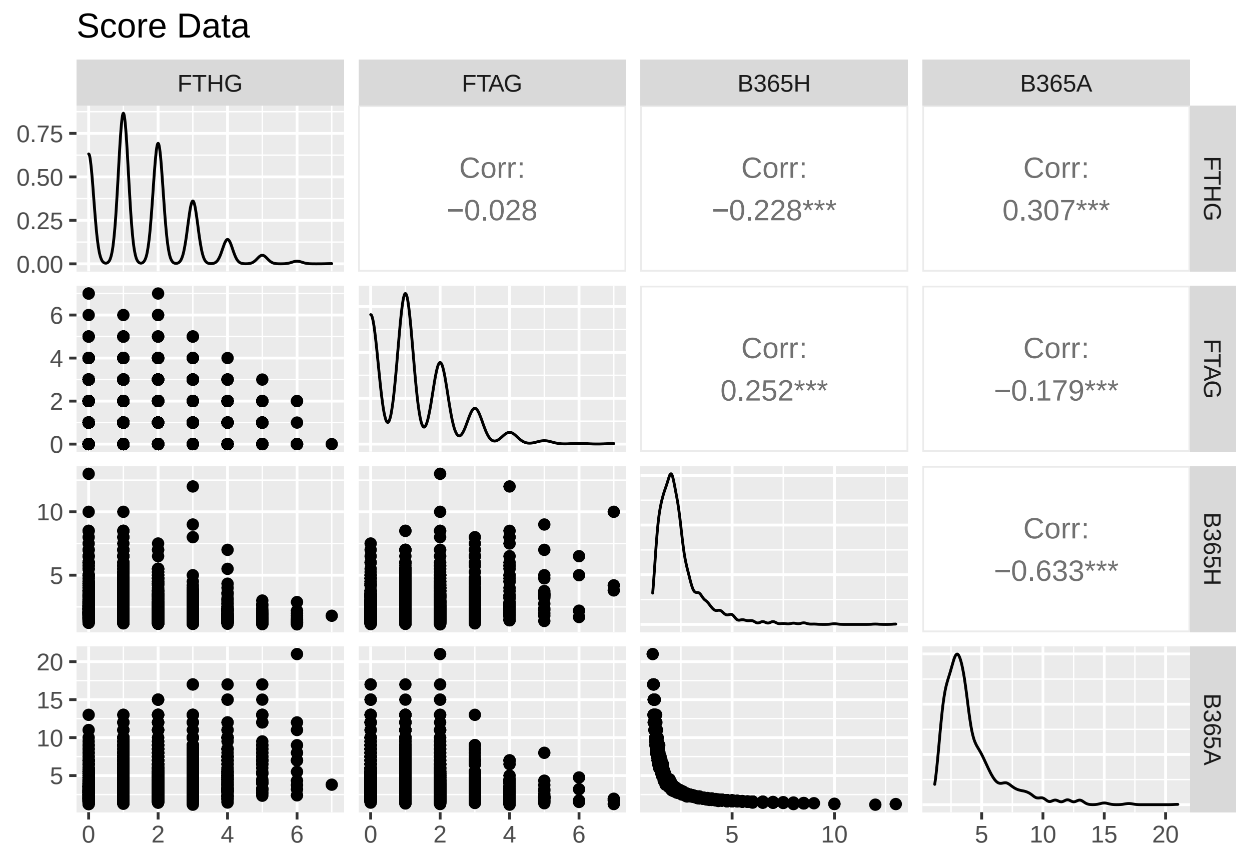
The chart reveals that, each season, the home team's are winning more frequently than the away teams, and there are some seasons that have matches more than others. I look into the data in more detail in order to compare them.

```
# See detailed data
glimpse(alldata)
```

```
## Rows: 2,183
## Columns: 25
## $ Div      <chr> "T1", "T1", "T1", "T1", "T1", "T1", "T1", "T1", "T1", "T1"...
## $ Date     <chr> "16/08/2019", "17/08/2019", "17/08/2019", "17/08/2019", "1...
## $ HomeTeam <chr> "Denizlispor", "Genclerbirligi", "Kayserispor", "Sivasspor...
## $ AwayTeam <chr> "Galatasaray", "Rizespor", "Alanyaspor", "Besiktas", "Anka...
## $ FTHG     <int> 2, 0, 0, 3, 0, 0, 1, 3, 5, 3, 1, 4, 1, 2, 2, 0, 1, 4, 0, 2...
## $ FTAG     <int> 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 2, 1, 1, 2, 1, 1, 1, 3...
## $ FTR      <chr> "H", "A", "A", "H", "D", "A", "D", "H", "H", "H", "D", "H"...
## $ HTHG     <int> 0, 0, 0, 1, 0, 0, 1, 0, 3, 1, 0, 1, 1, 0, 1, 0, 0, 2, 0, 1...
## $ HTAG     <int> 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0...
## $ HTR      <chr> "D", "D", "D", "H", "D", "A", "D", "D", "H", "H", "D", "D"...
## $ B365H    <dbl> 4.50, 2.10, 2.40, 2.75, 1.72, 1.85, 3.30, 3.50, 1.44, 1.44...
## $ B365D    <dbl> 3.75, 3.30, 3.20, 3.50, 3.50, 3.60, 3.50, 3.25, 4.20, 4.20...
## $ B365A    <dbl> 1.75, 3.50, 3.00, 2.40, 5.00, 4.00, 2.10, 2.10, 7.50, 7.50...
## $ BWH      <dbl> 4.25, 2.10, 2.35, 2.80, 1.75, 1.85, 3.10, 3.30, 1.44, 1.50...
## $ BWD      <dbl> 3.80, 3.30, 3.20, 3.40, 3.50, 3.60, 3.60, 3.30, 4.40, 4.10...
## $ BWA      <dbl> 1.75, 3.40, 3.00, 2.40, 4.75, 3.90, 2.15, 2.15, 6.75, 6.25...
## $ IWH      <dbl> 4.15, 2.00, 2.35, 2.70, 1.73, 1.80, 3.10, 3.40, 1.45, 1.47...
## $ IWD      <dbl> 3.70, 3.25, 3.20, 3.40, 3.45, 3.60, 3.60, 3.20, 4.35, 4.20...
## $ IWA      <dbl> 1.75, 3.55, 2.90, 2.40, 4.70, 3.95, 2.05, 2.10, 6.10, 6.10...
## $ WHH      <dbl> 4.00, 2.05, 2.35, 2.75, 1.73, 1.83, 3.10, 3.40, 1.44, 1.47...
## $ WHD      <dbl> 3.70, 3.40, 3.40, 3.50, 3.50, 3.60, 3.70, 3.30, 4.33, 4.33...
## $ WHA      <dbl> 1.83, 3.50, 2.88, 2.38, 4.80, 4.00, 2.10, 2.10, 7.00, 6.50...
## $ VCH      <dbl> 4.40, 2.05, 2.40, 2.75, 1.73, 1.83, 3.13, 3.40, 1.45, 1.45...
## $ VCD      <dbl> 3.80, 3.40, 3.30, 3.50, 3.60, 3.70, 3.75, 3.30, 4.60, 4.40...
## $ VCA      <dbl> 1.75, 3.60, 2.90, 2.45, 5.00, 4.10, 2.10, 2.15, 6.50, 7.00...
```

To compare different data sets, the data from one betting company's data is sufficient to see if there is a correlation or not. I include home goals (**FTHG**), away goals (**FTAG**), home win odds (**B265H**) and away win odds (**B365A**):

```
# Plot Matrix for understand and observe data
ggpairs(data=alldata, columns=c(5,6,11,13), title="Score Data",cardinality_threshold=NULL)
```



As expected, there is only a negative correlation with **B365H** and **B365A**, which are opposite odds for away and home teams.

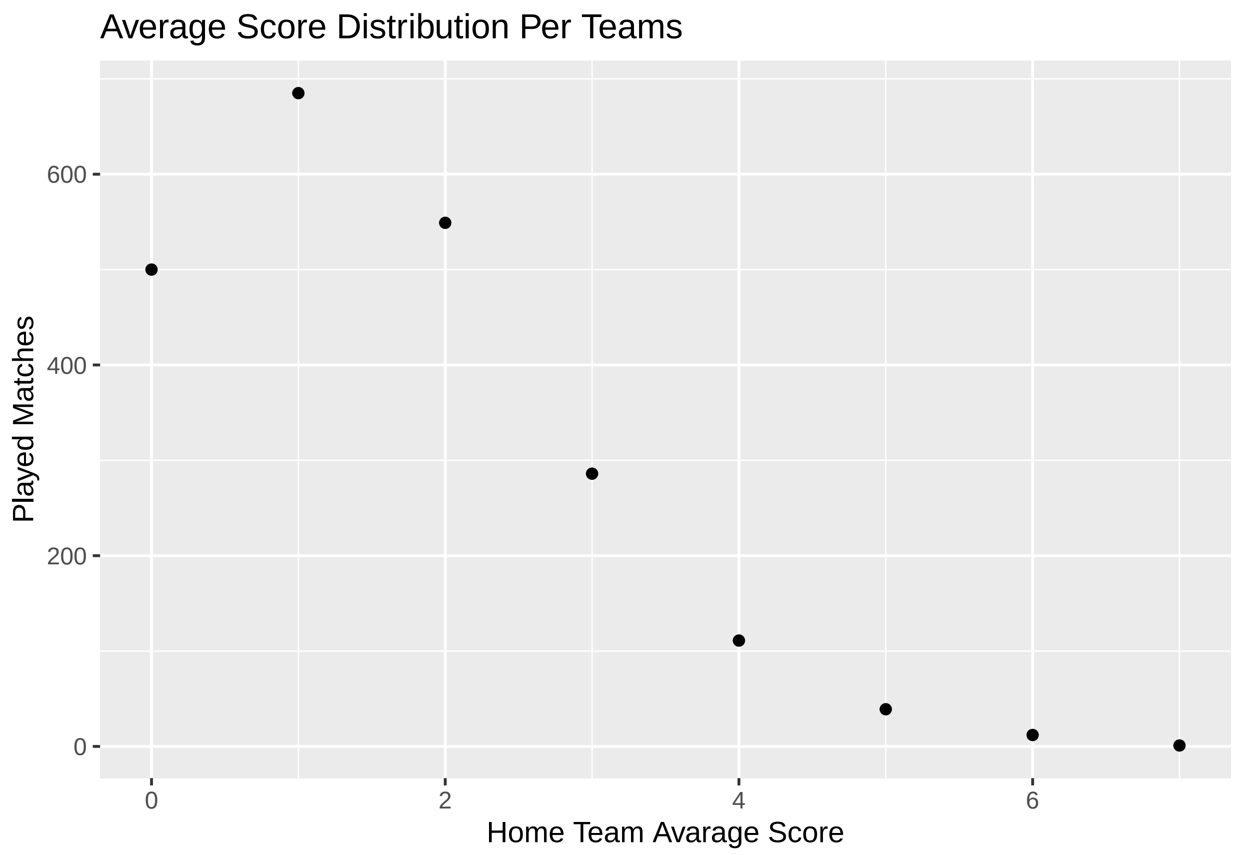
Now, I take a look at the Full Time Result (**FTR**) in order to understand the score distribution:

```
# Away, Draw, Home Means
FTR_means <- alldata %>%
  group_by(FTR) %>%
  summarise(mean_scored = mean(FTHG)) %>%
  print()
```

```
## # A tibble: 3 x 2
##   FTR   mean_scored
##   <chr>       <dbl>
## 1 A           0.626
## 2 D           0.979
## 3 H           2.43
```

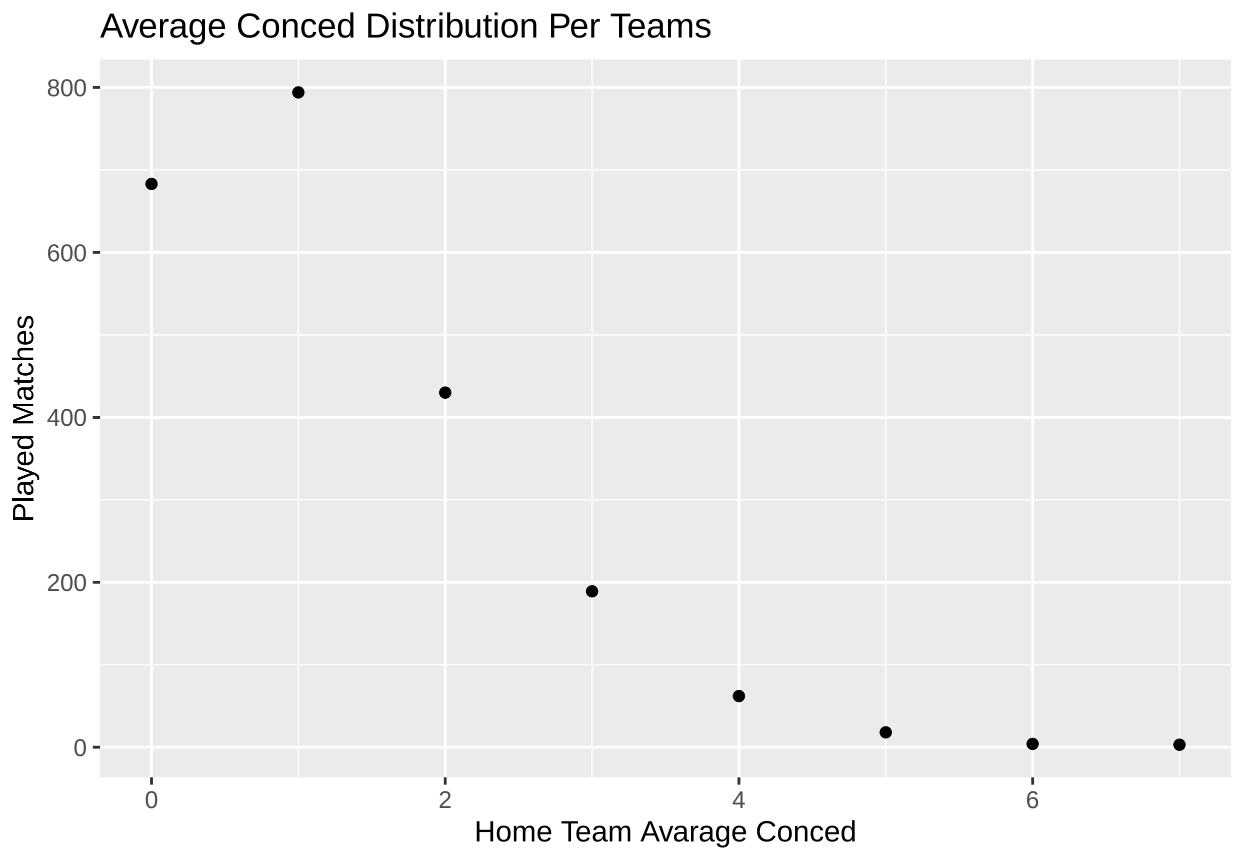
From the above analysis it can be seen that the home teams score average is **2.43**, while the away teams score average is **0.626**. The "home team effect" is apparent. Next, the scoring distribution per team is evaluated:

```
# Average Goal Score per Team
alldata %>%
  group_by(FTHG) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x = FTHG, y = count)) +
  geom_point() +
  xlab("Home Team Avarage Score") +
  ylab("Played Matches") +
  scale_y_continuous(labels = scales::label_number_si())+
  labs(title = "Average Score Distribution Per Teams")+
  theme()
```



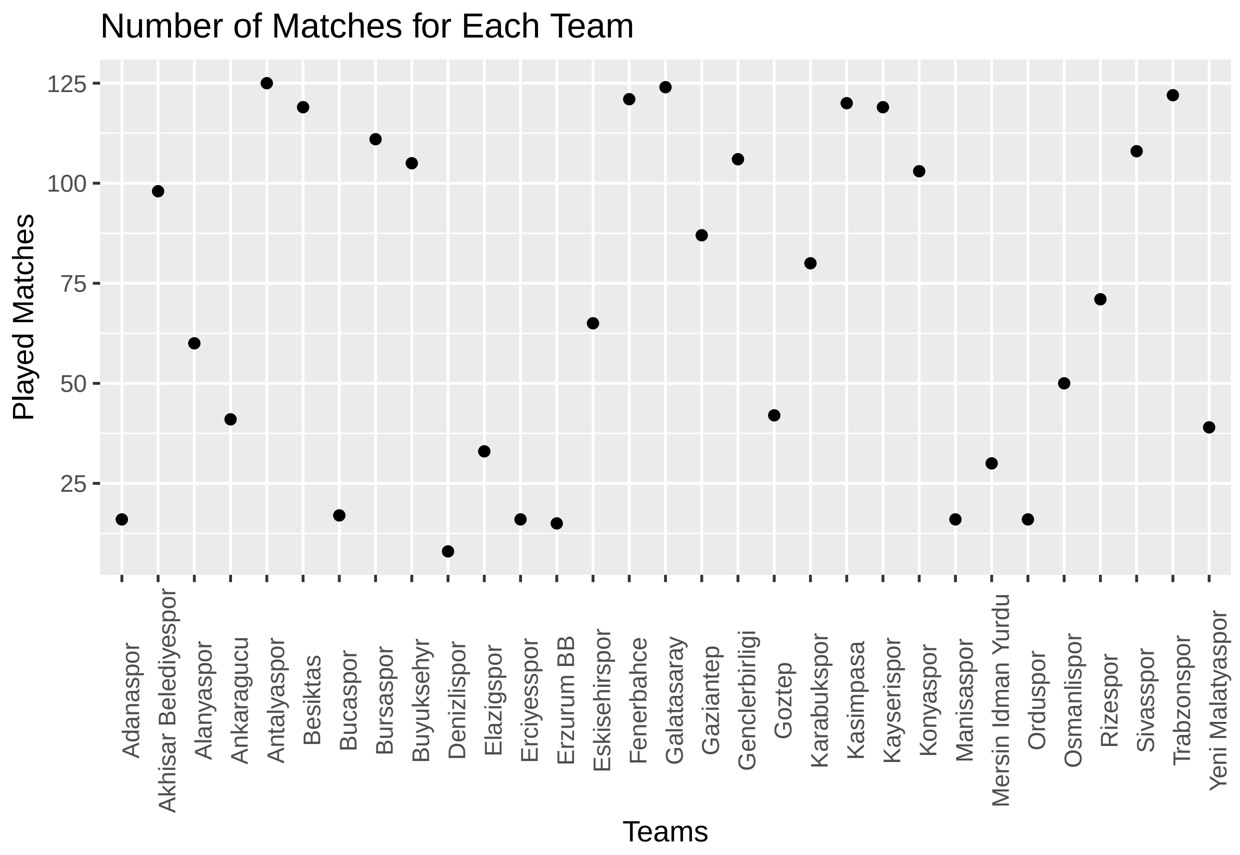
The average number of goals conceded per team is obtained from:

```
# Average Goal Conceded per Team
alldata %>%
  group_by(FTAG) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x = FTAG, y = count)) +
  geom_point() +
  xlab("Home Team Avarage Conced") +
  ylab("Played Matches") +
  scale_y_continuous(labels = scales::label_number_si())+
  labs(title = "Average Conced Distribution Per Teams")+
  theme()
```



From the above results, it became apparent that in almost **700** matches the home team scored by **1** goal, and in almost **800** matches, the home team conceded by **1** goal. So, we understood that soccer is a low scoring game and the probability for having more than seven goals is very low for "**Turkish Super Lig**".

```
# All teams number of played games
alldata %>%
  group_by(HomeTeam) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x = HomeTeam, y = count)) +
  geom_point() +
  xlab("Teams") +
  ylab("Played Matches") +
  labs(title = "Number of Matches for Each Team")+
  theme(axis.text.x  = element_text(angle= 90))
```



The "**Turkish Super Lig**" has **18** teams on each season. New additions are coming from the champions of the second division, and the teams of lower performance are descending to the second division for the next season. For this reason, some teams participate for a limited time, e.g. 2-3 seasons, and there are teams that have played more matches than others. I will consider this later.

I identify the best teams in the league according to the difference of scored mean and conceded mean:

```
# See the Best Teams
best_teams <- alldata %>% group_by(HomeTeam) %>%
  summarise(Scored_Mean = mean(FTHG),
            Conceeded_Mean = mean(FTAG)) %>%
  arrange(desc(Scored_Mean-Conceeded_Mean)) %>% head(20)
best_teams
```

```
## # A tibble: 20 x 3
##    HomeTeam            Scored_Mean Conceeded_Mean
##    <chr>                     <dbl>          <dbl>
##  1 Fenerbahce                 2.21          0.917
##  2 Galatasaray                2.19          0.919
##  3 Besiktas                   2.13          0.992
##  4 Buyuksehyr                 1.76          0.990
##  5 Trabzonspor                1.73          1.10
##  6 Yeni Malatyaspor           1.69          1.13
##  7 Sivasspor                  1.64          1.15
##  8 Alanyaspor                 1.8           1.33
##  9 Goztep                     1.55          1.24
## 10 Bursaspor                    1.29          1.08
## 11 Konyaspor                    1.21          1.01
## 12 Genclerbirligi               1.27          1.08
## 13 Eskisehirspor                1.38          1.22
## 14 Rizespor                     1.45          1.30
## 15 Antalyaspor                  1.48          1.37
## 16 Erzurum BB                   1.13          1.07
## 17 Kasimpasa                    1.6           1.53
## 18 Akhisar Belediyespor         1.30          1.23
## 19 Elazigspor                   1.06          1
## 20 Osmanlispor                  1.36          1.38
```

The data is filtered for the teams who participated in more than 100 matches:

```
# Filtered-Teams, Scored-Conceded Average per team
filtered_teams <- alldata %>% group_by(HomeTeam) %>%
  summarise(Scored_Mean = mean(FTHG),
            Conceeded_Mean = mean(FTAG),
            count=n()) %>% filter(count > 100)%>%
  arrange(desc(Scored_Mean-Conceeded_Mean)) %>% head(20)
filtered_teams
```

```
## # A tibble: 12 x 4
##    HomeTeam       Scored_Mean Conceeded_Mean count
##    <chr>                <dbl>          <dbl> <int>
##  1 Fenerbahce            2.21          0.917   121
##  2 Galatasaray           2.19          0.919   124
##  3 Besiktas              2.13          0.992   119
##  4 Buyuksehyr            1.76          0.990   105
##  5 Trabzonspor           1.73          1.10    122
##  6 Sivasspor             1.64          1.15    108
##  7 Bursaspor             1.29          1.08    111
##  8 Konyaspor             1.21          1.01    103
##  9 Genclerbirligi        1.27          1.08    106
## 10 Antalyaspor           1.48          1.37    125
## 11 Kasimpasa             1.6           1.53    120
## 12 Kayserispor           1.11          1.24    119
```

Note that some teams, such as "**Yeni Malatya**", were removed from the first list due to filtering the data. The number of played matches will affect the average.

Now time to see, average of game results:

```
# Average of Game Results
H <- alldata %>% group_by(HomeTeam) %>% filter(FTR=="H") %>%
  summarise(count = n())

A <- alldata %>% group_by(HomeTeam) %>% filter(FTR== "A") %>%
  summarise(count = n())

D <- alldata %>% group_by(HomeTeam) %>% filter(FTR=="D") %>%
  summarise(count = n())

# See the result
mean(D$count)
```

```
## [1] 18.32258
```

```
mean(A$count)
```

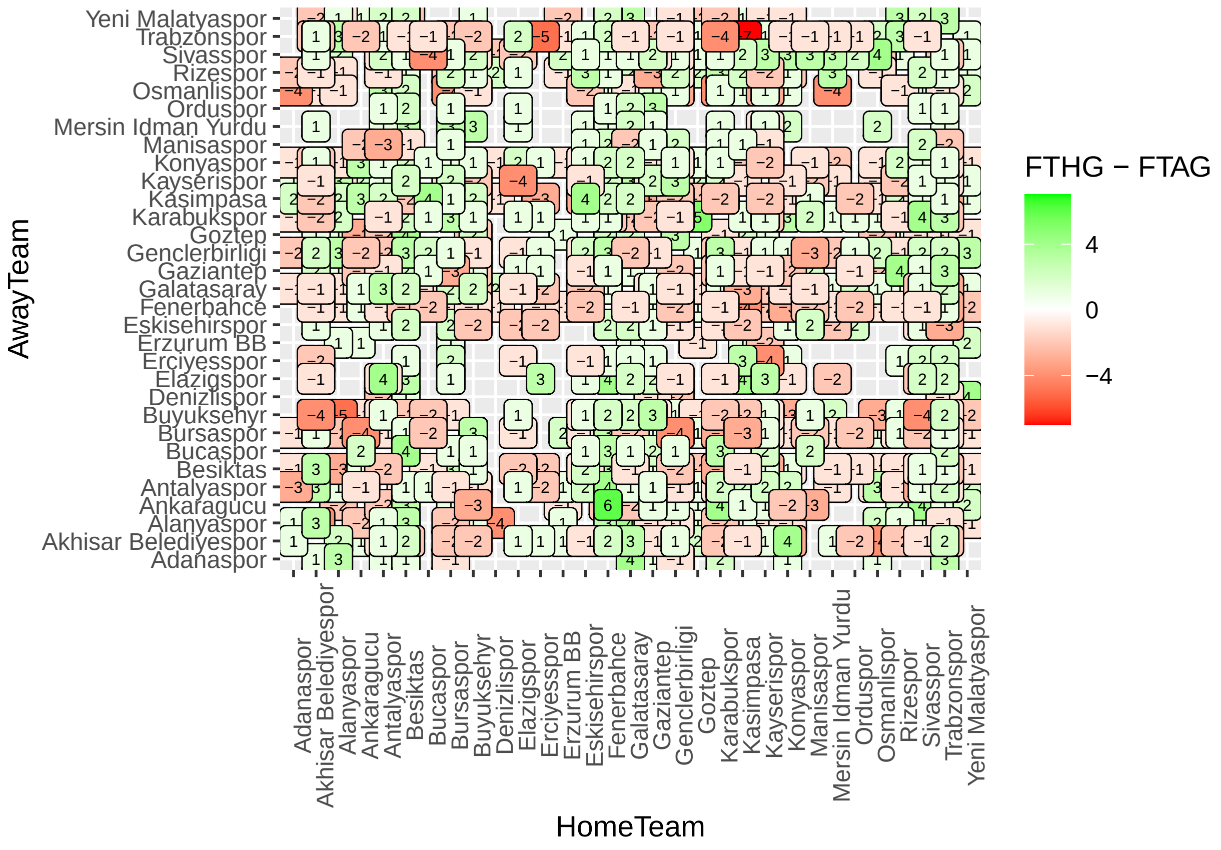
```
## [1] 20.03226
```

```
mean(H$count)
```

```
## [1] 32.06452
```

As expected, the home teams' winning average is higher than the draw teams' and away teams' averages. Below is the matrix of home-away scoring data:

```
# Plot Home Away Matrix
HomeAwayMatrix <- alldata %>%
  # Remove not played games
  filter(!is.na(FTHG) & !(FTHG-FTAG==0)) %>%
  ggplot(., aes(x = HomeTeam, y = AwayTeam, fill = FTHG-FTAG)) +
  geom_tile() +
  # Add the scorelines
  geom_label(aes(label = paste(FTHG-FTAG)), size = 2) +
  # Coloring - Where green shows home wins and red an away wins
  scale_fill_gradient2(low = "red", high = "green", midpoint = 0) +
  scale_x_discrete(limits = levels(alldata$HomeTeam)) +
  scale_y_discrete(limits = levels(alldata$AwayTeam)) +
  theme(axis.text.x = element_text(angle = 90))
# Plot the Matrix
HomeAwayMatrix
```



Each teams' net scores can be identified and characterized in the matrix. "**Red**" means that a team conceded more than scored, "**Green**" denotes the opposite.

I now evaluate the **Attack** and **Defense** Ratings of each team. While the attack rating of a team suggests how many goals a team has scored per game with respect to the total average of the league in which they are competing, the defense rating indicates how many goals a team has conceded per game with respect to the average.

```r
# Best Teams by attack and defense rating
rating <- alldata %>% group_by(HomeTeam) %>%
  summarise(Attack = mean(FTHG) / mean(alldata$FTHG),
            Defense = mean(FTAG)/ mean(alldata$FTAG)) %>%
  arrange(desc(Attack-Defense)) %>% head(20)
rating
```

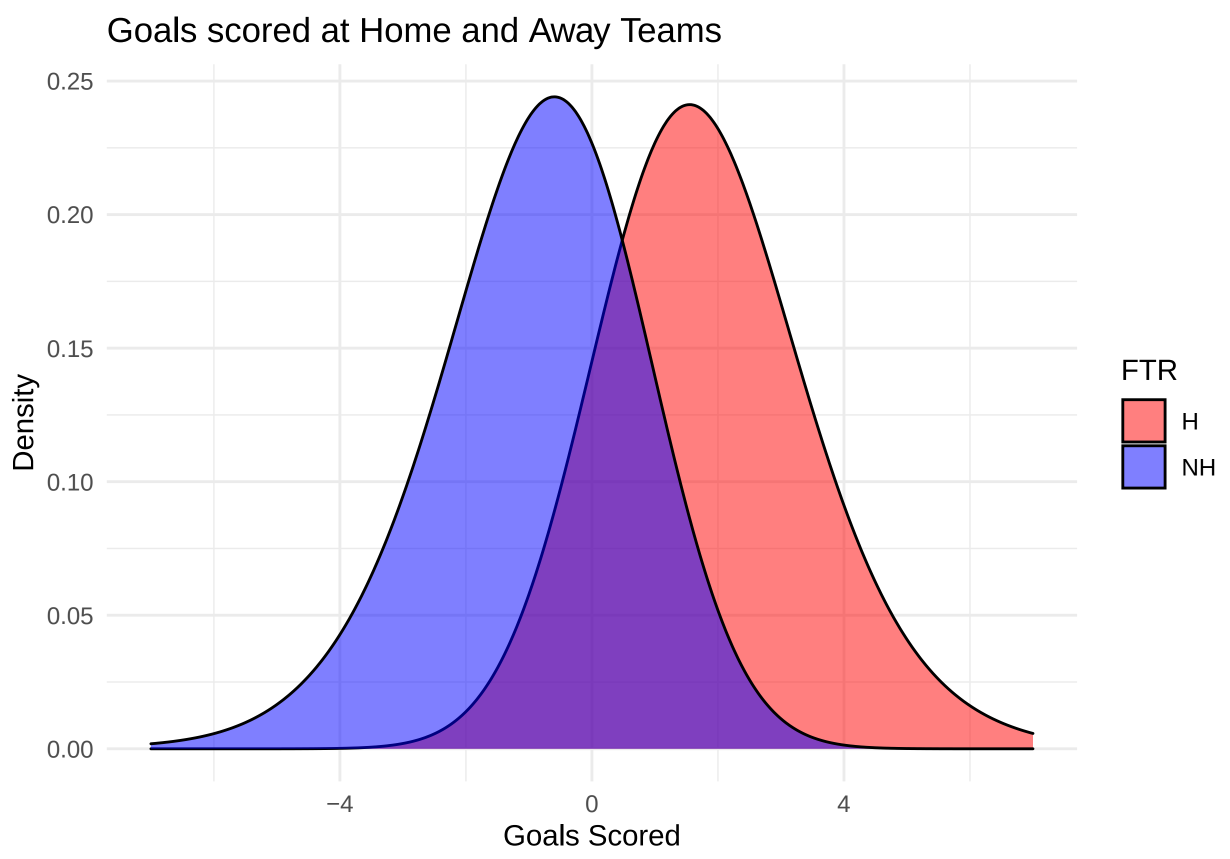
```
## # A tibble: 20 x 3
##    HomeTeam             Attack Defense
##    <chr>                 <dbl>   <dbl>
##  1 Fenerbahce             1.44   0.769
##  2 Galatasaray            1.43   0.771
##  3 Besiktas               1.39   0.831
##  4 Buyuksehyr             1.15   0.830
##  5 Trabzonspor            1.12   0.921
##  6 Yeni Malatyaspor       1.10   0.946
##  7 Sivasspor              1.07   0.963
##  8 Alanyaspor             1.17   1.12
##  9 Goztep                 1.01    1.04
## 10 Konyaspor              0.789   0.846
## 11 Bursaspor              0.837   0.906
## 12 Genclerbirligi         0.828   0.910
## 13 Eskisehirspor          0.900   1.02
## 14 Rizespor               0.943   1.09
## 15 Elazigspor             0.689   0.838
## 16 Erzurum BB             0.737   0.894
## 17 Antalyaspor            0.962   1.15
## 18 Akhisar Belediyespor   0.842   1.04
## 19 Kasimpasa              1.04    1.29
## 20 Osmanlispor            0.884   1.16
```

To make the predictions simpler and more accurate, I change the probability of the final result to two different results instead of three. The data is displayed as Home Win (**H**), and Not Home Win (**NH**):

```
# Change the data to Home and NotHome.
somedata <- alldata
# Replace "Away" with "Not Home"
somedata$FTR <- lapply(somedata$FTR, gsub, pattern = "A", replacement = "NH", fixed = TRUE)
# Replace "Draw" with "Not Home"
somedata$FTR <- lapply(somedata$FTR, gsub, pattern = "D", replacement = "NH", fixed = TRUE)
somedata$FTR <- unlist(somedata$FTR)
```

In the following the **Home** and **Not Home** goal density is evaluated:

```
# plot Home Goals scored Home - Not Home
HomeGoalsSocred <- somedata %>%
  ggplot(., aes(x = (FTHG-FTAG), fill = FTR)) +
  # smooth densities
  geom_density(adjust = 8, alpha = 0.5) +
  scale_fill_manual(values = c("red", "blue")) +
  labs(title = "Goals scored at Home and Away Teams",
       x = "Goals Scored",
       y = "Density") +
  theme_minimal()
# plot
HomeGoalsSocred
```



3 Modeling

3.1 Train and Test Data

Any training-test split which has more data in the training set will most likely lead to better accuracy as calculated on that test set. However, the accuracy of the test set is not of interest. I am interested in the "**real**" accuracy, which gets estimated by the test set. With data from ten seasons of the "**Turkish Super Lig.**", a **90%-10%** set for at least 1 season can be obtained to predict results. Let's start with split our data into 2 parts; as test and training:

```r
## Train and Test data
# Create "somedata" to not lose any info from "alldata" data set.
somedata = alldata
set.seed(2105, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
# Split data to train and test data by portion of %10-%90
test_index <- createDataPartition(y = somedata$HomeTeam, times = 1,
                                  p = 0.1, list = FALSE)
train <- somedata[-test_index,]
test <- somedata[test_index,]
rm(test_index)
```

Now, a **90%** portion of all data is used to train the models, and **10%** is used for testing the models and for determining the accuracy of the models.

3.2 Score Prediction and Winning Match Percentage

As mentioned in the introduction, the **Poisson** model will be used in the project to predict the results and the scores football matches.

$$P(x, \mu) = \frac{(e^{-\mu})(\mu^x)}{x!}, x = 0, 1, 2, ..., \mu > 0$$

The general focus of the Poisson distribution is a variable event; this event occurs at a certain time interval, and the number of events observed in this range is considered to be a random variable for the Poisson distribution. The expected value of the number of events occurring in this fixed range (the mean number of occurrences) is fixed as μ, and this mean value is proportional to the range length.

The function of x is the probability mass function for the **Poisson** distribution.

Poisson Distribution is a simple predictive model that does not allow for numerous factors. **Situational factors – such as club circumstances, game status, player performances etc. – and subjective evaluation of the change of each team during the transfer window are completely ignored**. Thus we will just calculating the scores without considering any other effects.

The constant e is the unique real number such that the value of the derivative (slope of the tangent line) of the function $f(x) = e^x$ at the point of $x = 0$ is equal to 1.

The exponential dispersion form can be retrieved as follows:

$$\begin{aligned} P(x, \mu) = \frac{(e^{-\mu})(\mu^x)}{x!} &= exp[log(e^{-\mu}) + log(\mu^x) - log(x!)] \\ &= exp(x.log(\mu) - \mu - log(x!)) \end{aligned}$$

A Poisson Regression model is a **Generalized Linear Model** (GLM) that is used to model count data and contingency tables. The output **Y** (count) is a value that follows the Poisson distribution. It assumes the logarithm of the expected values (mean) that can be modeled into a linear form by some unknown parameters:

$$Y_{u,i} = a + \beta_1 x_1 i + \beta_2 x_2 i + ... + \beta_p x_p i + e_i$$

The response variable y_i is modeled by a linear function of predictor variables and some error term. To transform the non-linear relationship to linear form, a link function is used which is the log for Poisson Regression. For that reason, a Poisson Regression model is also called log-linear model. The general mathematical form of Poisson Regression model is:

$$log(y) = a + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$$

y: is the response variable a and β: are numeric coefficients, a being the intercept, sometimes a also is represented by β_0 x: is the predictor/explanatory variable

Consider an equation with one predictor variables and one response variable:

$$log(y) = a + \beta(x)$$

This is equivalent to:

$$y = e^{(a+\beta(x))} = e^a + e^{\beta * x}$$

One of the most important characteristics for Poisson distribution and Poisson Regression is equidispersion, which means that the mean and variance of the distribution are equal. Variance measures the spread of the data. It is the "average of the squared differences from the mean". Variance (VAR) is equal to 0 if all values are identical. The greater the difference between the values, the greater the variance. Mean is the average of values of a dataset. Average is the sum of the values divided by the number of values.

Let us say that the mean μ is denoted by $E(X)$:

$$E(X) = \mu$$

For Poisson Regression, mean and variance are related as:

$$var(X) = \sigma^2 E(X)$$

Where σ^2 is the dispersion parameter. Since var(X)=E(X)(variance=mean) must hold for the Poisson model to be completely fit, σ^2 must be equal to 1.

Therefore, the expectation is equal to the variance as expected in the Poisson case.

For predicting football results, models that use an attack and a defense coefficient to estimate the expected number of goals for both the home team and the away team can be found in popular science. Since the two random variables are assumed to be independent, the bi-variate Poisson density will simply be the product of the two marginal Poisson densities. Mathematically, it can be expressed as follows:

X = "Number of home goals" ~ Poisson(μx) and Y = "Number of away goals" ~ Poisson(μy)

The expectations for each of the two random variables (i.e. μx and μy) are then calculated by assigning an attack and a defense coefficient for both the home and away team. The attack coefficient for the home team is estimated by taking the ratio of the average number of home goals scored by the home team this season, and the total average of all home goals scored this season. The attack coefficient for the away team is then estimated in an analogous manner, but by using the ratio of the average number of away goals scored by the away team and the total average of all away goals scored this season.

The defense coefficient for the home team is then calculated by taking the ratio of the average number of goals conceded by the home team at home and the total average of goals conceded at home this season. The defense coefficient for the away team is calculated by taking the ratio of the average number of goals conceded away for the away team and the total average of goals conceded away this season.

So, in order to predict how many goals the team can score, I use:

$$log(\mu.home) = a + \beta.attack.home + \beta.defense.away$$

I create the model as follows:

```r
# Poisson Distribution - Predict Games by Home and Away Team
# Create a Model
poisson_model <-
  rbind(
    # Calculate Home Attack Rating,
    data.frame(goals = train$FTHG,
               team = train$HomeTeam,
               opponent = train$AwayTeam,
               home = 1
               ),
    # Calculate Away Attack rating as Defense,
    data.frame(goals = train$FTAG,
               team = train$AwayTeam,
               opponent = train$HomeTeam,
               home=0)) %>%
  glm(goals ~ team + home + opponent, family=poisson(link=log),data=.)
summary(poisson_model)
```

```
##
## Call:
## glm(formula = goals ~ team + home + opponent, family = poisson(link = log),
##     data = .)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1853  -1.3125  -0.1513   0.5785   3.4929
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    0.133021   0.233070   0.571 0.568181
## teamAkhisar Belediyespor       0.179669   0.198345   0.906 0.365019
## teamAlanyaspor                 0.364178   0.203146   1.793 0.073022 .
## teamAnkaragucu                 0.239689   0.212815   1.126 0.260048
## teamAntalyaspor                0.204460   0.195765   1.044 0.296292
## teamBesiktas                   0.584266   0.192757   3.031 0.002437 **
## teamBucaspor                   0.068456   0.261092   0.262 0.793174
## teamBursaspor                  0.157178   0.197228   0.797 0.425489
## teamBuyuksehyr                 0.422512   0.194852   2.168 0.030130 *
## teamDenizlispor               -0.010876   0.334167  -0.033 0.974036
## teamElazigspor                -0.015320   0.227970  -0.067 0.946422
## teamErciyesspor                0.032201   0.260822   0.123 0.901743
## teamErzurum BB                 0.089433   0.263072   0.340 0.733889
## teamEskisehirspor              0.144877   0.204409   0.709 0.478473
## teamFenerbahce                 0.613082   0.192399   3.187 0.001440 **
## teamGalatasaray                0.596339   0.192284   3.101 0.001926 **
## teamGaziantep                  0.163430   0.200472   0.815 0.414944
## teamGenclerbirligi             0.163402   0.197349   0.828 0.407679
## teamGoztep                     0.163749   0.215702   0.759 0.447767
## teamKarabukspor                0.034556   0.203307   0.170 0.865033
## teamKasimpasa                  0.360171   0.194414   1.853 0.063940 .
## teamKayserispor                0.084591   0.196823   0.430 0.667355
## teamKonyaspor                  0.099795   0.198339   0.503 0.614856
## teamManisaspor                 0.270786   0.250152   1.082 0.279036
## teamMersin Idman Yurdu        -0.092061   0.237554  -0.388 0.698359
## teamOrduspor                  -0.001261   0.265468  -0.005 0.996211
## teamOsmanlispor                0.297070   0.206861   1.436 0.150979
## teamRizespor                   0.212717   0.201909   1.054 0.292098
## teamSivasspor                  0.318154   0.195894   1.624 0.104352
## teamTrabzonspor                0.420734   0.193860   2.170 0.029984 *
## teamYeni Malatyaspor           0.383345   0.210137   1.824 0.068113 .
## home                           0.257859   0.027695   9.311  < 2e-16 ***
## opponentAkhisar Belediyespor -0.238326   0.152915  -1.559 0.119102
## opponentAlanyaspor           -0.178268   0.160332  -1.112 0.266194
## opponentAnkaragucu           -0.018557   0.164547  -0.113 0.910210
## opponentAntalyaspor          -0.149122   0.149570  -0.997 0.318762
## opponentBesiktas             -0.435131   0.153469  -2.835 0.004578 **
## opponentBucaspor              0.053613   0.199892   0.268 0.788538
## opponentBursaspor            -0.301942   0.152308  -1.982 0.047429 *
## opponentBuyuksehyr           -0.515850   0.155917  -3.308 0.000938 ***
## opponentDenizlispor          -0.126788   0.263752  -0.481 0.630723
## opponentElazigspor           -0.119638   0.174788  -0.684 0.493676
## opponentErciyesspor          -0.072141   0.205301  -0.351 0.725295
## opponentErzurum BB           -0.460126   0.229772  -2.003 0.045227 *
## opponentEskisehirspor        -0.321108   0.161349  -1.990 0.046574 *
## opponentFenerbahce           -0.518270   0.154399  -3.357 0.000789 ***
## opponentGalatasaray          -0.401158   0.152120  -2.637 0.008362 **
## opponentGaziantep            -0.132554   0.154055  -0.860 0.389553
## opponentGenclerbirligi       -0.242113   0.151947  -1.593 0.111069
## opponentGoztep               -0.350218   0.174188  -2.011 0.044370 *
## opponentKarabukspor          -0.104768   0.154174  -0.680 0.496793
## opponentKasimpasa            -0.139721   0.149367  -0.935 0.349571
## opponentKayserispor          -0.142273   0.149075  -0.954 0.339896
## opponentKonyaspor            -0.385980   0.153822  -2.509 0.012098 *
## opponentManisaspor           -0.122901   0.208150  -0.590 0.554891
## opponentMersin Idman Yurdu   -0.084948   0.178613  -0.476 0.634360
## opponentOrduspor             -0.215037   0.210802  -1.020 0.307685
## opponentOsmanlispor          -0.258102   0.165922  -1.556 0.119812
## opponentRizespor             -0.189574   0.156380  -1.212 0.225410
## opponentSivasspor            -0.179199   0.151422  -1.183 0.236635
## opponentTrabzonspor          -0.382152   0.152377  -2.508 0.012144 *
## opponentYeni Malatyaspor     -0.314992   0.173874  -1.812 0.070046 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4859.7  on 3903  degrees of freedom
## Residual deviance: 4469.8  on 3842  degrees of freedom
## AIC: 11521
##
## Number of Fisher Scoring iterations: 5
```

The above code shows that there is a Home "**Estimate**", which means home teams have an advantage - as mentioned in the above sections. They are more likely to score than the away team.

$$e^{0.258238581} = 1.30$$

It can be seen that home teams generally score **1.30** times more likely goals than the away teams.

I demonstrate this model with 2 different teams: **Goztep** and **Besiktas**:

```
# Set Home and Away Team from the Data set
phome = "Goztep"
paway = "Besiktas"
```

The above charts and data show that soccer has a low probability of very high scores. In the "Turkish Super Lig", the maximum goals scored in the league was seven. A heat map chart will reveal all goal probabilities with the goal limit of seven:

The next step is to get the Poisson probabilities for the possible "goals scored" values, using the mean as the rate parameter, since we know that the mean of a Poisson distribution is also the event rate μ . R's **dpois** which takes as input the values for the number of successes and expected number of events, and returns the probability, is used to accomplished this.

Also, **dpois** is called density, distribution function, quantile function and random generation for the Poisson distribution with parameter lambda.

I start with predicting the home goal and away goals. "**Home**" has a value of 1 and "**Away**" has 0, taking into consideration the home advantage. Also, away and home goals are represented by μ in the Poisson model formula:

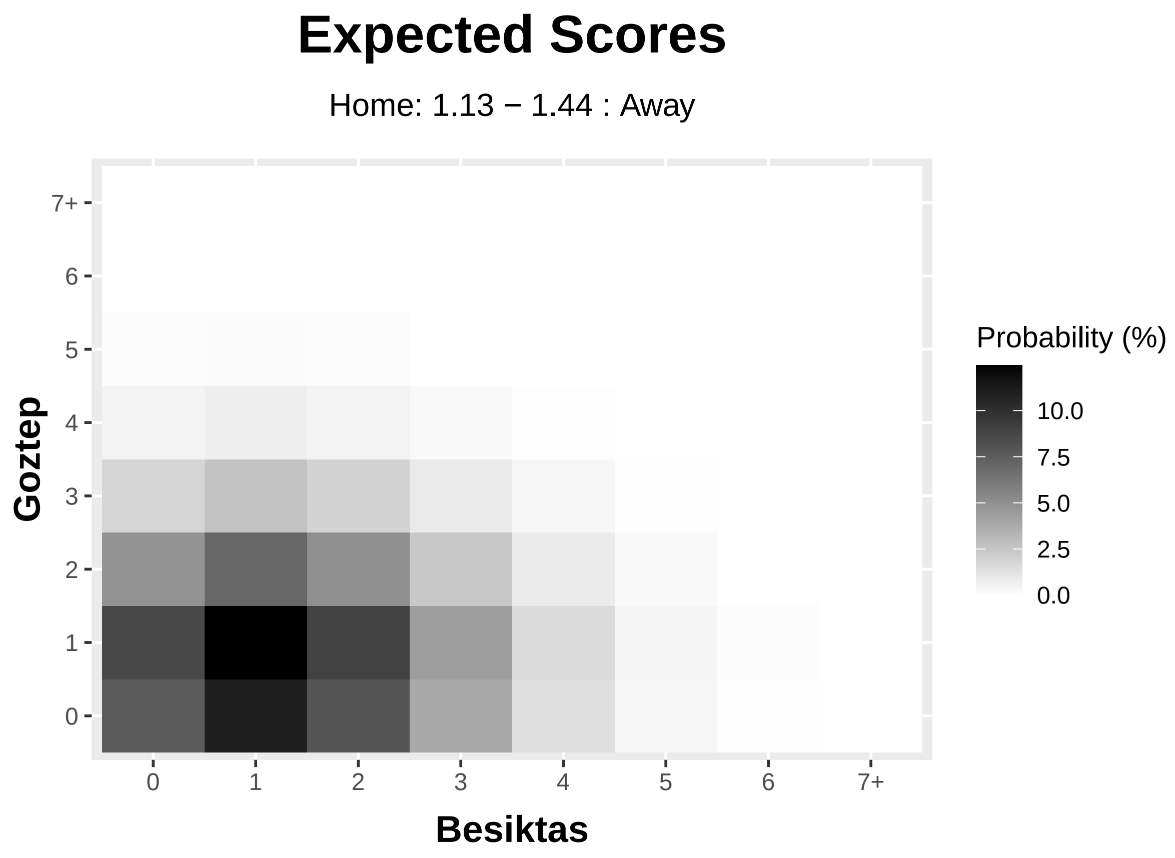
```r
# Predict Home Goal
homeXg <- predict(poisson_model,
              data.frame(home=1, team= phome,
                         opponent=paway)
              , type="response")
# Predict Away Goal
awayXg <- predict(poisson_model,
              data.frame(home=0, team=paway,
                         opponent=phome)
              , type="response")
```

For visualization I create a "**Expected Scores Heat Map**":

```r
# Create a function to see the probability of the score distribution.
ScoreGrid <- function(homeXg,awayXg){
  # Get the score data
  A <- as.numeric()
  B <- as.numeric()
  # Limit the score with 7 goals
  for(i in 0:6) {
    A[(i+1)] <- dpois(i,homeXg)
    B[(i+1)] <- dpois(i,awayXg)
  }
  # Built the grid for 7 goals
  A[8] <- 1 - sum(A[1:7])
  B[8] <- 1 - sum(B[1:7])
  name <- c("0","1","2","3","4","5","6","7+")
  zero <- mat.or.vec(8,1)
  C <- data.frame(row.names=name, "0"=zero, "1"=zero, "2"=zero, "3"=zero, "4"=zero,
                  "5"=zero, "6"=zero, "7+"=zero)
  for(j in 1:8) {
    for(k in 1:8) {
      C[j,k] <- A[k]*B[j]
    }
  }
  colnames(C) <- name
  return(round(C*100,4))
}

# Create a Score Heat Map to see the probability of the full time scores
ScoreHeatMap <- function(home,away,homeXg,awayXg){
  adjustedHome<-as.character(sub("_", " ", home))
  adjustedAway<-as.character(sub("_"," ",away))
  df <- ScoreGrid(homeXg,awayXg)
  # Use ScoreGrid function to create a heat map
  df %>%
    as_tibble(rownames = all_of(away)) %>%
    pivot_longer(cols = -all_of(away),
                 names_to = home,
                 values_to = "Probability") %>%
    mutate_at(vars(all_of(away), home),
              ~forcats::fct_relevel(.x, "7+", after = 7)) %>%
    # Use black and white colors to see the strong and weak probability
    ggplot() +
    geom_tile(aes_string(x=all_of(away), y=all_of(home), fill = "Probability")) +
    scale_fill_gradient2(mid="white", high = "black")+
    theme(plot.margin = unit(c(0.5,0.5,0.5,0.5),"cm"),
          plot.title = element_text(size=20,hjust = 0.5,face="bold",vjust =4),
          plot.subtitle = element_text(size=12,hjust = 0.5,vjust=4),
          axis.title.x=element_text(size=14,vjust=-0.5,face="bold"),
          axis.title.y=element_text(size=14, vjust =0.5,face="bold")
    )+
    labs(x=adjustedAway,y=adjustedHome,fill='Probability (%)')+
    ggtitle(label = "Expected Scores", subtitle = paste("Home:",round(homeXg,2),"-",round(awayXg,2),":

}
# Show the Heat Map
ScoreHeatMap(phome, paway, homeXg,awayXg)
```



As predicted above, and confirmed by the chart, with 12.5% probability the “**Goztep**” team has **1.13** score probability, while “**Besiktas**” team has **1.44**. So We might say that this match will end with a score of 1-1.

Also, with **10%** probability, **Besiktas** will win the game with a score of 0-1. The chart also shows the other probabilities for a lower ratio than the score of 1-1.

I now calculate the predictions of the result of the games (**FTR**), using the **Poisson** model.

```
# Predict Match Function - Set max goal 7
predict_match <- function(foot_model, HomeTeam, AwayTeam, max_goals=7){
  # Set Mu For Poisson Distribution
  home_goals_avg <- predict(foot_model,
                            data.frame(home=1, team=HomeTeam,
                                       opponent=AwayTeam), type="response")
  away_goals_avg <- predict(foot_model,
                            data.frame(home=0, team=AwayTeam,
                                       opponent=HomeTeam), type="response")
  # dpois(x, mu) is the probability of x successes in a period when the expected number of events is mu
  dpois(0:max_goals, home_goals_avg) %o% dpois(0:max_goals, away_goals_avg)
}
# Store the data generated by function
teams <- predict_match(poisson_model, phome, paway, max_goals=7)

# Store the results
H <- sum(teams[lower.tri(teams)])
D <- sum(diag(teams))
A <- sum(teams[upper.tri(teams)])

# Print the winning chance results by percentage
print (c(paste0("Home: %",round(H,digits=4)*100),
         paste0("Draw: %",round(D,digits=4)*100),
         paste0("Away: %",round(A,digits=4)*100)))
```

```
## [1] "Home: %29.49" "Draw: %26.17" "Away: %44.32"
```

The home team has a lower chance to win the match, and away team has a higher chance to win.

The actual final result of the above match was 1-2. Besiktas won the match, and with it the cup of the "**Turkish Super Lig**". In that match, Besiktas won by penalty, which was not taken into account in the expected score, because scores that comes from penalties are random scores - unrelated to the used data set. Thus, the above model's prediction can be considered successful.

I calculate the model's actual accuracy for the entire data set in the next section[3.6].

3.3 Data Manipulation

I manipulate the data set in order to achieve predictions with based on two probabilities, which are Home Wins **(H)** and Not Home Wins **(NH)**. I change the Full Time Result column from **A** and **D** to **NH**.

I also need to factorize the data for working on with **Binomial Models**.

```
##         HomeTeam            AwayTeam          FTHG          FTAG      FTR
##  Antalyaspor: 125   Galatasaray: 125   1      :685   1      :794   H : 994
##  Galatasaray: 124   Kayserispor: 125   2      :549   0      :683   NH:1189
##  Trabzonspor: 122   Besiktas   : 123   0      :500   2      :430
##  Fenerbahce : 121   Fenerbahce : 122   3      :286   3      :189
##  Kasimpasa  : 120   Trabzonspor: 122   4      :111   4      : 62
##  Besiktas   : 119   Antalyaspor: 120   5      : 39   5      : 18
##  (Other)    :1452   (Other)    :1446   (Other): 13   (Other):  7
##  HTHG     HTAG     HTR          B365H            B365D           B365A
##  0:1114   0:1303   H : 720   Min.   : 1.120   Min.   :3.000   Min.   : 1.170
##  1: 755   1: 661   NH:1463   1st Qu.: 1.730   1st Qu.:3.300   1st Qu.: 2.500
##  2: 249   2: 171             Median : 2.100   Median :3.400   Median : 3.400
##  3:  56   3:  41             Mean   : 2.448   Mean   :3.664   Mean   : 4.056
##  4:   8   4:   4             3rd Qu.: 2.750   3rd Qu.:3.750   3rd Qu.: 4.750
##  5:   1   5:   2             Max.   :13.000   Max.   :7.500   Max.   :21.000
##            6:   1
##       BWH              BWD             BWA              IWH
##  Min.   : 1.120   Min.   :2.800   Min.   : 1.180   Min.   : 1.120
##  1st Qu.: 1.720   1st Qu.:3.250   1st Qu.: 2.550   1st Qu.: 1.730
##  Median : 2.150   Median :3.400   Median : 3.250   Median : 2.100
##  Mean   : 2.395   Mean   :3.599   Mean   : 3.875   Mean   : 2.349
##  3rd Qu.: 2.650   3rd Qu.:3.700   3rd Qu.: 4.500   3rd Qu.: 2.600
##  Max.   :11.500   Max.   :8.500   Max.   :17.000   Max.   :13.000
##
##       IWD             IWA              WHH              WHD
##  Min.   :2.700   Min.   : 1.170   Min.   : 1.100   Min.   :2.880
##  1st Qu.:3.200   1st Qu.: 2.500   1st Qu.: 1.750   1st Qu.:3.250
##  Median :3.300   Median : 3.150   Median : 2.150   Median :3.400
##  Mean   :3.529   Mean   : 3.735   Mean   : 2.384   Mean   :3.598
##  3rd Qu.:3.600   3rd Qu.: 4.300   3rd Qu.: 2.700   3rd Qu.:3.750
##  Max.   :8.600   Max.   :20.000   Max.   :12.000   Max.   :8.500
##
##       WHA              VCH              VCD             VCA
##  Min.   : 1.200   Min.   : 1.100   Min.   :2.900   Min.   : 1.170
##  1st Qu.: 2.500   1st Qu.: 1.750   1st Qu.:3.300   1st Qu.: 2.550
##  Median : 3.200   Median : 2.150   Median :3.500   Median : 3.300
##  Mean   : 3.802   Mean   : 2.441   Mean   :3.734   Mean   : 4.039
##  3rd Qu.: 4.400   3rd Qu.: 2.750   3rd Qu.:3.800   3rd Qu.: 4.750
##  Max.   :21.000   Max.   :12.000   Max.   :9.000   Max.   :23.000
##
```

3.4 Naive Bayes

Naive Bayes Classifier is based on the Bayes Theorem. The Bayes Theorem says the conditional probability of an outcome x can be computed using the conditional probability of the cause of the outcome C.

Using the prior probability, the posterior probability can be computed, which is the probability that event C will occur given that x has occurred. The Naive Bayes classifier uses the input variable to choose the class with the highest posterior probability. The algorithm is called naive because it makes an assumption about the distribution of the data. The distribution can be Gaussian, Bernoulli or Multinomial.

I create a model for comparing the results with the main model:

```r
naive_bayes_model<-naiveBayes(FTR ~ ., data = train)
naive_bayes_predictions<-predict(naive_bayes_model, newdata=test)
naive_bayes_accuracy=round(mean(naive_bayes_predictions==test$FTR),2)*100
table(naive_bayes_predictions,test[,5])
```

```
##
## naive_bayes_predictions   H  NH
##                      H   62  18
##                      NH  49 102
```

```
# Accuracy of our model
result <- mean(naive_bayes_predictions==test[,5])
# Add to the Table
all_rmse <- data.frame()
all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "Naive Bayes Model",
                                 Model_Accuracy = round(result*100,2)))
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71

3.5 Support Vector

A support vector machine (SVM) is a classification and regression algorithm. It operates by identifying a hyper plane which separates the classes in the data. A hyper plane is a geometric entity which has a dimension of 1 less than its surrounding (ambient) space. If an SVM is utilized to classify a two-dimensional dataset, it will perform it with a one-dimensional hyper place (a line), classes in 3D data will be separated by a 2D plane and Nth dimensional data will be separated by a N-1 dimension line. A SVM is also called a margin classifier because it draws a margin between classes.

Sometime classes cannot be separated by a straight line in the present dimension. An SVM is capable of mapping the data in higher dimension such that it becomes separable by a margin. SVMs are powerful in situations where the number of features (columns) is more than the number of samples (rows). It is also effective in high dimensions.

I try this model to compare with the main model. A SVM has several methods to compute the probability, which I all explore.

3.5.1 Radial Kernel SVM Model

The Radial kernel support vector machine is a good approach when the data is not linearly separable. The idea behind generating non-linear decision boundaries is that I need to perform some nonlinear transformations on the features X_i which transforms them into a higher dimensional space.

```
## SVM Radial Model
svm_model_radial <- svm(FTR ~ ., kernel = "radial", data=train)
summary(svm_model_radial)
```

```
##
## Call:
## svm(formula = FTR ~ ., data = train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  1178
##
##  ( 594 584 )
##
##
## Number of Classes:  2
##
## Levels:
##  H NH
```

```r
prediction_radial <- predict(svm_model_radial, newdata = test)
# Create confusion Matrix
cm <- confusionMatrix(prediction_radial, test$FTR )
# Accuracy of our model
result <- cm$overall[[1]]
# Add to the Table
all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "SVM_Model_Radial",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42

3.5.2 Linear Kernel SVM Model

The Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most commonly used kernels. It is mostly used when there are a Large number of Features in a particular Data Set.

```r
## SVM Linear Model
svm_model_Linear <- svm(FTR ~ .,kernel = "linear", data=train)
summary(svm_model_Linear)
```

```
##
## Call:
## svm(formula = FTR ~ ., data = train, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
```

```
## Number of Support Vectors:  716
##
##  ( 369 347 )
##
##
## Number of Classes:  2
##
## Levels:
##  H NH
```

```
prediction_linear <- predict(svm_model_Linear, newdata = test)
# Create confusion Matrix
cm <- confusionMatrix(prediction_linear, test$FTR )
# Accuracy of our model
result <- cm$overall[[1]]
# Add to the Table
all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "SVM_Model_Linear",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27

3.5.3 Polynomial 3th Degree Kernel SVM Model

In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models. Intuitively, the polynomial kernel looks not only at given features of input samples in order to determine their similarity, but also at combinations of these. In the context of regression analysis, such combinations are known as interaction features. The (implicit) feature space of a polynomial kernel is equivalent to that of polynomial regression, but without the combinatorial blow-up in the number of parameters to be learned. When the input features are binary-valued, then the features correspond to logical conjunctions of input features. For degree-d polynomials, the polynomial kernel is defined as:

$$K(x, y) = (x^T y + c)^d$$

where x and y are vectors in the input space, i.e. vectors of features computed from training or test samples and c >= 0 is a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial. When $c = 0$, the kernel is called homogeneous. Below is the result for degree $(d) = 3$;

```
## SVM Polynomial Model
svm_model_poli <- svm(FTR ~ .,kernel = "polynomial",degree = 3, data=train)
summary(svm_model_poli)
```

```
##
## Call:
## svm(formula = FTR ~ ., data = train, kernel = "polynomial", degree = 3)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  1669
##
##  ( 837 832 )
##
##
## Number of Classes:  2
##
## Levels:
##  H NH
```

```
prediction_poli <- predict(svm_model_poli, newdata = test)
# Create confusion Matrix
cm <- confusionMatrix(prediction_poli, test$FTR )
# Accuracy of our model
result <- cm$overall[[1]]
# Add to the Table
all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "SVM_Model_Polinominal_3th",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27
SVM_Model_Polinominal_3th	58.01

3.5.4 Polynomial 5th Degree Kernel SVM Model

The following code used degree = 5:

```
## SVM Polynomial 5th Degree Model
svm_model_poli_5 <- svm(FTR ~ .,kernel = "polynomial",degree = 5, data=train)
summary(svm_model_poli_5)
```

```
##
```

```
## Call:
## svm(formula = FTR ~ ., data = train, kernel = "polynomial", degree = 5)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  5
##      coef.0:  0
##
## Number of Support Vectors:  1715
##
##  ( 864 851 )
##
##
## Number of Classes:  2
##
## Levels:
##  H NH
```

```
prediction_poli_5 <- predict(svm_model_poli_5, newdata = test)
# Create confusion Matrix
cm <- confusionMatrix(prediction_poli_5, test$FTR )
# Accuracy of our model
result <- cm$overall[[1]]
# Add to the Table
all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "SVM_Model_Polinominal_5th",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27
SVM_Model_Polinominal_3th	58.01
SVM_Model_Polinominal_5th	55.41

3.5.5 Quadratic Discriminant Analysis Model

Quadratic Discrimination is the general form of Bayesian discrimination. Discriminant analysis is used to determine which variables discriminate between two or more naturally occurring groups. QDA does not assume that the covariance of each of the classes is identical. To estimate the parameters required in quadratic discrimination, more computation effort and data is required than in the case of linear discrimination. If there is not a large difference in the group covariance matrices, then the latter will perform as well as quadratic discrimination.

```r
## QDA Model - Prediction Model
test = data.frame(lapply(test, function(x) as.numeric(x)))
train = data.frame(lapply(train, function(x) as.numeric(x)))
qda.fit = qda(FTR ~.,data=train)
summary(qda.fit)
```

```
##         Length Class  Mode
## prior     2    -none- numeric
## counts    2    -none- numeric
## means    44    -none- numeric
## scaling 968    -none- numeric
## ldet      2    -none- numeric
## lev       2    -none- character
## N         1    -none- numeric
## call      3    -none- call
## terms     3    terms  call
## xlevels   0    -none- list
```

```r
qda.class=predict(qda.fit,test)$class
table(qda.class,test[,5]) # 5th column FTR
```

```
##
## qda.class   1   2
##         1  94  15
##         2  17 105
```

```r
# Accuracy of our model
result <- mean(qda.class==test[,5]) # 5th column FTR (Accuracy)

all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "QDA_Model",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27
SVM_Model_Polinominal_3th	58.01
SVM_Model_Polinominal_5th	55.41
QDA_Model	86.15

3.5.6 Linear Discriminant Analysis

Linear Discriminant Analysis is a generalization of Fisher's linear discriminant, which is a technique for dimensionality reduction. It a method used in statistics and other fields for finding a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or typically, for dimensionality reduction before later classification. LDA is closely related to analysis of variance (ANOVA) and regression analysis, which also attempt to express one dependent variable as a linear combination of other features or measurements.

```
## LDA - Prediction Model
lda.fit = lda(FTR ~ .,data=train)
summary(lda.fit)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   44     -none- numeric
## scaling 22     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
lda.pred=predict(lda.fit, test[-5,])
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class=predict(lda.fit,test)$class
table(lda.class,test[,5]) # 5th column FTR
```

```
##
## lda.class   1   2
##         1 110   1
##         2   1 119
```

```
# Accuracy of our model
result <- mean(lda.class==test[,5]) # 5th column FTR (Accuracy)

all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "LDA_Model",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27
SVM_Model_Polinominal_3th	58.01
SVM_Model_Polinominal_5th	55.41
QDA_Model	86.15
LDA_Model	99.13

3.6 Generalized Linear Model with Poisson Regression

A Poisson Regression model is a Generalized Linear Model (**GLM**) that is used to model count data and contingency tables. The output Y (**count**) is a value that follows the Poisson distribution. It assumes the logarithm of expected values (mean) that can be modeled into a linear form by some unknown parameters. As we predict scores above, we will predict full time result (**FTR**) with our test and train data.

```
## GLM - Probability Model with poisson Regression
glm.fit = glm(FTR ~ .,data=train, family=poisson(link=log))
summary(glm.fit)
```

```
##
## Call:
## glm(formula = FTR ~ ., family = poisson(link = log), data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.92486  -0.23295   0.00707   0.20184   0.56306
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.3678377  0.2480927   1.483   0.1382
## HomeTeam    -0.0002092  0.0020858  -0.100   0.9201
## AwayTeam    -0.0002377  0.0021155  -0.112   0.9105
## FTHG        -0.1766616  0.0215108  -8.213  < 2e-16 ***
## FTAG         0.1270310  0.0214114   5.933 2.98e-09 ***
## HTHG         0.0602027  0.0443397   1.358   0.1745
## HTAG        -0.0475302  0.0352024  -1.350   0.1770
## HTR          0.1250120  0.0688923   1.815   0.0696 .
## B365H       -0.0282457  0.1112326  -0.254   0.7995
## B365D       -0.0073222  0.1485675  -0.049   0.9607
## B365A       -0.0078022  0.0575902  -0.135   0.8922
## BWH          0.0834098  0.1422488   0.586   0.5576
## BWD          0.0073358  0.1461473   0.050   0.9600
## BWA          0.0132680  0.0581277   0.228   0.8194
## IWH         -0.0382237  0.1018935  -0.375   0.7076
## IWD         -0.0106968  0.1374694  -0.078   0.9380
## IWA         -0.0211456  0.0571747  -0.370   0.7115
## WHH          0.0135944  0.1035794   0.131   0.8956
## WHD          0.0334943  0.1470129   0.228   0.8198
## WHA         -0.0021295  0.0563645  -0.038   0.9699
## VCH         -0.0368487  0.1407121  -0.262   0.7934
## VCD          -0.0129680  0.1606455  -0.081   0.9357
## VCA           0.0060618  0.0587513   0.103   0.9178
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 325.15  on 1951  degrees of freedom
## Residual deviance: 123.45  on 1929  degrees of freedom
## AIC: 4729.5
##
## Number of Fisher Scoring iterations: 4
```

```
glm.prob=predict(glm.fit,test,type="response")
glm.prob = ifelse(glm.prob>0.5,1,0)
table(glm.prob,test[,5])
```

```
##
## glm.prob   1   2
##        1 111 120
```

```
# Accuracy of our model
result <- mean(glm.prob==test[,5])

all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "GLM_Poisson_Regression_Model",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27
SVM_Model_Polinominal_3th	58.01
SVM_Model_Polinominal_5th	55.41
QDA_Model	86.15
LDA_Model	99.13
GLM_Poisson_Regression_Model	48.05

3.7 Neural Networks

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain functions. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

```
## Neural Networks
train = data.Normalization (train,type="n4",normalization="column")
test = data.Normalization (test,type="n4",normalization="column")

set.seed(2105)
# 5 neurons hidden layer
neural_networks_model = neuralnet(FTR ~ HomeTeam+AwayTeam+FTHG+FTAG+HTR,
              data = train,hidden = 5,linear.output = FALSE)
output <- compute(neural_networks_model, train[,-5])
p1 <- output$net.result
prediction_neural_networks_train <- ifelse(p1>0.5, 1, 0)
table_train <- table(prediction_neural_networks_train, train$FTR)
table_train
```

```
##
## prediction_neural_networks_train    0    1
##                                0  883    0
##                                1    0 1069
```

```
# Accuracy of our model
sum(diag(table_train))/sum(table_train)
```

```
## [1] 1
```

```
# Confusion Matrix - Testing data
output <- compute(neural_networks_model, test[,-5])
p2 <- output$net.result
prediction_neural_networks_test <- ifelse(p2>0.5, 1, 0)
table_test <- table(prediction_neural_networks_test, test$FTR)
table_test
```

```
##
## prediction_neural_networks_test   0   1
##                               0  97   0
##                               1  14 120
```

```
result <- sum(diag(table_test))/sum(table_test)

all_rmse <- bind_rows(all_rmse,
                      data.frame(Linear_Model = "Neural Networks",
                                 Model_Accuracy = round(result*100,2)))
# Write the result on the table
all_rmse %>%
  kable() %>% kable_styling(font_size = 12, position = "center",
                            latex_options = c("HOLD_position"))
```

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27
SVM_Model_Polinominal_3th	58.01
SVM_Model_Polinominal_5th	55.41
QDA_Model	86.15
LDA_Model	99.13
GLM_Poisson_Regression_Model	48.05
Neural Networks	93.94

4 Results

With the code produced I predicted soccer scores for 2 teams, visualized the results and utilized a number of models to compare our training and test data, and finally identified the best way to generate the machine learning model which is producing the best prediction result for the soccer game full time results(**FTR**).

Linear_Model	Model_Accuracy
Naive Bayes Model	71.00
SVM_Model_Radial	84.42
SVM_Model_Linear	98.27
SVM_Model_Polinominal_3th	58.01
SVM_Model_Polinominal_5th	55.41
QDA_Model	86.15
LDA_Model	99.13
GLM_Poisson_Regression_Model	48.05
Neural Networks	93.94

All models produced reasonably good to very good predictions. The **LDA**, and **SVM models with linear kernel** have been able to predict 231 matches in test data by over 95%, whereas other models have their test accuracy in the range of 55- 95%.

4.1 Discussion

I used the Poisson Distribution for score prediction, which I demonstrated to be quite successful, but not accurate for the full time result predictions.

For an improvement I converted the full time results to a simpler representation: Home winning(H) or Not Home Winning(NH). The **Linear SVM Model**, **QDA Model**, **LDA Model** and **Neural Networks** models provided the best results for the **Full Time Result (FTR)**, able to distinguish between Home win (**H**)-Not Home win (**NH**) decision by more than **80%** accuracy.

5 Conclusion

The main focus of this project was to provide a work flow for predicting game outcomes, and to propose various alternative models for the task. The target of **the Data Science: Capstone - Soccer Prediction** was to find an algorithm which provides the score predictions and best model to predict Full Time Results with Linear Regression and several more advance models, which was achieved. In this exercise, I used various statistical models to predict whether a home team will win or not win.

The Poisson Distribution model is able to produce very close results for the score predictions, but is not suitable to predict the full time results.

Additional feature engineering and selection, and alternative fitting strategies can potentially increase performance and are worth pursuing.