Fall 2018

BUAN 6356: Business Analytics with R

A Report On

Analysis of Olympics History

THE UNIVERSITY OF TEXAS AT DALLAS

Submitted by

Akansha Guruprasad - AXG180053

Ashwin Rathore - AXR180015

Pragati Mishra – PXM180012

Rajashree Kumkar - RSK180000

Tanmay Jain - TXJ180001

Under the Guidance of

Prof. Sourav Chatterjee

Introduction

November 29, 2018

"The goal of the Olympic Movement is to contribute to building a peaceful and better world by educating youth through sport practiced without discrimination of any kind and in the Olympic spirit, which requires mutual understanding with a spirit of friendship, solidarity and fair play."

Table of Contents:

Introduction to Olympics Games:

Olympic Games have been the world's biggest sporting event for over some 100 years. The start of the Olympic Movement amid the twentieth and 21st hundreds of years has brought about a lot of changes to the Olympic Games. A portion of these modifications incorporate the making of the Winter Olympic Games for snow and ice sports, the Paralympic Games for competitors with a handicap, the Youth Olympic Games for competitors matured 14 to 18, the five Continental amusements (Pan American, African, Asian, European, and Pacific), and the World Games for games that are not challenged in the Olympic Games.

The modern Olympic Games are driving international sporting events highlighting summer and winter sports in which a huge number of competitors from around the globe take an interest in the competition. The Olympic Games are viewed as the world's foremost sports rivalry with in excess of

200 countries taking an interest. The Olympic Games are held like clockwork of 4 years, with the summer and winter Games alternating by occurring every four years but two years apart.

Here, we have a dataset for the Olympic Games. We have information about all the athletes who have won medals for every Olympic Games since the inaugural games of 1896 till the last Olympic Games held in 2016.

Outline:-

Our project is divided into several parts: -

1. Data Collection:- Content, libraries used and data importing.
2. Data Munging:- Initial Exploration of data. Removing/Filling missing null values based on a certain pattern observed in the missing data.
3. Data Exploration:- Visualize the data in the form of charts, graphs and table as it will help in choosing a better prediction algorithm.
4. Prediction:- Prediction of which sports athlete might win gold/silver/bronze medals using the algorithm chosen.
5. Conclusion:- Results we found using our predictive model and the issues we faced while interpreting the results.

Data Collection:

The data set has been collected from Kaggle.

https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results

There are two csv files that is being used in the project. One data set has all the data about the Olympics (athelete_events.csv) with reference to the sports person and the second data set (noc_regions.csv) has regions of where these sports men belong.

The file athlete_events.csv contains 271116 rows and 15 columns. Each row corresponds to an individual athlete competing in an Individual/Team Olympic event (athlete-events). The columns are the following:

1. ID - Unique number for each athlete
2. Name - Athlete’s name
3. Sex - M or F
4. Age - Integer
5. Height - In centimeters
6. Weight - In kilograms
7. Team - Team name
8. NOC - National Olympic Committee 3-letter code
9. Games - Year and season
10. Year - Integer
11. Season - Summer or Winter
12. City - Host city
13. Sport - Sport
14. Event - Event
15. Medal - Gold, Silver, Bronze, or NA.

The file noc_regions.csv contains 230 rows and 3 columns. Each row corresponds to a country. The columns are the following:

1. NOC
2. Country
3. Notes

Data Importing and Loading of Libraries:

R offers a variety of packages that we can use for analysis and prediction. The libraries that we are using in our project are:

```r
library(data.table)
library(Hmisc)
library(dplyr)
library(ggplot2)
library(leaps)
library(gains)
library(gplots)
library(caret)
```

- Library data.table is used to import data from .csv files. As the number of records is enormous in athlete_events.csv file, we are using fread() function available in data.table package as they perform significantly faster than the basic read.csv() function.
- Libraries Hmisc and dplyr are used here for data cleaning.
- Library ggplot2 and gplots will be used for data exploring.
- Libraries leaps, gains and caret are used for prediction and model evaluation.

Using the in-built functions available in these libraries, we read the .csv file.

```r
athlete_df <- fread("athlete_events.csv")
regions.df <- fread("noc_regions.csv")
```

The above code uses fread() function in order to load the data into R. We use this function as it is more convenient and fast. The controls such as separation (sep), column classes (colClasses) and nrow (number of rows) is automatically detected.

```r
colnames(regions.df) <- c("NOC","Country","Notes")
```

Data Munging:

Initial Exploration of the dataset:

From the information obtained from Kaggle site, we know that there are 271,116 records and 15 columns in athlete_events.csv file and 230 records and 3 columns in noc_regions.csv file.

(i) To check if all the data is loaded using dim() function:

```r
dim(athlete_df)
## [1] 271116     15
```

```
dim(regions.df)
## [1] 230   3
```

The dim() function tells us the dimension of the table i.e, the number of rows and columns. This function helps us to know how much data is present in the data set.

From the result of dim(), we can see that there are 271116 records and 15 columns in athlete_df dataframe and 230 records and 3 columns in regions.df dataframe. Hence, all the data has been loaded in the dataframe.

(ii) To get overview of the dataframe using glimpse() function:

```
glimpse(athlete_df)

## Observations: 271,116
## Variables: 15
## $ ID     <int> 1, 2, 3, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 7...
## $ Name   <chr> "A Dijiang", "A Lamusi", "Gunnar Nielsen Aaby", "Edgar ...
## $ Sex    <chr> "M", "M", "M", "M", "F", "F", "F", "F", "F", "F", "M", ...
## $ Age    <int> 24, 23, 24, 34, 21, 21, 25, 25, 27, 27, 31, 31, 31, 31,...
## $ Height <int> 180, 170, NA, NA, 185, 185, 185, 185, 185, 185, 188, 18...
## $ Weight <dbl> 80, 60, NA, NA, 82, 82, 82, 82, 82, 82, 75, 75, 75, 75,...
## $ Team   <chr> "China", "China", "Denmark", "Denmark/Sweden", "Netherl...
## $ NOC    <chr> "CHN", "CHN", "DEN", "DEN", "NED", "NED", "NED", "NED",...
## $ Games  <chr> "1992 Summer", "2012 Summer", "1920 Summer", "1900 Summ...
## $ Year   <int> 1992, 2012, 1920, 1900, 1988, 1988, 1992, 1992, 1994, 1...
## $ Season <chr> "Summer", "Summer", "Summer", "Summer", "Winter", "Wint...
## $ City   <chr> "Barcelona", "London", "Antwerpen", "Paris", "Calgary",...
## $ Sport  <chr> "Basketball", "Judo", "Football", "Tug-Of-War", "Speed ...
## $ Event  <chr> "Basketball Men's Basketball", "Judo Men's Extra-Lightw...
## $ Medal  <chr> NA, NA, NA, "Gold", NA, NA, NA, NA, NA, NA, NA, NA, NA,...

glimpse(regions.df)

## Observations: 230
## Variables: 3
## $ NOC     <chr> "AFG", "AHO", "ALB", "ALG", "AND", "ANG", "ANT", "ANZ"...
## $ Country <chr> "Afghanistan", "Curacao", "Albania", "Algeria", "Andor...
## $ Notes   <chr> "", "Netherlands Antilles", "", "", "", "", "Antigua a...
```

The glimpse() function is used to take a look of the data. This function gives a transpose version of the print i.e., columns runs down the page and data run across. It also shows the number of observations as well as the number of variables. It also shows the data type of each rows like int for integer, char for character and dbl for double.

Data Merging:

From the initial exploration, we can see that both the datframes have a common column - NOC. So, we can merge both these dataframes into one single dataframe which we can use for our analysis.

merge() function can be used to join the dataframes based on a common attribute. The result of this function is similar to an inner-join query as the ‘all’ parameter of this funtion is by default F.

```
mergd.athlete.df <- merge(athlete_df, regions.df, by="NOC")
Full.athlete.df <- mergd.athlete.df[,c(2,3,4,5,6,7,8,10,11,12,13,14,1,16,15,1
7)]
Full.athlete.df$Medal <- as.factor(Full.athlete.df$Medal)
```

After rearranging the column attributes, the glimpse of the dataframe which we will be using for further analysis is:

```
glimpse(Full.athlete.df)

## Observations: 270,741
## Variables: 16
## $ ID      <int> 502, 1076, 1101, 1745, 4628, 5285, 5582, 5678, 5679, 5...
## $ Name    <chr> "Ahmad Shah Abouwi", "Jammal-ud-Din Affendi", "Mohamma...
## $ Sex     <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",...
## $ Age     <int> NA, 28, NA, 17, 22, NA, 17, NA, NA, 22, NA, 25, 21, 27...
## $ Height  <int> NA, NA, NA, 156, NA, NA, NA, NA, NA, NA, NA, 160, 176,...
## $ Weight  <dbl> NA, NA, NA, 48, NA, 52, NA, NA, NA, NA, NA, 52, 78, NA...
## $ Team    <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis...
## $ Year    <int> 1956, 1936, 1948, 1980, 1964, 1972, 1936, 1948, 1948, ...
## $ Season  <chr> "Summer", "Summer", "Summer", "Summer", "Summer", "Sum...
## $ City    <chr> "Melbourne", "Berlin", "London", "Moskva", "Tokyo", "M...
## $ Sport   <chr> "Hockey", "Hockey", "Football", "Wrestling", "Wrestlin...
## $ Event   <chr> "Hockey Men's Hockey", "Hockey Men's Hockey", "Footbal...
## $ NOC     <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG", "AFG"...
## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanis...
## $ Medal   <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ Notes   <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", ""...
```

The total number of observations 270,767 and the number of columns i.e., variables is 16.

Data Cleaning:

From the glimpse() function we can see that there are few missing values (NA) in the dataset. These NA values should be removed in order to proceed with the data exploration and prediction.

In order to remove these null values, we first need to know how many null values are present in each columns.

Let's check for the missing values sapply() and is.na() function.

```
miss.val <- data.frame(miss.val = sapply(Full.athlete.df, function(x) +   sum
((is.na(x)))))
miss.val

##         miss.val
## ID             0
## Name           0
## Sex            0
## Age         9462
## Height     60083
## Weight     62785
## Team           0
## Year            0
## Season          0
## City            0
## Sport           0
## Event           0
## NOC             0
## Country        21
## Medal      230993
## Notes           0
```

The sapply() function gives the list for every element and sum() function sums up the NA values returned is.na() function.

In this dataframe, Age, Height, Weight, Country, Medal attributes have missing values. Let's take a look at the summary of the variables: age, height and weight.

```
summary(Full.athlete.df$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.00   21.00   24.00   25.56   28.00   97.00    9462

summary(Full.athlete.df$Height)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   127.0   168.0   175.0   175.3   183.0   226.0   60083

summary(Full.athlete.df$Weight)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   25.00   60.00   70.00   70.71   79.00  214.00   62785
```

From the summary, it is clear that the mean of Age, Height and Weight is almost equal to the Median of Age, Height and Weight.

Hence, the distribution of Age, Height and Weight among athletes participating in the Olympics event is a normal distribution.

As the distribution is normal, the missing values in Age, Height and Weight can be replaced with the value of Median for Age, Height and Weight respectively.

Replacing the missing values with Median using impute function:

```
Full.athlete.df$Age <- as.numeric( impute( Full.athlete.df$Age, median))
Full.athlete.df$Height <- as.numeric( impute( Full.athlete.df$Height,   media
n))
Full.athlete.df$Weight <- as.numeric( impute( Full.athlete.df$Weight,   media
n))
```

Now, there are 21 values of countries that are missing. Let's check which are these countries and their medal tally.

```
country_NA <- filter(Full.athlete.df,is.na(Country))
unique(country_NA$Notes)
```

```
## [1] "Refugee Olympic Team" "Tuvalu"               "Unknown"
```

```
table(country_NA$Notes)
```

```
## 
## Refugee Olympic Team               Tuvalu              Unknown 
##                   12                    7                    2
```

```
table(country_NA$Medal)
```

```
## 
## Bronze   Gold Silver 
##      0      0      0
```

Using the filter() function, we found that the athletes of these 21 missing values belong to the Refugee Olympic Team, Tuvalu or Unknown. As the no of missing records is very small compared to the no of records we have and as their medal tally is also 0, they can be omitted.

But we cannot use na.omit() function on the entire dataset as the missing values of Medals are important for prediction.

So, we first fill the missing values of Medal with “Dnw” - “Did not win” and then omit the missing values in rest of the dataset.

As Full.athlete.df$Medal is a factor, we will have to change the label of NA level as given below to remove NAs in Medal and then use na.omit() function on the dataset.

```
Full.athlete.df$Medal <- factor( Full.athlete.df$Medal, exclude = NULL,
                          levels = c("Bronze", "Gold", "Silver", NA),
                          labels = c("Bronze", "Gold", "Silver", "Dnw"))

Full.athlete.df <- na.omit(Full.athlete.df)
miss.val <- data.frame(miss.val=sapply(Full.athlete.df, function(x) + sum((is
.na(x)))))
miss.val
```

```
##         miss.val
## ID             0
## Name           0
## Sex            0
## Age            0
## Height         0
## Weight         0
## Team           0
## Year           0
## Season         0
## City           0
## Sport          0
## Event          0
## NOC            0
## Country        0
## Medal           0
## Notes           0
```

Full.athlete.df dataframe is now free of missing values. And this dataframe is now ready for data exploration.

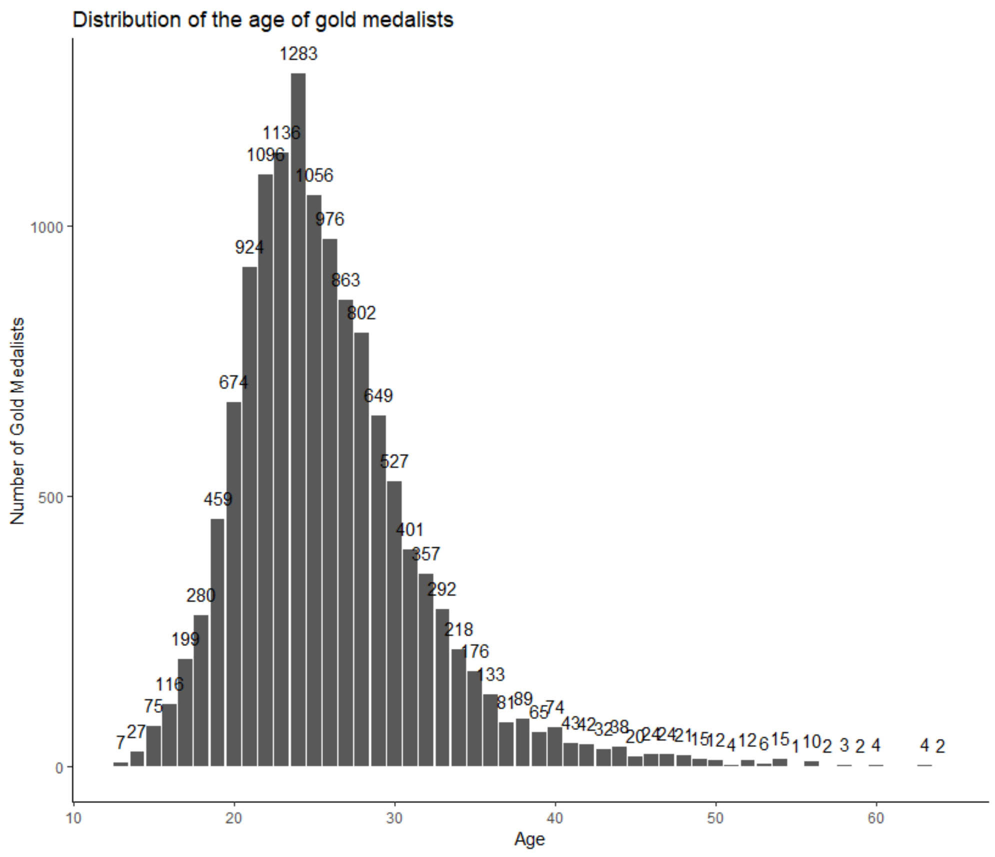
Data exploration

Distribution of the age of Medalists:

(a) Gold Medalists:

Let's first explore the distribution of age for athletes winning gold medal.

```
gold.medals.df <- filter( Full.athlete.df, Medal == "Gold")
ggplot(data=gold.medals.df, aes(x=Age)) + geom_bar() +
   geom_text(stat='count', aes(label=..count..), vjust=-1) +
   labs(x = "Age", y = "Number of Gold Medalists",
        title = "Distribution of the age of gold medalists") +
   theme(plot.title = element_text(size=10, hjust = 0.5),
         axis.title = element_text(size=10),
         axis.text = element_text(size=10)) +
   theme_classic()
```



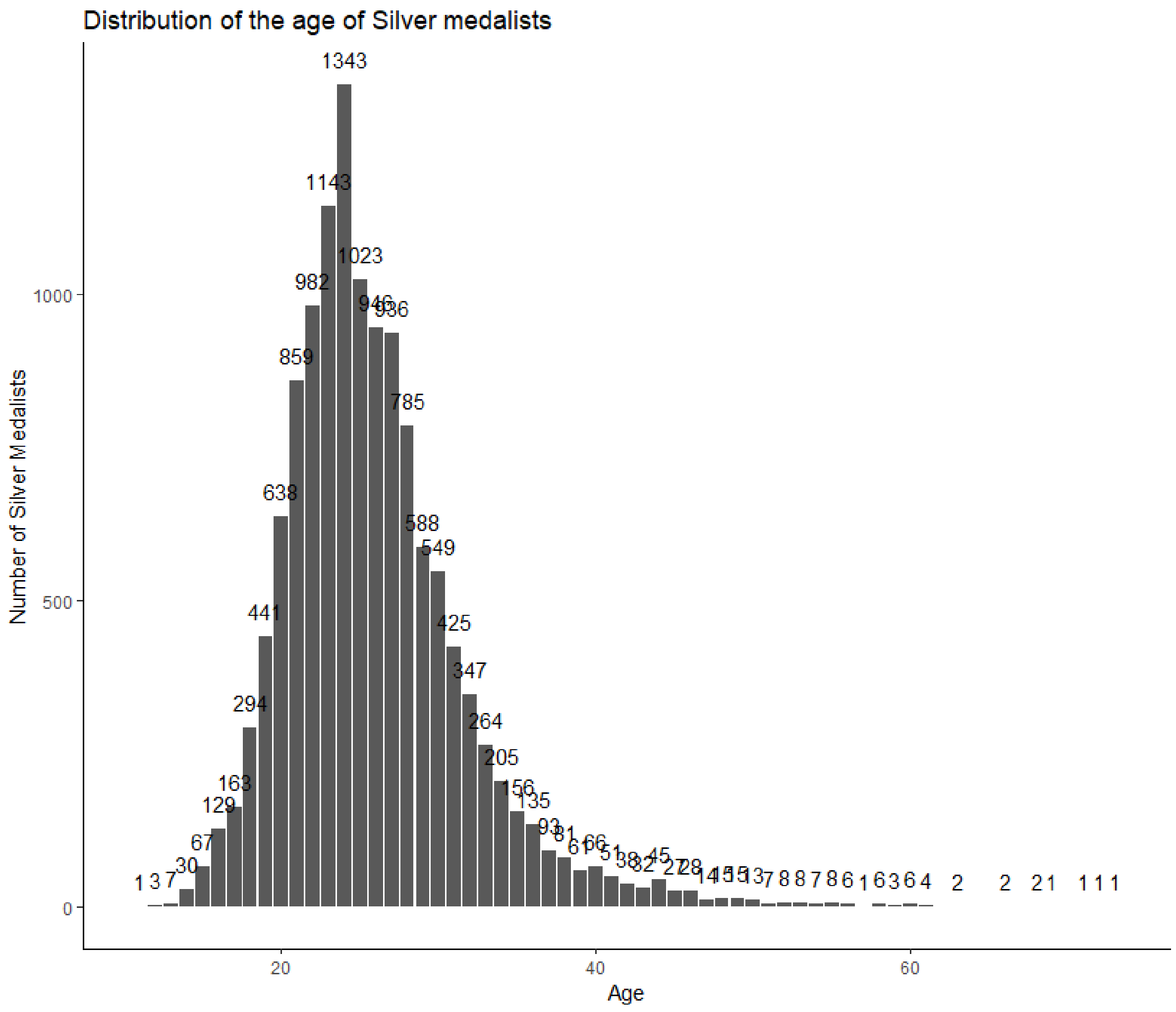
Observations:

We can see that the distribution of gold medals is normal.
There are in total 13371 Gold medals earned by players with majority of gold medals were earned by age group 20 to 30.
Players with age 24 has earned the maximum number of gold medals i.e. 1283. Another interesting thing to note is that there are many Gold Medallists who have age more than 50.

(b) Silver medalists:

Let's now explore the distribution of age for athletes winning silver medal:

```r
silver.medals.df <- filter(Full.athlete.df, Medal == "Silver")
 ggplot(data=silver.medals.df, aes(x=Age)) + geom_bar() +
   geom_text(stat='count', aes(label=..count..), vjust=-1) +
   labs(x = "Age", y = "Number of Silver Medalists",
        title = "Distribution of the age of Silver medalists") +
   theme(plot.title = element_text(size=10),
         axis.title = element_text(size=10),
         axis.text = element_text(size=10)) +
   theme_classic()
```

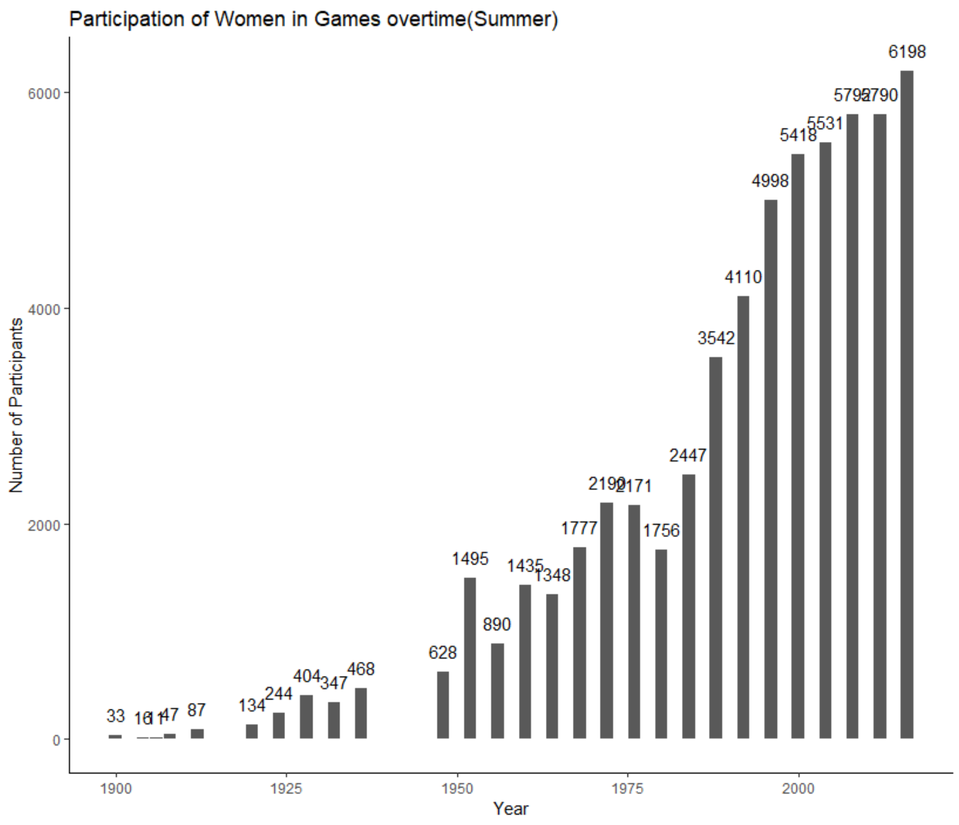


Observations:
The relation between age and number of silver medalists is normally distributed.
Cumulatively, there are 13112 Silver medals earned in Olympics with majority of silver medals were earned by age group 20-30.
From the graph above we can clearly see that players with age more than 60 are also earning Silver medals in Olympic games.
Also, Comparing the previous graph (Number of Gold Medallists and Age) and the above graph (Number of Silver Medallists and Age), We can say that there are more number of Silver medals earned when age is more than 60 group than gold.

Distribution of Women in Summer and Winter Games:

Let’s explore the participation of women in Summer games.

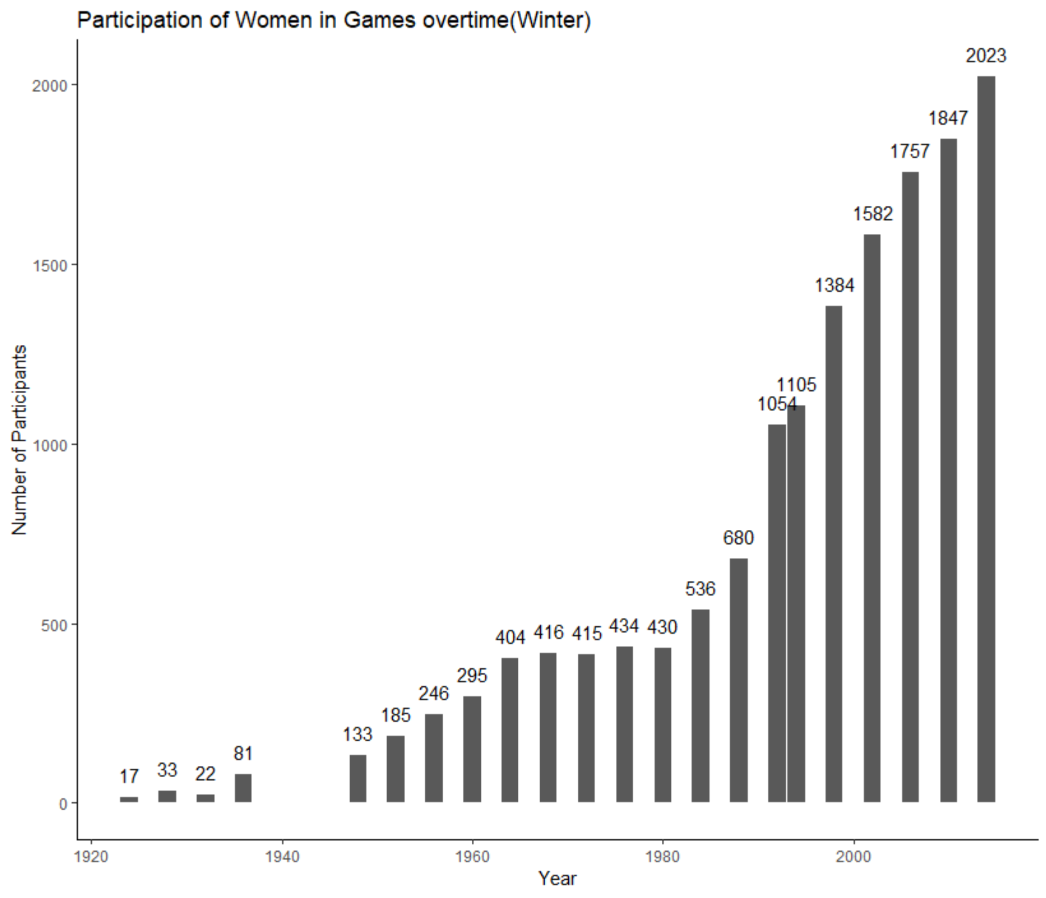
```
women.sum.olym.df <- Full.athlete.df[ which( Full.athlete.df$Sex == "F"
                                           & Full.athlete.df$Season == "Summer
") , ]
ggplot(data=women.sum.olym.df, aes(x=Year)) +
   geom_bar() +
   geom_text(stat='count', aes(label=..count..), vjust=-1) +
   labs(x = "Year", y = "Number of Participants",
        title = "Participation of Women in Games overtime(Summer)") +
   theme(plot.title = element_text(size=10),
         axis.title = element_text(size=10),
         axis.text = element_text(size=10)) +
   theme_classic()
```



Observations:
The above graph indicates the relation between Number of female participants in Summer season and the year.
There are in total 53262 women participated in summer Olympic games over a time period. From the graph above, we can say that the female players participation in summer Olympic games increased over the years. And the maximum womens participation in summer games was observed after year 2000, max being 6198.
Also, there was no participation for the years 1916, 1940 - 1944. This was because the olympic games was cancelled for these years, following the outbreak of World Wars.

Let's explore the participation of women in Winter games.

```
women.win.olym.df <- Full.athlete.df[ which( Full.athlete.df$Sex == "F"
                                             & Full.athlete.df$Season == "Wi
nter") , ]
 ggplot(data=women.win.olym.df, aes(x=Year)) +
   geom_bar() +
   geom_text(stat='count', aes(label=..count..), vjust=-1) +
   labs(x = "Year", y = "Number of Participants",
        title = "Participation of Women in Games overtime(Winter)") +
   theme(plot.title = element_text(size=10),
         axis.title = element_text(size=10),
         axis.text = element_text(size=10)) +
   theme_classic()
```



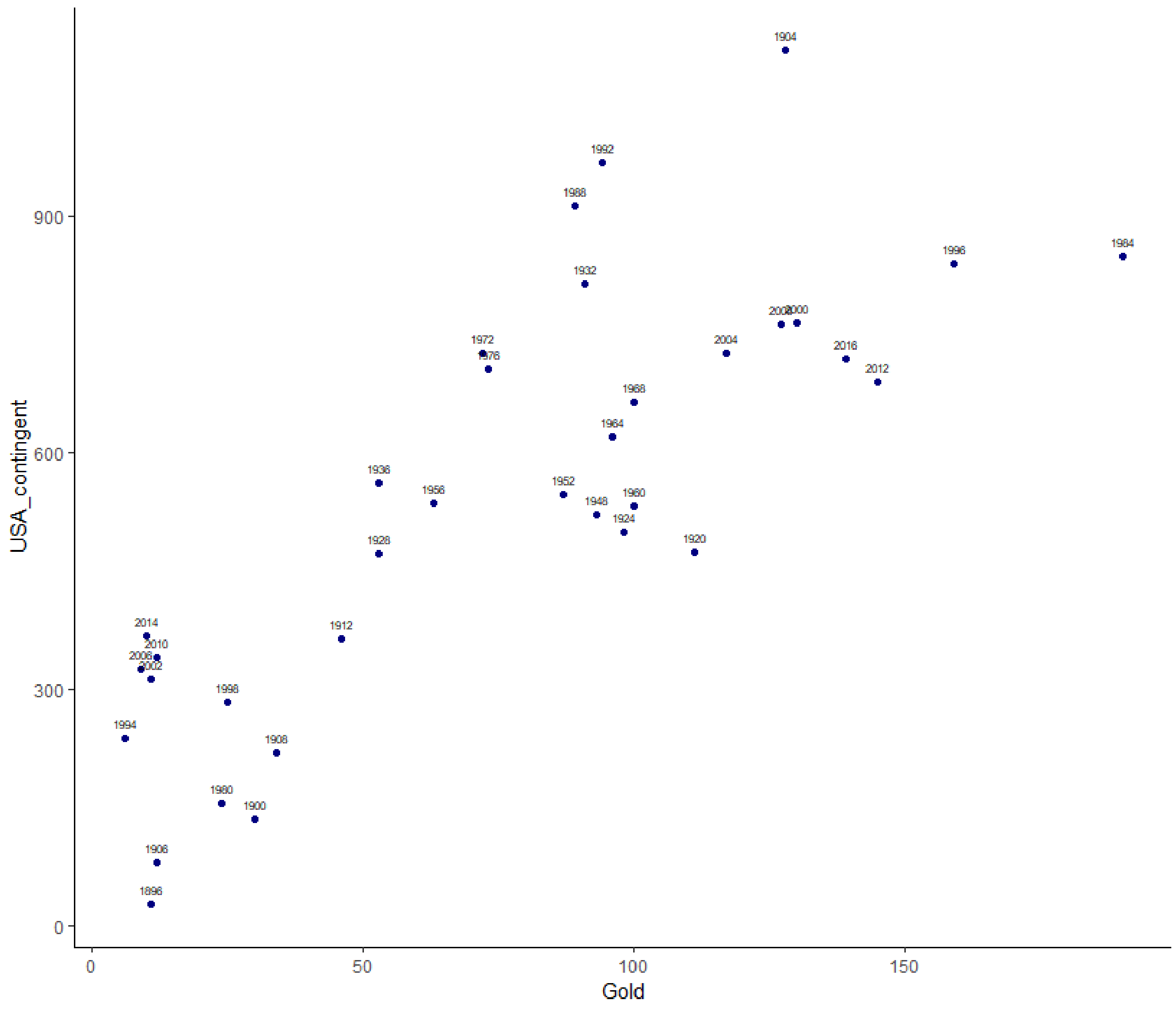
The participation of the women in the Olympics in winter is increasing in years. Thus, showing more participation of women in Olympics during the years.

Observations:
The above graph indicates the relation between Number of female participants in Winter season and its corresponding year.
There are in total 17079 women participated in winter Olympic over a time period which is far less than number of women participated in summer Olympic games.
From the graph above, we can say that the female players participation in winter Olympic games increased over the years. And the maximum number of women participation in winter games was observed after year 2000, max being 2023.
Similar to the the graph observed for summer games, we can see that there was no participation for the years between 1936-1944. This was because the olympic games was cancelled because of the outbreak of World Wars.

Relation between medals and size of contingent for USA across all years

As we analyze data for each Olympics event from the 1896 till the last one in 2016, we realize that there is a constant increase in the size of contingent that each country sends for the games. Here, we will determine whether the size of contingent has any relation with the number of medals that a country wins over the period of time. USA being the best performing nation in comparison to other nations overall, we will analyze data on USA for now.

```
gold.medals.usa.df <- filter(gold.medals.df, NOC == "USA")
gm.usa.df <- data.frame(head(sort(table(gold.medals.usa.df$Event),decreasing
= T),15))
colnames(gm.usa.df) <- c("Sport Event","No of Gold Medals")

full.usa.df <- filter(Full.athlete.df, NOC == "USA")
goldcount <- as.data.frame(table(gold.medals.usa.df$Medal=="Gold",gold.medals
.usa.df$Year))
players <- as.data.frame(table(full.usa.df$Year))
final.usa.df <- data.frame(Year = goldcount$Var2, Gold = goldcount$Freq, USA_
contingent = players$Freq)

Golds = goldcount$Freq
USA_Contingent_Size = players$Freq

ggplot(final.usa.df, aes(x= Gold, y= USA_contingent,  label=Year))+
  geom_point(color = "navy") +geom_text(aes(label=Year),hjust=0.5, vjust=-1,
size = 2)
```



Observations:
The plot here determines the number of medals USA has won in every year.
As we can see there is a positive linear relationship between the size of contingent that US sends and the number of medals they win in every game. With every Olympics game, the size the of contingent generally increases for USA and with that there is an increase in the number of medals that they win.
There are also a few exceptions like in 1920 and 1924 when, even with a smaller contingent size relatively, USA managed to win good number of medals.

Correlation between age, height and weight in different events of Olympics.

(a) Correlation between age, height and weight in athletics:

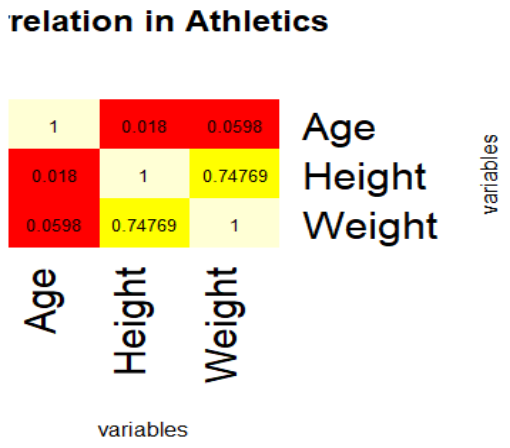
Filtering the data for the sport of athletics and pulling out only age, height and weight variables..

```
data.for.athletics.df <- filter(Full.athlete.df,
                                Sport == "Athletics")
```

```
correlation.in.athletics.df <- data.for.athletics.df[,c(4,5,6)]
```

Performing correlation on these three variables age, height and weight. Also, plotting the heat map for a better view of the data.

```
cor.olympics.athletics <- cor(correlation.in.athletics.df)

heatmap.2(cor(correlation.in.athletics.df), Rowv = FALSE, Colv = FALSE,
          dendrogram = "none",cellnote = round(cor(correlation.in.athletics.d
f),5),
          notecol = "black", key = FALSE, trace = 'none', margins = c(10,10),
          main = "correlation in Athletics",xlab = "variables",ylab="variable
s")
```



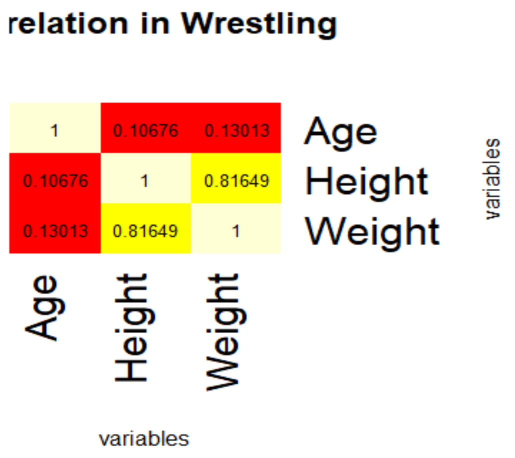
Interpretation:
The correlation shows that there is strong positive correlation between height and weight. Athletics is the sport in which these two variables play a very important role in the Olympics that is why they are strongly correlated.

(b) Correlation between age, height and weight in wrestling:
Filtering the data for the sport of Wrestling and pulling out only age, height and weight variables.

```
data.for.wrestling.df <- filter(Full.athlete.df,
                                Sport == "Wrestling")

correlation.in.wrestling.df <- data.for.wrestling.df[,c(4,5,6)]
```

Performing correlation on these three variables age, height and weight and plotting heat map.

```
cor.olympics.wrestling <- cor(correlation.in.wrestling.df)
heatmap.2(cor(correlation.in.wrestling.df), Rowv = FALSE, Colv = FALSE,
          dendrogram = "none",cellnote = round(cor(correlation.in.wrestling.d
f),5),
          notecol = "black", key = FALSE, trace = 'none', margins = c(10,10),
          main = "correlation in Wrestling",xlab = "variables",ylab="variable
s")
```



Interpretation:
The correlation shows that there is strong positive correlation between height and weight. Wrestling is the sport in which these two variables play a very important role in the Olympics that is why they are strongly correlated.

Model Prediction

There various types of events being held in Olympics. We have divided our model into predicting 3 types of situation which are possible when considering participation.

(i) Events having individual participation of athletes from different countries.

(ii) Events having team participation of athletes representing their countries like Hockey, Football

(iii) Events having team participation of athletes with countries sending multiple teams in a single event like Tennis Doubles, Athletics Relay.

But what are we trying to predict?

Athletes participate in Olympic games to win for a medal. Here, we are trying to predict the possibility of an athlete or team to win in an event.

How do we do it?

There are various algorithms which can be used for prediction. But before that let's choose the column having relevant information as all the columns in the dataframe are not required for prediction.

The independent variables available in the dataset of an athlete which we can use for prediction are Age, Weight, Height and Country. Age, Height and Weight are important factors, but it varies from one event to another. For gymnasts, shorter height is preferable but for events like basketball and volleyball, taller heights are preferred. Age and Weight could be an important factor in events like athletics and Wrestling respectively. We have seen this in our data exploration section as well where we were checking for correlation between these parameters.

Country can also be an important category. The facilities provided by the country for training an athlete in an event could increase the possibility of winning.

Hence, the predictors having relevant information in our model are Age, Weight, Height and Country.

The dependent variable, which we are predicting, is whether an athlete might win or not. It's value will be 1 if the athlete wins and 0 if the athlete loses i.e. we are predicting a binary outcome of win/lose.

As we are predicting a binary outcome, we have chosen Logistic regression as our algorithm for prediction.

- Why we cannot use Linear Regression for predicting our model: There should be a linear relationship between the independent variables and dependent variable i.e. with the increase in the value of independent variables, the value of dependent variable should increase. But as we have seen in the data exploration section, where we plotted the graph of Age vs Medal Winners, the relationship was not linear. The possibility of winning increases with age but, after a point, it starts decreasing.

  Also, linear regression can only be used on numerical variables. In our model, one of the independent(Country) varible is a categorical non-numeric variable. Hence for the above two reasons we cannot use linear regression for prediction.

- Why we cannot use KNN for predicting our model: KNN is best suitable for classification related prediction. We could use KNN for prediction as we are classifying whether an athlete might win or not. But it does not handle categorical variables. Also, it has an issue with normalization. It predicts based on the measure of the distance calculated. If the data is not normalized, the distance calculated could be biased towards a particular classification.

- Why we cannot use LDA for predicting our model: LDA can be one of the best algorithms that can be used for classification models when we use categorical variables as predictors and the predictors are normally distributed. However, Logistic Regression outperforms this algorithm as it is a very powerful classification method when it comes to predicting binary outcomes.

Also, we have not implemented dimension reduction techniques in our model as we do not have many independent variables as predictors.

Logistic Regression for prediction:

Logistic regression is a powerful statistical way of modeling a binary outcome (takes the value 0 or 1). In our case, the dependent variable has two possible values which is win/lose. Logistic regression combines both binomial and normal distribution to predict results. So, the independent variables can be binary, ordinal or continuous. Also, logistic regression provides a quantified value for the strength of the association adjusting for other variables i.e. it removes confounding effects.

(I) Prediction of Events having individual participation:

There are a lot of events in Olympics having individual participations. Let's look at a few of them:

(a) Prediction of winner in 100-meter Freestyle Men's Swimming:

Using Logistic regression, we will predict which athletes might win in 100-meter Freestyle Men's Swimming.

Step 1: Create the training data frame of all the participants of 100-meter free style swimming event excluding 2016 data only. (As 2016-year data will be used for prediction later)

```
swimming.df <- Full.athlete.df[which( Full.athlete.df$Sport == "Swimming"
                                      & Full.athlete.df$Event == "Swimming Me
n's 100 metres Freestyle"
                                      & Full.athlete.df$Year != 2016), ]
```

Step 2: To implement logistic regression, the outcome should be binary so converting the Medal variable into binary values.

```
for (i in seq_along(swimming.df$Medal)){
  if (swimming.df$Medal[i] == "Dnw") {
    swimming.df$Outcome[i] <- 0
  } else {
    swimming.df$Outcome[i] <- 1
  }
}
```

Step 3: Data is now ready to run the logistic regression on Outcome variable depending on Age, Weight, Height and NOC(nationality of Candidate).

```
swimming.log <- glm(Outcome ~ Age + Weight + Height + NOC, data = swimming.df
, family = "binomial")
options(scipen=999)
```

Step 4: Model is ready. Summarizing the results of model and analyzing it.

```
summary(swimming.log)

##
## Call:
## glm(formula = Outcome ~ Age + Weight + Height + NOC, family = "binomial",
##     data = swimming.df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.42443  -0.33308  -0.00006  -0.00004   2.78614
##
## Coefficients:
##                  Estimate   Std. Error z value Pr(>|z|)
## (Intercept)   -33.394249  8790.432515  -0.004  0.99697
## Age            -0.009682     0.044205  -0.219  0.82664
## Weight         -0.027713     0.027793  -0.997  0.31871
## Height          0.079928     0.030754   2.599  0.00935 **
## NOCALB          0.391324 15153.841708   0.000  0.99998
## NOCALG          0.506249 11371.747520   0.000  0.99996
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 580.81  on 1226  degrees of freedom
## Residual deviance: 375.72  on 1082  degrees of freedom
## AIC: 665.72
##
## Number of Fisher Scoring iterations: 19
```

(**NOTE**: We have not included all the coefficients that was shown in the summary in this report.)

Interpretation:
Results of summary show that the Height is the one of the significant variable on which the candidate’s ability to win depends.

Step 5: Predict the outcome on the trained model.

```
swimming.log.pred <- predict(swimming.log, swimming.df, type = "response")
```

Step 6: Create the confusion matrix to choose a cut-off value for this event.

```
swimming.cm <- t(table(swimming.df$Outcome , swimming.log.pred > 0.185, dnn =
c("Actual Class","Predictive Class")))

rownames(swimming.cm) <- colnames(swimming.cm)

confusionMatrix(swimming.cm, positive = "1", dnn = c("Actual Class","Predicti
ve Class"))

## Confusion Matrix and Statistics
##
##                 Actual Class
## Predictive Class    0    1
##                0 1065   33
##                1   84   45
##
##                Accuracy : 0.9046
##                  95% CI : (0.8868, 0.9205)
##     No Information Rate : 0.9364
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3861
##  Mcnemar's Test P-Value : 0.000003791
##
##             Sensitivity : 0.57692
##             Specificity : 0.92689
##          Pos Pred Value : 0.34884
##          Neg Pred Value : 0.96995
##              Prevalence : 0.06357
##          Detection Rate : 0.03667
##    Detection Prevalence : 0.10513
##       Balanced Accuracy : 0.75191
##
```

```
##        'Positive' Class : 1
##
```

Interpretation:

- Sensitivity of this model i.e. the ability of the model to identify true positive is 57.692%.
- Specificity of this model i.e. the ability of the model to identify true negative is 92.689%.
- Model is doing good in terms of reducing the false negative values. However, the model is not that good in terms of reducing the false positive values.

Step 7: Model has been built and read to be used on validation data. Create the validation data frame by taking 2016 year data only.

```
swimming.df.16 <- Full.athlete.df[which( Full.athlete.df$Sport == "Swimming"
            & Full.athlete.df$Event == "Swimming Men's 100 metres Freestyle"
                                      & Full.athlete.df$Year == 2016), ]
```

Step 8: Create Outcome binary variables from Medal variable to check if the model is correctly able predict the winners.

```
for (i in seq_along(swimming.df.16$Medal)){
  if (swimming.df.16$Medal[i] == "Dnw") {
    swimming.df.16$Outcome[i] <- 0
  } else {
    swimming.df.16$Outcome[i] <- 1
  }
}
```

Step 9: Run the prediction model created using with training data on the validation dataframe and create confusion matrix.

```
swimming.df.pred.16 <- predict(swimming.log, swimming.df.16, type = "response
")

swimming.df.predicted <- t(t(swimming.df.pred.16))

for (i in seq_along(swimming.df.predicted)){
  swimming.df.16$Predicted[i] <- as.numeric(swimming.df.pred.16[i])
}

swimming.cm.16 <- t(table(swimming.df.16$Outcome , swimming.df.16$Predicted >
0.185, dnn = c("Actual Class","Predictive Class")))
rownames(swimming.cm.16) <- colnames(swimming.cm)

confusionMatrix(swimming.cm.16, positive = "1", dnn = c("Actual Class","Predi
ctive Class"))

## Confusion Matrix and Statistics
##
##                 Actual Class
## Predictive Class  0  1
##                0 45  1
##                1  6  2
##
##                Accuracy : 0.8704
##                  95% CI : (0.751, 0.9463)
##     No Information Rate : 0.9444
##     P-Value [Acc > NIR] : 0.9904
##
##                   Kappa : 0.3077
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.66667
##             Specificity : 0.88235
##          Pos Pred Value : 0.25000
##          Neg Pred Value : 0.97826
##              Prevalence : 0.05556
##          Detection Rate : 0.03704
##    Detection Prevalence : 0.14815
##       Balanced Accuracy : 0.77451
##
##        'Positive' Class : 1
##
```

Interpretation:

- From the Confusion matrix, we can see that our model is predicting 8 probable winners in the 100 meters Men's Swimming Freestyle Event for 2016.
- Out of the 8 predicted possible winners, 2 have actually won medals in this event.
- The accuracy of the model is 87.04% in prediction.
- The chosen cut-off value for this event is 0.185.

Step 10: Get the names of the predicted players who can win the medals in the event.

```
swimming.16.pred.winners <- swimming.df.16[which(swimming.df.16$Predicted >=
0.185),]
swimming.16.winners <- swimming.df.16[which(swimming.df.16$Outcome == 1),]
```

Using our model, we have predicted the following athletes might win a medal in the 2016 Swimming Men's 100 metres Freestyle Event

```
as.character(swimming.16.pred.winners$Name)

## [1] "Kyle Chalmers"                "Cameron McEvoy"
## [3] "Dominik Kozma"                "Katsumi Nakamura"
## [5] "Shinri Shioura"               "Andrey Vladimirovich Grechin"
## [7] "Nathan Ghar-Jun Adrian"       "Caeleb Remel Dressel"
```

The following athletes have actually won

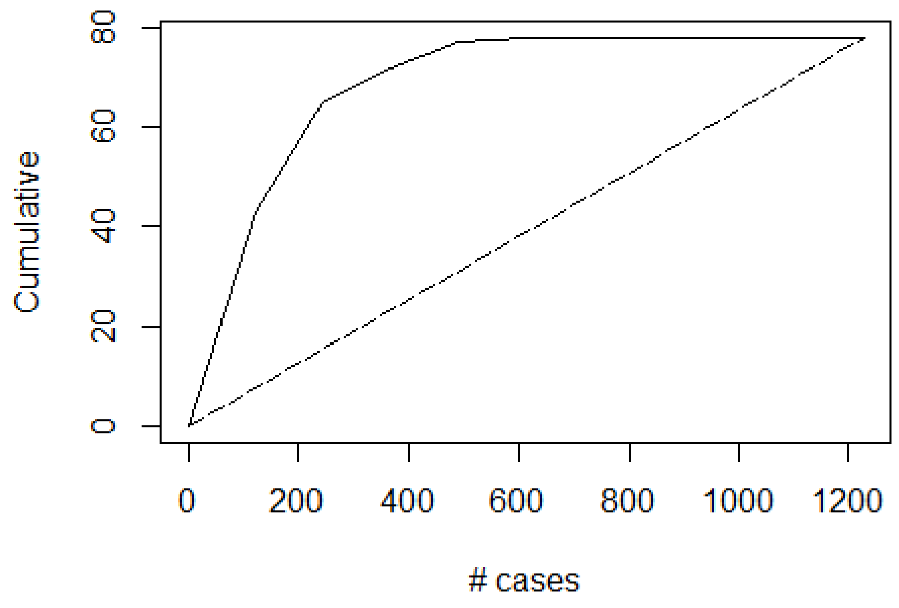
```
as.character(swimming.16.winners$Name)

## [1] "Kyle Chalmers"          "Pieter Timmers"
## [3] "Nathan Ghar-Jun Adrian"
```

Result: Our model was correctly able to predict that Kyle Chalmers and Nathan Ghar-Jun Adrian might win in the 2016 100-meters Men's Swimming Freestyle.

Step 11: Performance Evaluation on Training data

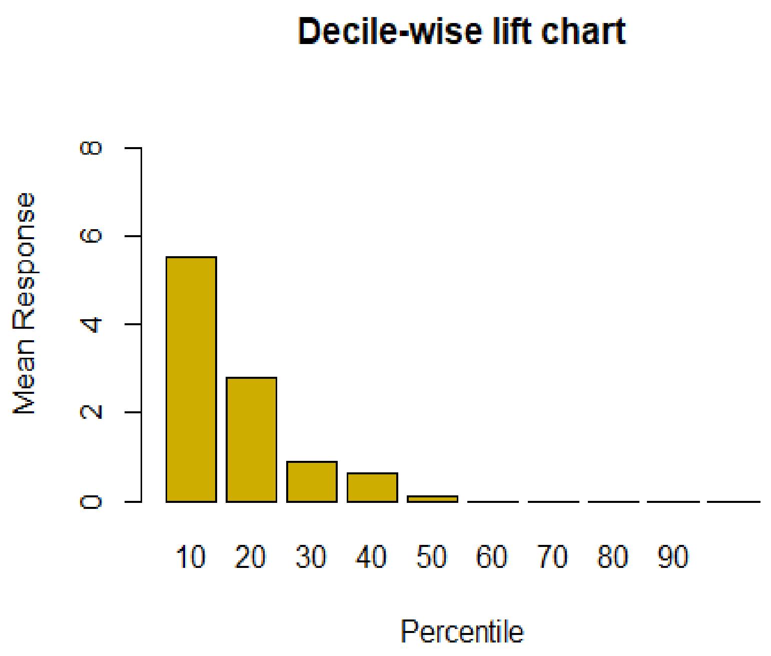
A. Plotting the lift chart of training data set

```
library(gains)
gain <- gains(swimming.df$Outcome, swimming.log.pred, groups = 10)
### Plot Lift Chart
plot(c(0,gain$cume.pct.of.total*sum(swimming.df$Outcome))~c(0,gain$cume.obs),
     xlab = "# cases", ylab = "Cumulative", main = "", type = "l")
lines(c(0,sum(swimming.df$Outcome))~c(0, dim(swimming.df)[1]), lty = 5)
```



B. Plotting the decile chart

```
heights <- gain$mean.resp/mean(swimming.df$Outcome)
midpoints <- barplot(heights, names.arg = gain$depth,  ylim = c(0,9), col = "
gold3",
                     xlab = "Percentile", ylab = "Mean Response",
                     main = "Decile-wise lift chart")
```



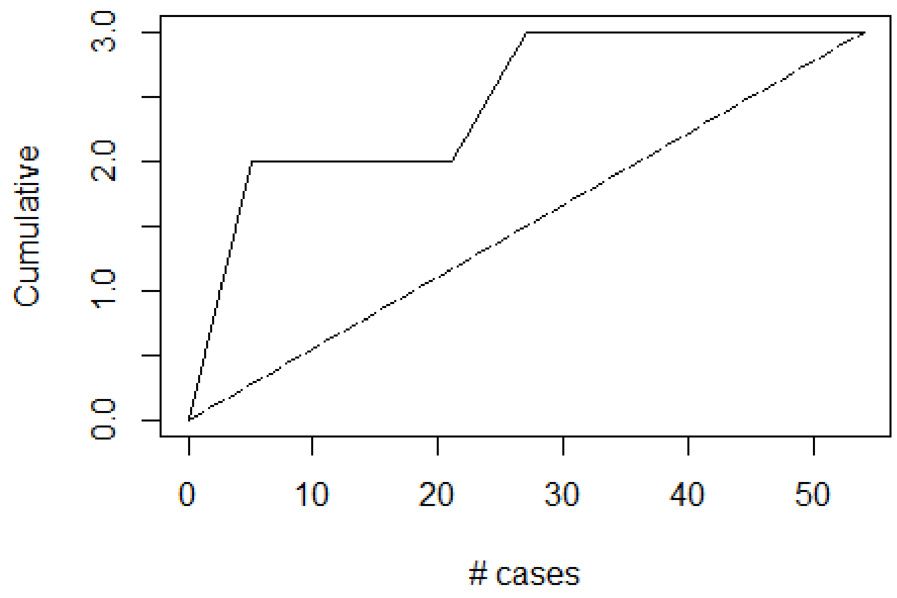
Interpretation: - From the Lift chart and Decile chart it can be said that: (i) The mean response of the top 10% in the decile chart is approx 6 times more the naive value. (ii) The Decile chart shows a decrease trend in the percentile showing that the model is good. (iii) The cumulative value of the Lift Chart is steeply increasing for the first few set of cases showing that the model is good.

Step 13: - Performance evaluation on Validation dataset

The Lift chart and Decile chart run on a validation dataset will always give good results. Let's check how good our model's outcome is on a validation dataset.

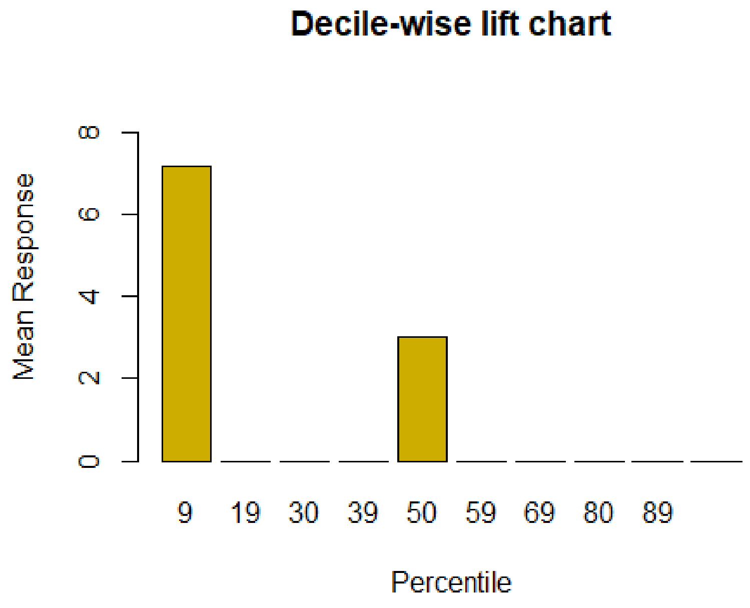
A. Plotting the lift chart

```
gain_valid <- gains(swimming.df.16$Outcome,swimming.df.pred.16,groups = 10)
### Plot Lift Chart
plot(c(0,gain_valid$cume.pct.of.total*sum(swimming.df.16$Outcome))~c(0,gain_v
alid$cume.obs),
     xlab = "# cases", ylab = "Cumulative", main = "", type = "l")
lines(c(0,sum(swimming.df.16$Outcome))~c(0, dim(swimming.df.16)[1]), lty = 5)
```



B. Plotting the decile chart for Validation data set

```
heights <- gain_valid$mean.resp/mean(swimming.df.16$Outcome)
midpoints <- barplot(heights, names.arg = gain_valid$depth,  ylim = c(0,9), c
ol = "gold3",
                     xlab = "Percentile", ylab = "Mean Response",
                     main = "Decile-wise lift chart")
```



Interpretation: - From the Lift chart and Decile chart run on validation dataframe it can be said that:

(i) The mean response of the top 10% in the decile chart is approx 7 times more the naive value.
(ii) The cumulative value of the Lift Chart is steeply increasing for the first few set of cases showing that the model is good.

Let us look at one more prediction model on an individual participation event before we start predicting on team participation events.

(b) Prediction of winner in 100-meter Freestyle Men's Athletics:

Step 1: Create the training dataframe of all the participants of 100-meter Men's Athletics event excluding 2016 data only. (As 2016-year data will be used for validation of the prediction model)

```
Athletics_100_meter_df <- Full.athlete.df [ which( Full.athlete.df$Sport == "
Athletics"
                    & Full.athlete.df$Event == "Athletics Men's 100 metres"
                    & Full.athlete.df$Year != 2016), ]
```

Step 2: To implement logistic regression, the outcome should be binary so converting the Medal variable into binary values. If an atlete has won Gold/Silver/Bronze medal, then the outcome will be 1 else it will be 0.

```
for (i in seq_along(Athletics_100_meter_df$Medal)){
  if (Athletics_100_meter_df$Medal[i] == "Dnw") {
    Athletics_100_meter_df$Outcome[i] <- 0
  } else {
    Athletics_100_meter_df$Outcome[i] <- 1
  }
}
table(Athletics_100_meter_df$Outcome)

##
##    0    1
## 1758   85
```

Step 3: Training data is now ready to run the logistic regression on Outcome variable depending on Age, Weight, Height and NOC(nationality of Candidate).

```
Athletics_100_meter_log<- glm(Outcome ~ Age + Weight + Height + NOC, data = A
thletics_100_meter_df, family = "binomial")
options(scipen=999)
```

Step 4: Model is ready. Summarizing the results of model and analyzing it.

```
summary(Athletics_100_meter_log)

##
## Call:
## glm(formula = Outcome ~ Age + Weight + Height + NOC, family = "binomial",
##     data = Athletics_100_meter_df)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -1.27283  -0.25806  -0.00003  -0.00003   2.75899
##
## Coefficients:
##                   Estimate    Std. Error z value Pr(>|z|)
## (Intercept)   -29.3874069 11906.3053598  -0.002    0.998
## Age             0.0024351     0.0369206   0.066    0.947
## Weight          0.0076222     0.0253248   0.301    0.763
## Height          0.0421077     0.0309863   1.359    0.174
## NOCAHO         -0.3146054 17672.1987383   0.000    1.000
## NOCALB         -0.5427736 31564.1492660   0.000    1.000
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 689.02  on 1842  degrees of freedom
## Residual deviance: 427.30  on 1650  degrees of freedom
## AIC: 813.3
##
## Number of Fisher Scoring iterations: 20
```

(**NOTE**: We have not included all the coefficients that was shown in the summary in this report.)

Step 5: Predict the outcome on the training dataframe.

```
Athletics_100_meter_log_pred <- predict(Athletics_100_meter_log, Athletics_10
0_meter_df, type = "response")
```

Step 6: Create confusion matrix to choose a cutoff value for this event.

```
Athletics_100_meter_cm <- t(table(Athletics_100_meter_df$Outcome , Athletics_
100_meter_log_pred > 0.175, dnn = c("Actual Class","Predictive Class")))
rownames(Athletics_100_meter_cm) <- colnames(Athletics_100_meter_cm)

myConfusion <- confusionMatrix(Athletics_100_meter_cm, positive = "1", dnn =
c("Actual Class","Predictive Class"))
myConfusion

## Confusion Matrix and Statistics
##
##                  Actual Class
## Predictive Class    0    1
##                 0 1657   34
##                 1  101   51
##
##                Accuracy : 0.9267
##                  95% CI : (0.9139, 0.9382)
##     No Information Rate : 0.9539
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3946
##  Mcnemar's Test P-Value : 0.00000011
##
##             Sensitivity : 0.60000
##             Specificity : 0.94255
##          Pos Pred Value : 0.33553
##          Neg Pred Value : 0.97989
##              Prevalence : 0.04612
##          Detection Rate : 0.02767
##    Detection Prevalence : 0.08247
##       Balanced Accuracy : 0.77127
##
##        'Positive' Class : 1
##
```

Interpretation:

- Sensitivity of this model i.e. the ability of the model to identify true positive is 60%.
- Specificity of this model i.e. the ability of the model to identify true negative is 94.255%.
- Model is doing good in terms of reducing the false negative values. However, the model is not that good in terms of reducing the false positive values.

Step 7: Model has been built and read to be used on validation data. Create validation data frame by taking 2016 year data only.

```
Athletics_100_meter_Year_16_df <- Full.athlete.df[which( Full.athlete.df$Spor
t == "Athletics"
                      & Full.athlete.df$Event == "Athletics Men's 100 metres"
                      & Full.athlete.df$Year == 2016), ]
```

Step 8: Creating Outcome binary variables from Medal variable to check if the model is correctly able predict the winners.

```
for (i in seq_along(Athletics_100_meter_Year_16_df$Medal)){
  if (Athletics_100_meter_Year_16_df$Medal[i] == "Dnw") {
    Athletics_100_meter_Year_16_df$Outcome[i] <- 0
  } else {
    Athletics_100_meter_Year_16_df$Outcome[i] <- 1
  }
}
table(Athletics_100_meter_Year_16_df$Outcome)

##
##  0  1
## 77  3
```

Step 9: Run the prediction model created using training data on the validation dataframe and create confusion matrix.

```
Athletics_100_meter_16_pred <- predict(Athletics_100_meter_log, Athletics_100
_meter_Year_16_df, type = "response")
Athletics_100_meter_16_predicted <- t(t(Athletics_100_meter_16_pred))

for (i in seq_along(Athletics_100_meter_16_predicted)){
  Athletics_100_meter_Year_16_df$Predicted[i] <- as.numeric(Athletics_100_met
er_16_pred[i])
}

Athletics_100_meter_Year_16_cm <- t(table(Athletics_100_meter_Year_16_df$Outc
ome , Athletics_100_meter_Year_16_df$Predicted > 0.175, dnn = c("Actual Class
","Predictive Class")))
rownames(Athletics_100_meter_Year_16_cm) <- colnames(Athletics_100_meter_Year
_16_cm)

confusionMatrix(Athletics_100_meter_Year_16_cm, positive = "1", dnn = c("Actu
al Class","Predictive Class"))

## Confusion Matrix and Statistics
##
##                 Actual Class
## Predictive Class  0  1
##                0 71  1
##                1  6  2
##
##                Accuracy : 0.9125
##                  95% CI : (0.828, 0.9641)
##     No Information Rate : 0.9625
##     P-Value [Acc > NIR] : 0.9897
##
##                   Kappa : 0.3269
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.6667
##             Specificity : 0.9221
##          Pos Pred Value : 0.2500
##          Neg Pred Value : 0.9861
##              Prevalence : 0.0375
##          Detection Rate : 0.0250
##    Detection Prevalence : 0.1000
##       Balanced Accuracy : 0.7944
##
##        'Positive' Class : 1
##
```

Interpretation:

- From the Confusion matrix, we can see that our model is predicting 8 probable winners in the 100 meters Men's Athletics Event for 2016.
- Out of the 8 predicted possible winners, 2 have actually won medals in this event.
- The accuracy of the model is 91.25% in prediction.
- The chosen cut-off value for this event is 0.175.

Step 10: Get the names of the predicted players who can win the medals in the event.

```
Athletics_100_meter_16.pred.winners <- Athletics_100_meter_Year_16_df[which(A
thletics_100_meter_Year_16_df$Predicted >= 0.175),]
Athletics_100_meter_16.winner <- Athletics_100_meter_Year_16_df[which(Athleti
cs_100_meter_Year_16_df$Outcome == 1),]
```

Using our model, we have predicted the following athletes might win a medal in the 2016 Men's Freestyle 100-metres Event:

```
as.character(Athletics_100_meter_16.pred.winners$Name)

##  [1] "Aaron Brown"

##  [2] "Nickel Ashmeade"
##  [3] "Yohan Blake"
##  [4] "Usain St. Leo Bolt"
##  [5] "Richard Thompson"
##  [6] "Marvin Bracy"
##  [7] "Trayvon Jaquez Bromell"
##  [8] "Justin Alexander Gatlin"
```

The following have actually won:

```
as.character(Athletics_100_meter_16.winner$Name)

## [1] "Andre De Grasse"         "Usain St. Leo Bolt"
## [3] "Justin Alexander Gatlin"
```

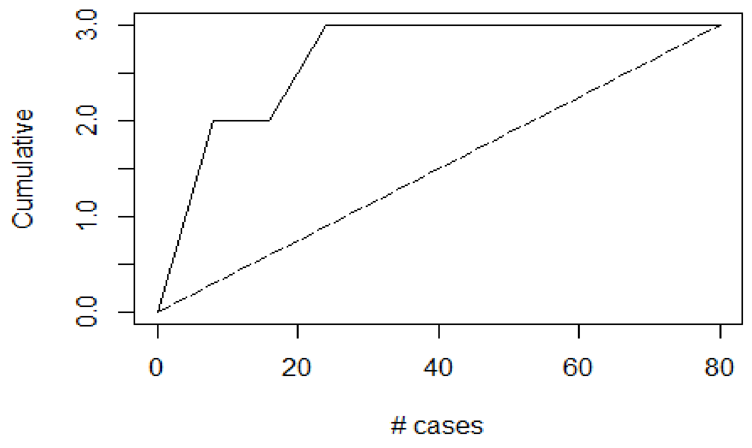
Result: Our model was correctly able to predict that Usain St. Leo Bolt and Justin Alexander Gatlin might win in the 2016 100-meters Men's Athletics.

Step 11 : Performance evaluation on Validation dataset

Here, we are directly evaluating our model on the validation dataset as we know that the lift chart and decile chart is going to be good on Training dataset.

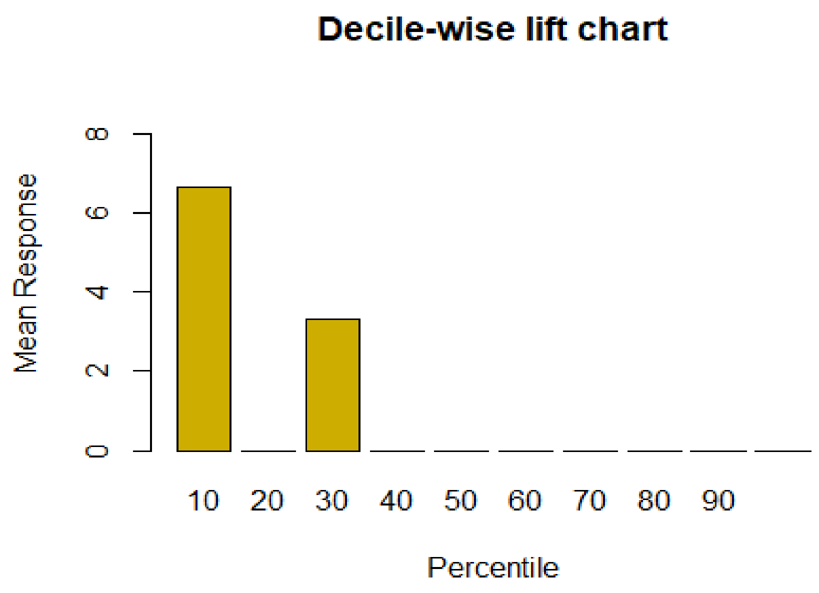
A. Plotting the lift chart

```
gain_valid <- gains(Athletics_100_meter_Year_16_df$Outcome,Athletics_100_mete
r_16_pred,groups = 10)
plot(c(0,gain_valid$cume.pct.of.total*sum(Athletics_100_meter_Year_16_df$Outc
ome))~c(0,gain_valid$cume.obs),
     xlab = "# cases", ylab = "Cumulative", main = "", type = "l")
lines(c(0,sum(Athletics_100_meter_Year_16_df$Outcome))~c(0, dim(Athletics_100
_meter_Year_16_df)[1]), lty = 5)
```



B. Plotting the decile chart for Validation data set

```
heights <- gain_valid$mean.resp/mean(Athletics_100_meter_Year_16_df$Outcome)
midpoints <- barplot(heights, names.arg = gain_valid$depth,  ylim = c(0,9), c
ol = "gold3",
                     xlab = "Percentile", ylab = "Mean Response",
                     main = "Decile-wise lift chart")
```



Interpretation:From the Lift chart and Decile chart run on validation dataframe it can be said that:

(i) The mean response of the top 10% in the decile chart is approx 7 times more the naive value.

(ii) The cumulative value of the Lift Chart is steeply increasing for the first few set of cases showing that the model is good.

(II) Prediction of Events having Team participation:

Till now, we have looked how we can predict which athlete might win in an individual participation event. Let's see how we can use logistic regression for predicting the winning country in a team participation event.

Prediction of Winning Countries in 2016 Men's Football event:

The steps used for predicting the winning team of a Team event will be similar to that done for an individual event. However, the way training dataframe and validation dataframe will be created for running the algorithm will be a bit different as the variables will have to be grouped by Country and Year.

Step 1: Create a dataframe for football with data grouped by Year and Country.

In team events like Football, there is only one team representing the entire country for that particular year. Let's take a look at the data for germany for the year 2016.

```
Football.germany <- Full.athlete.df[which(Full.athlete.df$Event == "Football
Men's Football"
                      & Full.athlete.df$Country == "Germany"
                      & Full.athlete.df$Year == 2016),]
Football.germany[,c(2,3,4,5,6,14,15)]

##                       Name Sex Age Height Weight Country  Medal
##  1:           Robert Bauer   M  21    183     76 Germany Silver
##  2:            Lars Bender   M  27    184     80 Germany Silver
##  3:            Sven Bender   M  27    185     80 Germany Silver
##  4:          Julian Brandt   M  20    183     83 Germany Silver
##  5:       Max Christiansen   M  19    187     84 Germany Silver
##  6: Matthias Lukas Ginter   M  22    190     88 Germany Silver
##  7:      Serge David Gnabry   M  21    173     74 Germany Silver
##  8:          Leon Goretzka   M  21    189     79 Germany Silver
##  9:              Timo Horn   M  23    191     76 Germany Silver
## 10:      Lukas Klostermann   M  20    189     83 Germany Silver
## 11:            Philipp Max   M  22    177     76 Germany Silver
## 12:        Maximilian Meyer   M  20    173     60 Germany Silver
## 13:          Nils Petersen   M  27    188     80 Germany Silver
## 14:          Grischa Prmel   M  21    182     78 Germany Silver
## 15:            Davie Selke   M  21    192     82 Germany Silver
## 16:             Niklas Sle   M  20    195     95 Germany Silver
## 17:          Jeremy Toljan   M  21    182     72 Germany Silver
```

We can see that if a team is winning an event, the entire team is credited for it and not just one individual athlete. So, we will have to recreate the values of the predictors(Age, Height and Weight) as one value(Mean value) for the entire team.

```
Football.df <- Full.athlete.df[which(Full.athlete.df$Sport == "Football"
                              & Full.athlete.df$Event == "Football Men's Foot
ball"),]
data.football.age<- data.frame(Football.df %>%
        group_by(Country, Year) %>%
          summarise(mean = mean(Age)))
colnames(data.football.age) <- c("Country","Year","Mean Age")
head(data.football.age)

##       Country Year Mean Age
## 1 Afghanistan 1948 24.00000
## 2     Algeria 1980 23.64286
## 3     Algeria 2016 22.66667
## 4   Argentina 1928 24.93750
## 5   Argentina 1960 20.92308
## 6   Argentina 1964 20.92857
```

Here, we are using %>% funtion to create subsets. %>% is used to create a subset of the data given on the lhs of %>% based on the condition mentioned on the rhs of %>%.

In the above code, Football.df is the dataframe created for the event: “Men’s Football”. We are then creating a subset of it by first grouping the data according to Year and Country and then calculating the mean age on the grouped data.

Similarly, calculating mean height and weight on the data frame Football.df grouped by Country and Year.

```
data.football.height<- data.frame(Football.df %>%
                                 group_by(Country, Year) %>%
                                 summarise(mean = mean(Height)))
colnames(data.football.height) <- c("Country","Year","Mean Height")

data.football.weight<- data.frame(Football.df %>%
                                    group_by(Country, Year) %>%
                                    summarise(mean = mean(Weight)))
colnames(data.football.weight) <- c("Country","Year","Mean Weight")
```

Create a dataframe for medal grouped by Country and Year and merge the above dataframes with it using merge() function.

```
data.football.medal <-Football.df[,c(8,14,15)]
data.football.medal<- unique(data.football.medal)

mergd.football.df <- merge(data.football.age,data.football.medal, by = c("Yea
r", "Country"))

mergd.football.df <- merge(mergd.football.df,data.football.height, by = c("Ye
ar", "Country"))

mergd.football.df <- merge(mergd.football.df,data.football.weight, by = c("Ye
ar", "Country"))
```

```r
#Rearranging the columns:
mergd.football.df <- mergd.football.df[, c(1,2,3,5,6,4)]

colnames(mergd.football.df) <- c("Year", "Country", "Age", "Height", "Weight"
, "Medal")
tail(mergd.football.df)
```

```
##     Year      Country      Age   Height   Weight  Medal
## 382 2016       Mexico 23.11111 177.8889 70.16667    Dnw
## 383 2016      Nigeria 21.27778 172.7222 75.72222 Bronze
## 384 2016     Portugal 22.00000 181.7647 77.52941    Dnw
## 385 2016 South Africa 22.11111 178.0556 69.88889    Dnw
## 386 2016  South Korea 22.41176 181.7647 74.05882    Dnw
## 387 2016       Sweden 22.40000 181.6000 75.33333    Dnw
```

We have created the dataframe for each country having a single entry the event. Now, this dataframe can be treated as a dataframe with each row as an individual participation.

Step 2: Create the training data frame of all the teams participating in summer Men's Football event excluding the data for 2016. (2016 year data will be used for prediction later)

Also, to implement logistic regression, the outcome should be binary so converting the Medal variable into binary values.

```r
train.football.df <- mergd.football.df[which(mergd.football.df$Year != 2016),
]

for (i in seq_along(train.football.df$Medal)){
  if (train.football.df$Medal[i] == "Dnw") {
    train.football.df$Outcome[i] <- 0
  } else {
    train.football.df$Outcome[i] <- 1
  }
}
```

Step 3: Data is now ready to run the logistic regression on Outcome variable depending on Age, Weight, Height and Country

```r
train.football.log <- glm(Outcome ~ Age + Weight + Height + Country,
                      data = train.football.df,
                      family = "binomial")
options(scipen=999)
```

Step 4: Model is ready. Predict the outcome on the trained model.

```r
train.football.log.pred <- predict(train.football.log, train.football.df,
                                type = "response")
```

Step 5: Create the confusion matrix to choose a cut-off value for this event.

```r
train.football.cm <- t(table(train.football.df$Outcome , train.football.log.p
red > 0.28, dnn = c("Actual Class","Predictive Class")))
rownames(train.football.cm) <- colnames(train.football.cm)
```

```
confusionMatrix(train.football.cm, positive = "1", dnn = c("Actual Class","Pr
edictive Class"))

## Confusion Matrix and Statistics
## 
##                   Actual Class
## Predictive Class   0   1
##                  0 203  17
##                  1  86  65
##                                           
##                Accuracy : 0.7224          
##                  95% CI : (0.6738, 0.7674)
##     No Information Rate : 0.779           
##     P-Value [Acc > NIR] : 0.9957          
##                                           
##                   Kappa : 0.3805          
##  Mcnemar's Test P-Value : <0.0000000000000002
##                                           
##             Sensitivity : 0.7927          
##             Specificity : 0.7024          
##          Pos Pred Value : 0.4305          
##          Neg Pred Value : 0.9227          
##              Prevalence : 0.2210          
##          Detection Rate : 0.1752          
##    Detection Prevalence : 0.4070          
##       Balanced Accuracy : 0.7476          
##                                           
##        'Positive' Class : 1               
## 
```

Interpretation:

- Sensitivity of this model i.e. the ability of the model to identify true positive is 79.27%. From this value of sensitivity, it can be said that our model doing good in predicting the probable winners of this event.
- The specificity of this model i.e. the ability of the model to identify true negative is 70.24%.
- This model is doing good in terms of reducing the false negative values. However, the model is not that good in terms of reducing the false positive values.

Step 7: Model has been built and read to be used on validation data. Creating the validation data frame by taking 2016 year data only.

```
valid.football.16 <- mergd.football.df[which( mergd.football.df$Year == 2016)
, ]
```

Step 8: Create Outcome binary variables from Medal variable to check if the model is correctly able predict the winners.

```
for (i in seq_along(valid.football.16$Medal)){
  if (valid.football.16$Medal[i] == "Dnw") {
    valid.football.16$Outcome[i] <- 0
  } else {
    valid.football.16$Outcome[i] <- 1
  }
}
```

Step 9: Run prediction model created using training data on the validation dataframe and create confusion matrix.

```
valid.football.16.pred <- predict(train.football.log, valid.football.16, type
= "response")
valid.football.16.predicted <- t(t(valid.football.16.pred))

for (i in seq_along(valid.football.16.predicted)){
  valid.football.16$Predicted[i] <- as.numeric(valid.football.16.pred[i])
}

valid.football.16.cm <- t(table(valid.football.16$Outcome , valid.football.16
$Predicted > 0.28, dnn = c("Actual Class","Predictive Class")))
rownames(valid.football.16.cm) <- colnames(valid.football.16.cm)

confusionMatrix(valid.football.16.cm, positive = "1", dnn = c("Actual Class",
"Predictive Class"))

## Confusion Matrix and Statistics
## 
##                 Actual Class
## Predictive Class  0 1
##                0  9 0
##                1  3 3
## 
##                Accuracy : 0.8
##                  95% CI : (0.5191, 0.9567)
##     No Information Rate : 0.8
##     P-Value [Acc > NIR] : 0.6482
## 
##                   Kappa : 0.5455
##  Mcnemar's Test P-Value : 0.2482
## 
##             Sensitivity : 1.0000
##             Specificity : 0.7500
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.2000
##          Detection Rate : 0.2000
##    Detection Prevalence : 0.4000
##       Balanced Accuracy : 0.8750
## 
##        'Positive' Class : 1
## 
```

Results:

- From the Confusion matrix, we can see that our model is correctly predict all 3 probable Countries that won Men's Football Event in 2016 giving a sensitivity of 1.000.
- The accuracy of the model is 80% in prediction.
- The chosen cut-off value for this event is 0.28.

Step 10: Get the names of the predicted countries who can win medals in this event.

```
valid.football.16.pred.winners <- valid.football.16[which(valid.football.16$P
redicted >= 0.28),]
valid.football.16.winners <- valid.football.16[which(valid.football.16$Outcom
e == 1),]
```

Using our model, we have predicted the following countries might win a medal in the 2016 Men's Football Event:

```
as.character(valid.football.16.pred.winners$Country)

## [1] "Denmark"   "Germany"   "Nigeria"   "Sweden"  "Argentina"   "Brazil"
```

The following countries have actually won:

```
as.character(valid.football.16.winners$Country)

## [1] "Brazil"  "Germany" "Nigeria"
```

Conclusion: Our model was correctly able to predict all 3 countries that won in the 2016 Men's Football Event which are Germany, Nigeria and Brazil.

(III) Prediction Events having Team Participation of Athletes with Countries sending multiple teams:

In events such as Football and Hockey, there is only one representation of a country as a team for that year. But there are other events as well such as Tennis and Relay in which there are multiple participation of teams belonging to the same country. In such a case, if we group by country then there will only be one entry for that year instead of multiple entries. So instead of grouping by Country, we will group by Team for predicting such events.

Prediction of winner in Men's Tennis doubles:

Step 1: Create a dataframe for Men's Tennis with data grouped by Year and Team. We are using aggregate() function.

Here, instead of using %>% function to group by Team and Year to find the mean Age, Height and Weight and then merge them to create a dataframe for Men's Tennis.

```
Tennis.df <- Full.athlete.df[which(Full.athlete.df$Sport == "Tennis"
                    & Full.athlete.df$Event == "Tennis Men's Doubles"), ]



data.tennis.age <- data.frame(aggregate(Tennis.df$Age,
                                        by = list(Tennis.df$Team,Tennis.df$
Year),
                                        FUN = mean, drop = FALSE))
```

```
colnames(data.tennis.age) <- c("Team","Year","Mean Age")

data.tennis.age <- data.tennis.age[!is.nan(data.tennis.age$`Mean Age`),]
head(data.tennis.age)
```

```
##                        Team Year Mean Age
## 7   Australia/Great Britain 1896     22.5
## 59    Great Britain/Germany 1896     22.5
## 61                 Greece-1 1896     20.0
## 62                 Greece-2 1896     20.5
## 63                 Greece-3 1896     24.0
## 184                France-1 1900     35.5
```

```
data.tennis.height <- data.frame(aggregate(Tennis.df$Height,
                                             by = list(Tennis.df$Team,Tennis.
df$Year),
                                             FUN = mean, drop = FALSE))

colnames(data.tennis.height) <- c("Team","Year","Mean Height")
data.tennis.height <- data.tennis.height[!is.nan(data.tennis.height$`Mean Hei
ght`),]


data.tennis.weight <- data.frame(aggregate(Tennis.df$Weight,
                                             by = list(Tennis.df$Team,Tennis.
df$Year),
                                             FUN = mean, drop = FALSE))
colnames(data.tennis.weight) <- c("Team","Year","Mean Weight")
data.tennis.weight <- data.tennis.weight[!is.nan(data.tennis.weight$`Mean Wei
ght`),]


data.tennis.medal <-Tennis.df[,c(7,8,14,15)]
data.tennis.medal<- unique(data.tennis.medal)


mergd.tennis.df <- merge(data.tennis.age,data.tennis.medal, by = c("Year", "T
eam"))
mergd.tennis.df <- merge(mergd.tennis.df,data.tennis.height, by = c("Year", "
Team"))
mergd.tennis.df <- merge(mergd.tennis.df,data.tennis.weight, by = c("Year", "
Team"))
mergd.tennis.df <- mergd.tennis.df[, c(1,2,4,3,6,7,5)]
colnames(mergd.tennis.df) <- c("Year", "Team" ,"Country", "Age", "Height", "W
eight", "Medal")
tail(mergd.tennis.df)
```

```
##     Year         Team  Country  Age Height Weight  Medal
## 374 2016      Spain-1    Spain 31.0  179.0   74.5    Dnw
## 375 2016      Spain-2    Spain 32.0  180.0   75.5   Gold
## 376 2016     Thailand Thailand 34.0  175.0   70.5    Dnw
## 377 2016          Ukraine  Ukraine 28.5  187.5   80.0    Dnw
## 378 2016 United States-1      USA 31.5  192.0   80.5    Dnw
## 379 2016 United States-2      USA 24.5  189.5   85.0 Bronze
```

Step 2: Create the training data frame of all the teams participating in summer Men's Tennis event excluding the data for 2016. (2016 year data will be used for prediction later) Also, to implement logistic regression, the outcome should be binary so converting the Medal variable into binary values.

```
train.tennis.df <- mergd.tennis.df[which(mergd.tennis.df$Year != 2016), ]

for (i in seq_along(train.tennis.df$Medal)){
  if (train.tennis.df$Medal[i] == "Dnw") {
    train.tennis.df$Outcome[i] <- 0
  } else {
    train.tennis.df$Outcome[i] <- 1
  }
}
table(train.tennis.df$Outcome)

## 
##   0   1 
## 295  52
```

Step 3: Data is now ready to run the logistic regression on Outcome variable depending on Age, Weight, Height and Country.

```
train.tennis.log <- glm(Outcome ~ Age + Weight + Height + Country,
                          data = train.tennis.df,
                          family = "binomial")
options(scipen=999)
```

Step 4: Model is ready. Predict the outcome on the trained model.

```
train.tennis.log.pred <- predict(train.tennis.log, train.tennis.df,
                                   type = "response")
```

Step 5: Create confusion matrix to choose a cut-off value for this event.

```
train.tennis.cm <- t(table(train.tennis.df$Outcome , train.tennis.log.pred >
0.13, dnn = c("Actual Class","Predictive Class")))
rownames(train.tennis.cm) <- colnames(train.tennis.cm)
train.tennis.cm

##                  Actual Class
## Predictive Class   0   1
##                 0 181   6
##                 1 114  46

confusionMatrix(train.tennis.cm, positive = "1", dnn = c("Actual Class","Pred
ictive Class"))

## Confusion Matrix and Statistics
## 
##                  Actual Class
## Predictive Class   0   1
##                0 181   6
##                1 114  46
##
##                Accuracy : 0.6542
##                  95% CI : (0.6015, 0.7042)
##     No Information Rate : 0.8501
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2685
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.8846
##             Specificity : 0.6136
##          Pos Pred Value : 0.2875
##          Neg Pred Value : 0.9679
##              Prevalence : 0.1499
##          Detection Rate : 0.1326
##    Detection Prevalence : 0.4611
##       Balanced Accuracy : 0.7491
##
##        'Positive' Class : 1
##
```

Interpretation:

- Sensitivity of this model i.e. the ability of the model to identify true positive is 88.46%.
- Also, specificity of this model i.e. the ability of the model to identify true negative is 61.36% which is also good.
- This model is doing good in terms of reducing the false negative values. However, the model is not that good in terms of reducing the false positive values as seen in the previous predictive models as well.

Step 7: - Model has been built and read to be used on validation data. Creating the validation data frame by taking 2016 year data only.

```
valid.tennis.16 <- mergd.tennis.df[which( mergd.tennis.df$Year == 2016), ]
```

Step 8: - Create Outcome binary variables from Medal variable to check if the model is correctly able predict the winners.

```
for (i in seq_along(valid.tennis.16$Medal)){
  if (valid.tennis.16$Medal[i] == "Dnw") {
    valid.tennis.16$Outcome[i] <- 0
  } else {
    valid.tennis.16$Outcome[i] <- 1
  }
}
```

Step 9: Run prediction model created using training data on the validation dataframe and create confusion matrix.

```
valid.tennis.16.pred <- predict(train.tennis.log, valid.tennis.16, type = "re
sponse")
```

```
valid.tennis.16.predicted <- t(t(valid.tennis.16.pred))
for (i in seq_along(valid.tennis.16.predicted)){
  valid.tennis.16$Predicted[i] <- as.numeric(valid.tennis.16.pred[i])
}
valid.tennis.16.cm <- t(table(valid.tennis.16$Outcome , valid.tennis.16$Predi
cted > 0.13, dnn = c("Actual Class","Predictive Class")))
rownames(valid.tennis.16.cm) <- colnames(valid.tennis.16.cm)
confusionMatrix(valid.tennis.16.cm, positive = "1", dnn = c("Actual Class","P
redictive Class"))

## Confusion Matrix and Statistics
##
##                  Actual Class
## Predictive Class  0  1
##                 0 17  1
##                 1 10  2
##
##                Accuracy : 0.6333
##                  95% CI : (0.4386, 0.8007)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : 0.99998
##
##                   Kappa : 0.127
##  Mcnemar's Test P-Value : 0.01586
##
##             Sensitivity : 0.66667
##             Specificity : 0.62963
##          Pos Pred Value : 0.16667
##          Neg Pred Value : 0.94444
##              Prevalence : 0.10000
##          Detection Rate : 0.06667
##    Detection Prevalence : 0.40000
##       Balanced Accuracy : 0.64815
##
##        'Positive' Class : 1
##
```

Interpretation:

- From the Confusion matrix, we can see that our model is predicting 12 probable teams that might win Men's Tennis Event in 2016. Out of these 12 predicted teams, 2 have actually won medals in this event giving a sensitivity of 66.67%.
- The accuracy of the model is 63.3% in prediction.
- The chosen cut-off value for this event is 0.13.

Step 10: Get the names of the predicted countries who can win medals in this event.

```
valid.tennis.16.pred.winners <- valid.tennis.16[which(valid.tennis.16$Predict
ed >= 0.13),]
valid.tennis.16.winners <- valid.tennis.16[which(valid.tennis.16$Outcome == 1
),]
```

Using our model, we have predicted the following countries might win a medal in the 2016 Men's Tennis Event:

```
as.character(valid.tennis.16.pred.winners$Team)

##  [1] "Australia"       "Austria"         "Chile"
##  [4] "Croatia"         "France-1"        "France-2"
##  [7] "Great Britain-1" "Great Britain-2" "Spain-1"
## [10] "Spain-2"         "United States-1" "United States-2"
```

The following teams have actually won

```
as.character(valid.tennis.16.winners$Team)

## [1] "Romania"         "Spain-2"         "United States-2"
```

Result: Our model was correctly able to predict the following teams - Spain-2 and United States-2 who actually won in the 2016 Men's Tennis Event.

All the predictions that we have done till now were based on Summer games. Let us now do a prediction on an event of Winter Olympic Games.

Prediction of winner in Figure Skating Women's Singles(Winter)

This is an individual participation event, So the steps followed for prediction will be similar to that of an Individual participation sport.

Step 1: Create the training data frame of all the participants of Figure Skating Women's Singles event excluding 2016 data only. (As 2016-year data will be used for prediction later)

```
Figure_Skating_Women_Single.df = Full.athlete.df[which(Full.athlete.df$Sport
== "Figure Skating"
                                                 & Full.athlete.df$Event == "Figure S
kating Women's Singles"
                                                 & Full.athlete.df$Year != 2014),]
```

Step 2: To implement logistic regression, the outcome should be binary so converting the Medal variable into binary values.

```
for (i in seq_along(Figure_Skating_Women_Single.df$Medal)){
  if (Figure_Skating_Women_Single.df$Medal[i] == "Dnw") {
    Figure_Skating_Women_Single.df$Outcome[i] <- 0
  } else {
    Figure_Skating_Women_Single.df$Outcome[i] <- 1
  }
}
table(Figure_Skating_Women_Single.df$Outcome)

##
##   0   1
## 456  69
```

Step 3: Data is now ready to run the logistic regression on Outcome variable depending on Age, Weight, Height and NOC(nationality of Candidate).

```
Figure_Skating_Women_Single_log<- glm(Outcome ~ Age + Weight + Height + NOC,
data = Figure_Skating_Women_Single.df, family = "binomial")
options(scipen=999)
```

Step 4: Model is ready. Summarizing the results of model and analyzing it.

```
summary(Figure_Skating_Women_Single_log)

##
## Call:
## glm(formula = Outcome ~ Age + Weight + Height + NOC, family = "binomial",
##     data = Figure_Skating_Women_Single.df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.72549  -0.53739  -0.31259  -0.00007   2.72270
##
## Coefficients:
##                Estimate  Std. Error z value Pr(>|z|)
## (Intercept)   -14.80572 10754.01295  -0.001 0.998902
## Age             0.14742     0.04035   3.653 0.000259 ***
## Weight          0.01315     0.02792   0.471 0.637630
## Height         -0.05498     0.03333  -1.650 0.099019 .
## NOCAUS          0.86222 11207.50790   0.000 0.999939
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 408.55  on 524  degrees of freedom
## Residual deviance: 309.53  on 474  degrees of freedom
## AIC: 411.53
##
## Number of Fisher Scoring iterations: 18
```

Interpretation: - Results show that the Age and Height are significant variables on which the candidate's ability to win depends.

Step 5: Predict the outcome on the trained model.

```
Figure_Skating_Women_Single_pred = predict(Figure_Skating_Women_Single_log,Fi
gure_Skating_Women_Single.df, type = "response")
```

Step 6: Create confusion matrix to choose a cutoff value for this event.

```
Figure_Skating_Women_Single_cm <- t(table(Figure_Skating_Women_Single.df$Outc
ome , Figure_Skating_Women_Single_pred > 0.11, dnn = c("Actual Class","Predic
tive Class")))
Figure_Skating_Women_Single_cm

##                  Actual Class
## Predictive Class   0   1
##            FALSE 319  12
##            TRUE  137  57

rownames(Figure_Skating_Women_Single_cm) <- colnames(Figure_Skating_Women_Sin
gle_cm)
```

```
myConfusion = confusionMatrix(Figure_Skating_Women_Single_cm, positive = "1",
dnn = c("Actual Class","Predictive Class"))
myConfusion
```

```
## Confusion Matrix and Statistics
## 
##                  Actual Class
## Predictive Class   0   1
##                 0 319  12
##                 1 137  57
## 
##                 Accuracy : 0.7162
##                   95% CI : (0.6755, 0.7544)
##      No Information Rate : 0.8686
##      P-Value [Acc > NIR] : 1
## 
##                    Kappa : 0.2972
##   Mcnemar's Test P-Value : <0.0000000000000002
## 
##              Sensitivity : 0.8261
##              Specificity : 0.6996
##           Pos Pred Value : 0.2938
##           Neg Pred Value : 0.9637
##               Prevalence : 0.1314
##           Detection Rate : 0.1086
##     Detection Prevalence : 0.3695
##        Balanced Accuracy : 0.7628
## 
##         'Positive' Class : 1
## 
```

Interpretation:

- Sensitivity of this model i.e. the ability of the model to identify true positive is 82.61%.
- Specificity of this model i.e. the ability of the model to identify true negative is 69.96%.
- Model is doing good in terms of reducing the false negative values. However, the model is not that good in terms of reducing the false positive values.

Step 7: Model has been built and read to be used on validation data. Creating the validation data frame by taking 2016 year data only.

```
Figure_Skating_Women_Single_14.df = Full.athlete.df[which(Full.athlete.df$Spo
rt == "Figure Skating"
                                                          & Full.athlete.df$Even
t == "Figure Skating Women's Singles"
                                                          & Full.athlete.df$Year
== 2014), ]
```

Step 8: Create Outcome binary variables from Medal variable to check if the model is correctly able predict the winners.

```
for (i in seq_along(Figure_Skating_Women_Single_14.df$Medal)){
  if (Figure_Skating_Women_Single_14.df$Medal[i] == "Dnw") {
    Figure_Skating_Women_Single_14.df$Outcome[i] <- 0
  } else {
    Figure_Skating_Women_Single_14.df$Outcome[i] <- 1
  }
}
```

Step 9: Run prediction model created using with training data on the validation dataframe and create confusion matrix.

```
Figure_Skating_Women_Single_14_pred <- predict(Figure_Skating_Women_Single_lo
g, Figure_Skating_Women_Single_14.df, type = "response")
Figure_Skating_Women_Single_14_predicted <- t(t(Figure_Skating_Women_Single_1
4_pred))

for (i in seq_along(Figure_Skating_Women_Single_14_predicted)){
  Figure_Skating_Women_Single_14.df$Predicted[i] <- as.numeric(Figure_Skating
_Women_Single_14_pred[i])
}

Figure_Skating_Women_Single_14_cm <- t(table(Figure_Skating_Women_Single_14.d
f$Outcome , Figure_Skating_Women_Single_14.df$Predicted > 0.11, dnn = c("Actu
al Class","Predictive Class")))

rownames(Figure_Skating_Women_Single_14_cm) <- colnames(Figure_Skating_Women_
Single_14_cm)

confusionMatrix(Figure_Skating_Women_Single_14_cm, positive = "1", dnn = c("A
ctual Class","Predictive Class"))

## Confusion Matrix and Statistics
## 
##                 Actual Class
## Predictive Class  0  1
##                0 13  1
##                1 13  2
##                                           
##                Accuracy : 0.5172          
##                  95% CI : (0.3253, 0.7055)
##     No Information Rate : 0.8966          
##     P-Value [Acc > NIR] : 1.000000        
##                                           
##                   Kappa : 0.0602          
##  Mcnemar's Test P-Value : 0.003283        
##                                           
##             Sensitivity : 0.66667         
##             Specificity : 0.50000         
##          Pos Pred Value : 0.13333         
##          Neg Pred Value : 0.92857         
##              Prevalence : 0.10345         
##          Detection Rate : 0.06897         
##    Detection Prevalence : 0.51724         
##        Balanced Accuracy : 0.58333
##
##         'Positive' Class : 1
##
```

Interpretation:

- From the Confusion matrix, we can see that our model was correctly able to predict two winners.
- The accuracy of the model is 51.72% in prediction.
- The chosen cut-off value for this event is 0.11.

Step 10: Get the names of the predicted players who can win the medals in the event.

```
Figure_Skating_Women_Single_14_pred_winner <- Figure_Skating_Women_Single_14.
df[which(Figure_Skating_Women_Single_14.df$Predicted >= 0.11),]
Figure_Skating_Women_Single_14_winner <- Figure_Skating_Women_Single_14.df[wh
ich(Figure_Skating_Women_Single_14.df$Outcome == 1),]
```

Using our model, we have predicted the following athletes might win a medal in the 2014 Figure Skating Women's Singles Event

```
as.character(Figure_Skating_Women_Single_14_pred_winner$Name)

##  [1] "Kerstin Frank"
##  [2] "Li Zijun"
##  [3] "Zhang Kexin"
##  [4] "Jenna McCorkell"
##  [5] "Mao Asada"
##  [6] "Akiko Suzuki"
##  [7] "Yu-Na Kim"
##  [8] "Park So-Yeon"
##  [9] "Anne Line Gjersem"
## [10] "Yuliya Vyacheslavovna Lipnitskaya"
## [11] "Adelina Dmitriyevna Sotnikova"
## [12] "Hanna Viktoria Helgesson"
## [13] "Polina Edmunds"
## [14] "Grace Elizabeth \"\"Gracie\"\" Gold"
## [15] "Ashley Elisabeth Wagner"
```

The following have actually won:

```
as.character(Figure_Skating_Women_Single_14_winner$Name)

## [1] "Carolina Kostner"              "Yu-Na Kim"
## [3] "Adelina Dmitriyevna Sotnikova"
```

Result: Our model was correctly able to predict that Yu-Na Kim and Adelina Dmitriyevna Sotnikova might win in the 2014

Conclusion:

We were able to predict winners in the different events held in Olympics using Logistic Regression. However, there were a few crucial points that was observed for while predicting the winners for every event. Some of them are:

(i) **Low Cut-Off Value**: Normally, for a binary outcome, the cut-off value is 0.5. But the cut-off values for all events that were predicted was very low. This is due to the fact that there is only three spots for winning and the no of participants is very high.

(ii) **Changing Cut-Off Value for each Event**: When starting the prediction with logistic regression, we had thought of keeping the cut-off value constant across all events. But the significance of predictors changes from event to event and also the no of participation differs from event to event. Hence, keeping a constant cut-off value for all event will not give good prediction results.

(iii) **Lift and Decile Chart**: As there are only 3 possibilities of the actual outcome to be 1, the chart was sometimes not of the desired curve. However, in most of the cases, we got a steep lift curve for the initial cases and 5-6 times the mean response from the naïve model in decile chart. So, we can conclude that top performers were identified by our model.

(iv) **Accuracy**: The accuracy of prediction on the validation dataset for the different events lied in the range of 50% to 80% which is good. Our model is doing good in terms of Specificity, Sensitivity and having less False Negative values. However, our model was good with having a less False Positive values.

(v) **P-Value**: For few predictions that were done above, the P-value did not have a significant value to decide whether that predictor is important enough in predicting the outcome variable or not. This could be because of the number of countries in the categorical variable - NOC. As the number of countries participating in an event was large, the model might have run into overfitting issue. This can be observed by the fact – When we predicted for the winter games (Figure Skating), we got predictors age and height having significant P-value as the number of countries participating in winter games is less.

Distribution of Content:

The following were the contribution of each member of the group:

(i) Ashwin
- Data Exploration – Distribution of Age of Medalists
- Model Prediction – Prediction of winner in Men's Tennis Doubles
- Create collaborated report

(ii) Akanksha
- Data Munging
- Model Prediction – Prediction of winner in Men's Football
- Conclusion
- Create .Rmd file

(iii) Pragati
- Introduction
- Data Exploration – Correlation between age, height and weight in different events of Olympics
- Model prediction – Prediction of winner in 100-meter Freestyle Men's Swimming

(iv) Rajashree
- Data Exploration - Distribution of Women in Summer and Winter Games
- Model Prediction - Prediction of winner in Figure Skating Women's Singles(Winter)

(v) Tanmay
- Data Exploration - Relation between medals and size of contingent for USA across all years

- Model Prediction - Prediction of winner in 100-meter Freestyle Men’s Athletics

R-Code:

~~~
---
title: "R Project - Analysis of Olympics Data"
author: "Ashwin, Akanksha, Pragati, Rajashree, Tanmay"
date: "December 3, 2018"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

```{r}
library(data.table)
library(Hmisc)
library(dplyr)
library(ggplot2)
library(leaps)
library(gains)
library(gplots)
library(caret)

athlete_df <- fread("athlete_events.csv")
regions.df <- fread("noc_regions.csv")

colnames(regions.df) <- c("NOC","Country","Notes")
```

```
dim(athlete_df)
dim(regions.df)

glimpse(athlete_df)
glimpse(regions.df)

mergd.athlete.df <- merge(athlete_df, regions.df, by="NOC")
Full.athlete.df <- mergd.athlete.df[,c(2,3,4,5,6,7,8,10,11,12,13,14,1,16,15,17)]
Full.athlete.df$Medal <- as.factor(Full.athlete.df$Medal)

glimpse(Full.athlete.df)

miss.val <- data.frame(miss.val = sapply(Full.athlete.df, function(x) +   sum((is.na(x)))))
miss.val

summary(Full.athlete.df$Age)
summary(Full.athlete.df$Height)
summary(Full.athlete.df$Weight)

Full.athlete.df$Age <- as.numeric( impute( Full.athlete.df$Age, median))
Full.athlete.df$Height <- as.numeric( impute( Full.athlete.df$Height,   median))
Full.athlete.df$Weight <- as.numeric( impute( Full.athlete.df$Weight,   median))

country_NA <- filter(Full.athlete.df,is.na(Country))
unique(country_NA$Notes)
table(country_NA$Notes)
table(country_NA$Medal)

Full.athlete.df$Medal <- factor( Full.athlete.df$Medal, exclude = NULL,
                  levels = c("Bronze", "Gold", "Silver", NA),
                  labels = c("Bronze", "Gold", "Silver", "Dnw"))

Full.athlete.df <- na.omit(Full.athlete.df)
miss.val <- data.frame(miss.val=sapply(Full.athlete.df, function(x) + sum((is.na(x)))))
miss.val

gold.medals.df <- filter( Full.athlete.df, Medal == "Gold")
ggplot(data=gold.medals.df, aes(x=Age)) + geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
  labs(x = "Age", y = "Number of Gold Medalists",
     title = "Distribution of the age of gold medalists") +
  theme(plot.title = element_text(size=10, hjust = 0.5),
      axis.title = element_text(size=10),
      axis.text = element_text(size=10)) +
  theme_classic()

silver.medals.df <- filter(Full.athlete.df, Medal == "Silver")
 ggplot(data=silver.medals.df, aes(x=Age)) + geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
  labs(x = "Age", y = "Number of Silver Medalists",
     title = "Distribution of the age of Silver medalists") +
  theme(plot.title = element_text(size=10),
      axis.title = element_text(size=10),
      axis.text = element_text(size=10)) +
  theme_classic()

 women.sum.olym.df <- Full.athlete.df[ which( Full.athlete.df$Sex == "F"
                            & Full.athlete.df$Season == "Summer") , ]
ggplot(data=women.sum.olym.df, aes(x=Year)) +
  geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
  labs(x = "Year", y = "Number of Participants",
       title = "Participation of Women in Games overtime(Summer)") +
  theme(plot.title = element_text(size=10),
        axis.title = element_text(size=10),
        axis.text = element_text(size=10)) +
  theme_classic()

women.win.olym.df <- Full.athlete.df[ which( Full.athlete.df$Sex == "F"
                                      & Full.athlete.df$Season == "Winter") , ]
 ggplot(data=women.win.olym.df, aes(x=Year)) +
  geom_bar() +
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
  labs(x = "Year", y = "Number of Participants",
       title = "Participation of Women in Games overtime(Winter)") +
  theme(plot.title = element_text(size=10),
        axis.title = element_text(size=10),
        axis.text = element_text(size=10)) +
  theme_classic()

 gold.medals.usa.df <- filter(gold.medals.df, NOC == "USA")
gm.usa.df <- data.frame(head(sort(table(gold.medals.usa.df$Event),decreasing = T),15))
colnames(gm.usa.df) <- c("Sport Event","No of Gold Medals")


full.usa.df <- filter(Full.athlete.df, NOC == "USA")
goldcount <- as.data.frame(table(gold.medals.usa.df$Medal=="Gold",gold.medals.usa.df$Year))
players <- as.data.frame(table(full.usa.df$Year))
final.usa.df <- data.frame(Year = goldcount$Var2, Gold = goldcount$Freq, USA_contingent =
players$Freq)
```

```
Golds = goldcount$Freq

USA_Contingent_Size = players$Freq


ggplot(final.usa.df, aes(x= Gold, y= USA_contingent,  label=Year))+
  geom_point(color = "navy") +geom_text(aes(label=Year),hjust=0.5, vjust=-1, size = 2)


data.for.athletics.df <- filter(Full.athlete.df,
                         Sport == "Athletics")


correlation.in.athletics.df <- data.for.athletics.df[,c(4,5,6)]


cor.olympics.athletics <- cor(correlation.in.athletics.df)


heatmap.2(cor(correlation.in.athletics.df), Rowv = FALSE, Colv = FALSE,
        dendrogram = "none",cellnote = round(cor(correlation.in.athletics.df),5),
        notecol = "black", key = FALSE, trace = 'none', margins = c(10,10),
        main = "correlation in Athletics",xlab = "variables",ylab="variables")


data.for.wrestling.df <- filter(Full.athlete.df,
                         Sport == "Wrestling")


correlation.in.wrestling.df <- data.for.wrestling.df[,c(4,5,6)]


cor.olympics.wrestling <- cor(correlation.in.wrestling.df)
heatmap.2(cor(correlation.in.wrestling.df), Rowv = FALSE, Colv = FALSE,
        dendrogram = "none",cellnote = round(cor(correlation.in.wrestling.df),5),
        notecol = "black", key = FALSE, trace = 'none', margins = c(10,10),
        main = "correlation in Wrestling",xlab = "variables",ylab="variables")
```

```
swimming.df <- Full.athlete.df[which( Full.athlete.df$Sport == "Swimming"
                    & Full.athlete.df$Event == "Swimming Men's 100 metres Freestyle"
                    & Full.athlete.df$Year != 2016), ]


for (i in seq_along(swimming.df$Medal)){
  if (swimming.df$Medal[i] == "Dnw") {
    swimming.df$Outcome[i] <- 0
  } else {
    swimming.df$Outcome[i] <- 1
  }
}


swimming.log <- glm(Outcome ~ Age + Weight + Height + NOC, data = swimming.df, family =
"binomial")
options(scipen=999)


summary(swimming.log)


swimming.log.pred <- predict(swimming.log, swimming.df, type = "response")


swimming.cm <- t(table(swimming.df$Outcome , swimming.log.pred > 0.185, dnn = c("Actual
Class","Predictive Class")))


rownames(swimming.cm) <- colnames(swimming.cm)


confusionMatrix(swimming.cm, positive = "1", dnn = c("Actual Class","Predictive Class"))



swimming.df.16 <- Full.athlete.df[which( Full.athlete.df$Sport == "Swimming"
                    & Full.athlete.df$Event == "Swimming Men's 100 metres Freestyle"
```

```
                                   & Full.athlete.df$Year == 2016), ]

for (i in seq_along(swimming.df.16$Medal)){
  if (swimming.df.16$Medal[i] == "Dnw") {
    swimming.df.16$Outcome[i] <- 0
  } else {
    swimming.df.16$Outcome[i] <- 1
  }
}

swimming.df.pred.16 <- predict(swimming.log, swimming.df.16, type = "response")

swimming.df.predicted <- t(t(swimming.df.pred.16))

for (i in seq_along(swimming.df.predicted)){
  swimming.df.16$Predicted[i] <- as.numeric(swimming.df.pred.16[i])
}

swimming.cm.16 <- t(table(swimming.df.16$Outcome , swimming.df.16$Predicted > 0.185, dnn =
c("Actual Class","Predictive Class")))
rownames(swimming.cm.16) <- colnames(swimming.cm)

confusionMatrix(swimming.cm.16, positive = "1", dnn = c("Actual Class","Predictive Class"))

swimming.16.pred.winners <- swimming.df.16[which(swimming.df.16$Predicted >= 0.185),]
swimming.16.winners <- swimming.df.16[which(swimming.df.16$Outcome == 1),]

as.character(swimming.16.pred.winners$Name)

as.character(swimming.16.winners$Name)
```

```
library(gains)
gain <- gains(swimming.df$Outcome, swimming.log.pred, groups = 10)
### Plot Lift Chart
plot(c(0,gain$cume.pct.of.total*sum(swimming.df$Outcome))~c(0,gain$cume.obs),
   xlab = "# cases", ylab = "Cumulative", main = "", type = "l")
lines(c(0,sum(swimming.df$Outcome))~c(0, dim(swimming.df)[1]), lty = 5)

heights <- gain$mean.resp/mean(swimming.df$Outcome)
midpoints <- barplot(heights, names.arg = gain$depth,  ylim = c(0,9), col = "gold3",
                 xlab = "Percentile", ylab = "Mean Response",
                 main = "Decile-wise lift chart")

gain_valid <- gains(swimming.df.16$Outcome,swimming.df.pred.16,groups = 10)
### Plot Lift Chart
plot(c(0,gain_valid$cume.pct.of.total*sum(swimming.df.16$Outcome))~c(0,gain_valid$cume.obs),
   xlab = "# cases", ylab = "Cumulative", main = "", type = "l")
lines(c(0,sum(swimming.df.16$Outcome))~c(0, dim(swimming.df.16)[1]), lty = 5)

heights <- gain_valid$mean.resp/mean(swimming.df.16$Outcome)
midpoints <- barplot(heights, names.arg = gain_valid$depth,  ylim = c(0,9), col = "gold3",
                 xlab = "Percentile", ylab = "Mean Response",
                 main = "Decile-wise lift chart")

Athletics_100_meter_df <- Full.athlete.df[which( Full.athlete.df$Sport == "Athletics"
                                       & Full.athlete.df$Event == "Athletics Men's 100 metres"
                                       & Full.athlete.df$Year != 2016), ]

for (i in seq_along(Athletics_100_meter_df$Medal)){
  if (Athletics_100_meter_df$Medal[i] == "Dnw") {
    Athletics_100_meter_df$Outcome[i] <- 0

  } else {

    Athletics_100_meter_df$Outcome[i] <- 1

  }

}

table(Athletics_100_meter_df$Outcome)


Athletics_100_meter_log<- glm(Outcome ~ Age + Weight + Height + NOC, data =
Athletics_100_meter_df, family = "binomial")

options(scipen=999)


summary(Athletics_100_meter_log)


Athletics_100_meter_log_pred <- predict(Athletics_100_meter_log, Athletics_100_meter_df, type =
"response")


Athletics_100_meter_cm <- t(table(Athletics_100_meter_df$Outcome ,
Athletics_100_meter_log_pred > 0.175, dnn = c("Actual Class","Predictive Class")))

rownames(Athletics_100_meter_cm) <- colnames(Athletics_100_meter_cm)


myConfusion <- confusionMatrix(Athletics_100_meter_cm, positive = "1", dnn = c("Actual
Class","Predictive Class"))

myConfusion


Athletics_100_meter_Year_16_df <- Full.athlete.df[which( Full.athlete.df$Sport == "Athletics"

                                   & Full.athlete.df$Event == "Athletics Men's 100 metres"

                                   & Full.athlete.df$Year == 2016), ]


for (i in seq_along(Athletics_100_meter_Year_16_df$Medal)){

  if (Athletics_100_meter_Year_16_df$Medal[i] == "Dnw") {

    Athletics_100_meter_Year_16_df$Outcome[i] <- 0
  } else {
    Athletics_100_meter_Year_16_df$Outcome[i] <- 1
  }
}
table(Athletics_100_meter_Year_16_df$Outcome)


Athletics_100_meter_16_pred <- predict(Athletics_100_meter_log, Athletics_100_meter_Year_16_df,
type = "response")
Athletics_100_meter_16_predicted <- t(t(Athletics_100_meter_16_pred))


for (i in seq_along(Athletics_100_meter_16_predicted)){
  Athletics_100_meter_Year_16_df$Predicted[i] <- as.numeric(Athletics_100_meter_16_pred[i])
}


Athletics_100_meter_Year_16_cm <- t(table(Athletics_100_meter_Year_16_df$Outcome ,
Athletics_100_meter_Year_16_df$Predicted > 0.175, dnn = c("Actual Class","Predictive Class")))
rownames(Athletics_100_meter_Year_16_cm) <- colnames(Athletics_100_meter_Year_16_cm)


confusionMatrix(Athletics_100_meter_Year_16_cm, positive = "1", dnn = c("Actual Class","Predictive
Class"))


Athletics_100_meter_16.pred.winners <-
Athletics_100_meter_Year_16_df[which(Athletics_100_meter_Year_16_df$Predicted >= 0.175),]
Athletics_100_meter_16.winner <-
Athletics_100_meter_Year_16_df[which(Athletics_100_meter_Year_16_df$Outcome == 1),]


as.character(Athletics_100_meter_16.pred.winners$Name)


as.character(Athletics_100_meter_16.winner$Name)


gain_valid <- gains(Athletics_100_meter_Year_16_df$Outcome,Athletics_100_meter_16_pred,groups
= 10)
```

```
plot(c(0,gain_valid$cume.pct.of.total*sum(Athletics_100_meter_Year_16_df$Outcome))~c(0,gain_va
lid$cume.obs),

    xlab = "# cases", ylab = "Cumulative", main = "", type = "l")

lines(c(0,sum(Athletics_100_meter_Year_16_df$Outcome))~c(0,
dim(Athletics_100_meter_Year_16_df)[1]), lty = 5)


heights <- gain_valid$mean.resp/mean(Athletics_100_meter_Year_16_df$Outcome)

midpoints <- barplot(heights, names.arg = gain_valid$depth,  ylim = c(0,9), col = "gold3",

                  xlab = "Percentile", ylab = "Mean Response",

                  main = "Decile-wise lift chart")




Football.germany <- Full.athlete.df[which(Full.athlete.df$Event == "Football Men's Football"

                   & Full.athlete.df$Country == "Germany"

                   & Full.athlete.df$Year == 2016),]

Football.germany[,c(2,3,4,5,6,14,15)]


Football.df <- Full.athlete.df[which(Full.athlete.df$Sport == "Football"

                          & Full.athlete.df$Event == "Football Men's Football"),]

data.football.age<- data.frame(Football.df %>%

       group_by(Country, Year) %>%

         summarise(mean = mean(Age)))

colnames(data.football.age) <- c("Country","Year","Mean Age")

head(data.football.age)


data.football.height<- data.frame(Football.df %>%

                            group_by(Country, Year) %>%

                            summarise(mean = mean(Height)))

colnames(data.football.height) <- c("Country","Year","Mean Height")
```

```
data.football.weight<- data.frame(Football.df %>%
                                  group_by(Country, Year) %>%
                                  summarise(mean = mean(Weight)))
colnames(data.football.weight) <- c("Country","Year","Mean Weight")

data.football.medal <-Football.df[,c(8,14,15)]
data.football.medal<- unique(data.football.medal)

mergd.football.df <- merge(data.football.age,data.football.medal, by = c("Year", "Country"))

mergd.football.df <- merge(mergd.football.df,data.football.height, by = c("Year", "Country"))

mergd.football.df <- merge(mergd.football.df,data.football.weight, by = c("Year", "Country"))

#Rearranging the columns:
mergd.football.df <- mergd.football.df[, c(1,2,3,5,6,4)]

colnames(mergd.football.df) <- c("Year", "Country", "Age", "Height", "Weight", "Medal")
tail(mergd.football.df)

train.football.df <- mergd.football.df[which(mergd.football.df$Year != 2016), ]

for (i in seq_along(train.football.df$Medal)){
  if (train.football.df$Medal[i] == "Dnw") {
    train.football.df$Outcome[i] <- 0
  } else {
    train.football.df$Outcome[i] <- 1
  }
}
```

```
train.football.log <- glm(Outcome ~ Age + Weight + Height + Country,
                    data = train.football.df,
                    family = "binomial")
options(scipen=999)


train.football.log.pred <- predict(train.football.log, train.football.df,
                            type = "response")


train.football.cm <- t(table(train.football.df$Outcome , train.football.log.pred > 0.28, dnn = c("Actual
Class","Predictive Class")))
rownames(train.football.cm) <- colnames(train.football.cm)


confusionMatrix(train.football.cm, positive = "1", dnn = c("Actual Class","Predictive Class"))


valid.football.16 <- mergd.football.df[which( mergd.football.df$Year == 2016), ]


for (i in seq_along(valid.football.16$Medal)){
  if (valid.football.16$Medal[i] == "Dnw") {
    valid.football.16$Outcome[i] <- 0
  } else {
    valid.football.16$Outcome[i] <- 1
  }
}




valid.football.16.pred <- predict(train.football.log, valid.football.16, type = "response")
valid.football.16.predicted <- t(t(valid.football.16.pred))


for (i in seq_along(valid.football.16.predicted)){
  valid.football.16$Predicted[i] <- as.numeric(valid.football.16.pred[i])
```

```
}


valid.football.16.cm <- t(table(valid.football.16$Outcome , valid.football.16$Predicted > 0.28, dnn =
c("Actual Class","Predictive Class")))

rownames(valid.football.16.cm) <- colnames(valid.football.16.cm)


confusionMatrix(valid.football.16.cm, positive = "1", dnn = c("Actual Class","Predictive Class"))


valid.football.16.pred.winners <- valid.football.16[which(valid.football.16$Predicted >= 0.28),]

valid.football.16.winners <- valid.football.16[which(valid.football.16$Outcome == 1),]


as.character(valid.football.16.pred.winners$Country)


as.character(valid.football.16.winners$Country)



Tennis.df <- Full.athlete.df[which(Full.athlete.df$Sport == "Tennis"
                          & Full.athlete.df$Event == "Tennis Men's Doubles"), ]



data.tennis.age <- data.frame(aggregate(Tennis.df$Age,
                             by = list(Tennis.df$Team,Tennis.df$Year),
                             FUN = mean, drop = FALSE))

colnames(data.tennis.age) <- c("Team","Year","Mean Age")


data.tennis.age <- data.tennis.age[!is.nan(data.tennis.age$`Mean Age`),]

head(data.tennis.age)


data.tennis.height <- data.frame(aggregate(Tennis.df$Height,
                               by = list(Tennis.df$Team,Tennis.df$Year),
                                 FUN = mean, drop = FALSE))

colnames(data.tennis.height) <- c("Team","Year","Mean Height")
data.tennis.height <- data.tennis.height[!is.nan(data.tennis.height$`Mean Height`),]
head(data.tennis.height)

data.tennis.weight <- data.frame(aggregate(Tennis.df$Weight,
                                 by = list(Tennis.df$Team,Tennis.df$Year),
                                 FUN = mean, drop = FALSE))
colnames(data.tennis.weight) <- c("Team","Year","Mean Weight")
data.tennis.weight <- data.tennis.weight[!is.nan(data.tennis.weight$`Mean Weight`),]
head(data.tennis.weight)
colnames(Tennis.df)
data.tennis.medal <-Tennis.df[,c(7,8,14,15)]
data.tennis.medal<- unique(data.tennis.medal)
head(data.tennis.medal)
mergd.tennis.df <- merge(data.tennis.age,data.tennis.medal, by = c("Year", "Team"))
mergd.tennis.df <- merge(mergd.tennis.df,data.tennis.height, by = c("Year", "Team"))
mergd.tennis.df <- merge(mergd.tennis.df,data.tennis.weight, by = c("Year", "Team"))
mergd.tennis.df <- mergd.tennis.df[, c(1,2,4,3,6,7,5)]
colnames(mergd.tennis.df) <- c("Year", "Team" ,"Country", "Age", "Height", "Weight", "Medal")
tail(mergd.tennis.df)

train.tennis.df <- mergd.tennis.df[which(mergd.tennis.df$Year != 2016), ]

for (i in seq_along(train.tennis.df$Medal)){
  if (train.tennis.df$Medal[i] == "Dnw") {
    train.tennis.df$Outcome[i] <- 0
  } else {
    train.tennis.df$Outcome[i] <- 1
```

```
 }
}
table(train.tennis.df$Outcome)


train.tennis.log <- glm(Outcome ~ Age + Weight + Height + Country,
                        data = train.tennis.df,
                        family = "binomial")
options(scipen=999)


train.tennis.log.pred <- predict(train.tennis.log, train.tennis.df,
                                 type = "response")


train.tennis.cm <- t(table(train.tennis.df$Outcome , train.tennis.log.pred > 0.13, dnn = c("Actual
Class","Predictive Class")))
rownames(train.tennis.cm) <- colnames(train.tennis.cm)
train.tennis.cm
confusionMatrix(train.tennis.cm, positive = "1", dnn = c("Actual Class","Predictive Class"))


valid.tennis.16 <- mergd.tennis.df[which( mergd.tennis.df$Year == 2016), ]


for (i in seq_along(valid.tennis.16$Medal)){
 if (valid.tennis.16$Medal[i] == "Dnw") {
   valid.tennis.16$Outcome[i] <- 0
 } else {
   valid.tennis.16$Outcome[i] <- 1
 }
}



valid.tennis.16.pred <- predict(train.tennis.log, valid.tennis.16, type = "response")
valid.tennis.16.predicted <- t(t(valid.tennis.16.pred))

for (i in seq_along(valid.tennis.16.predicted)){

  valid.tennis.16$Predicted[i] <- as.numeric(valid.tennis.16.pred[i])

}

valid.tennis.16.cm <- t(table(valid.tennis.16$Outcome , valid.tennis.16$Predicted > 0.13, dnn =
c("Actual Class","Predictive Class")))

rownames(valid.tennis.16.cm) <- colnames(valid.tennis.16.cm)

valid.tennis.16.cm

confusionMatrix(valid.tennis.16.cm, positive = "1", dnn = c("Actual Class","Predictive Class"))


valid.tennis.16.pred.winners <- valid.tennis.16[which(valid.tennis.16$Predicted >= 0.13),]

valid.tennis.16.winners <- valid.tennis.16[which(valid.tennis.16$Outcome == 1),]


as.character(valid.tennis.16.pred.winners$Team)


as.character(valid.tennis.16.winners$Team)


Figure_Skating_Women_Single.df = Full.athlete.df[which(Full.athlete.df$Sport == "Figure Skating"

                                & Full.athlete.df$Event == "Figure Skating Women's Singles"

                                & Full.athlete.df$Year != 2014),]


for (i in seq_along(Figure_Skating_Women_Single.df$Medal)){

  if (Figure_Skating_Women_Single.df$Medal[i] == "Dnw") {

    Figure_Skating_Women_Single.df$Outcome[i] <- 0

  } else {

    Figure_Skating_Women_Single.df$Outcome[i] <- 1

  }

}

table(Figure_Skating_Women_Single.df$Outcome)
```

```
Figure_Skating_Women_Single_log<- glm(Outcome ~ Age + Weight + Height + NOC, data =
Figure_Skating_Women_Single.df, family = "binomial")

options(scipen=999)


Figure_Skating_Women_Single_pred =
predict(Figure_Skating_Women_Single_log,Figure_Skating_Women_Single.df, type = "response")


Figure_Skating_Women_Single_cm <- t(table(Figure_Skating_Women_Single.df$Outcome ,
Figure_Skating_Women_Single_pred > 0.11, dnn = c("Actual Class","Predictive Class")))

Figure_Skating_Women_Single_cm

rownames(Figure_Skating_Women_Single_cm) <- colnames(Figure_Skating_Women_Single_cm)


myConfusion = confusionMatrix(Figure_Skating_Women_Single_cm, positive = "1", dnn = c("Actual
Class","Predictive Class"))

myConfusion


Figure_Skating_Women_Single_14.df = Full.athlete.df[which(Full.athlete.df$Sport == "Figure
Skating"

                            & Full.athlete.df$Event == "Figure Skating Women's Singles"

                            & Full.athlete.df$Year == 2014), ]


for (i in seq_along(Figure_Skating_Women_Single_14.df$Medal)){

 if (Figure_Skating_Women_Single_14.df$Medal[i] == "Dnw") {

  Figure_Skating_Women_Single_14.df$Outcome[i] <- 0

 } else {

  Figure_Skating_Women_Single_14.df$Outcome[i] <- 1

 }

}

table(Figure_Skating_Women_Single_14.df$Outcome)


Figure_Skating_Women_Single_14_pred <- predict(Figure_Skating_Women_Single_log,
Figure_Skating_Women_Single_14.df, type = "response")
```

```
Figure_Skating_Women_Single_14_predicted <- t(t(Figure_Skating_Women_Single_14_pred))


for (i in seq_along(Figure_Skating_Women_Single_14_predicted)){

  Figure_Skating_Women_Single_14.df$Predicted[i] <-
as.numeric(Figure_Skating_Women_Single_14_pred[i])

}


Figure_Skating_Women_Single_14_cm <- t(table(Figure_Skating_Women_Single_14.df$Outcome ,
Figure_Skating_Women_Single_14.df$Predicted > 0.11, dnn = c("Actual Class","Predictive Class")))


rownames(Figure_Skating_Women_Single_14_cm) <-
colnames(Figure_Skating_Women_Single_14_cm)


confusionMatrix(Figure_Skating_Women_Single_14_cm, positive = "1", dnn = c("Actual
Class","Predictive Class"))


Figure_Skating_Women_Single_14_pred_winner <-
Figure_Skating_Women_Single_14.df[which(Figure_Skating_Women_Single_14.df$Predicted >=
0.11),]

Figure_Skating_Women_Single_14_winner <-
Figure_Skating_Women_Single_14.df[which(Figure_Skating_Women_Single_14.df$Outcome == 1),]


as.character(Figure_Skating_Women_Single_14_pred_winner$Name)


as.character(Figure_Skating_Women_Single_14_winner$Name)
```