Pokemon Go - Analysis

Zohaib Sheikh

12/11/2021

Loading relevant libraries

```
library(tidyverse)
library(lubridate)
library(caret)
library(readxl)
library("ggplot2")
library("sqldf")
library(ggrepel)
library(rpart)
library(pROC)
library('glmnet')
library(ROCR)
library(e1071)
```

1 - Reading the Data

```
df<-read_excel('C:/Users/Zohaib Sheikh/Desktop/Summer Int/Assignments/pokemon_data_science.xlsx')
```

2 - Understanding the Data

```
dim(df)
```

```
## [1] 721  23
```

```
str(df)
```

```
## tibble [721 x 23] (S3: tbl_df/tbl/data.frame)
##  $ Number          : num [1:721] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Name            : chr [1:721] "Bulbasaur" "Ivysaur" "Venusaur" "Charmander" ...
##  $ Type_1          : chr [1:721] "Grass" "Grass" "Grass" "Fire" ...
##  $ Type_2          : chr [1:721] "Poison" "Poison" "Poison" NA ...
##  $ Total           : num [1:721] 318 405 525 309 405 534 314 405 530 195 ...
##  $ HP              : num [1:721] 45 60 80 39 58 78 44 59 79 45 ...
##  $ Attack          : num [1:721] 49 62 82 52 64 84 48 63 83 30 ...
##  $ Defense         : num [1:721] 49 63 83 43 58 78 65 80 100 35 ...
##  $ Sp_Atk          : num [1:721] 65 80 100 60 80 109 50 65 85 20 ...
##  $ Sp_Def          : num [1:721] 65 80 100 50 65 85 64 80 105 20 ...
##  $ Speed           : num [1:721] 45 60 80 65 80 100 43 58 78 45 ...
```

```
##  $ Generation      : num [1:721] 1 1 1 1 1 1 1 1 1 1 ...
##  $ isLegendary     : logi [1:721] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Color           : chr [1:721] "Green" "Green" "Green" "Red" ...
##  $ hasGender       : logi [1:721] TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ Pr_Male         : num [1:721] 0.875 0.875 0.875 0.875 0.875 0.875 0.875 0.875 0.875 0.5 ...
##  $ Egg_Group_1     : chr [1:721] "Monster" "Monster" "Monster" "Monster" ...
##  $ Egg_Group_2     : chr [1:721] "Grass" "Grass" "Grass" "Dragon" ...
##  $ hasMegaEvolution: logi [1:721] FALSE FALSE TRUE FALSE FALSE TRUE ...
##  $ Height_m        : num [1:721] 0.71 0.99 2.01 0.61 1.09 1.7 0.51 0.99 1.6 0.3 ...
##  $ Weight_kg       : num [1:721] 6.9 13 100 8.5 19 90.5 9 22.5 85.5 2.9 ...
##  $ Catch_Rate      : num [1:721] 45 45 45 45 45 45 45 45 45 255 ...
##  $ Body_Style      : chr [1:721] "quadruped" "quadruped" "quadruped" "bipedal_tailed" ...
```

```
head(df)
```

```
## # A tibble: 6 x 23
##   Number Name       Type_1 Type_2 Total    HP Attack Defense Sp_Atk Sp_Def Speed
##    <dbl> <chr>      <chr>  <chr>  <dbl> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
## 1      1 Bulbasaur  Grass  Poison   318    45     49      49     65     65    45
## 2      2 Ivysaur    Grass  Poison   405    60     62      63     80     80    60
## 3      3 Venusaur   Grass  Poison   525    80     82      83    100    100    80
## 4      4 Charmander Fire   <NA>     309    39     52      43     60     50    65
## 5      5 Charmeleon Fire   <NA>     405    58     64      58     80     65    80
## 6      6 Charizard  Fire   Flying   534    78     84      78    109     85   100
## # ... with 12 more variables: Generation <dbl>, isLegendary <lgl>, Color <chr>,
## #   hasGender <lgl>, Pr_Male <dbl>, Egg_Group_1 <chr>, Egg_Group_2 <chr>,
## #   hasMegaEvolution <lgl>, Height_m <dbl>, Weight_kg <dbl>, Catch_Rate <dbl>,
## #   Body_Style <chr>
```

```
summary(df)
```

```
##      Number        Name              Type_1            Type_2
##  Min.   :  1   Length:721         Length:721         Length:721
##  1st Qu.:181   Class :character   Class :character   Class :character
##  Median :361   Mode  :character   Mode  :character   Mode  :character
##  Mean   :361
##  3rd Qu.:541
##  Max.   :721
##
##      Total             HP             Attack          Defense
##  Min.   :180.0   Min.   :  1.00   Min.   :  5.00   Min.   :  5.00
##  1st Qu.:320.0   1st Qu.: 50.00   1st Qu.: 53.00   1st Qu.: 50.00
##  Median :424.0   Median : 65.00   Median : 74.00   Median : 65.00
##  Mean   :417.9   Mean   : 68.38   Mean   : 75.01   Mean   : 70.81
##  3rd Qu.:499.0   3rd Qu.: 80.00   3rd Qu.: 95.00   3rd Qu.: 85.00
##  Max.   :720.0   Max.   :255.00   Max.   :165.00   Max.   :230.00
##
##      Sp_Atk           Sp_Def           Speed          Generation
##  Min.   : 10.00   Min.   : 20.00   Min.   :  5.00   Min.   :1.000
##  1st Qu.: 45.00   1st Qu.: 50.00   1st Qu.: 45.00   1st Qu.:2.000
##  Median : 65.00   Median : 65.00   Median : 65.00   Median :3.000
##  Mean   : 68.74   Mean   : 69.29   Mean   : 65.71   Mean   :3.323
##  3rd Qu.: 90.00   3rd Qu.: 85.00   3rd Qu.: 85.00   3rd Qu.:5.000
```

```
##  Max.   :154.00   Max.   :230.00   Max.   :160.00   Max.   :6.000
##
##  isLegendary        Color           hasGender          Pr_Male
##  Mode :logical   Length:721         Mode :logical   Min.   :0.0000
##  FALSE:675       Class :character   FALSE:77        1st Qu.:0.5000
##  TRUE :46        Mode  :character   TRUE :644       Median :0.5000
##                                                     Mean   :0.5534
##                                                     3rd Qu.:0.5000
##                                                     Max.   :1.0000
##                                                     NA's   :77
##  Egg_Group_1        Egg_Group_2        hasMegaEvolution    Height_m
##  Length:721         Length:721         Mode :logical    Min.   : 0.100
##  Class :character   Class :character   FALSE:675        1st Qu.: 0.610
##  Mode  :character   Mode  :character   TRUE :46         Median : 0.990
##                                                        Mean   : 1.145
##                                                        3rd Qu.: 1.400
##                                                        Max.   :14.500
##
##    Weight_kg        Catch_Rate     Body_Style
##  Min.   :  0.10   Min.   :  3.0   Length:721
##  1st Qu.:  9.40   1st Qu.: 45.0   Class :character
##  Median : 28.00   Median : 65.0   Mode  :character
##  Mean   : 56.77   Mean   :100.2
##  3rd Qu.: 61.00   3rd Qu.:180.0
##  Max.   :950.00   Max.   :255.0
##
```

```
colSums(is.na(df))
```

```
##           Number             Name           Type_1           Type_2
##                0                0                0              371
##            Total               HP           Attack          Defense
##                0                0                0                0
##          Sp_Atk           Sp_Def            Speed       Generation
##                0                0                0                0
##      isLegendary            Color        hasGender          Pr_Male
##                0                0                0               77
##      Egg_Group_1      Egg_Group_2 hasMegaEvolution         Height_m
##                0              530                0                0
##        Weight_kg       Catch_Rate       Body_Style
##                0                0                0
```

```
df%>%select_if(is.character)%>%select(-c(Name))%>%sapply(unique)
```

```
## $Type_1
##  [1] "Grass"    "Fire"     "Water"    "Bug"      "Normal"   "Poison"
##  [7] "Electric" "Ground"   "Fairy"    "Fighting" "Psychic"  "Rock"
## [13] "Ghost"    "Ice"      "Dragon"   "Dark"     "Steel"    "Flying"
##
## $Type_2
##  [1] "Poison"   NA         "Flying"   "Ground"   "Fairy"    "Grass"
##  [7] "Fighting" "Psychic"  "Steel"    "Ice"      "Rock"     "Water"
## [13] "Electric" "Fire"     "Dragon"   "Dark"     "Ghost"    "Bug"
```

```
## [19] "Normal"
##
## $Color
##  [1] "Green"  "Red"    "Blue"   "White"  "Brown"  "Yellow" "Purple" "Pink"
##  [9] "Grey"   "Black"
##
## $Egg_Group_1
##  [1] "Monster"      "Bug"          "Flying"       "Field"        "Undiscovered"
##  [6] "Fairy"        "Grass"        "Water_1"      "Human-Like"   "Water_3"
## [11] "Mineral"      "Amorphous"    "Water_2"      "Ditto"        "Dragon"
##
## $Egg_Group_2
##  [1] "Grass"      "Dragon"     "Water_1"    NA           "Fairy"
##  [6] "Field"      "Water_3"    "Water_2"    "Flying"     "Bug"
## [11] "Human-Like" "Amorphous"  "Mineral"    "Monster"
##
## $Body_Style
##  [1] "quadruped"        "bipedal_tailed"   "insectoid"        "serpentine_body"
##  [5] "four_wings"       "two_wings"        "bipedal_tailless" "head_legs"
##  [9] "head_base"        "multiple_bodies"  "several_limbs"    "head_arms"
## [13] "with_fins"        "head_only"
```

The Dataset has 12 categorical variables and 11 numeric variables. Our response variable is 'isLegendary' having 6.4% of total Pokemon as Legendary Pokemons(our class of Interest). The dataset looks structured which we will explore further.

3 - Data Cleaning - Handling Null Values

```
colSums(is.na(df))
```

```
##           Number             Name           Type_1           Type_2
##                0                0                0              371
##            Total               HP           Attack          Defense
##                0                0                0                0
##          Sp_Atk           Sp_Def            Speed       Generation
##                0                0                0                0
##      isLegendary            Color        hasGender          Pr_Male
##                0                0                0               77
##      Egg_Group_1      Egg_Group_2 hasMegaEvolution         Height_m
##                0              530                0                0
##        Weight_kg       Catch_Rate       Body_Style
##                0                0                0
```

- Here we looked at the null values across the dataset. Three variables have null values: 'Type 2', 'Egg_group 2' and 'Pr_male'.
- We will handle the three variables separately.
  - Type 2: These are 371 pokemons with undefined type_2. Hence we will replace the null values for these with the value 'No Type 2'.
  - Egg_group 2: These are 530 pokemons with undefined Egg group 2. Hence we will replace the null values for these with the value 'No Egg_Group 2'.

```
df$Type_2[is.na(df$Type_2)]<-'No Type 2'
unique(df$Type_2)
```

```
##  [1] "Poison"    "No Type 2" "Flying"    "Ground"    "Fairy"     "Grass"
##  [7] "Fighting"  "Psychic"   "Steel"     "Ice"       "Rock"      "Water"
## [13] "Electric"  "Fire"      "Dragon"    "Dark"      "Ghost"     "Bug"
## [19] "Normal"
```

```
df$Egg_Group_2[is.na(df$Egg_Group_2)]<-'No Egg_Group 2'
```

- Pr_male : This is a tricky column to handle the null values. There are 77 pokemons null 'pr_male' values. To handle these, let's dig a bit deeper into the data. Looking at these 77 records, we see that 40 of 77 are legendary pokemons which is our class of interest. As we have only 6.4% fill rate for our class of interest, we can't remove these. Also, when we look at the column 'hasGender', we observe that there are 77 pokemons with undefined gender and these 77 pokemons are the same pokemons which have null values for their 'Pr_male' column. So, in order to handle these, we impute these null values with 0.5 as the gender for these is undefined.

```
df%>%group_by(hasGender)%>%summarise(n=n())
```

```
## # A tibble: 2 x 2
##   hasGender     n
##   <lgl>     <int>
## 1 FALSE        77
## 2 TRUE        644
```

```
mean(df$hasGender[is.na(df$Pr_Male)])
```

```
## [1] 0
```

```
unique(df$hasGender[is.na(df$Pr_Male)])
```

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

```
unique(df$isLegendary[is.na(df$Pr_Male)])
```

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

```
df%>%filter(is.na(df$Pr_Male)==TRUE)%>%group_by(isLegendary)%>%summarise(n=n())
```

```
## # A tibble: 2 x 2
##   isLegendary     n
##   <lgl>       <int>
## 1 FALSE          37
## 2 TRUE           40
```

```
df$Pr_Male[is.na(df$Pr_Male)]<-0.5
```

Next we will convert all our categorical variables as Factors with different labels.

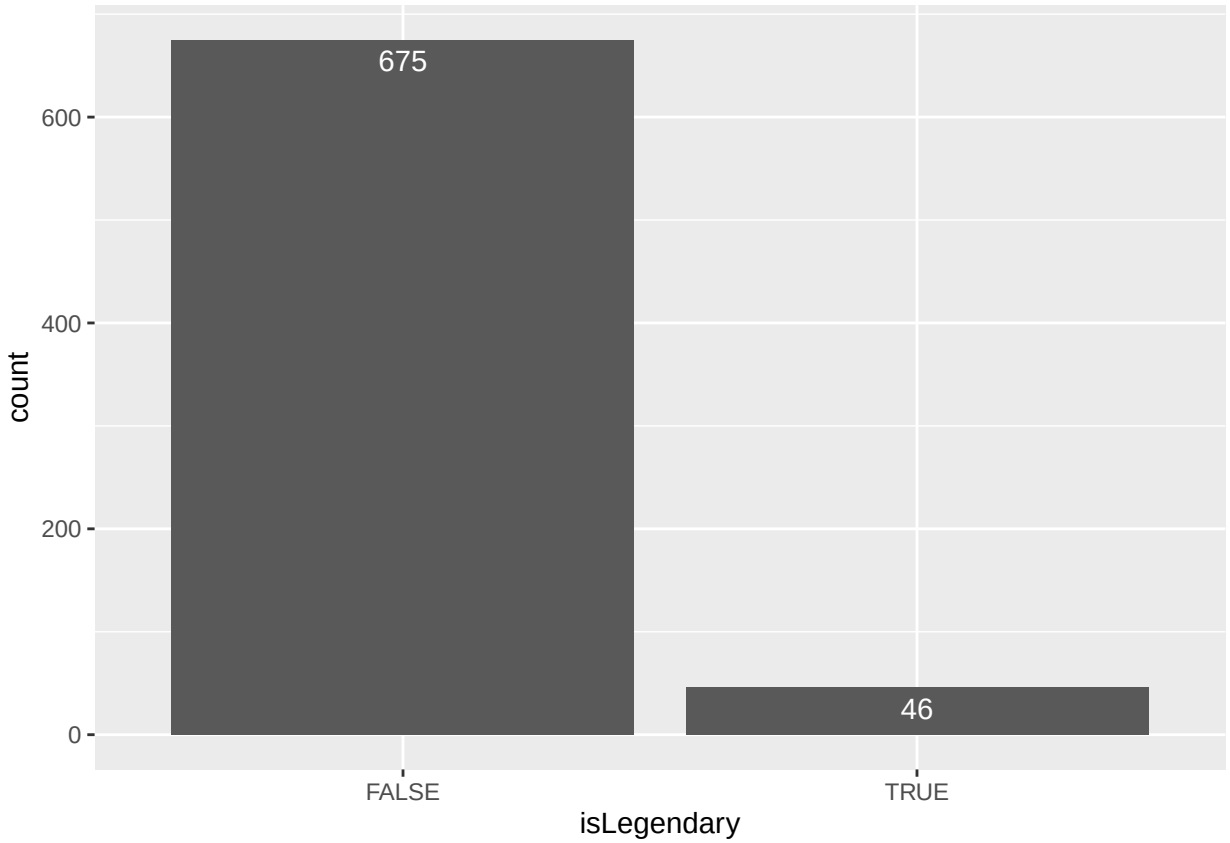
4 - Data Cleaning - Changing the char and logical variables to Factors

```
df <- as.data.frame(unclass(df),stringsAsFactors = TRUE)
df$isLegendary<-as.factor(df$isLegendary)
df$hasMegaEvolution<-as.factor(df$hasMegaEvolution)
df$hasGender<-as.factor(df$hasGender)
df$Number<-as.factor(df$Number)
df$Generation<-as.factor(df$Generation)
```

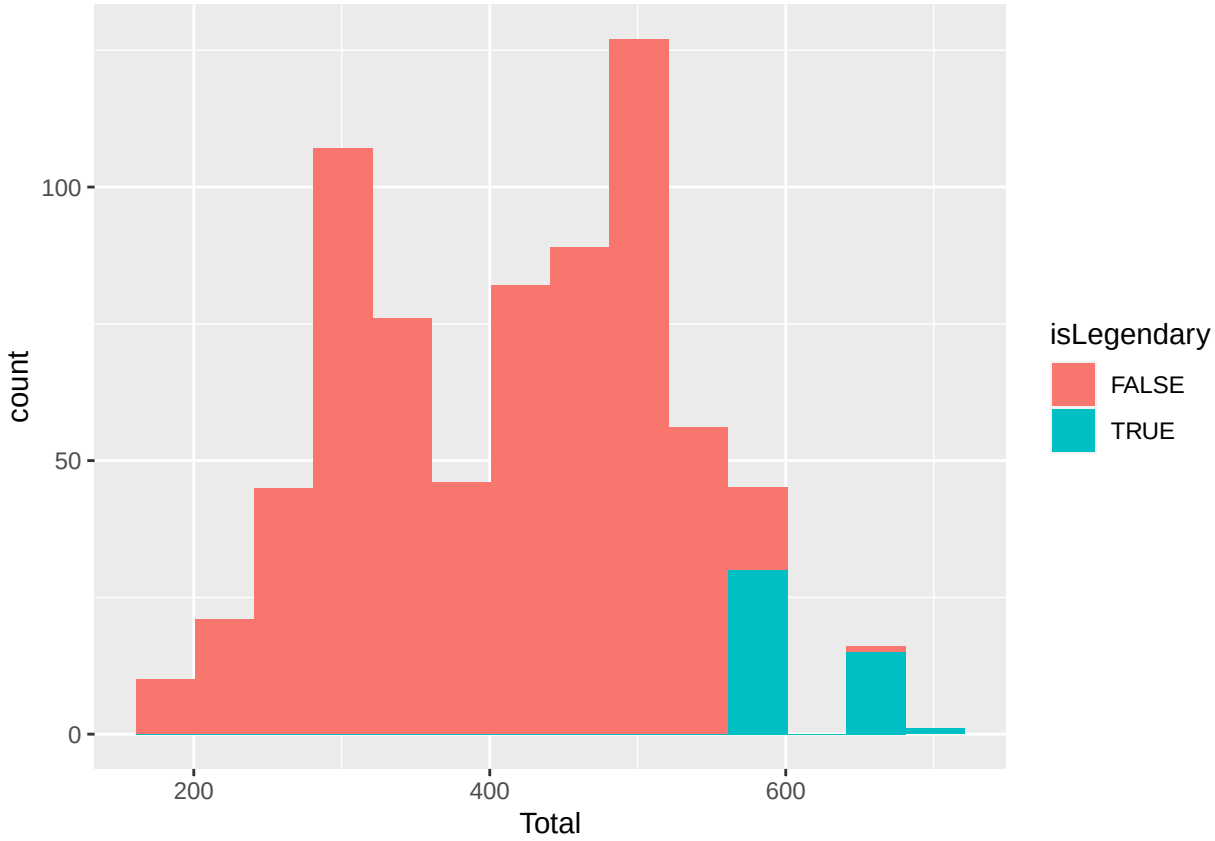
Next, we will explore the data more by slicing, dicing and viewing from different angles. We will also look at some quick visualizations.

5 - Data Exploration - Visualization

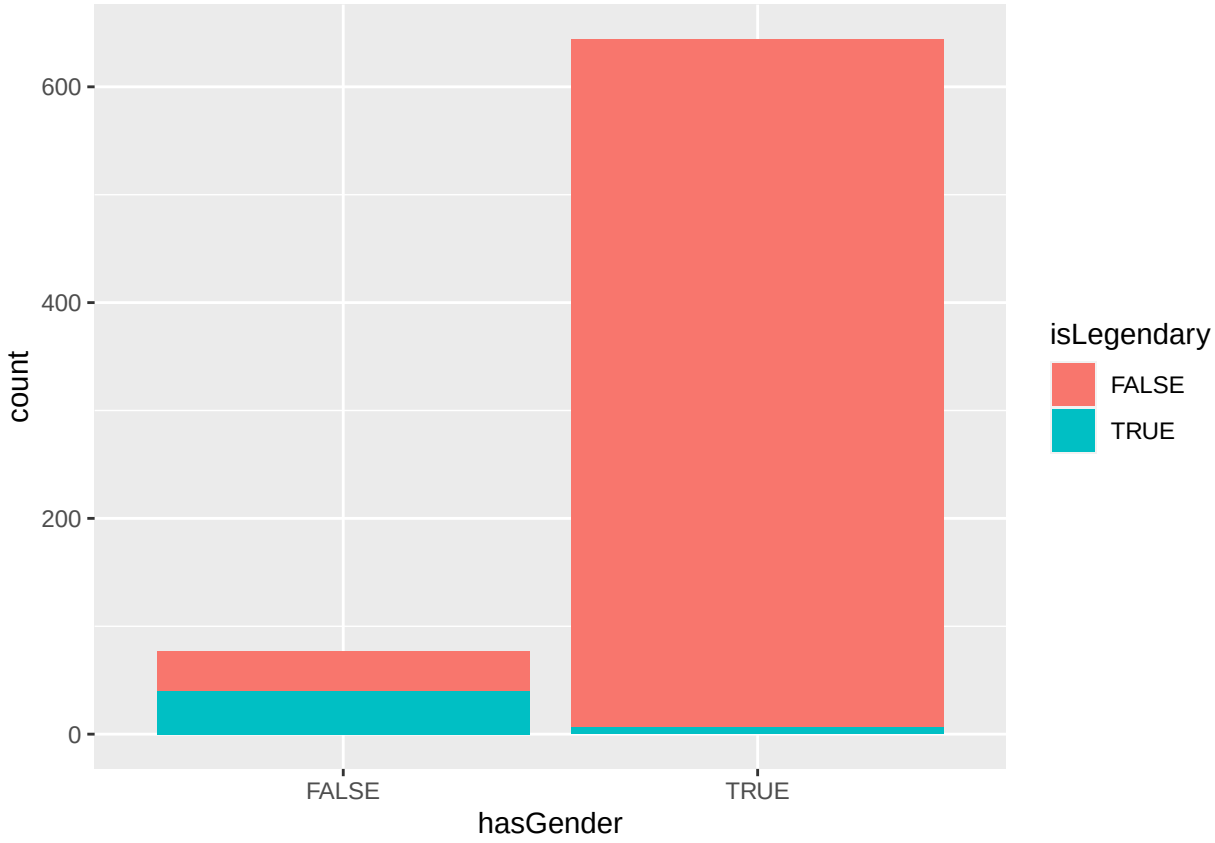
```
ggplot(df, aes(isLegendary)) +
 geom_bar()+ geom_text(aes(label = ..count..), stat = "count", vjust = 1.5, colour = "white")
```



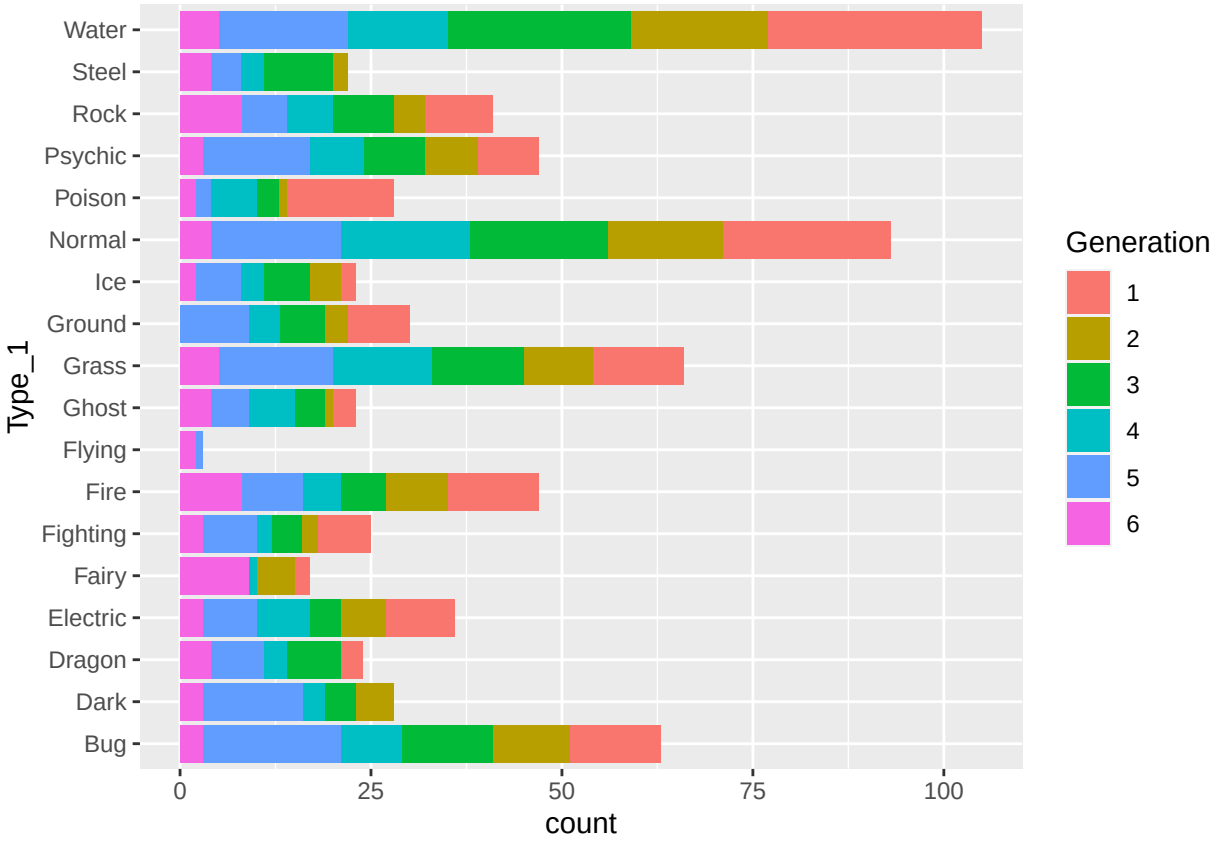

```
ggplot(df, aes(Total, fill = isLegendary)) +
  geom_histogram(binwidth = 40,boundary = 1)
```



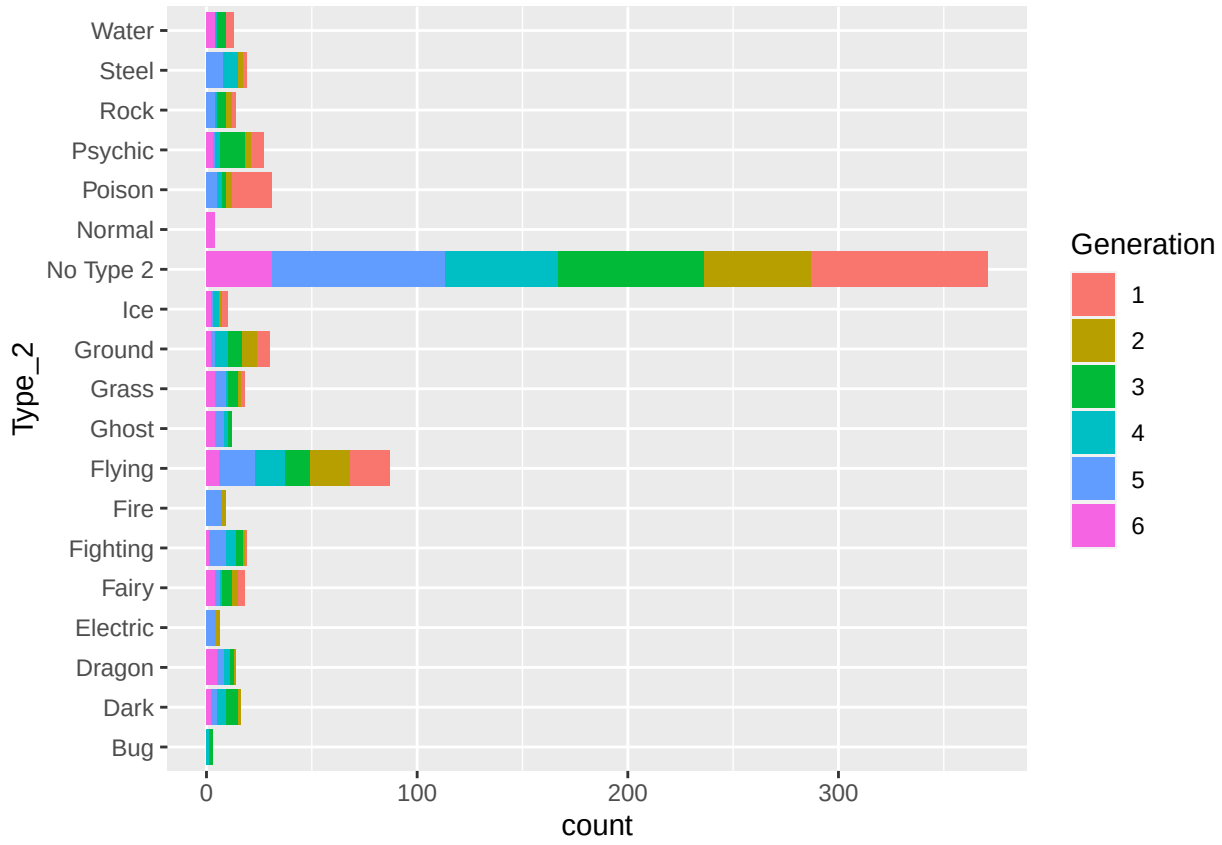

```
ggplot(df, aes(hasGender, fill = isLegendary)) +
 geom_bar()
```



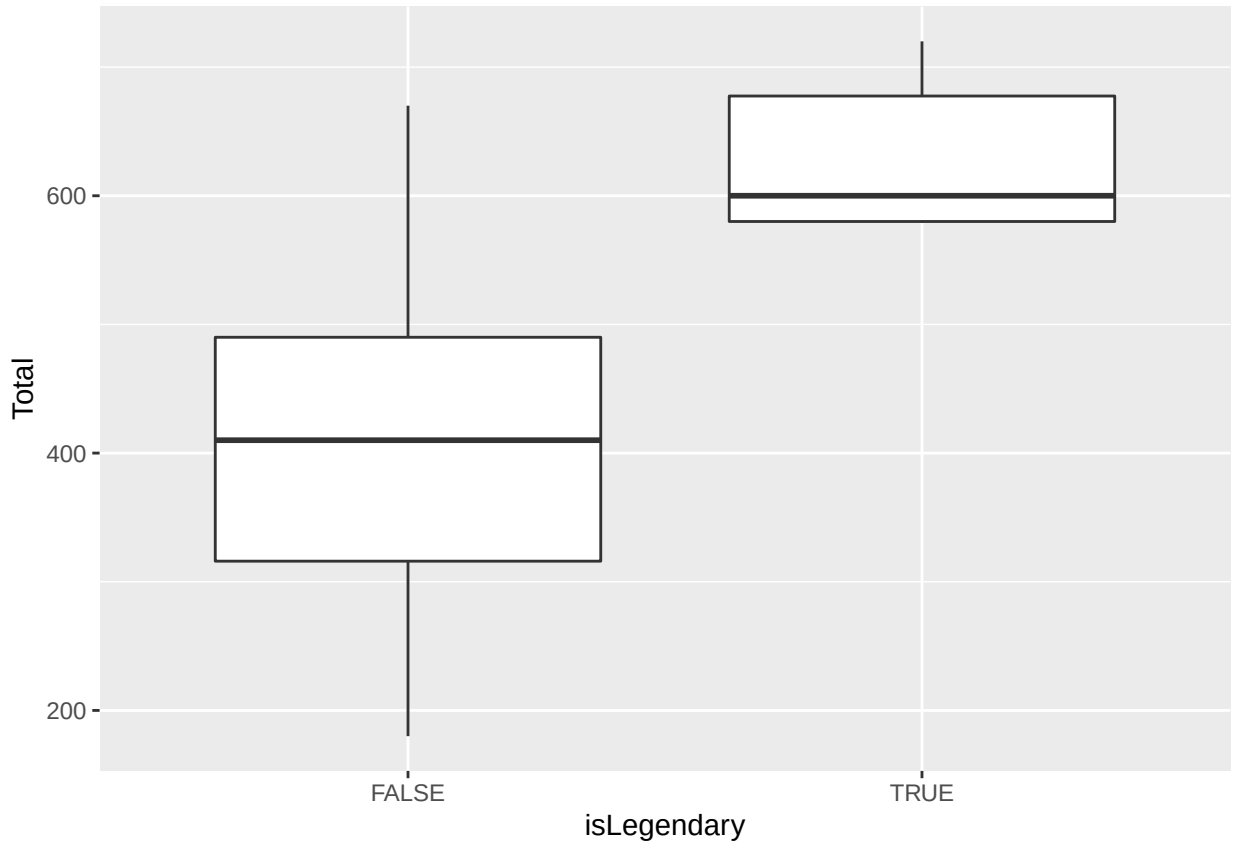

```
df %>% ggplot(aes(x= Type_1, fill = Generation)) + geom_bar() + coord_flip()
```



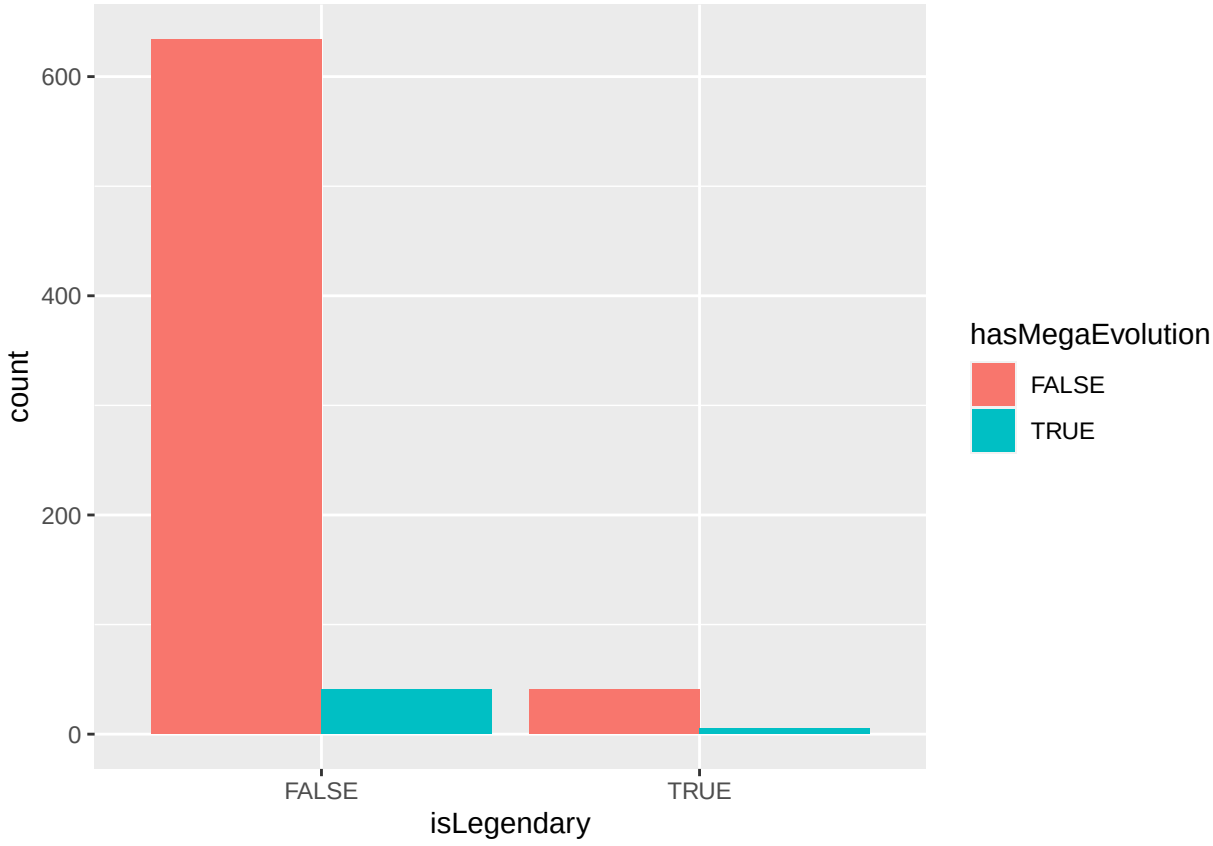

```
df %>% ggplot(aes(x= Type_2, fill = Generation)) + geom_bar() + coord_flip()
```



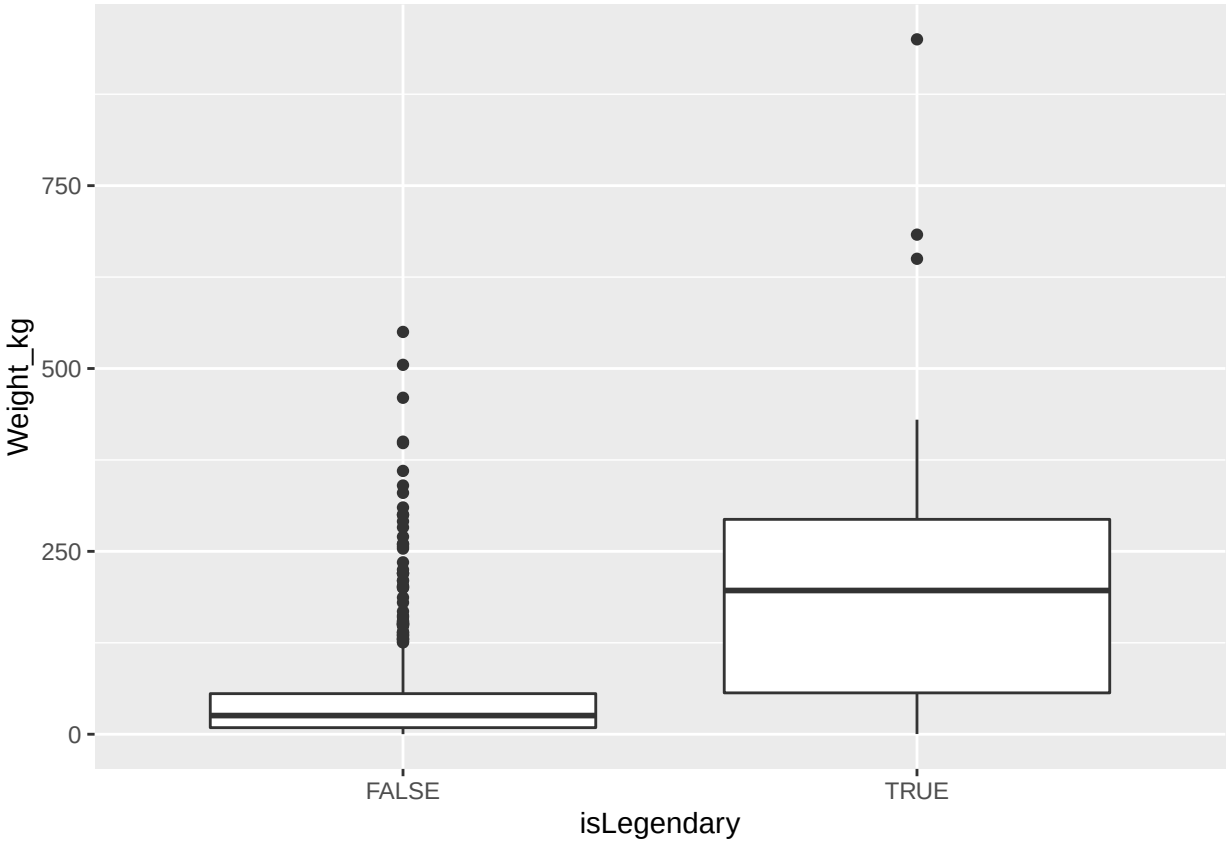

```
df %>% ggplot(aes(x= isLegendary, y = Total)) + geom_boxplot()
```



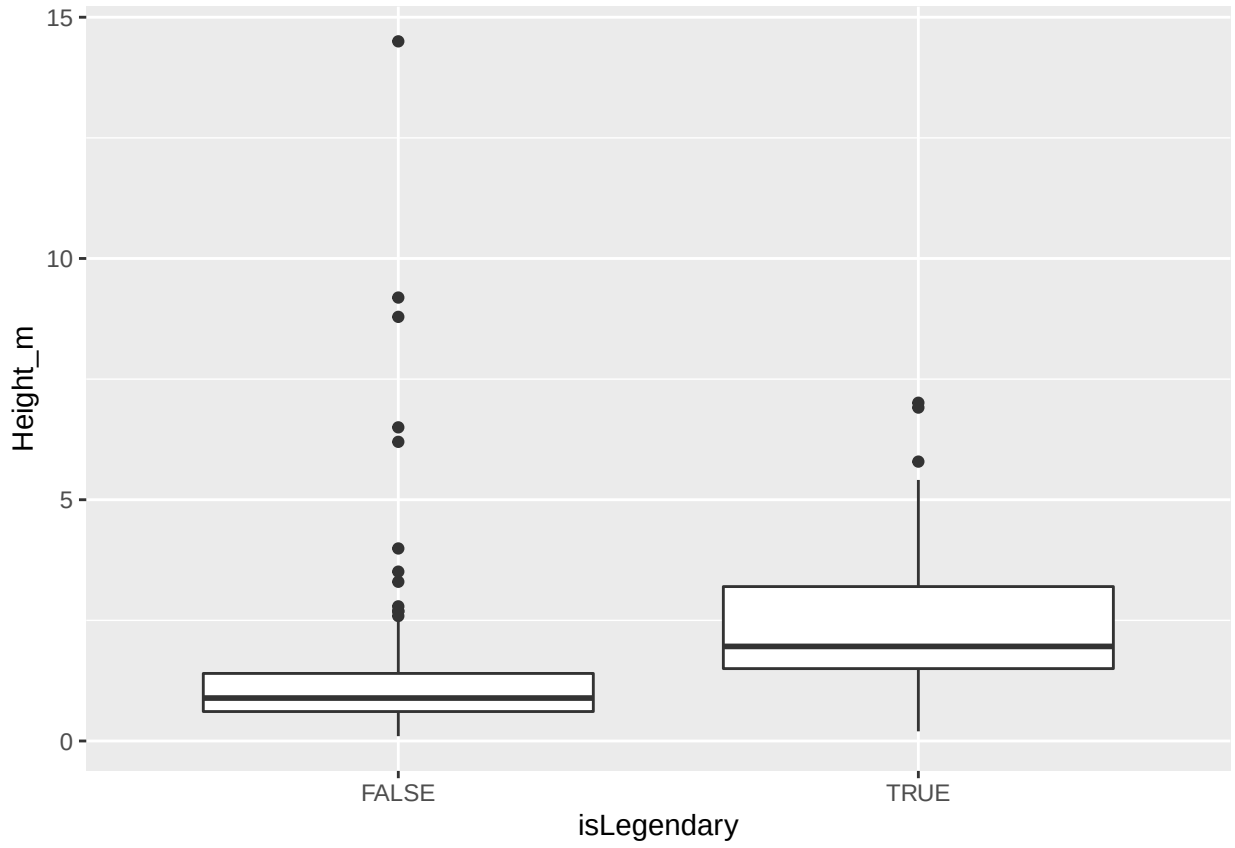

```
ggplot(df, aes(isLegendary, ..count..)) + geom_bar(aes(fill = hasMegaEvolution), position = "dodge")
```



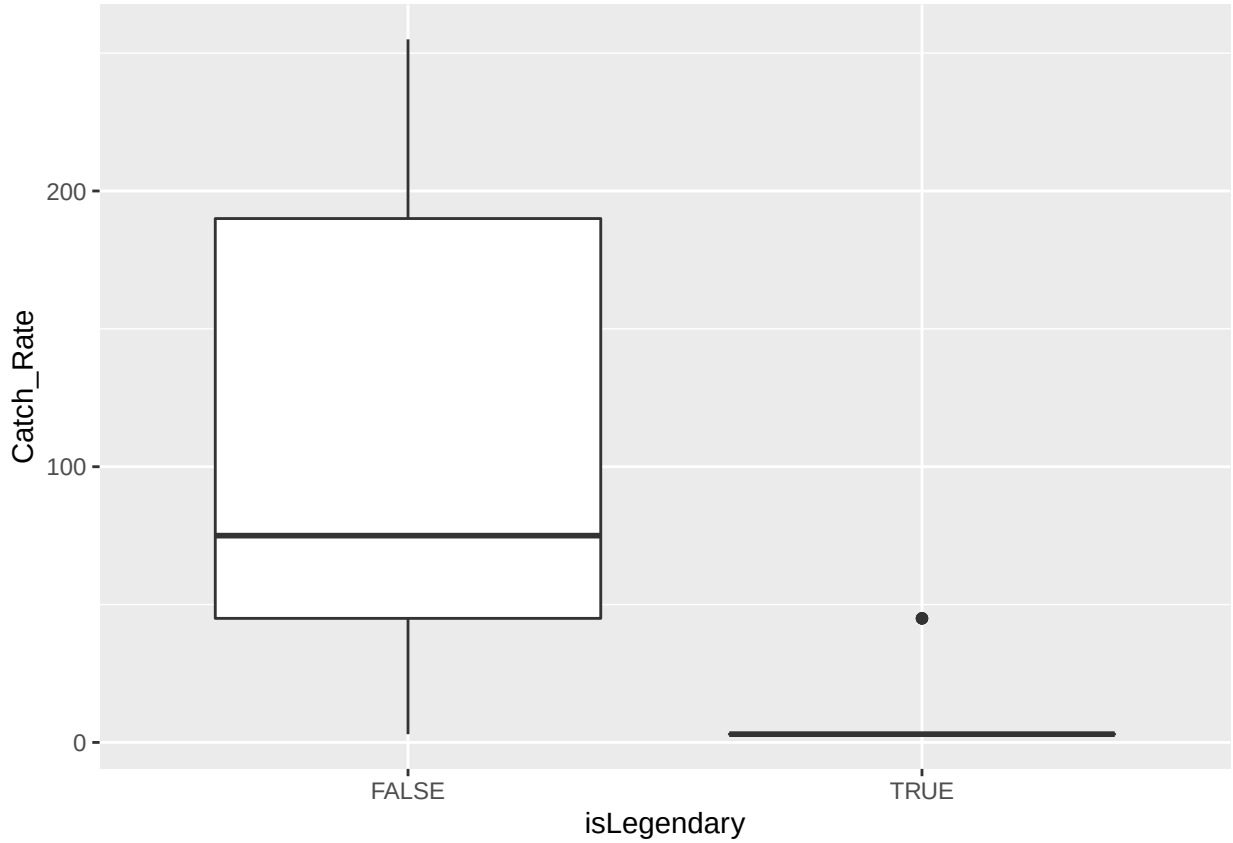

```
df %>% ggplot(aes(x= isLegendary, y = Weight_kg)) + geom_boxplot()
```




```
df %>% ggplot(aes(x= isLegendary, y = Height_m)) + geom_boxplot()
```




```
df %>% ggplot(aes(x= isLegendary, y = Catch_Rate)) + geom_boxplot()
```




```
df%>%group_by(isLegendary)%>%summarise(avg_rate=mean(Catch_Rate))
```

```
## # A tibble: 2 x 2
##   isLegendary avg_rate
##   <fct>          <dbl>
## 1 FALSE         107.
## 2 TRUE            6.65
```

```
df%>%filter(isLegendary == 'TRUE')%>%group_by(Generation,isLegendary)%>%summarise(n=n())%>%arrange(desc
```

```
## # A tibble: 6 x 3
## # Groups:   Generation [6]
##   Generation isLegendary     n
##   <fct>      <fct>       <int>
## 1 4          TRUE           11
## 2 3          TRUE           10
## 3 5          TRUE           10
## 4 6          TRUE            6
## 5 2          TRUE            5
## 6 1          TRUE            4
```

Some Insights from Data

- Our dataset is imbalanced as our class of interest (isLegendary = True) is only 6.4%.

- Legendary Pokemons have highest total stats among all pokemons (>580).
- Only 13% of Legendary pokemons have their gender defined.
- Water is the most common Pokemon type (15%).
- Legendary Pokemons are heavier and taller.
- Legendary Pokemons have the lowest catch rate of 6.7%.

Answers to Assignment Questions

Q1. What is the number of raid battles per player divided by the number of raid battles per battler?

a - Number of Raid battles = 4
b - Number of Players = 4
d -Number of Raid battles per player = a/b = 1
e - Number of battlers = 3
f -Number of Raid battles per battler = a/e = 1.3333333
Number of raid battles per player divided by the number of raid battles per battler - d/f = 0.75

Q2. Suppose the Pokémon Dataset is a SQL table called 'PokemonStats'. In a SQL dialect you are most comfortable with, find...

A. The number of distinct primary types present across Pokemon

```
sqldf("select Type_1  as Primary_Types, count(Number) as number_of_pokemon from df group by 1 order by 1
```

```
##    Primary_Types number_of_pokemon
## 1          Water               105
## 2         Normal                93
## 3          Grass                66
## 4            Bug                63
## 5        Psychic                47
## 6           Fire                47
## 7           Rock                41
## 8       Electric                36
## 9         Ground                30
## 10        Poison                28
## 11          Dark                28
## 12      Fighting                25
## 13        Dragon                24
## 14           Ice                23
## 15         Ghost                23
## 16         Steel                22
## 17         Fairy                17
## 18        Flying                 3
```

B. The average Total stats for each Pokemon generation

```
sqldf("select Generation, avg(Total) as average_total_stats from df group by 1 order by 1 asc")
```

```
##   Generation average_total_stats
## 1          1            407.0795
## 2          2            406.1800
## 3          3            402.0593
## 4          4            445.7570
## 5          5            425.3077
## 6          6            429.5833
```

C. The white Pokemon with the highest Total stats

```
sqldf("select Name,Color,Total from df where Color ='White' and Total = (select max(Total) from df where
```

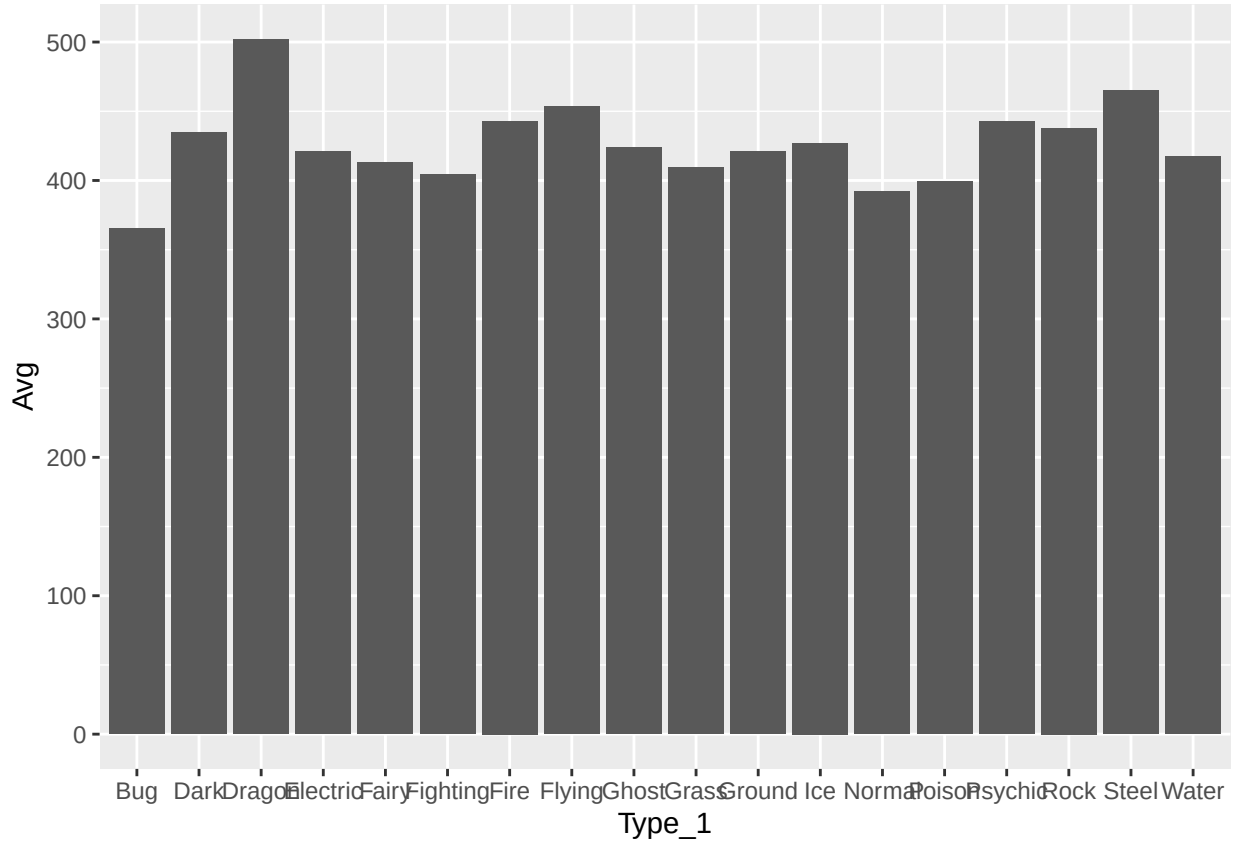
```
##       Name Color Total
## 1    Lugia White   680
## 2   Dialga White   680
## 3 Reshiram White   680
```

Q3. Imagine a new Pokemon game where you are only allowed to collect ONE type of Pokemon. Similar to other Pokemon games, your goal is to have the strongest battlers and defenders for battles and raids. Which type will you pick? Why?

```
df%>%group_by(Type_1)%>%summarize(Avg=mean(Total),Avg_attack=mean(Attack),Avg_defense=mean(Defense),Avg_
```

```
## # A tibble: 18 x 8
##    Type_1     Avg Avg_attack Avg_defense Avg_splattack Avg_Spldefense Avg_hp
##    <fct>    <dbl>      <dbl>       <dbl>          <dbl>           <dbl>  <dbl>
##  1 Dragon    502.      103.         79.1           82.5            83.0   78.0
##  2 Steel     465.       83.5       119.            63.5            79.4   64.8
##  3 Flying    453.       71.7        61.7           89              66.7   68
##  4 Fire      443.       82.2        64.9           83.5            69.5   68.6
##  5 Psychic   442.       61.3        65.0           90.6            82.4   70.6
##  6 Rock      438.       88.5        99.8           60.0            73.1   64.5
##  7 Dark      435.       86.2        67.9           70.5            67.5   67.2
##  8 Ice       427.       70.7        71.0           75.7            76.1   71.7
##  9 Ghost     424.       67.5        76.0           81.3            76.5   61.6
## 10 Ground    421        91.3        82.2           51.7            61.3   72.4
## 11 Electric  421.       67.8        59.1           83.1            67.7   59.5
## 12 Water     417.       71.0        70.5           71.7            68.1   70.9
## 13 Fairy     413.       61.5        65.7           78.5            84.7   74.1
## 14 Grass     410.       70.9        69.4           74.3            68.9   66.2
## 15 Fighting  404.       94.7        64.3           48.6            63.7   70.2
## 16 Poison    399.       74.7        68.8           60.4            64.4   67.2
## 17 Normal    392.       71.7        57.8           54.5            62     76.5
## 18 Bug       365.       65.2        67.7           54.0            62.2   56.0
## # ... with 1 more variable: Avg_Speed <dbl>
```

```
df%>%group_by(Type_1)%>%summarize(Avg=mean(Total),Avg_attack=mean(Attack),Avg_defense=mean(Defense))%>%
```




```
df3<-df%>%group_by(Type_1,isLegendary)%>%summarise(n=n())
df3<-df3%>%group_by(Type_1)%>%mutate(prop=n/sum(n))
df3%>%filter(isLegendary=='TRUE')%>%arrange(desc(prop))
```

```
## # A tibble: 15 x 4
## # Groups:   Type_1 [15]
##    Type_1   isLegendary     n   prop
##    <fct>    <fct>       <int>  <dbl>
##  1 Flying   TRUE            1 0.333
##  2 Dragon   TRUE            7 0.292
##  3 Steel    TRUE            4 0.182
##  4 Psychic  TRUE            8 0.170
##  5 Fire     TRUE            5 0.106
##  6 Ice      TRUE            2 0.0870
##  7 Electric TRUE            3 0.0833
##  8 Rock     TRUE            3 0.0732
##  9 Dark     TRUE            2 0.0714
## 10 Ground   TRUE            2 0.0667
## 11 Fairy    TRUE            1 0.0588
## 12 Ghost    TRUE            1 0.0435
## 13 Grass    TRUE            2 0.0303
## 14 Water    TRUE            3 0.0286
## 15 Normal   TRUE            2 0.0215
```

If I am allowed to collect only one type of Pokemon, I would choose to collect the type 'Dragon'. Pokemons of type 'Dragon' are the best Attackers and one of the best Defenders. Among all the types, Dragon have

the highest average stats, highest average attack stats, highest average special defense and highest average health points. Also, they are in top 3 in terms of their average defense stats, average special attack stats, and average speed. Hence, the 'Dragon' would be the optimal choice as both strong battlers and defenders. This is also evident by the fact that the type 'Dragon' have the highest proportion (30%) of Legendary Pokemons.

Q4. Model Building

Let's look at our dataset that will be used for building our predictive Models.

```
str(df)
```

```
## 'data.frame':    721 obs. of  23 variables:
##  $ Number          : Factor w/ 721 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Name            : Factor w/ 721 levels "Abomasnow","Abra",..: 68 295 670 87 88 86 595 688 54 82 .
##  $ Type_1          : Factor w/ 18 levels "Bug","Dark","Dragon",..: 10 10 10 7 7 7 18 18 18 1 ...
##  $ Type_2          : Factor w/ 19 levels "Bug","Dark","Dragon",..: 15 15 15 13 13 8 13 13 13 13 ...
##  $ Total           : num  318 405 525 309 405 534 314 405 530 195 ...
##  $ HP              : num  45 60 80 39 58 78 44 59 79 45 ...
##  $ Attack          : num  49 62 82 52 64 84 48 63 83 30 ...
##  $ Defense         : num  49 63 83 43 58 78 65 80 100 35 ...
##  $ Sp_Atk          : num  65 80 100 60 80 109 50 65 85 20 ...
##  $ Sp_Def          : num  65 80 100 50 65 85 64 80 105 20 ...
##  $ Speed           : num  45 60 80 65 80 100 43 58 78 45 ...
##  $ Generation      : Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ isLegendary     : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Color           : Factor w/ 10 levels "Black","Blue",..: 4 4 4 8 8 8 2 2 2 4 ...
##  $ hasGender       : Factor w/ 2 levels "FALSE","TRUE": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Pr_Male         : num  0.875 0.875 0.875 0.875 0.875 0.875 0.875 0.875 0.875 0.5 ...
##  $ Egg_Group_1     : Factor w/ 15 levels "Amorphous","Bug",..: 11 11 11 11 11 11 11 11 11 2 ...
##  $ Egg_Group_2     : Factor w/ 14 levels "Amorphous","Bug",..: 7 7 7 3 3 3 12 12 12 11 ...
##  $ hasMegaEvolution: Factor w/ 2 levels "FALSE","TRUE": 1 1 2 1 1 2 1 1 2 1 ...
##  $ Height_m        : num  0.71 0.99 2.01 0.61 1.09 1.7 0.51 0.99 1.6 0.3 ...
##  $ Weight_kg       : num  6.9 13 100 8.5 19 90.5 9 22.5 85.5 2.9 ...
##  $ Catch_Rate      : num  45 45 45 45 45 45 45 45 45 255 ...
##  $ Body_Style      : Factor w/ 14 levels "bipedal_tailed",..: 10 10 10 1 1 1 1 1 1 8 ...
```

Checking variable Importance

All our Numeric Variables have high importance. However, the numerical variables are correlated which we will hanlde during building when predictive Model

```
aucAll<- sapply(df %>% select_if(is.numeric), auc, response=df$isLegendary)
aucAll
```

```
##      Total         HP     Attack    Defense     Sp_Atk     Sp_Def      Speed
##  0.9888406  0.8424799  0.8353945  0.8291787  0.8978583  0.8762319  0.8255395
##    Pr_Male   Height_m  Weight_kg Catch_Rate
##  0.4732367  0.7922061  0.7721900  0.9754589
```

Splitting the train test data

Instead of K-fold cross validation,we will use a 70-30 split for our train and test set.
We remove the leakage variable (Catch Rate) from our dataset.

```
df<-df%>%select(-Catch_Rate)
trn=0.7
nr<-nrow(df)
trnd<-sample(1:nr,trn*nr,replace=FALSE)
train_data<-df[trnd,]
test_data<-df[-trnd,]
dim(train_data)
```

```
## [1] 504  22
```

```
dim(test_data)
```

```
## [1] 217  22
```

Selecting different models to build

- Initially we will use all the variables for building the model. Our Dataset has few collinear variables. We will handle multi-collinearity using Lasso Regression.
- We can choose multiple classification models for predicting the Legendary status. We can use regression models, regression models using different kinds of regularization, decision trees, ensemble models like Random Forest, GBM etc.
- However, given our dataset is small, there will be higher chances for advanced models with large number of parameters like GBM, Random Forest, Decision Trees being prone to overfit and over-learn.
- In general, the simpler the machine learning algorithm, the better it will learn from small datasets.
- From an ML perspective, small data requires models that have low complexity (or high bias) to avoid overfitting the model to the data.
- Hence, we will rely on 3 simpler models:
  - Logistic Regression without Regularization
  - Logistic Regression with Regularization (Lasso) - Will handle Multi-Colliniearity
  - Naive Bayes Classifier with Laplace Smoothing
- For evaluating each Model, we will look at their in sample accuracy as well as the confusion matrix, the AUC value, the ROC Curve.

Regression Model - Without Regularization

```
y_train_data<-factor(if_else(train_data$isLegendary == "TRUE", '1', '0'))
x_train_data<-train_data%>%select(-c(isLegendary,Number,Name))%>% data.matrix() %>% as.data.frame()
glm_basic<-glm(formula = y_train_data ~ ., data = x_train_data, family="binomial")
summary(glm_basic)
```

```
##
## Call:
## glm(formula = y_train_data ~ ., family = "binomial", data = x_train_data)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
```

```
## -1.413e-03  -2.000e-08  -2.000e-08  -2.000e-08   1.409e-03
##
## Coefficients: (1 not defined because of singularities)
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -5.893e+03  2.528e+05  -0.023    0.981
## Type_1            3.260e+01  1.448e+03   0.023    0.982
## Type_2            1.819e+01  1.239e+03   0.015    0.988
## Total             6.753e+00  4.375e+02   0.015    0.988
## HP               -8.175e+00  5.209e+02  -0.016    0.987
## Attack            3.248e+00  5.687e+02   0.006    0.995
## Defense          -6.934e+00  4.423e+02  -0.016    0.987
## Sp_Atk           -9.188e+00  5.718e+02  -0.016    0.987
## Sp_Def            3.290e+00  2.046e+02   0.016    0.987
## Speed                    NA         NA      NA       NA
## Generation        1.068e+01  3.484e+03   0.003    0.998
## Color             1.255e+01  1.182e+03   0.011    0.992
## hasGender        -1.505e+03  4.217e+04  -0.036    0.972
## Pr_Male           3.375e+03  7.050e+04   0.048    0.962
## Egg_Group_1       1.571e+02  3.833e+03   0.041    0.967
## Egg_Group_2       8.607e+01  7.521e+03   0.011    0.991
## hasMegaEvolution -2.583e+02  2.423e+04  -0.011    0.991
## Height_m          1.041e+02  5.376e+03   0.019    0.985
## Weight_kg         1.585e-01  2.603e+01   0.006    0.995
## Body_Style        2.006e+01  1.450e+03   0.014    0.989
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.3836e+02  on 503  degrees of freedom
## Residual deviance: 1.4977e-05  on 485  degrees of freedom
## AIC: 38
##
## Number of Fisher Scoring iterations: 25
```

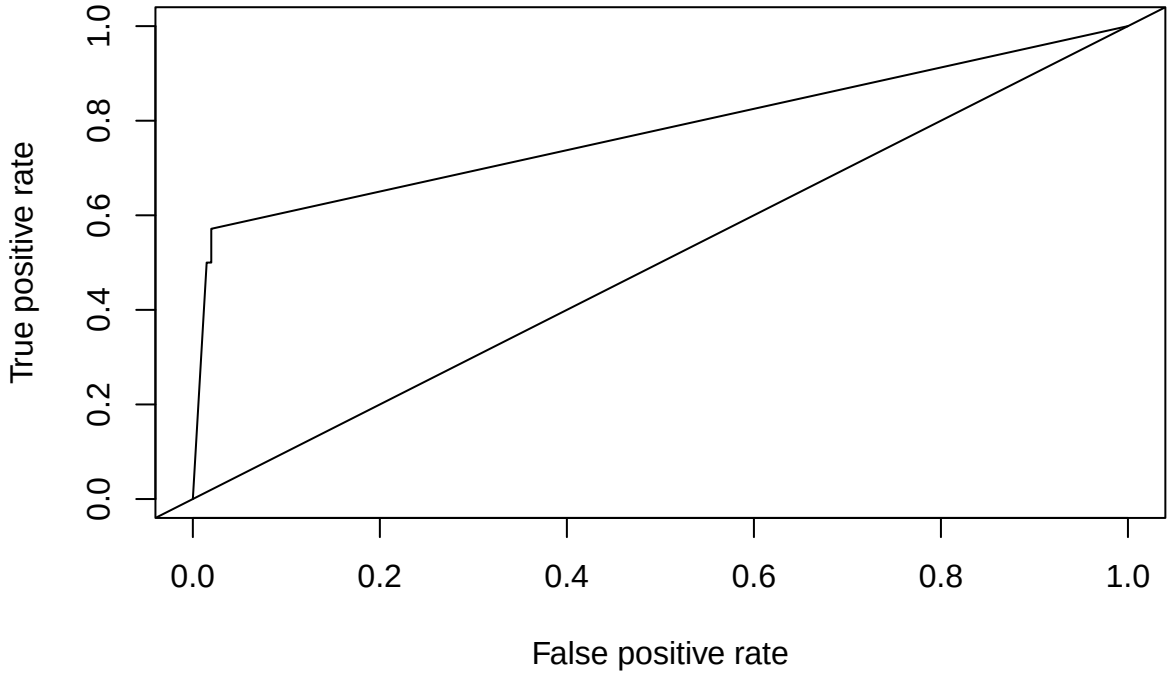
```
#Basic Evaluation
fitModel <-as.data.frame(if_else(glm_basic$fitted.values > 0.5, 1, 0))
table(Actual = y_train_data, Predicted = fitModel[,1])
```

```
##       Predicted
## Actual   0   1
##      0 472   0
##      1   0  32
```

```
#Evaluation on test data
y_test_data<-factor(if_else(test_data$isLegendary == "TRUE", '1', '0'))
x_test_data<-test_data %>%select(-c(isLegendary,Number,Name))%>% data.matrix() %>% as.data.frame()
predModel <- predict(glm_basic, newdata = x_test_data, type = "response")
predModelCls <-as.data.frame(if_else(predModel > 0.5, 1, 0))
table(Actual = y_test_data, Predicted = predModelCls[,1])
```

```
##       Predicted
## Actual   0   1
##      0 199   4
##      1   7   7
```

```
#ROC
prediction(predModel, y_test_data) %>% performance(measure = "tpr", x.measure = "fpr") %>% plot()
abline(a=0, b= 1)
```

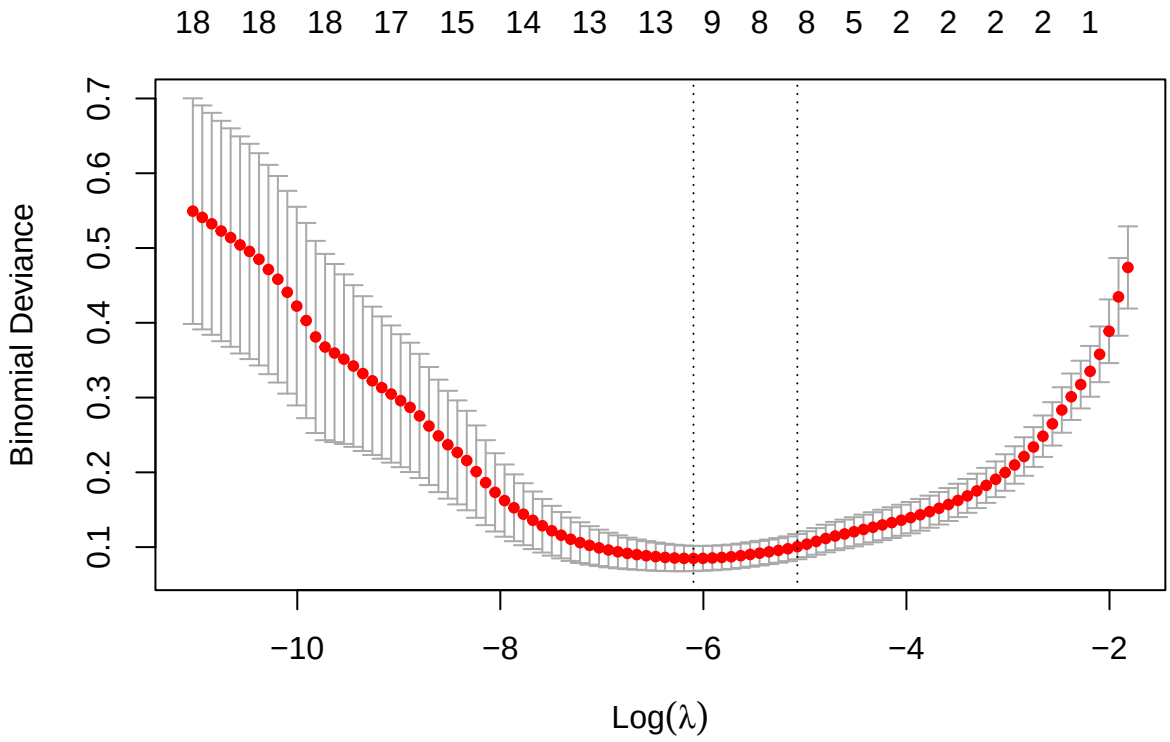



```
#AUC
prediction(predModel, y_test_data) %>%
  performance(measure = "auc") %>%
  .@y.values
```

```
## [[1]]
## [1] 0.7763899
```

Regression Model - With Regularization

```
y_train_data<-factor(if_else(train_data$isLegendary == "TRUE", '1', '0'))
x_train_data<-train_data%>%select(-c(isLegendary,Number,Name))
glm_cv<-cv.glmnet(data.matrix(x_train_data),y_train_data, family="binomial", alpha=1)
plot(glm_cv)
```




```
coef(glm_cv, s = glm_cv$lambda.min)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)      -24.775879131
## Type_1             .
## Type_2            -0.050355714
## Total              0.029529308
## HP                 .
## Attack             .
## Defense            .
## Sp_Atk             .
## Sp_Def             0.016955642
## Speed              0.011430445
## Generation         .
## Color             -0.012797336
## hasGender         -2.943822980
## Pr_Male            4.509569754
## Egg_Group_1        0.321558270
## Egg_Group_2        0.306668150
## hasMegaEvolution  -0.180692247
## Height_m           0.009765141
## Weight_kg          .
## Body_Style         0.002274167
```

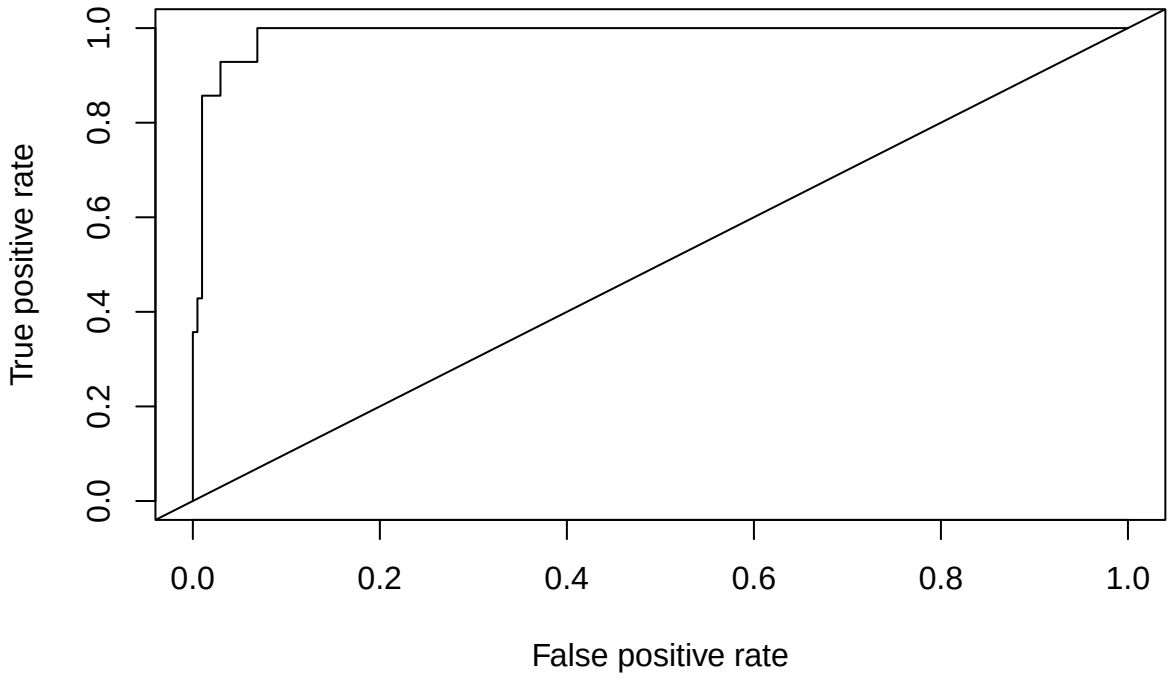
```
#Basic Evaluation
PredTrn <- predict(glm_cv, data.matrix(x_train_data), type="class",s="lambda.min")
table(actuals=train_data$isLegendary, preds=PredTrn)
```

```
##        preds
## actuals   0   1
##   FALSE 468   4
##   TRUE    1  31
```

```
#Evaluation on test data
y_test_data<-factor(if_else(test_data$isLegendary == "TRUE", '1', '0'))
x_test_data<-test_data%>%select(-c(isLegendary,Number,Name))

glmPredls_1=predict(glm_cv,data.matrix(x_test_data), s="lambda.min" )
glmPredls_p=predict(glm_cv,data.matrix(x_test_data), s="lambda.min", type="response" )
predsauc <- prediction(glmPredls_p, test_data$isLegendary, label.ordering = c("FALSE", "TRUE"))


#ROC
prediction(glmPredls_p, y_test_data) %>% performance(measure = "tpr", x.measure = "fpr") %>% plot()
abline(a=0, b= 1)
```




```
#AUC
prediction(glmPredls_p, y_test_data) %>%
```

```
  performance(measure = "auc") %>%
  .@y.values
```

```
## [[1]]
## [1] 0.9883885
```

```
PredTbl <- predict(glm_cv, data.matrix(x_test_data), type="class")
table(actuals=test_data$isLegendary, preds=PredTbl)
```

```
##        preds
## actuals   0   1
##   FALSE 201   2
##   TRUE    3  11
```

Naive Bayes Classifier

```
nbModel <- naiveBayes(isLegendary ~ ., data=train_data %>% select(-c(isLegendary,Number,Name),"isLegenda
nbPredTrn <- predict(nbModel, newdata = train_data, type="class")
table(actuals=train_data$isLegendary, preds=nbPredTrn)
```

```
##        preds
## actuals FALSE TRUE
##   FALSE   455   17
##   TRUE      0   32
```

```
nbPredTst <- predict(nbModel, newdata = test_data, type="class")
table(actuals=test_data$isLegendary, preds=nbPredTst)
```

```
##        preds
## actuals FALSE TRUE
##   FALSE   198    5
##   TRUE      0   14
```

Model Selection

- 1. Regression without Regularization - This model fits the training data well.However, due to multi-collinear variables present in our dataset, the model's interpretability is not good.
- 2. Naive Bayes - This Model does good work in fitting our training data as well as generalizing to our test data. However, its accuracy on our class of interest is lower than the regularized regression.
- 3. Regression with Regularization - This model fits the training data well with greater accuracy on our class of interest (Legentary = True). Also, using lasso regression has eliminated few of the multi-collinear variables from out data. Hence, model's interpretability is better and generalizes well on the test data.

Hence, we shall pick Regression using Regularization (Lasso) as our Final Model

- We have used all the variables in building our final model except Sp_Atk , Color, hasMegaEvolution,HP and weight

- Accuracy on Training Data = 98.6%
- Precision = 0.991
- Recall = 0.994
- F1 Score = 0.992
- Accuracy on Test Data = 99.1%
- Precision (Test Data) = 0.98
- Recall (Test Data) = 1
- F1 Score (Test Data) = 0.99
- AUC = 99.3