The ETL Process of Fifa 23 Player Ratings

Liam Frank

Last compiled on December 15, 2023

Contents

1 Introduction

Soccer is the most popular sport in the world, The Fifa video game franchise by EA Sports, now titled EAFC has captured the large market of soccer enthusiasts as well as video game connoisseurs. Fifa 23 contains more than 19,000 players, 700 plus teams, and 30 plus leagues from countries around the world. Fifa 23, EA's second latest iteration of the game sold 10.3 million units worldwide at an average selling price of 70 US dollars. This equates to a little over 720 million US dollars in gross revenue for EA's Fifa franchise in the 2022 year alone. The highest selling game to date was Fifa 18, which sold 26.4 million units worldwide at an average selling price of 60 US dollars, leading to a gross revenue of 1.5 billion US dollars in 2017. The profound success of the EA Sports Fifa franchise can't be argued. In 2010 EA released Ultimate Team, a game mode that allows users to build their own team featuring different players from teams around the globe. Users can then competitively compete online with other users via a division system, win enough games in a given season and your team will move up a division. Divisions span from 10 to 1 with 1 being the highest level of play. Other game modes exist such as squad battles, drafts, and friendly matches. The Ultimate Team system contains an online marketplace where users can buy and sell different player cards. All 19,000 plus players in the game possess a base card, either Bronze, Silver, or Gold. Bronze cards are allotted to players rated 64 and below, Silver for a player rating between 65 and 74, and finally Gold is given to players who exceed the rating of 75. Player ratings are comprised of a multitude of attributes that will be touched on later in this study. With good performance, players are given special cards, the resale value on the online marketplace for special card typically far exceeds the price of base cards. Some examples of special card types are Team of the Season, Team of the Year, and Hero. In this paper we will scrape data from a popular online Fifa card resource that is frequented by Ultimate Team players, clean and transform the data into a form that is suitable for analysis, and finally explore subset selection, shrinkage, and dimensionality reduction methods in R.

2 Background

During the research process, 4 previous studies were found that conducted analysis utilizing a data set of Fifa players similar to the one scraped for this study. The first study titled "Predict The Value of Football Players Using Fifa Video Game Data and Machine Learning Techniques", employed multiple linear regression, decision trees, and random forrest to predict a players real life transfer market value based upon different variables affiliated with the Fifa video game such as rating. In this study the multiple linear regression model has an R squared of .56, the decision tree had an R squared of .87, and the random forrest had an R squared of .95. The second study titled "Using Fifa Soccer Video Game Data for Soccer Analytics", compared and contrasted the 2014 Brazilian and German national teams by analyzing player ratings and metrics on each players Fifa Card from their respective team. The third study titled "Fifa 23 Exploratory Data Analysis", listed summary statistics and created plots to visualize relationships between variables. The player ratings for all 19,000 plus players in the game form a binomial or standard normal distribution. The distribution of player age vs frequency is right skewed indicating a greater number of younger players in the game, with the mode being left of the median and mean in the distribution plot. A high correlation between real life player transfer market value and in game Fifa rating was also found. The last study titled "Machine Learning on Fifa 20" classified players with multiple popular classification algorithms including K-nearest neighbor, decision tree, support vector machine, and logistic regression.

3 Data and Source

The data for this study was acquired from https://www.futbin.com/23/players?page=1 via a web scraper built utilizing the rvest library in R. The website futbin is an online database used frequently by Ultimate Team players for price history of player cards. The user interface of futbin is extremely user friendly and allows for sorting of cards by a variety of measures including player rating, card type, position, Nation, League, Club team, individual card statistics, price, attributes, and even height and weight. For Fifa 23, the website contains 735 pages with 30 cards per page, for a total of just over 22,000 total player cards. It is important to note that for this study, only the first 34 pages of the highest rated player cards were scraped, for a total of 1,020 cards. This was due to constraints faced such as rate limit erros, when scraping the futbin website as well as the project requirement of 1,000 total observations.

```r
get_player = function(link) {
  prac <- read_html(link)
  pname = prac %>% html_nodes("tr:nth-child(1) .table-row-text") %>% html_text()
  print(pname)
  return(pname)
}
get_cardtype = function(link) {
  prac <- read_html(link)
  cardtype = prac %>% html_nodes("tr:nth-child(2) .table-row-text") %>% html_text()
  print(cardtype)
  return(cardtype)
}
fifa_df3 = data.frame()
for (page_result in seq(from = 1, to = 34, by = 1)) {
  link = paste0("https://www.futbin.com/23/players?page=",
              page_result)
  page = read_html(link)
  player_links = page %>% html_nodes(".get-tp") %>% html_attr("href") %>%  paste0("https://www.futbin.c
  pname1 = page %>% html_nodes(".get-tp") %>% html_text()
  print(pname1)
  Sys.sleep(30)
  card = page %>% html_nodes(".mobile-hide-table-col div:nth-child(1)") %>% html_text()
  print(card)
  Sys.sleep(30)
  pprice = page %>% html_nodes("span.font-weight-bold") %>% html_text()
  print(pprice)
  Sys.sleep(30)
  player = sapply(player_links, FUN = get_player, USE.NAMES = FALSE)
  Sys.sleep(30)
  cardtype = sapply(player_links, FUN = get_cardtype, USE.NAMES = FALSE)
  Sys.sleep(120)
  fifa_df3 = rbind(fifa_df3, data.frame(player,
                                        pname1,
                                        pprice,
                                        card,
                                        cardtype,
                                        stringsAsFactors = FALSE))
}

write.csv(fifa_df3, "fifa_df3.csv")
```

The web scraping code from above is a sample of the full code used to scrap the data from the futbin website. The full code is in the appendix of the paper. The web scraper was built with the rvest library in R. To reach 1,000 total observations, 34 pages were scraped. Using a for loop for page results 1 through 34, the links were created for each page were player name, card type, and card price were obtained. Next, using the href attribute in the source code of the website, the links to individual player pages were obtained. Functions were then created for each other variable and the data was extracted using the css selectors for the element found through the SelectorGadget chrome extension tool. Sapply was used to call each individual function in the for loop, and each time through the loop the results of each function were mapped into a data frame that was continuously updated through out the scraping process. Sys.sleep was used to avoid rate limit errors that would other wise stem from the web scraper making too many requests to the futbin website in a given amount of time. Lastly, the newly created data frame was written to a csv file.

4 Cleaning and Transformations

Below is a transposition of the first two observations from the un-cleaned scraped data from futbin, the only modification being the use of trimws to remove excess whitespace for finer formatting and visual display.

```
##                  1                           2
## X.1              "1"                         "2"
## X                "1"                         "2"
## player           "Edson Arantes Nascimento"  "Kevin De Bruyne"
## pname1           "Pelé"                      "Kevin De Bruyne"
## pprice           "2.88M"                     "0"
## cardtype         "Explosive"                 "Controlled"
## position         "LW"                        "CM"
## age              "83 years old"              "32 years old"
## rating           "99"                        "99"
## club             "FUT ICONS"                 "Manchester City"
## league           "Icons"                     "Premier League"
## nation           "Brazil"                    "Belgium"
## height           "173cm | 5'8\""             "181cm | 5'11\""
## weight           "70"                        "70"
## foot             "Right"                     "Right"
## weakfoot         "5"                         "5"
## skills           "5"                         "5"
## attackworkrate   "High"                      "High"
## defenseworkrate  "Med"                       "High"
## pace             "96"                        "88"
## dribbling        "99"                        "97"
## shooting         "97"                        "95"
## defending        "61"                        "91"
## passing          "94"                        "99"
## physical         "78"                        "99"
## card             "Shapeshifters ICON"        "Level Up Obj"
```

```r
fifa_tidy <- fifa[, -which(names(fifa) == "X.1")]
fifa_tidy <- fifa %>% filter(trimws(foot) %in% c("Right","Left"))
nrow(fifa_tidy)
```

```
## [1] 1013
```

```r
nrow(fifa)
```

```
## [1] 1020
```

The code above first removes the repetitive column of X.1. Next, utilizing the dplyr library for piping and filtering, mis-formatted observations are removed from the data frame. Calling nrow, of the original data frame and tidy data frame, it is evident that 7 total observations were mis-formatted via css selector issues from web scraping. These 7 observations have been filtered out of the data frame.

```
fifa_tidy <- fifa_tidy %>% mutate_all(trimws)
fifa_tidy$height <- parse_number(fifa_tidy$height)
```

The next step in the cleaning process was removing all excess and unnecessary whitespace from all observations in the data frame. The height variable contained a labeled measurement in both centimeters as well as feet and inches, using the parse_number command only the measurement in centimeters was recorded.

```
names(fifa_tidy)[names(fifa_tidy) == "height"] <- "heightcm"
names(fifa_tidy)[names(fifa_tidy) == "weight"] <- "weightkg"
names(fifa_tidy)[names(fifa_tidy) == "pprice"] <- "price"
names(fifa_tidy)[names(fifa_tidy) == "pname1"] <- "playername"
names(fifa_tidy)[names(fifa_tidy) == "cardtype"] <- "attribute"
names(fifa_tidy)[names(fifa_tidy) == "card"] <- "cardtype"
```

Next, variable names were re-assigned. Starting, height and weight were re-assignment to contain the measurement unit for the recorded data. This was done in an attempt to have variable names be as precise and descriptive as possible to reduce errors. Next, when building the web scraper pprice and ppname1 were variable names choosen, those have since been updated to price and playername respectively. Also when building the web scraper, attribute was named cardtype and cardtype was in turn named card. These recorded variables names were fixed for readability, understanding, and error minimization.

```
fifa_tidy[1,8]
```

```
## [1] "83 years old"
```

```
fifa_tidy[101,8]
```

```
## [1] "22-09-1976"
```

```
standardize_age <- function(age) {
  if (grepl("\\d{2}-\\d{2}-\\d{4}", age)) {
    birth_date <- as.Date(age, format = "%d-%m-%Y")
    age_in_years <- as.numeric(difftime(Sys.Date(), birth_date, units = "days") / 365)
    return(paste(round(age_in_years), "years old"))
  } else {
    return(age)
  }
}
fifa_tidy$age <- sapply(fifa_tidy$age, standardize_age)
fifa_tidy$age <- parse_number(fifa_tidy$age)
fifa_tidy[1,8]
```

```
## [1] 83
```

```
fifa_tidy[101,8]
```

```
## [1] 47
```

Next, prior to cleaning the age recorded was in one of two formats, either "___ years old" or as a birth date in the form "day-month-year". Naturally, recorded data for one variable in two vastly differing formats is impossible for analysis. So a function was created, if the recorded value was in the form of a birth date, then the function would convert the birth date for the observation to the format of "___ years old". Lastly, parse_number was used to extract solely the number for the players age. Displayed above is an example of the code execution for observations 1 and 101. 83 years old was recorded as 83 and 22-09-1976 was passed into the function and changed to an age of 47.

```r
fifa_tidy <- separate(data = fifa_tidy, col = price, into = c("value", "label"), sep = -1)
fifa_tidy <- fifa_tidy %>%
  mutate(value = case_when(
    label == "M" ~ as.numeric(value) * 1e+06,
    label == "K" ~ as.numeric(value) * 1e+03,
    TRUE ~ as.numeric(value)
  ))
fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "label")]
fifa_tidy$value[fifa_tidy$value == 0] <- NA
```

Next in the cleaning process was player card price. As seen in the original scraped data frame, price is recorded as decimal value followed by a label M or K, indicating million or thousand. Cards without a price were recorded as a 0 when scraped, in reality that 0 represents an NA value. To convert the recorded value to an integer suitable for analysis the separate command from the tidyr library was used. A column was created titled "value" for the decimal and "label" for M or K. Next, utilizing case_when, the decimal value was converted to an integer, if label was an M the decimal was multiplied by one million and if label was a K the decimal was multiplied by one thousand. The recorded value was then converted to a numeric data type and the label column was then removed as it then served no valuable purpose.

```r
fifa_tidy$cardtype[fifa_tidy$cardtype =="" ] <- NA

fifa_tidy$rating <- as.numeric(fifa_tidy$rating)
rank<-fifa_tidy %>% group_by(league) %>% summarise(rank = mean(rating)) %>% arrange(desc(rank))

fifa_tidy <- fifa_tidy %>%
  mutate(league_rank = case_when(
    league == "Chinese FA Super L. (CHN 1)" ~ 1,
    league == "LaLiga Santander" ~ 2,
    league == "Ligue 1" ~ 3,
    league == "Serie A TIM" ~ 4,
    league == "MBS Pro League (SAU 1)" ~ 5,
    league == "Premier League" ~ 6,
    league == "Eliteserien (NOR 1)" ~ 7,
    league == "Italy Serie B (2)" ~ 8,
    league == "Bundesliga" ~ 9,
    league == "Major League Soccer" ~ 10,
    league == "Süper Lig (TUR 1)" ~ 11,
    league == "Liga NOS (POR 1)" ~ 12,
    league == "3F Superliga (DEN 1)" ~ 13,
    league == "1A Pro League (BEL 1)" ~ 14,
    league == "EFL League One (ENG 3)" ~ 15,
    league == "Polski Ekstraklasa (POL 1)r" ~ 16,
    league == "Scottish Premiership (SPFL)" ~ 17,
    league == "A-League (AUS 1)" ~ 18,
    league == "Eredivisie" ~ 19,
    league == "Icons" ~ 20,
    league == "EFL Championship (ENG 2)" ~ 21,
    league == " Ligue 2 (FRA 2)" ~ 22,
    league == "National League (ENG 5)" ~ 23,
    league == "World Cup" ~ 24,
    league == "EFL League Two (ENG 4)" ~ 25,
    league == "Hellas Liga (GRE 1)" ~ 26,
    TRUE ~ NA_real_
  ))
```

The next step in the cleaning process as seen above was first any cardtype was was recorded as blank was converted to an NA value. Next, for each of the 26 leagues in the data frame the mean rating of players in the league was recorded. These means were then arranged in descending order. A new variable was created titled league_rank, and for each observation the league rank was recorded for the corresponding league the player plays in by utilizing the case_when command. As evident from the code above, a 1 symbolizes the league with the highest mean player rating, and a 26 symbolizes the league with the lowest mean player rating of the players in the data frame.

```
cardtype_ranks <- fifa_tidy %>%
  group_by(cardtype) %>%
  summarise(rank = mean(rating)) %>%
  arrange(desc(rank)) %>%
  mutate(cardtype_rank = row_number())

add_cardtype_rank <- function(fifa_observation) {
  result <- fifa_observation %>%
    left_join(cardtype_ranks %>% select(cardtype, cardtype_rank), by = "cardtype")
  return(result)
}

fifa_tidy <- fifa_tidy %>%
  group_split() %>%
  map_df(add_cardtype_rank)
```

A similar process to the one carried out above for league was also completed for cardtype, this time instead of utilizing case_when, a function was created as their are 93 different unique cardtypes and using case_when would be exhaustive. Cardtypes were ranked in descending order based apon the mean player rating for all cards of that given type. A new variable was created titled cardtype_rank, for each observation the cardtype_rank was recorded for the cards corresponding cardtype. With a 1 symbolizing the cardtype with the highest mean player rating, and a 93 symbolizing the cardtype with the lowest mean player rating.

```
fifa_tidy <- fifa_tidy %>%
  group_by(rating, cardtype) %>%
  mutate(value = ifelse(is.na(value), median(value, na.rm = TRUE), value))
fifa_tidy <- fifa_tidy %>% filter(value != "NaN")
```

Continuing with cleaning of the value variable, values were imputed for observations recorded as NA. If an observation contained an NA for value, the NA was replaced with the median value for cards of the same rating and cardtype. Median is less susceptible to influence for outlier values and that is why it was used instead of mean. Cards with an NA value that did not feature any other cards of the same rating and cardtype were recorded as NaN. Values recorded as Nan were then removed from the data frame resulting in a decrease of observations from 1013 to 927.

```r
fifa_tidy$rating <- as.numeric(fifa_tidy$rating)
fifa_tidy$league_rank <- as.numeric(fifa_tidy$league_rank)
fifa_tidy$cardtype_rank <- as.numeric(fifa_tidy$cardtype_rank)
fifa_tidy$weightkg<- as.numeric(fifa_tidy$weightkg)
fifa_tidy$weakfoot<- as.numeric(fifa_tidy$weakfoot)
fifa_tidy$skills<- as.numeric(fifa_tidy$skills)
fifa_tidy$pace<- as.numeric(fifa_tidy$pace)
fifa_tidy$dribbling<- as.numeric(fifa_tidy$dribbling)
fifa_tidy$shooting<- as.numeric(fifa_tidy$shooting)
fifa_tidy$defending<- as.numeric(fifa_tidy$defending)
fifa_tidy$passing<- as.numeric(fifa_tidy$passing)
fifa_tidy$physical<- as.numeric(fifa_tidy$physical)

fifa_tidy$attribute <- as.factor(fifa_tidy$attribute)
fifa_tidy$position <- as.factor(fifa_tidy$position)
fifa_tidy$foot <- as.factor(fifa_tidy$foot)
fifa_tidy$attackworkrate <- as.factor(fifa_tidy$attackworkrate)
fifa_tidy$defenseworkrate <- as.factor(fifa_tidy$defenseworkrate)

fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "X.1")]
fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "X")]
```

Lastly, certain variables were converted to numeric values or factor values for future analysis methods. Variables X.1 and X were also remove as they are not needed in the data frame.

5 Variables

After the cleaning process the tidy data set that will be used for analysis featured 26 variables and 927 observation.

1) player: Birth name of the player. Character data type.
2) playername: Name on the card, typically what most people know the player as. Character data type.
3) value: Current selling price of the card. Numeric data type.
4) attribute: Player attribute. 3 level factor data type of either explosive, controlled, or lengthy
5) position: Position for player on the card, important to note that some cards are created that have players assigned to a position different from their natural or usual position. 15 level factor data type.
6) age: Age of player, important to note that deceased players age is recorded as how old they would be if still alive today. Numeric data type.
7) rating: Player card rating is integer between 1 and 99 comprised or calculated from weighted card variables based on position. Numeric data type.
8) Club: Club team that the player currently plays for. Character data type.
9) League: League that players club team currently competes in. Character data type.
10) Nation: Nationality of player. Character data type.
11) Height: Player height recorded in centimeters. Numeric data type.
12) Weight: Player weight recorded in kilograms. Numeric data type.
13) Foot: Players preferred foot. 2 level factor variable, either right or left
14) Weakfoot: Rating out of five for the players non-dominate foot. Numeric data type.
15) Skills: Rating out of five for the players skill and dribbling moves they can perform. Numeric data type.
16) Attacking Work Rate: Players work rate on offense. 3 level factor variable, either high, med, or low.
17) Defensive Work Rate: Players work rate on defense. 3 level factor variable, either high, med, or low

18) Pace: Measure of player speed as an integer between 1 and 99. Numeric data type.
19) Dribbling: Measure of player dribbling ability as an integer between 1 and 99. Numeric data type.
20) Shooting: Measure of player shooting ability as an integer between 1 and 99. Numeric data type.
21) Defending: Dribbling: Measure of player defensive ability as an integer between 1 and 99. Numeric data type.
22) Passing: Measure of player passing ability as an integer between 1 and 99. Numeric data type.
23) Physical: Measure of player physical statute as an integer between 1 and 99. Numeric data type.
24) Cardtype: Type of player card. Character data type.
25) League Rank: Derived from the mean player rating in a given league with 1 representing the league with the highest mean player rating. Numeric data type.
26) cardtype Rank: Derived from the mean player rating for a given cardtype with 1 representing the cardtype with the highest mean player rating. Numeric data type.

5.1 Observations

```
##                 1
## player          "Edson Arantes Nascimento"
## playername      "Pelé"
## value           "2880000"
## attribute       "Explosive"
## position        "LW"
## age             "83"
## rating          "99"
## club            "FUT ICONS"
## league          "Icons"
## nation          "Brazil"
## heightcm        "173"
## weightkg        "70"
## foot            "Right"
## weakfoot        "5"
## skills          "5"
## attackworkrate  "High"
## defenseworkrate "Med"
## pace            "96"
## dribbling       "99"
## shooting        "97"
## defending       "61"
## passing         "94"
## physical        "78"
## cardtype        "Shapeshifters ICON"
## league_rank     "20"
## cardtype_rank   "15"
```

The observation above is a transposition of the first observation in the tidy data set. The players birth name is Edson Arantes Nascimento, but is known as and displayed on the player card as Pele. The marketplace value on Fifa 23 for Pele is 2,880,000. His attribute is explosive and he plays left wing. Pele recently passed away in 2022 at the age of 82, but as previously mentioned the age variable accounts for if the player was still alive today, so the recorded age is 83. This card for Pele features a 99 rating with him playing for FUT ICONS in the ICONS League. Pele's nationality is Brazilian, and he measures in at a height of 173 centimeters with a weight of 70 kilograms. His preferred foot is right, but his weak foot is rated at 5 out of 5 as well as his skills. He has an attacking work rate denoted high and his defensive work rate is med. Pele's main stats are as follows: pace of 96, dribbling of 99, shooting of 97, defending of 61, passing of 94, and physical of 78. This specific Pele card is of type Shapeshifters ICON with a ranking of 15. The Icons league has a ranking of 20.

6 Models

As discussed above, the data frame features 26 different variable of of varying data types including numeric, factor, and character data. In the field of data science and machine learning we are often interested in building predictive models. As the number of features or variables increases past 10 in a given data set, the possibility of over-fitting to the data set increases exponentially and the predictive model becomes less useful. With the addition of more variables to the predictive model it also become harder to explain how each variable is influencing the model prediction or outcome. Another issue of data sets that contain a large number of features or variables is that it is unlikely that every variable is important in predicting the model outcome. The models developed in this study include a best subset selection, lasso regression, and principal component analysis. For all models developed a subset of the tidy data frame was created containing only numeric data. This resulted in a total of 15 observations: value, age, rating, height, weight, weakfoot, skills, pace, dribbling, shooting, defending, passing, physical, league_rank, and cardtype_rank.

6.1 Subset Selection

A best subset selection was carried out utilizing the leaps library in R. The goal of best subset selection is to find a small subset of predictors from the original data frame that result in the greatest prediction accuracy of the response variable for a linear model. This will remove explanatory variables that are irrelevant or less significant in predicting the response variable. Best subset selection builds linear models with every possible variable combination from the imputed data frame resulting in 2^n possible models where n represents the number of variables in the data frame. For selecting the best model, the adjusted R squared value was used. Using the previously created numeric_data, using best subsets every possible model will be created for predicting rating.

```
## Subset selection object
## Call: regsubsets.formula(rating ~ ., data = numeric_data)
## 14 Variables  (and intercept)
##               Forced in Forced out
## value             FALSE      FALSE
## age               FALSE      FALSE
## heightcm          FALSE      FALSE
## weightkg          FALSE      FALSE
## weakfoot          FALSE      FALSE
## skills            FALSE      FALSE
## pace              FALSE      FALSE
## dribbling         FALSE      FALSE
## shooting          FALSE      FALSE
## defending         FALSE      FALSE
## passing           FALSE      FALSE
## physical          FALSE      FALSE
## league_rank       FALSE      FALSE
## cardtype_rank     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          value age heightcm weightkg weakfoot skills pace dribbling shooting
## 1  ( 1 ) " "   " " " "      " "      " "      " "    " "  " "       " "
## 2  ( 1 ) " "   " " " "      "*"      " "      " "    " "  " "       " "
## 3  ( 1 ) "*"   " " " "      " "      "*"      " "    " "  " "       " "
## 4  ( 1 ) "*"   " " " "      " "      "*"      " "    " "  "*"       " "
## 5  ( 1 ) "*"   " " " "      " "      "*"      " "    " "  "*"       " "
## 6  ( 1 ) "*"   "*" " "      " "      "*"      " "    " "  "*"       " "
## 7  ( 1 ) "*"   "*" " "      " "      "*"      " "    " "  "*"       " "
## 8  ( 1 ) "*"   "*" " "      " "      "*"      " "    " "  "*"       " "
##          defending passing physical league_rank cardtype_rank
## 1  ( 1 ) " "       " "     " "      " "         "*"
## 2  ( 1 ) " "       " "     " "      " "         "*"
## 3  ( 1 ) " "       " "     " "      " "         "*"
## 4  ( 1 ) " "       " "     " "      " "         "*"
## 5  ( 1 ) " "       " "     "*"      " "         "*"
## 6  ( 1 ) " "       " "     "*"      " "         "*"
## 7  ( 1 ) " "       " "     "*"      "*"         "*"
## 8  ( 1 ) "*"       " "     "*"      "*"         "*"
```

```
## [1] 8
```

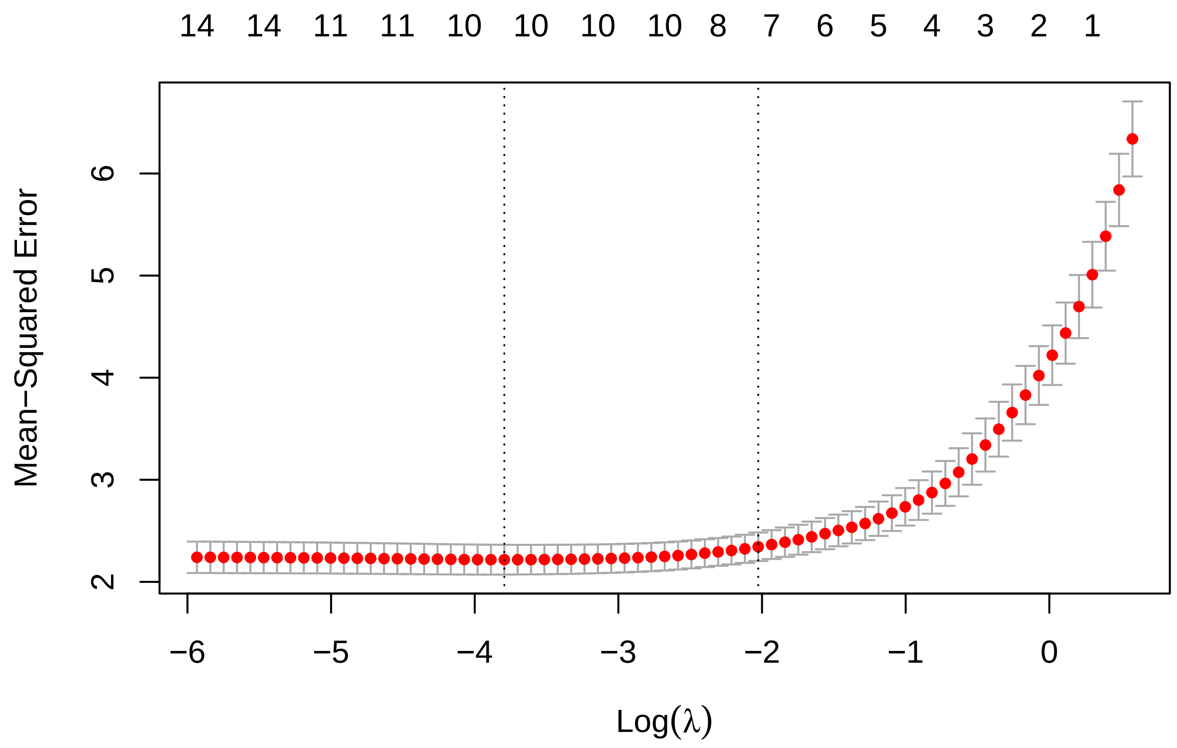
```
## [1] 0.6754143
```

```
##   (Intercept)         value           age      weakfoot     dribbling
##  7.805672e+01  6.766768e-07  3.452439e-02  5.925446e-01  9.148227e-02
##     defending      physical   league_rank cardtype_rank
##  8.422565e-03  5.244909e-02 -2.956854e-02 -5.336696e-02
```

As seen above in the model summary, the best model containing only one variable to predict rating featured cardtype_rank as the explanatory variable, indicating that it is the most significant variable in predicting rating. The model containing the highest adjusted R squared value of .675 was created on 8 out of the 15 variables in the numeric data frame, those variables being value, age, weakfoot, dribbling, defending, physical, league_rank, and cardtype_rank.

6.2 Shrinkage Methods

Similar to subset selection, shrinkage methods aim to reduce the number of variables contributing to the predictive outcome. This study employed LASSO regression as a method of shrinkage using the glmnet library in R. Unlike subset selection algorithms such as best, which completely remove certain variables from the model, shrinkage methods keep all variables in the model but reduce the coefficients of less important explanatory variables towards zero. In the case of LASSO regression coefficients are able to be set equal to zero for variables in the model eliminating relationships between explanatory and dependent variables. As the penalty term tuning parameter lambda in the LASSO regression equation is increased, so is the potential bias of the model. With the correct lambda selection the prediction variance will decrease. Rating will again be predicted, this time using LASSO regression and comparison to the results of best subset selection.



```
## [1] 0.02250347
```

```
## [1] 0.681213
```

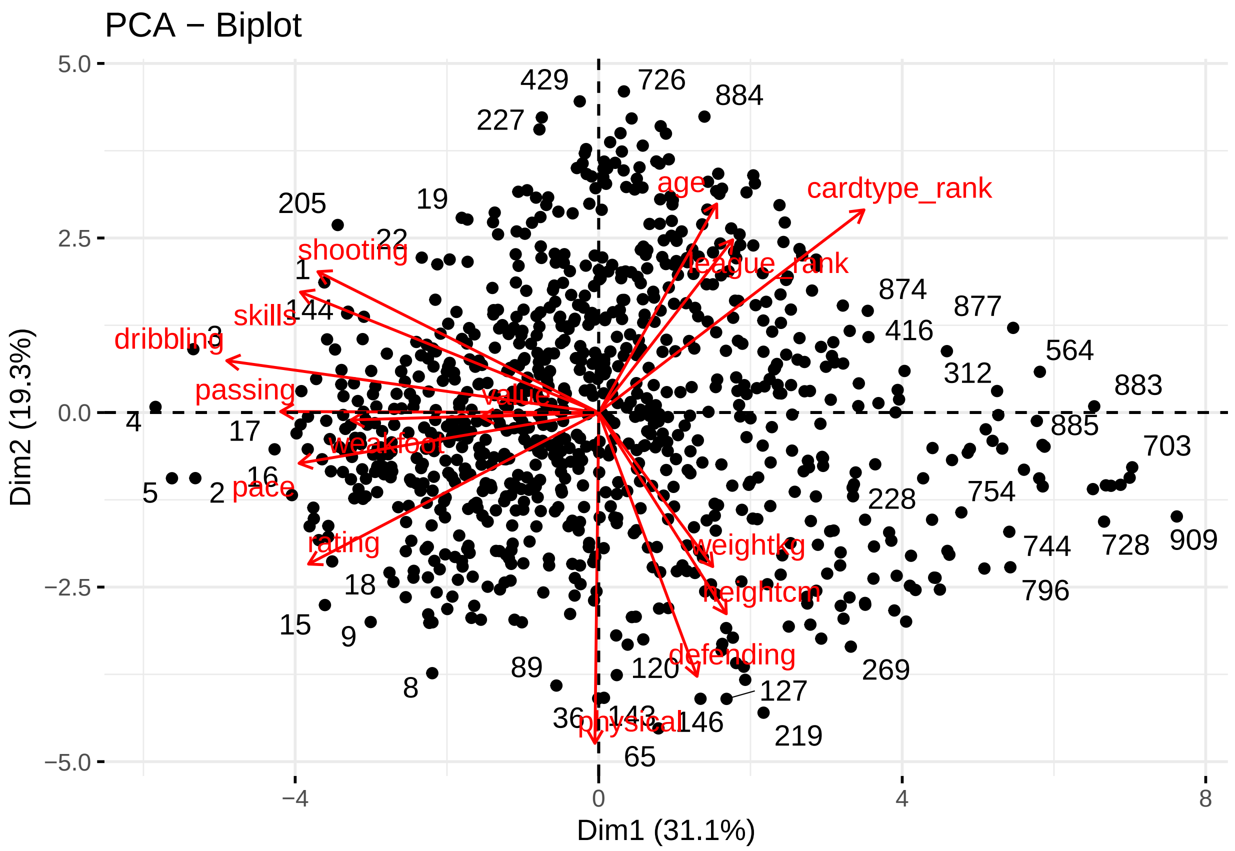
```
##   (Intercept)         value           age      heightcm      weightkg 
##  7.781714e+01  6.379030e-07  3.010908e-02  0.000000e+00  5.003348e-04 
##      weakfoot        skills          pace     dribbling      shooting 
##  5.467975e-01  7.947321e-02  1.871981e-02  5.698786e-02  0.000000e+00 
##     defending       passing      physical   league_rank cardtype_rank 
##  5.590223e-03  1.899040e-02  5.323290e-02 -2.239512e-02 -5.124429e-02
```

The LASSO regression resulting in the highest adjusted R squared value of .681 was produced using a lambda value of .03207641 for the model tuning parameter. As seen from the coefficient estimates for the model produced using the best lambda value, 2 out of the 15 total coefficients, shooting and height were shrunk to zero resulting in no effect on the models predictive outcome.

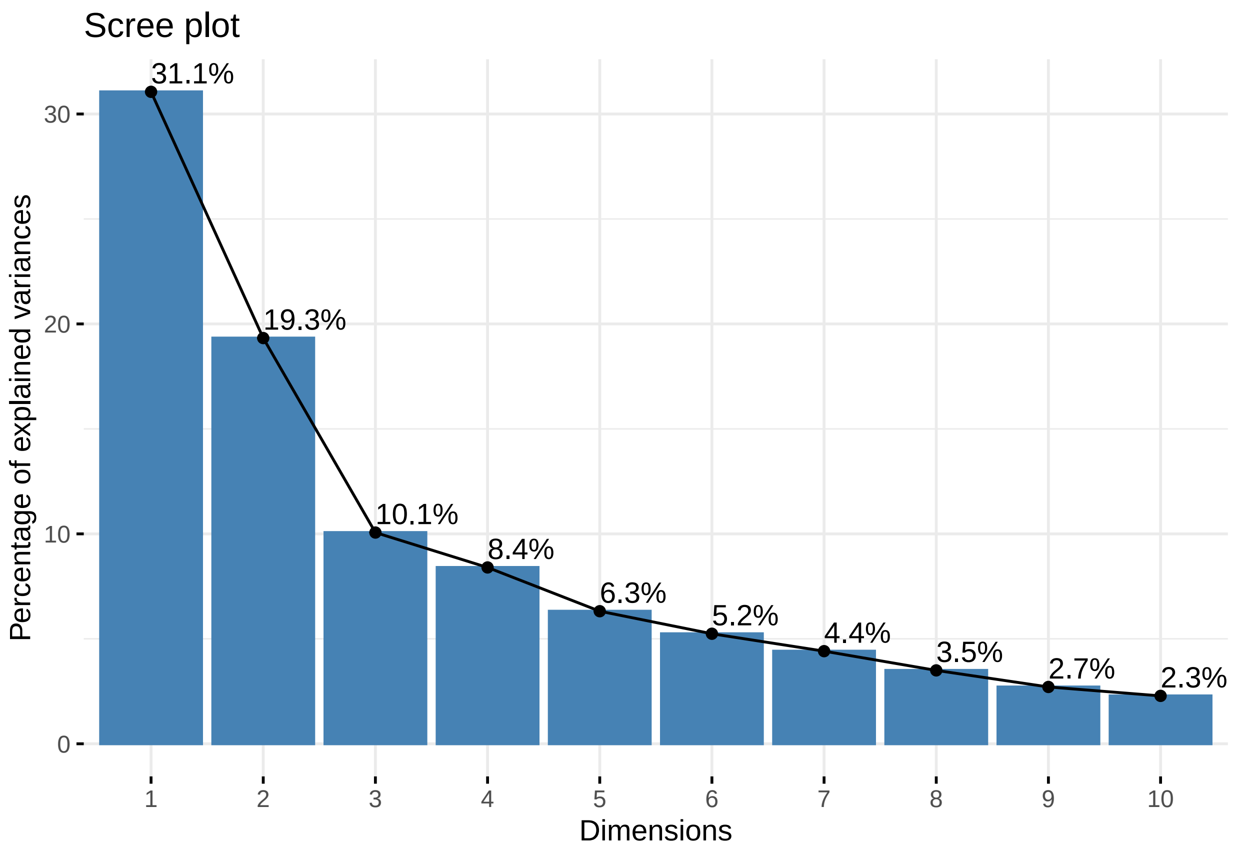
6.3 Dimensionality Reduction

Dimensionality reduction is the process of transforming data from a high dimensional space into a lower dimensional space while retaining the as much information as possible from the original data set. In dimensionality reduction the number of features or variables in the data set is reduced while preserving and accounting for the greatest percentage of the variation from the original data set. Downsides of highly dimensional data include immense computational power and time, difficultly in visualization development, and the curse of dimensionality. The curse of dimensionality refers the a phenomena with highly dimensional data that occurs in the classification algorithms such as K nearest neighbor. As the number of features is increased in a data set the feature space becomes increasingly sparse and the number of observations required to both occupy a given percentage of the feature space and train a successful model grows exponentially. As the number of features is increased to ∞, all training data observations become equidistant from each other observations using euclidean distance and resemble the structure of a hypersphere. This study will employ Principal Component Analysis using the prcomp function in base R and visualizations will be created with the factoextra library.

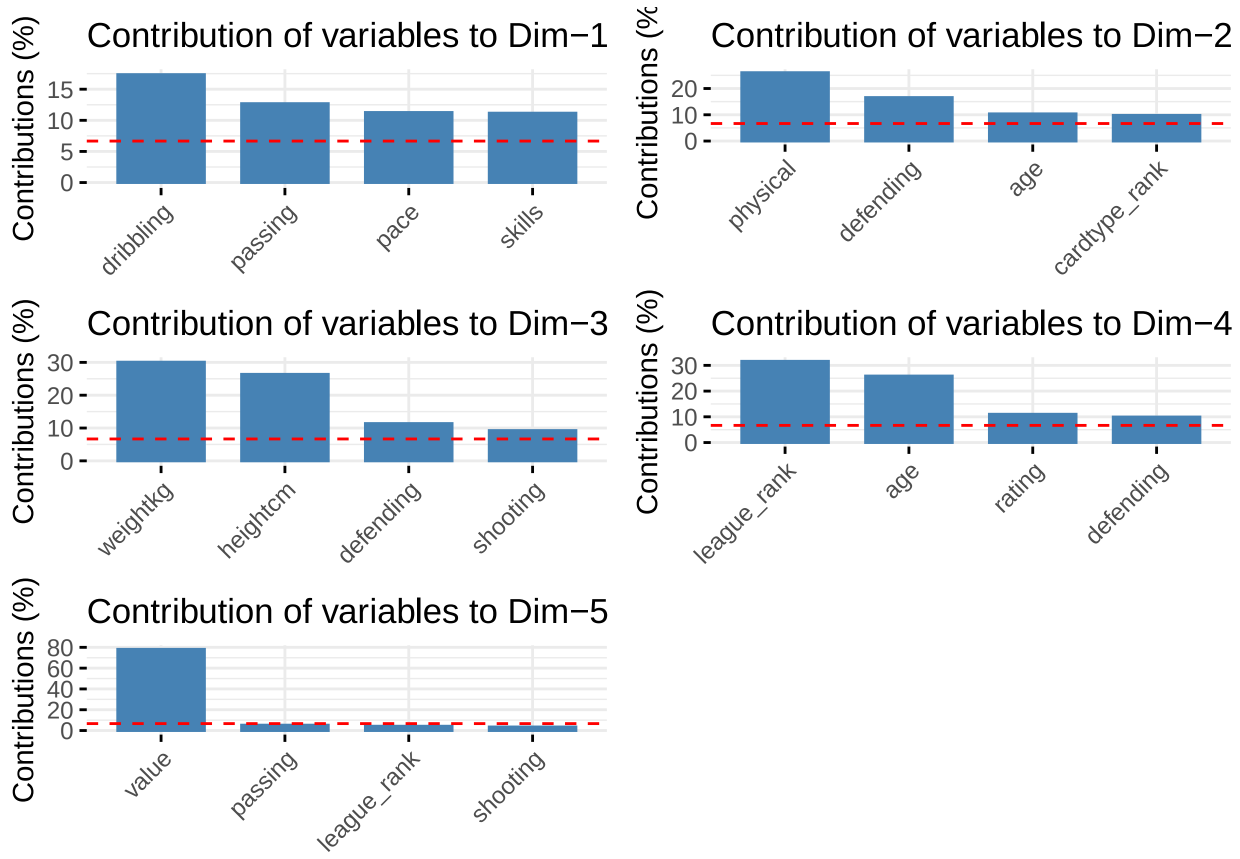
```
##                       PC1          PC2           PC3         PC4         PC5
## value        -0.131802282 -0.006503092  0.2273626074 -0.17996138  0.88377045
## age           0.132098378  0.321804458  0.2651413393 -0.50853488 -0.11131412
## rating       -0.324887363 -0.233671870  0.1042167941 -0.33165179  0.08097818
## heightcm      0.142839565 -0.310369517  0.5133812540  0.15128202 -0.11168626
## weightkg      0.127725363 -0.237625525  0.5482780041  0.17218851 -0.10130057
## weakfoot     -0.277180708 -0.011396872  0.1726582260 -0.06601607 -0.08194047
## skills       -0.333922188  0.185772706  0.0327659926  0.07610951 -0.01979131
## pace         -0.335531084 -0.078316615  0.0007938586  0.05479809  0.02988031
## dribbling    -0.416440454  0.079996920  0.0176391653  0.07265180 -0.13118791
## shooting     -0.314427890  0.217267000  0.3037907768  0.14638690 -0.18503515
## defending     0.110114756 -0.406924190 -0.3370838446 -0.31504754 -0.08979843
## passing      -0.356101419  0.001604531 -0.1184205982 -0.10841071 -0.22738618
## physical     -0.004240696 -0.510231487  0.0892818173 -0.13462799 -0.12084920
## league_rank   0.150085366  0.266389375  0.2056277787 -0.56198975 -0.20361320
## cardtype_rank 0.297036291  0.312623183  0.0398501022  0.24829842  0.04458389
```



The loading's for PC1 through PC5 are displayed above, with the loading for PC6 through PC15 in the appendix of the paper. Scale = TRUE was used to convert every variable to have a total variance of 1, which is required for variables of different units. By examining the loading's for each principal component we can derive a interpretation of what each new dimension represents. In PC1, the loading's with the largest magnitude are dribbling, passing, skills, and pace, these loading's also all occur in the same direction. The negative coefficients indicate that high values for the original variables are mapped to low values in the PC1 dimension. An examination of the biplot above also confirms that the largest loading's from PC1 occur in the same direction. In PC2, physical contained the highest loading magnitude. Thus, the vectors for dribbling from PC1 and physical from PC2 are orthogonal.



The scree plot above details the percentage of variance in the original numeric data set that can be accounted for in each of the newly created principal component dimensions. A higher percentage of retained variance corresponds to a smaller loss of information. Via the scree plot, you can see that the first 5 principal components contain over 75% of the total variability in the original data set. In order to interpret the newly derived dimensions and name them correctly as well as accordingly, a plot will be created below that displays the top 4 contributing variables to each of the 5 principal components.



The three greatest contributors to PC1 are dribbling, passing, pace, and skills. These are common traits that are synonymous with players who typically play in attacking positions on the field, thus PC1 can be understood as attacking traits. The three greatest contributors to PC2 are physical, defending, and age. These traits often represent defensive players as they are typically larger is stature and older in age then their attacking counterparts. PC2 will be understood as defensive traits. The three greatest contributors to PC3 are weight, height, and defending. Like previously mentioned defenders are usually larger in stature, but with weight and height containing a much larger contribution percentage than defending in this principal component, PC3 can be understood as player size. The three greatest contributors to PC4 are league rank, age, and rating. The traits league rank and age are influential in representing the league a player plays in as leagues around the world and in Fifa 23 have different mean player ages. For example the average age of a player card in the Icons league is 55 because these cards are given to since retired players. Thus, PC4 can be understood as league. Finally, the three greatest contributors to PC5 are value, passing, and league rank. But, since the contribution percentage for value far outweighs other loadings, PC5 can be understood as value.

```
k <- 5
reduced_data <- as.matrix(numeric_data) %*% P$rotation[, 1:5]
reduced_data <- as.data.frame(reduced_data)
names(reduced_data)[names(reduced_data) == "PC1"] <- "attacking"
names(reduced_data)[names(reduced_data) == "PC2"] <- "defending"
names(reduced_data)[names(reduced_data) == "PC3"] <- "playersize"
names(reduced_data)[names(reduced_data) == "PC4"] <- "league"
names(reduced_data)[names(reduced_data) == "PC5"] <- "value"
head(reduced_data)
```

```
##    attacking defending playersize     league   value
## 1  -379704.7 -18827.76   654976.1  -518346.3 2545163
## 2 -1291784.0 -63852.77  2228332.9 -1763638.8 8660860
## 3  -784351.3 -38802.11  1352967.7 -1070794.6 5258346
## 4 -1098041.5 -54289.81  1894087.9 -1499098.7 7361722
## 5 -1098040.9 -54297.55  1894095.8 -1499092.3 7361721
## 6  -229452.1 -11450.74   395775.1  -313185.1 1537661
```

Above is the newly formed reduced data frame employing principal components one through five. The principal components have since been renamed to the understood interpretation of composition found above. This new reduced data frame consists of 5 variables as compared to the original numeric data frame that consisted of 15 variables. The new reduced data frame retained 75% of the variance from the original numeric data frame.

7 Results

7.1 Comparision of Best Subset Selection and LASSO Regression

Using both best subset selection and LASSO regression a multiple linear model was built to predict player rating using the numeric data frame created from the cleaned tidy fifa data frame. The best subset selection method model with the highest adjusted R squared of 0.675 was built using 8 variables. The LASSO regression utilizing all 15 numeric variables shrunk 2 variables to zero, resulting in a model that essentially uses 13 variables excluding height and shooting. The LASSO regression model had an adjusted R sqaured of 0.681, barely better than the best subset selection model. The 8 variables used in the best subset selection model were value, age, weakfoot, dribbling, defending, physical, league_rank, and cardtype_rank. The 13 variables used in the LASSO regression model were value, age, weight, weakfoot, skills, pace, dribbling, defending, passing, physical, league_rank, and cardtype_rank. Comparing coefficient estimates below..

```
##   (Intercept)         value           age      weakfoot     dribbling
##  7.805672e+01  6.766768e-07  3.452439e-02  5.925446e-01  9.148227e-02
##     defending      physical   league_rank cardtype_rank
##  8.422565e-03  5.244909e-02 -2.956854e-02 -5.336696e-02
```

```
##   (Intercept)         value           age      heightcm      weightkg
##  7.781714e+01  6.379030e-07  3.010908e-02  0.000000e+00  5.003348e-04
##      weakfoot        skills          pace     dribbling      shooting
##  5.467975e-01  7.947321e-02  1.871981e-02  5.698786e-02  0.000000e+00
##     defending       passing      physical   league_rank cardtype_rank
##  5.590223e-03  1.899040e-02  5.323290e-02 -2.239512e-02 -5.124429e-02
```

It is evident that the coefficient estimates between variables used in both models are very similar. The following 5 variables used in the LASSO regression that were not used in the best subset selection: weight, skills, pace, shooting, and passing all have small coefficient values indicating that a one unit increase in the coefficient value does not effect the predictive rating value much. Since the best subset selection model was built on 8 variables as compared to 13, and the adjusted R squared was less than 0.01 lower, that will be the model selected. The increase of 0.01 in adjusted R squared from the LASSO regression model is not worth the added complexity of 5 extra explanatory variables.

Diving into the interpretation of the coefficient values for the chosen best subset selection model, an intercept of 7.805672e+01 indicates that if every explanatory variable was equal to zero the mean responses variable value would be 7.805672e+01. The coefficient for value indicates that for a one unit increase in value, a 6.766768e-07 unit increase in rating can be expected. The coefficient for age indicates that for a one unit increase in age, a 3.452439e-02 unit increase in rating can be expected. The coefficient for weakfoot indicates that for a one unit increase in weakfoot, a 5.925446e-01 unit increase in rating can be expected.

The coefficient for dribbling indicates that for a one unit increase in dribbling, a 9.148227e-02 unit increase in rating can be expected. The coefficient for defending indicates that for a one unit increase in defending, a 8.422565e-03 unit increase in rating can be expected. The coefficient for physical indicates that for a one unit increase in physicality, a 5.244909e-02 unit increase in rating can be expected. The coefficient for league_rank indicates that for a one unit increase in league_rank, a -2.956854e-02 unit decrease in rating can be expected. This makes sense as recalling from earlier, league_rank is comprised of the mean player rating in the given league with 1 representing the league with the high mean player rating. Finally, The coefficient for cardtype_rank indicates that for a one unit increase in cardtype_rank, a -5.336696e-02 unit increase in rating can be expected. This also check out as similar to league_rank, cardtype_rank is comprised of the mean player rating for a given cardtype, with 1 corresponding to the league with the highest mean player rating.

7.2 Principal Component Analysis Results

As discussed above in the section were the principal component analysis was conducted. A new data frame was created utilizing the first 5 principal components. The values for each observation for a given principal component 1 through 5 were calculated based on the loading's for each variable in the original data set for the respective principal component. All 15 principal component loading's can be found in the appendix of the paper. The new reduced data frame features 5 variables (principal components) as compared to the original numeric data frame that consisted of 15 variables. The reduced data frame omitted 10 features or dimensions while only ridding of 25% of total variance. Principal Component 1 through 5 in the reduced data frame were renamed to the understood interpretation from the contribution plots. The five principal components were renamed to attacking, defending, playersize, league, and value. Observation one is shown below.

```
##    attacking defending playersize    league   value
## 1 -379704.7 -18827.76   654976.1 -518346.3 2545163
```

8 Discussion

8.1 Furture Work and Use Cases

An expansion on the work conducted in this study would entail the exploration of model creation using the reduced data from the principal component analysis. A multiple linear regression model to predict rating could again be conduced, this time using the reduced data frame and a comparison could be made to the the best subset selection model and the LASSO regression model created in the study. Another potential use case or area of future work would be creating a function that prompts the user for a max budget or price they are willing to spend to build a full team of 11 players. Building an algorithm the 11 highest rated players, one from each position, for the budgeted price could be found, resulting in the best deal or team for that given price. Constraints would be built into the model, for example the median value would be collected for each position. Using those median values a ratio for each positions value could be calculated by comparing it to the median value for a complete team created. This ratio would allow the algorithm to calculate a max price from each position in the given budget, then select the highest rated player at each position under the allotted max price. This essentially applies a simplified money ball technique to the Fifa video game Ultimte Team, allowing players with a smaller budget to remain competitive in online matches among user.

9 References

Al-Asadi, Mustafa. “Predict the Value of Football Players Using FIFA Video Game Data and Machine Learning Techniques.” Ieexplore, 25 Feb. 2022, ieeexplore.ieee.org/document/9721908/.

Cotta, Leonardo. “Using FIFA Soccer Video Game Data for Soccer Analytics.” Universidade Federal De Minas Gerais, Brazil, homepages.dcc.ufmg.br/~fabricio/download/lssa_fifa_CR.pdf. Accessed 13 Dec. 2023.

“EA SportsTM FIFA 23 - Official Site - Electronic Arts.” Ea, www.ea.com/games/fifa/fifa-23. Accessed 13 Dec. 2023.

Futbin. “FIFA 23 Players.” FUTBIN, www.futbin.com/23/players?page=1. Accessed 13 Dec. 2023.

Mbu-Ogar, Seyi. “FIFA 23 Exploratory Data Analysis.” Medium, Medium, 5 July 2023, medium.com/@seyiogar/fifa-23-exploratory-data-analysis-ed56ea424f48.

Rodriguez V., Erick Daniel Rodriguez. “Machine Learning on FIFA 20.” Medium, Towards Data Science, 13 Nov. 2020, towardsdatascience.com/fifa-20-player-clustering-f500cf0792c5.

Silvestri, Christian. “Top 10 Best Selling FIFA Games of All Time.” FIFA Infinity, 10 Aug. 2023, www.fifa-infinity.com/fifa-23/top-10-best-selling/#:~:text=FIFA%2023%20%E2%80%93%2010.3%2B%20million%20units%20sold&tex

Smith, Kieran. “When Was the First Ever FIFA Ultimate Team?” GiveMeSport, 16 Jan. 2023, www.givemesport.com/88105399-when-was-the-first-ever-fifa-ultimate-team/.

10 Appendix

10.1 All Principal Component Analysis Loadings

```
## Standard deviations (1, .., p=15):
##  [1] 2.1583245 1.7026412 1.2286050 1.1224995 0.9733604 0.8867202 0.8135930
##  [8] 0.7242311 0.6372065 0.5851275 0.5542447 0.5261392 0.4373568 0.3561838
## [15] 0.3200964
##
## Rotation (n x k) = (15 x 15):
##                        PC1          PC2           PC3         PC4         PC5
## value         -0.131802282 -0.006503092  0.2273626074 -0.17996138  0.88377045
## age            0.132098378  0.321804458  0.2651413393 -0.50853488 -0.11131412
## rating        -0.324887363 -0.233671870  0.1042167941 -0.33165179  0.08097818
## heightcm       0.142839565 -0.310369517  0.5133812540  0.15128202 -0.11168626
## weightkg       0.127725363 -0.237625525  0.5482780041  0.17218851 -0.10130057
## weakfoot      -0.277180708 -0.011396872  0.1726582260 -0.06601607 -0.08194047
## skills        -0.333922188  0.185772706  0.0327659926  0.07610951 -0.01979131
## pace          -0.335531084 -0.078316615  0.0007938586  0.05479809  0.02988031
## dribbling     -0.416440454  0.079996920  0.0176391653  0.07265180 -0.13118791
## shooting      -0.314427890  0.217267000  0.3037907768  0.14638690 -0.18503515
## defending      0.110114756 -0.406924190 -0.3370838446 -0.31504754 -0.08979843
## passing       -0.356101419  0.001604531 -0.1184205982 -0.10841071 -0.22738618
## physical      -0.004240696 -0.510231487  0.0892818173 -0.13462799 -0.12084920
## league_rank    0.150085366  0.266389375  0.2056277787 -0.56198975 -0.20361320
## cardtype_rank  0.297036291  0.312623183  0.0398501022  0.24829842  0.04458389
##                         PC6         PC7          PC8         PC9        PC10
## value         -0.0580228490  0.24491390 -0.066619163  0.10838591 -0.12325454
## age            0.0235091149 -0.06964990  0.046428907 -0.48427460 -0.28478674
## rating        -0.0008899856 -0.10941363  0.039401982 -0.31548643  0.30927835
## heightcm       0.0925479600  0.10482882 -0.196003494  0.30557028  0.02542631
## weightkg      -0.1884374759  0.07478024  0.530228928 -0.25043462 -0.02735386
## weakfoot      -0.6237400275 -0.52356167 -0.371231841  0.10409477 -0.09792904
## skills        -0.2346147900 -0.05493724  0.571270182  0.40525860 -0.17153472
## pace           0.5431920858 -0.38271533  0.142146363  0.02610844 -0.46968639
## dribbling      0.1589140523  0.20997890 -0.084075837 -0.10121294 -0.09841246
## shooting       0.0888183558  0.23988946 -0.291590882 -0.03624383  0.02274597
## defending     -0.2421555713  0.16181524  0.105813545  0.06463204 -0.33691321
## passing       -0.2114405065  0.57989900 -0.002083241 -0.03762173 -0.08272621
## physical       0.1370132193  0.05701118 -0.231335171  0.07285478 -0.33327124
## league_rank    0.1737118453  0.04243736  0.033784878  0.54412321  0.09275404
## cardtype_rank -0.1655762521  0.11769155 -0.173127523 -0.06187135 -0.54480078
##                     PC11         PC12         PC13         PC14         PC15
## value          0.05241130 -0.101384617 -0.093704159  0.021571561 -0.021388146
## age           -0.09154895  0.244680235 -0.380724847 -0.028939189  0.021681735
## rating        -0.14642915  0.315269678  0.614452721  0.041601014  0.060045413
## heightcm       0.18779202  0.593399169 -0.136309509  0.119255387  0.110296977
## weightkg       0.17729825 -0.391085740  0.104269201  0.009922064 -0.075089475
## weakfoot       0.19654868 -0.134490043 -0.052986956 -0.030238811 -0.017591020
## skills        -0.41059585  0.298618626 -0.061474941 -0.063844575 -0.055124104
## pace           0.28020177 -0.083275824  0.064459170  0.095691312  0.309205734
## dribbling      0.28150493  0.103679886 -0.005897859  0.007263785 -0.781635097
## shooting      -0.42627290 -0.282346428 -0.008398528  0.512837020  0.170826982
## defending      0.04280019  0.017644588 -0.005661544  0.615454088 -0.081709444
## passing        0.30730098  0.005870392 -0.022550017 -0.327106385  0.452194281
## physical      -0.48255870 -0.191757816 -0.008920501 -0.465663723 -0.143082998
## league_rank    0.15715520 -0.242859742  0.279348956 -0.022901516 -0.083108516
## cardtype_rank -0.01471236  0.156923211  0.591405584 -0.047904425 -0.005974548
```

10.2 Code

```r
knitr::opts_chunk$set(echo = TRUE)
library(rvest) # used for scraping data
library(dplyr) # used for piping, mutate, filter
library(tidyverse) # used for data cleaning
library(tidyr) # used for the separate command
library(leaps) # used for subset selection
library(glmnet) # used for shrinkage methods
library(factoextra) # used for PCA visualizations
library(gridExtra) # used for arranging visualizations
set.seed(1)
# Load cleaned data here. If cleaning was required make sure to have a separate r file for the cleaning
fifa_tidy<-read.csv("fifa_tidy.csv")
fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "X")]
get_player = function(link) {
  prac <- read_html(link)
  pname = prac %>% html_nodes("tr:nth-child(1) .table-row-text") %>% html_text()
  print(pname)
  return(pname)
}

get_cardtype = function(link) {
  prac <- read_html(link)
  cardtype = prac %>% html_nodes("tr:nth-child(2) .table-row-text") %>% html_text()
  print(cardtype)
  return(cardtype)
}

get_position = function(link) {
  prac <- read_html(link)
  position = prac %>% html_nodes(".pcdisplay-pos") %>% html_text()
  print(position)
  return(position)
}

get_age = function(link) {
  prac <- read_html(link)
  age = prac %>% html_nodes(".info_tr_1 .table-row-text") %>% html_text()
  print(age)
  return(age)
}

get_rating = function(link) {
  prac <- read_html(link)
  rating = prac %>% html_nodes(".pcdisplay-rat") %>% html_text()
  print(rating)
  return(rating)
}

get_club = function(link) {
  prac <- read_html(link)
  club = prac %>% html_nodes("tr:nth-child(3) .table-row-text") %>% html_text()
  print(club)
  return(club)
}

get_league = function(link) {
  prac <- read_html(link)
  league = prac %>% html_nodes("tr:nth-child(5) .table-row-text") %>% html_text()
  print(league)
  return(league)
}

get_nation = function(link) {
  prac <- read_html(link)
  nation = prac %>% html_nodes("tr:nth-child(4) .table-row-text") %>% html_text()
  print(nation)
  return(nation)
}

get_height = function(link) {
  prac <- read_html(link)
  height = prac %>% html_nodes("tr:nth-child(10) .table-row-text") %>% html_text()
  print(height)
  return(height)
}

get_weight = function(link) {
  prac <- read_html(link)
  weight = prac %>% html_nodes("tr:nth-child(11) .table-row-text") %>% html_text()
  print(weight)
  return(weight)
}

get_foot = function(link) {
  prac <- read_html(link)
  foot = prac %>% html_nodes("tr:nth-child(9) .table-row-text") %>% html_text()
  print(foot)
  return(foot)
}

get_weakfoot = function(link) {
  prac <- read_html(link)
  weakfoot = prac %>% html_nodes("tr:nth-child(7) .table-row-text") %>% html_text()
  print(weakfoot)
  return(weakfoot)
}

get_attackworkrate = function(link) {
  prac <- read_html(link)
  attackworkrate = prac %>% html_nodes("tr:nth-child(13) .table-row-text") %>% html_text()
  print(attackworkrate)
  return(attackworkrate)
}
```

```r
get_defenseworkrate = function(link) {
  prac <- read_html(link)
  defenseworkrate = prac %>% html_nodes("tr:nth-child(14) .table-row-text") %>% html_text()
  print(defenseworkrate)
  return(defenseworkrate)
}

get_pace = function(link) {
  prac <- read_html(link)
  pace = prac %>% html_nodes("#main-pace-val-0 .stat_val") %>% html_text()
  print(pace)
  return(pace)
}

get_dribbling = function(link) {
  prac <- read_html(link)
  dribbling = prac %>% html_nodes("#main-dribblingp-val-0 .stat_val") %>% html_text()
  print(dribbling)
  return(dribbling)
}

get_shooting = function(link) {
  prac <- read_html(link)
  shooting = prac %>% html_nodes("#main-shooting-val-0 .stat_val") %>% html_text()
  print(shooting)
  return(shooting)
}

get_defending = function(link) {
  prac <- read_html(link)
  defending = prac %>% html_nodes("#main-defending-val-0 .stat_val") %>% html_text()
  print(defending)
  return(defending)
}

get_passing = function(link) {
  prac <- read_html(link)
  passing = prac %>% html_nodes("#main-passing-val-0 .stat_val") %>% html_text()
  print(passing)
  return(passing)
}

get_physical = function(link) {
  prac <- read_html(link)
  physical = prac %>% html_nodes("#main-heading-val-0 .stat_val") %>% html_text()
  print(physical)
  return(physical)
}

get_skills = function(link) {
  prac <- read_html(link)
  skills = prac %>% html_nodes("tr:nth-child(6) .table-row-text") %>% html_text()
  print(skills)
  return(skills)
}


fifa_df3 = data.frame()

for (page_result in seq(from = 1, to = 34, by = 1)) {
  link = paste0("https://www.futbin.com/23/players?page=",
                page_result)

  page = read_html(link)

  player_links = page %>% html_nodes(".get-tp") %>% html_attr("href") %>%  paste0("https://www.futbin.c

  pname1 = page %>% html_nodes(".get-tp") %>% html_text()
  print(pname1)

  Sys.sleep(30)

  card = page %>% html_nodes(".mobile-hide-table-col div:nth-child(1)") %>% html_text()
  print(card)

  Sys.sleep(30)

  pprice = page %>% html_nodes("span.font-weight-bold") %>% html_text()
  print(pprice)

  Sys.sleep(30)

  player = sapply(player_links, FUN = get_player, USE.NAMES = FALSE)

  Sys.sleep(30)

  cardtype = sapply(player_links, FUN = get_cardtype, USE.NAMES = FALSE)

  Sys.sleep(30)

  position = sapply(player_links, FUN = get_position, USE.NAMES = FALSE)

  Sys.sleep(30)

  age = sapply(player_links, FUN = get_age, USE.NAMES = FALSE)

  Sys.sleep(30)

  rating = sapply(player_links, FUN = get_rating, USE.NAMES = FALSE)

  Sys.sleep(30)

  club = sapply(player_links, FUN = get_club, USE.NAMES = FALSE)

  Sys.sleep(30)
```

```
league = sapply(player_links, FUN = get_league, USE.NAMES = FALSE)

Sys.sleep(30)

nation = sapply(player_links, FUN = get_nation, USE.NAMES = FALSE)

Sys.sleep(30)

height = sapply(player_links, FUN = get_height, USE.NAMES = FALSE)

Sys.sleep(30)

weight = sapply(player_links, FUN = get_weight, USE.NAMES = FALSE)

Sys.sleep(30)

foot = sapply(player_links, FUN = get_foot, USE.NAMES = FALSE)

Sys.sleep(30)

weakfoot = sapply(player_links, FUN = get_weakfoot, USE.NAMES = FALSE)

Sys.sleep(30)

attackworkrate = sapply(player_links, FUN = get_attackworkrate, USE.NAMES = FALSE)

Sys.sleep(30)

defenseworkrate = sapply(player_links, FUN = get_defenseworkrate, USE.NAMES = FALSE)

Sys.sleep(30)

pace = sapply(player_links, FUN = get_pace, USE.NAMES = FALSE)

Sys.sleep(30)

dribbling = sapply(player_links, FUN = get_dribbling, USE.NAMES = FALSE)

Sys.sleep(30)

shooting = sapply(player_links, FUN = get_shooting, USE.NAMES = FALSE)

Sys.sleep(30)

defending = sapply(player_links, FUN = get_defending, USE.NAMES = FALSE)

Sys.sleep(30)

passing = sapply(player_links, FUN = get_passing, USE.NAMES = FALSE)

Sys.sleep(30)

physical = sapply(player_links, FUN = get_physical, USE.NAMES = FALSE)
```

```
  Sys.sleep(30)

  skills = sapply(player_links, FUN = get_skills, USE.NAMES = FALSE)

  Sys.sleep(120)


  fifa_df3 = rbind(fifa_df3, data.frame(player,
                                        pname1,
                                        pprice,
                                        card,
                                        cardtype,
                                        position,
                                        age,
                                        rating,
                                        club,
                                        league,
                                        nation,
                                        height,
                                        weight,
                                        foot,
                                        weakfoot,
                                        skills,
                                        attackworkrate,
                                        defenseworkrate,
                                        pace,
                                        dribbling,
                                        shooting,
                                        defending,
                                        passing,
                                        physical,
                                        stringsAsFactors = FALSE))
}

write.csv(fifa_df3, "fifa_df3.csv")
get_player = function(link) {
  prac <- read_html(link)
  pname = prac %>% html_nodes("tr:nth-child(1) .table-row-text") %>% html_text()
  print(pname)
  return(pname)
}
get_cardtype = function(link) {
  prac <- read_html(link)
  cardtype = prac %>% html_nodes("tr:nth-child(2) .table-row-text") %>% html_text()
  print(cardtype)
  return(cardtype)
}
fifa_df3 = data.frame()
for (page_result in seq(from = 1, to = 34, by = 1)) {
  link = paste0("https://www.futbin.com/23/players?page=",
                page_result)
  page = read_html(link)
  player_links = page %>% html_nodes(".get-tp") %>% html_attr("href") %>%  paste0("https://www.futbin.c
```

```r
  pname1 = page %>% html_nodes(".get-tp") %>% html_text()
  print(pname1)
  Sys.sleep(30)
  card = page %>% html_nodes(".mobile-hide-table-col div:nth-child(1)") %>% html_text()
  print(card)
  Sys.sleep(30)
  pprice = page %>% html_nodes("span.font-weight-bold") %>% html_text()
  print(pprice)
  Sys.sleep(30)
  player = sapply(player_links, FUN = get_player, USE.NAMES = FALSE)
  Sys.sleep(30)
  cardtype = sapply(player_links, FUN = get_cardtype, USE.NAMES = FALSE)
  Sys.sleep(120)
  fifa_df3 = rbind(fifa_df3, data.frame(player,
                                        pname1,
                                        pprice,
                                        card,
                                        cardtype,
                                        stringsAsFactors = FALSE))
}

write.csv(fifa_df3, "fifa_df3.csv")
fifa <- read.csv("fifa_df.csv")
fifa <- fifa %>% mutate_all(trimws)
transposed_fifa <- t(fifa[1:2,])
transposed_fifa
fifa_tidy <- fifa[, -which(names(fifa) == "X.1")]
fifa_tidy <- fifa %>% filter(trimws(foot) %in% c("Right","Left"))
nrow(fifa_tidy)
nrow(fifa)
fifa_tidy <- fifa_tidy %>% mutate_all(trimws)
fifa_tidy$height <- parse_number(fifa_tidy$height)
names(fifa_tidy)[names(fifa_tidy) == "height"] <- "heightcm"
names(fifa_tidy)[names(fifa_tidy) == "weight"] <- "weightkg"
names(fifa_tidy)[names(fifa_tidy) == "pprice"] <- "price"
names(fifa_tidy)[names(fifa_tidy) == "pname1"] <- "playername"
names(fifa_tidy)[names(fifa_tidy) == "cardtype"] <- "attribute"
names(fifa_tidy)[names(fifa_tidy) == "card"] <- "cardtype"
fifa_tidy[1,8]
fifa_tidy[101,8]
standardize_age <- function(age) {
  if (grepl("\\d{2}-\\d{2}-\\d{4}", age)) {
    birth_date <- as.Date(age, format = "%d-%m-%Y")
    age_in_years <- as.numeric(difftime(Sys.Date(), birth_date, units = "days") / 365)
    return(paste(round(age_in_years), "years old"))
  } else {
    return(age)
  }
}
fifa_tidy$age <- sapply(fifa_tidy$age, standardize_age)
fifa_tidy$age <- parse_number(fifa_tidy$age)
fifa_tidy[1,8]
fifa_tidy[101,8]
```

```r
fifa_tidy <- separate(data = fifa_tidy, col = price, into = c("value", "label"), sep = -1)
fifa_tidy <- fifa_tidy %>%
  mutate(value = case_when(
    label == "M" ~ as.numeric(value) * 1e+06,
    label == "K" ~ as.numeric(value) * 1e+03,
    TRUE ~ as.numeric(value)
  ))
fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "label")]
fifa_tidy$value[fifa_tidy$value == 0] <- NA
fifa_tidy$cardtype[fifa_tidy$cardtype =="" ] <- NA

fifa_tidy$rating <- as.numeric(fifa_tidy$rating)
rank<-fifa_tidy %>% group_by(league) %>% summarise(rank = mean(rating)) %>% arrange(desc(rank))

fifa_tidy <- fifa_tidy %>%
  mutate(league_rank = case_when(
    league == "Chinese FA Super L. (CHN 1)" ~ 1,
    league == "LaLiga Santander" ~ 2,
    league == "Ligue 1" ~ 3,
    league == "Serie A TIM" ~ 4,
    league == "MBS Pro League (SAU 1)" ~ 5,
    league == "Premier League" ~ 6,
    league == "Eliteserien (NOR 1)" ~ 7,
    league == "Italy Serie B (2)" ~ 8,
    league == "Bundesliga" ~ 9,
    league == "Major League Soccer" ~ 10,
    league == "Süper Lig (TUR 1)" ~ 11,
    league == "Liga NOS (POR 1)" ~ 12,
    league == "3F Superliga (DEN 1)" ~ 13,
    league == "1A Pro League (BEL 1)" ~ 14,
    league == "EFL League One (ENG 3)" ~ 15,
    league == "Polski Ekstraklasa (POL 1)r" ~ 16,
    league == "Scottish Premiership (SPFL)" ~ 17,
    league == "A-League (AUS 1)" ~ 18,
    league == "Eredivisie" ~ 19,
    league == "Icons" ~ 20,
    league == "EFL Championship (ENG 2)" ~ 21,
    league == " Ligue 2 (FRA 2)" ~ 22,
    league == "National League (ENG 5)" ~ 23,
    league == "World Cup" ~ 24,
    league == "EFL League Two (ENG 4)" ~ 25,
    league == "Hellas Liga (GRE 1)" ~ 26,
    TRUE ~ NA_real_
  ))
cardtype_ranks <- fifa_tidy %>%
  group_by(cardtype) %>%
  summarise(rank = mean(rating)) %>%
  arrange(desc(rank)) %>%
  mutate(cardtype_rank = row_number())

add_cardtype_rank <- function(fifa_observation) {
  result <- fifa_observation %>%
    left_join(cardtype_ranks %>% select(cardtype, cardtype_rank), by = "cardtype")
  return(result)
}

fifa_tidy <- fifa_tidy %>%
  group_split() %>%
  map_df(add_cardtype_rank)
fifa_tidy <- fifa_tidy %>%
  group_by(rating, cardtype) %>%
  mutate(value = ifelse(is.na(value), median(value, na.rm = TRUE), value))
fifa_tidy <- fifa_tidy %>% filter(value != "NaN")
fifa_tidy$rating <- as.numeric(fifa_tidy$rating)
fifa_tidy$league_rank <- as.numeric(fifa_tidy$league_rank)
fifa_tidy$cardtype_rank <- as.numeric(fifa_tidy$cardtype_rank)
fifa_tidy$weightkg<- as.numeric(fifa_tidy$weightkg)
fifa_tidy$weakfoot<- as.numeric(fifa_tidy$weakfoot)
fifa_tidy$skills<- as.numeric(fifa_tidy$skills)
fifa_tidy$pace<- as.numeric(fifa_tidy$pace)
fifa_tidy$dribbling<- as.numeric(fifa_tidy$dribbling)
fifa_tidy$shooting<- as.numeric(fifa_tidy$shooting)
fifa_tidy$defending<- as.numeric(fifa_tidy$defending)
fifa_tidy$passing<- as.numeric(fifa_tidy$passing)
fifa_tidy$physical<- as.numeric(fifa_tidy$physical)

fifa_tidy$attribute <- as.factor(fifa_tidy$attribute)
fifa_tidy$position <- as.factor(fifa_tidy$position)
fifa_tidy$foot <- as.factor(fifa_tidy$foot)
fifa_tidy$attackworkrate <- as.factor(fifa_tidy$attackworkrate)
fifa_tidy$defenseworkrate <- as.factor(fifa_tidy$defenseworkrate)

fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "X.1")]
fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "X")]
fifa_tidy<-read.csv("fifa_tidy.csv")
fifa_tidy <- fifa_tidy[, -which(names(fifa_tidy) == "X")]
transposed_fifa_tidy <- t(fifa_tidy[1,])
transposed_fifa_tidy
fifa_tidy2 <- na.omit(fifa_tidy)
numeric_cols <- sapply(fifa_tidy2, is.numeric)
numeric_data <- fifa_tidy2[, numeric_cols]
best<-regsubsets(rating~., data = numeric_data)
bestsum<-summary(best)
bestsum
which.max(bestsum$adjr2)
max(bestsum$adjr2)
coef(best,8)
x<-model.matrix(rating~.,numeric_data)[,-1]
y<-numeric_data$rating

train <- sample (1: nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]

cv.out <- cv.glmnet(x[train , ], y[train], alpha = 1)
plot(cv.out)
```

```r
bestlam <- cv.out$lambda.min
bestlam
grid<-10^seq(10,-2,length=100)
out <- glmnet(x, y, alpha = 1, lambda = grid)
max(out$dev.ratio)
lasso.coef <- predict(out , type = "coefficients",s = bestlam)[1:15, ]
lasso.coef
P <- prcomp(numeric_data, scale = TRUE)
loadings <- P$rotation[, 1:5]
loadings
fviz_pca_biplot(P, repel = TRUE,
                col.var = "red",
                col.ind = "black",
                max.overlaps =Inf)
fviz_eig(P, addlabels = TRUE)
plot1<-fviz_contrib(P, choice = "var", axes = 1, top = 4)
plot2<-fviz_contrib(P, choice = "var", axes = 2, top = 4)
plot3<-fviz_contrib(P, choice = "var", axes = 3, top = 4)
plot4<-fviz_contrib(P, choice = "var", axes = 4, top = 4)
plot5<-fviz_contrib(P, choice = "var", axes = 5, top = 4)
grid.arrange(plot1, plot2, plot3, plot4, plot5, ncol = 2)
k <- 5
reduced_data <- as.matrix(numeric_data) %*% P$rotation[, 1:5]
reduced_data <- as.data.frame(reduced_data)
names(reduced_data)[names(reduced_data) == "PC1"] <- "attacking"
names(reduced_data)[names(reduced_data) == "PC2"] <- "defending"
names(reduced_data)[names(reduced_data) == "PC3"] <- "playersize"
names(reduced_data)[names(reduced_data) == "PC4"] <- "league"
names(reduced_data)[names(reduced_data) == "PC5"] <- "value"
head(reduced_data)
coef(best,8)
lasso.coef
reduced_data[1,]
P
```