# Project Paper: Soccer Player Valuation Prediction

Firass Elhouat, Lex Brunett, Marco Palmisano

2024-11-26

## Introduction

In our exploratory data analysis (EDA) section, we conducted an in-dept exploration of the football dataset, which includes various variables related to player evaluation and the clubs to which these players are affiliated with. Furthermore, we focused on key features such as age, foot preference, height, player position, and nationality, as these are expected to contribute significantly to estimating the response variable (market_value_in_eur).

However, to fully capture the complexity of player valuation, it was crucial to incorporate additional variables that are aimed at providing deeper insights into player performance. In particular, we identified the need for position - specific performance metrics that better reflect the contributions of players in different roles, such as goalkeepers, midfielders, and defenders.

While the dataset we initially introduced in our EDA section did contain variables that captured such metrics such as goals, assists, and yellow/red cards, these metrics are more relevant for attacking players. For other positions, such as defense and midfield, these variables alone may not adequately represent a player's value. Therefore, expanding the dataset to include detailed performances metrics, such as blocks, interceptions, clearances, passing accuracy, and other position specific actions, will be essential for capturing the full range of actors that influence a player's market value.

## Data Cleaning and Feature Engineering

During the data cleaning process, we identified and removed specific columns that were not essential for this study. For instance, we decided to drop duplicate columns that were found on both data sets after merging, player names, highly correlated variables such as highest_market_value_in_eur which can be almost similar to our response variable.

Furthermore, dropping high cardinality categorical variables such as country_of_citizenship, city_of_birth, and last_name. This was a crucial step in simplifying the dataset, by minimizing redundancy, and reducing the risk for having more variables than observations, which could negatively impact the robustness and interpretability of our model.

Although the players dataset contains 32,405 observations and 23 columns, this would further be reduced to 2,135 observations and 240 columns. This is because the players data sets contains player information of several leagues from various nations. However, in this study we decided to focus players on the top five leagues in the world which are Premier League, Ligue 1, Bundesliga, Serie A and La Liga.

The original players dataset consisted of 32,405 observations and 23 columns, however, after filtering and feature engineering, the dataset was reduced to 2,135 observations and 240 columns. This reduction was all done in due to our decision to focus exclusively on players from the top five leagues in the world: the Premier League, Ligue 1, Bundesliga, Serie A, and La Liga. By narrowing our scope to these specific leagues, we ensured that the study concentrated on the most competitive and influential football leagues, enhancing the relevance and consistency of the analysis, and reducing the computational expense in processing a complex dataset.

```r
# Define the columns to drop that are not important for our analysis
columns_to_drop <- c("name", "first_name", "last_name", "last_season", "current_club_id",
                     "player_code", "city_of_birth", "image_url", "url",
                     "current_club_name", "country_of_birth", "date_of_birth",
                     "current_club_domestic_competition_id","player_id",
                     "sub_position", "highest_market_value_in_eur",'Pos',"Nation"
                     ,"country_of_citizenship")

# Ensure that only the columns that exist in merged_data are selected
columns_to_drop <- columns_to_drop[columns_to_drop %in% names(Football_DATA)]


# Drop the columns from the dataset set
Football_DATA <- Football_DATA[, !names(Football_DATA) %in% columns_to_drop]
```

## Feature Engineering

As part of our analysis, feature engineering plays a critical role in preparing the dataset for analysis and modeling. This streamlines the process in deriving meaningful insights and optimizing the dataset for better performance. This process involved filling in missing values, creating new variables, and refining existing columns to enhance the relevance and utility of the data. For instance, creating dummy variables which can be easy handled by our models in the next stage, such as neural networks, which is often difficult to train with a categorical variables.

1. "foot" variable: Handled NA values by defining them as "Unspecified", to ensure no data was left undefined.

2. "height_in_cm" variable: The median value was used to fill in the missing values.
3. "agent_name" variable: Initially, we considered dropping this column due to its high cardinality, with 2,834 unique agent names. Instead, we opted to transform this, by labeling players without an agent with a placeholder "No Agent". Next, a binary variable, "has_agent" was create to indicate whether a player had an agent, this preserved the valuable information without over complicating the model, as we could say that a player with an agent may be able to have an impact on their market value.

The next part consisted of creating a new variable called League_Tier, this categorized 98 of the teams into the league finish, during 2022-2023 football season. This variable aims in capturing the competitive level and performance of the teams in their respective leagues.

For instance, during a football season, in the la liga league, "Barcelona", "Real Madrid", "Atlético Madrid", and "Real Sociedad" qualified for the highest league competition. Similarly, we extended this categorization to account for teams that finished other tiers, such as: Europa League, Conference League, Not Qualified, and Relegation. In terms of Not Qualified and Relegation, these are often teams that did not secure qualification any of the European competition or notable stage, while Relegation, is the result of teams being demoted to a lower division.

This multi-tiered categorization provides an added layer of information, particularly for evaluating the impact of team success or failure on player market valuation, as often, teams who had qualified or relegated to a lower competitive division, may impact a player value on the market.

The next step involved applying a log transformation to the response variable "market_value_in_eur". This transformation was aimed to to reduce the skewness of the variable, ensuring a more normal-like distribution and improving the performance of the models. By transforming this variable, we can better capture the relative differences in player market values across the dataset.

Before creating dummy variables, we want to consider a player's contract expiration data, in this case we decided to extract just the year in which the contract is expected to expire and how this may impact their market value. The final stage of the feature engineering involved generating dummy variables for every categorical variable retained or created. This approach is beneficial in modeling, especially in machine learning algorithms such as Neural Networks.

```
# fill NA's in "foot" column
Football_DATA <- Football_DATA %>%
  mutate(foot = ifelse(is.na(foot), "Unspecified", foot))

# fill NA's in "height_in_cm" column
Football_DATA$height_in_cm[is.na(Football_DATA$height_in_cm)] <- median(Football_DATA$height_in_cm, na.

# print the length of unique agent_name
length(unique(players$agent_name))
```

```
## [1] 2834
```

```
length(unique(football_stats$Squad))
```

```
## [1] 98
```

```
# Replace NA values with a placeholder "No Agent"
Football_DATA$agent_name[is.na(Football_DATA$agent_name)] <- "No Agent"

# Create binary column to indicate presence of an agent
Football_DATA$has_agent <- ifelse(Football_DATA$agent_name == "No Agent" | Football_DATA$agent_name ==

# Drop the original agent_name column
Football_DATA$agent_name <- NULL

# Defining clubs that were in the Champions league qualification stage
champions_league <- c("Barcelona", "Real Madrid", "Atlético Madrid", "Real Sociedad",
                      "Manchester City", "Arsenal", "Manchester United",
                      "Newcastle Utd", "Paris S-G", "Lens", "Bayern Munich",
                      "Dortmund", "RB Leipzig", "Union Berlin","Napoli", "Lazio",
                      "Inter", "Milan")

# Defining clubs that were in the Europa league qualification stage
europa_league <- c("Villarreal","Real Betis", "Liverpool", "Brighton", "Marseille",
                   "Freiburg", "Leverkusen","Atalanta", "Roma")

# Defining clubs that were in the Conference league qualification stage
conference_league <- c("Osasuna", "Aston Villa", "Rennes",
                       "Eint Frankfurt", "Juventus")

# Defining clubs that were in the Relegation stage in their league
relegation <- c("Espanyol", "Valladolid", "Elche", "Leicester City",
                "Leeds United", "Southampton", "Auxerre", "Ajaccio",
                "Troyes", "Angers", "Stuttgart", "Schalke 04",
                "Hertha","Spezia", "Cremonese", "Sampdoria")
```

```r
# Assign the League_Tier using str_detect
Football_DATA$League_Tier <- case_when(
  str_detect(Football_DATA$Squad, paste(champions_league, collapse = "|")) ~ "Champions League",
  str_detect(Football_DATA$Squad, paste(europa_league, collapse = "|")) ~ "Europa League",
  str_detect(Football_DATA$Squad, paste(conference_league, collapse = "|")) ~ "Conference League",
  str_detect(Football_DATA$Squad, paste(relegation, collapse = "|")) ~ "Relegation",
  TRUE ~ "Not Qualified")

# Transform 'market_value_in_eur' to log scale
Football_DATA$log_market_value_in_eur <- log(Football_DATA$market_value_in_eur)

# Drop the original 'market_value_in_eur' column
Football_DATA <- Football_DATA[, !names(Football_DATA) %in% "market_value_in_eur"]

# Defining the contract_expiration_date as a Data column
Football_DATA$contract_expiration_date <- as.Date(Football_DATA$contract_expiration_date)

# Extract the year from the contract_expiration_date, and creating a contract_expiration_year column
Football_DATA <- Football_DATA %>%
  mutate(
    contract_expiration_year = as.numeric(format(contract_expiration_date, "%Y")))

# Calculate the median year, excluding NAs
median_year <- median(Football_DATA$contract_expiration_year, na.rm = TRUE)

# Fill NA values with the median year
Football_DATA$contract_expiration_year[is.na(Football_DATA$contract_expiration_year)] <- median_year

# Dropping the contract_expiration_date column
Football_DATA <- Football_DATA[, !names(Football_DATA) %in% "contract_expiration_date"]

# Removing NA values
Football_DATA_C <- na.omit(Football_DATA)

# Create a dummy variable transformation for categorical variables
dummy_transform <- dummyVars("~ .", data = Football_DATA_C)

# Applying the transformation to generate dummy variables
Football_DATA_C <- predict(dummy_transform, newdata = Football_DATA_C)

# Converting back to a data frame
Football_DATA_C <- as.data.frame(Football_DATA_C)

# removing an unnecessary spaces in within the dummy variables column names
colnames(Football_DATA_C) <- gsub("'", "", colnames(Football_DATA_C))
colnames(Football_DATA_C) <- gsub(" ", "_", colnames(Football_DATA_C))
```

## Feature Selection

In the context of feature selection, its is essential to preprocess the data before the feature selection stage. This is done by scaling the numerical variables and ensuring that the response variable is handled properly. By scaling, this ensures that all the numerical features are on the same scale, preventing certain variables with a larger ranges from dominating the model. This is especially essential with k-nearest neighbors (KNN) and Neural Networks (NN).

In terms of NN, it allows the model to speed by the convergence by ensuring equal contribution from all the variables during the gradient descent, and improves the training efficient and weights optimization process. While in most models, this can also be beneficial in improving the models performance in predictions and accuracy.

The next section involves splitting the dataset into 80% training set and 20% testing set. Typically, the split of a 80/20 is often common, in which the training data is used to train the model, and the remaining testing set is then used to evaluate the model's performance on unseen data, providing an insight into its generalization ability. In order to asses this performance, we would compute the Mean Squared Error (MSE) and compare it across the different models trained.

This is generally done to help determine if the a model is overfitting with very low training and high testing MSE or underfitting with high MSE on both training and testing sets. This evaluation stage helps us determine if the model is too complex, or too simple, and whether adjustments and further tuning of the model is need to better capture the relationship between the predictors and the response variable.

```
# Select only numerical columns for scaling
num_columns <- sapply(Football_DATA_C, is.numeric)

# Excluding the log-transformed response variable ('log_market_value_in_eur') from scaling
num_columns_without_target <- num_columns
num_columns_without_target[which(names(Football_DATA_C) == "log_market_value_in_eur")] <- FALSE

# Applying Min-Max scaling to the numerical columns (excluding 'log_market_value_in_eur')
maxs <- apply(Football_DATA_C[, num_columns_without_target], 2, max, na.rm = TRUE)
mins <- apply(Football_DATA_C[, num_columns_without_target], 2, min, na.rm = TRUE)

# Scaling the numerical columns (excluding the target variable)
scaled_numeric <- as.data.frame(scale(Football_DATA_C[, num_columns_without_target], center = mins, scal

# Adding the non-numerical columns back without scaling
scaled_data <- cbind(scaled_numeric, Football_DATA_C[, !num_columns])

# Ensuring 'log_market_value_in_eur' is kept in the data
scaled_data$log_market_value_in_eur <- Football_DATA_C$log_market_value_in_eur

# Move the response variable (column 'log_market_value_in_eur') to the last position
scaled_data <- scaled_data[, c(setdiff(1:ncol(scaled_data), which(names(scaled_data) == "log_market_valu

# Confirming the new structure
colnames(scaled_data)[ncol(scaled_data)]
```

```
## [1] "log_market_value_in_eur"
```

```
# Split the data into training and testing sets
set.seed(209) # Ensure reproducibility

# indexing and randomly sampling, splitting 80% training and 20% testing set
train_index <- sample(1:nrow(scaled_data), 0.8 * nrow(scaled_data))
train_data <- scaled_data[train_index, ]
test_data <- scaled_data[-train_index, ]
```

### Step-wise feature selection

We employed Step-wise Regression to try and identify which predictors would be the most useful in both explaining and predicting our log transformed response variable: Market Value in Euros. This initially entails fitting a model with no predictors present. We also defined a "full model" where all predictors in the data set are present. The step wise selection method employed here will perform both forwards and backwards selection. The output of the Step-wise model is included below, and the specific predictors selected by the step wise regression process are outlined.

Subsequently, of the predictors that should be included based on the aforementioned selection methods (step-wise formula), we took it a step further and included some select interaction effects in our "LM Model". The R^2 of the LM Model with select interactions included rose to 0.6604 from the observed R^2 value of 0.6554 in our Step-wise model. Further analysis was conducted in which outline values were removed from the training data set (i.e. observations with standardized residuals greater than 3). This amended training set was used to run our second version of the LM Model and resulted in an R^2 value of 0.7179. We elected to remove the outliers in this specific instance since the QQ plot produced by the original LM Model run on the original test data exhibited non-normality. However, we did not elect to use the outlier free training data for the remainder of this analysis. Furthermore, utilizing the variance inflation factor (VIF) function, we noted some of our selected predictors exhibited high multicollinearity: contract_expiration_year, League_TierChampions_League, CompPremiere_League, Age.

With the exception of the variables selected and model produced by Multivariate Adaptive Regression Splines (MARS) later on in this analysis, all other models discussed with be fitted with the variables found in the original step wise function below. However, the following section dedicated to the linear model will utilize LM Model 1 which includes interaction effects.

```r
# Stepwise model using the scaled data
initial_model <- lm(log_market_value_in_eur ~ 1, data = train_data)
full_model <- lm(log_market_value_in_eur ~ ., data = train_data)

# Perform step wise selection (both directions)
stepwise_model <- stepAIC(initial_model, scope = list(lower = initial_model, upper = full_model), direc
stepwise_formula <- formula(stepwise_model)

# Summary and diagnostics of the step wise model
summary(stepwise_model)
```

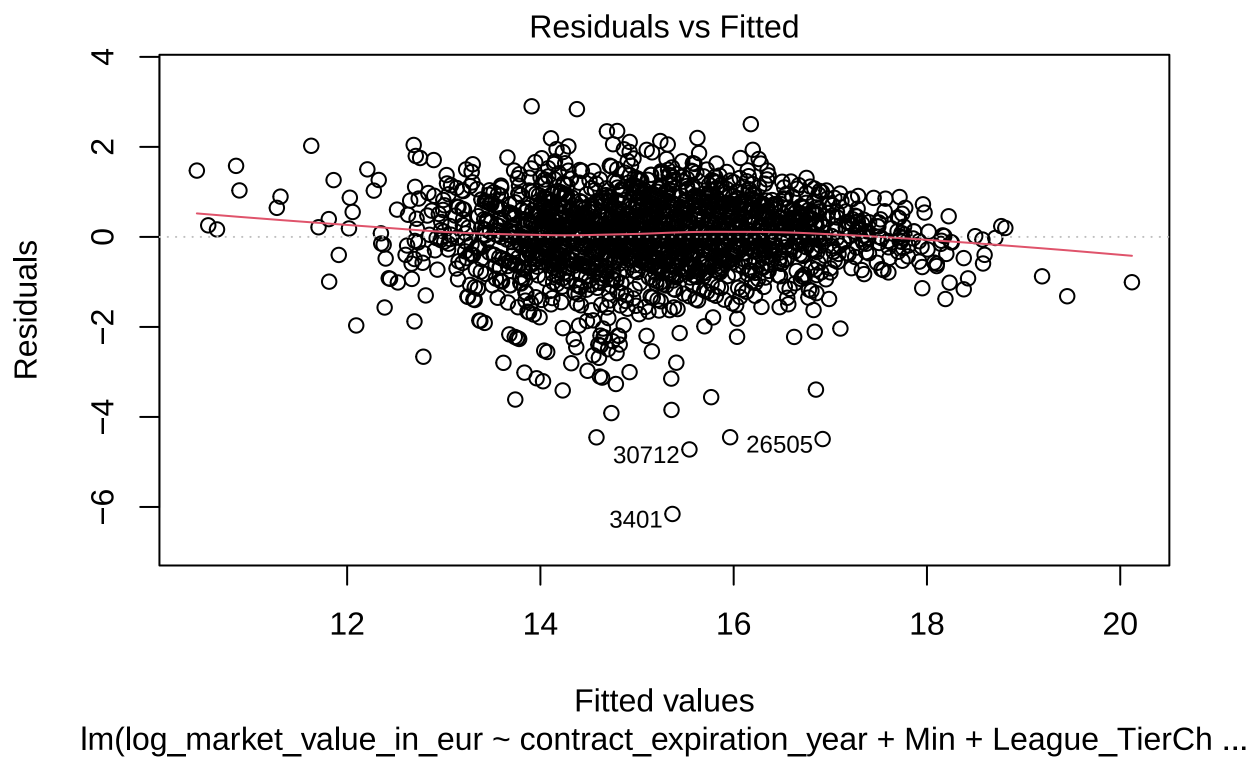
```
##
## Call:
## lm(formula = log_market_value_in_eur ~ contract_expiration_year +
##     MP + League_TierChampions_League + CompPremier_League + Age +
##     footUnspecified + Goals + League_TierRelegation + League_TierNot_Qualified +
##     PasMedCmp. + has_agent + PasAss + Starts + SquadOsasuna +
##     SquadLens + SquadCádiz + SquadNice + positionMissing + SquadManchester_Utd +
##     TB + TklDriPast + positionGoalkeeper + SquadBayern_Munich +
##     SquadClermont_Foot + SquadBochum + Fld + height_in_cm + PasShoCmp. +
##     X2CrdY + SquadLille + ScaFld + SquadNewcastle_Utd + SquadUnion_Berlin +
##     SquadAthletic_Club + PasProg + Pas3rd + CkOut + Fls + positionAttack +
##     SquadLazio + SquadLyon + PasMedAtt + Sw + SquadBournemouth +
##     TouDefPen + PasFK + SquadWolfsburg + CrdY + SquadVillarreal +
##     SquadAlmería + PasOff + SquadReal_Sociedad + ScaPassDead +
##     SquadMainz_05 + SquadArsenal + Rk + PasLonCmp. + PasLonCmp +
##     SquadWest_Ham + SquadTroyes + SquadAjaccio + Rec + SquadToulouse +
##     SquadAngers + SquadAuxerre + PasTotDist + SquadMilan + SquadBarcelona +
##     SquadTorino + SquadMonaco + GcaSh + ScaSh + SoT. + SoT +
##     TklMid3rd + G.SoT + SquadMarseille, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6437 -0.4543  0.0715  0.5745  2.7001
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 13.04387    0.21260  61.355  < 2e-16 ***
## contract_expiration_year     3.95538    0.17195  23.003  < 2e-16 ***
## MP                           0.16859    0.16494   1.022 0.306852
## League_TierChampions_League  0.49377    0.10759   4.589 4.74e-06 ***
## CompPremier_League           0.72769    0.06222  11.696  < 2e-16 ***
## Age                         -2.19994    0.14689 -14.977  < 2e-16 ***
## footUnspecified             -1.75662    0.20705  -8.484  < 2e-16 ***
## Goals                        1.57262    0.36452   4.314 1.69e-05 ***
## League_TierRelegation       -0.76562    0.09037  -8.472  < 2e-16 ***
## League_TierNot_Qualified    -0.56615    0.07639  -7.411 1.89e-13 ***
## PasMedCmp.                   0.57625    0.16321   3.531 0.000425 ***
## has_agent                    0.26895    0.04914   5.473 5.03e-08 ***
## PasAss                       2.01807    0.57203   3.528 0.000429 ***
## Starts                       1.01775    0.17394   5.851 5.75e-09 ***
## SquadOsasuna                -1.03293    0.24690  -4.184 3.00e-05 ***
## SquadLens                   -0.99450    0.23667  -4.202 2.77e-05 ***
## SquadCádiz                  -0.72259    0.21327  -3.388 0.000718 ***
## SquadNice                    0.65564    0.19070   3.438 0.000599 ***
## positionMissing             -3.22314    0.93146  -3.460 0.000552 ***
## SquadManchester_Utd          0.58311    0.19895   2.931 0.003421 **
## TB                           1.69673    0.65082   2.607 0.009205 **
## TklDriPast                  -0.42280    0.25554  -1.655 0.098185 .
## positionGoalkeeper          -0.80841    0.17375  -4.653 3.51e-06 ***
## SquadBayern_Munich           0.54746    0.23032   2.377 0.017556 *
## SquadClermont_Foot          -0.70445    0.25206  -2.795 0.005247 **
## SquadBochum                 -0.53694    0.21825  -2.460 0.013975 *
## Fld                          0.75100    0.40180   1.869 0.061768 .
## height_in_cm                 0.44189    0.14388   3.071 0.002163 **
## PasShoCmp.                   0.36651    0.18101   2.025 0.043028 *
## X2CrdY                       2.20083    1.00011   2.201 0.027888 *
## SquadLille                   0.64903    0.21321   3.044 0.002367 **
## ScaFld                       0.71049    0.46619   1.524 0.127667
## SquadNewcastle_Utd          -0.69199    0.22112  -3.130 0.001778 **
## SquadUnion_Berlin           -0.54572    0.23670  -2.306 0.021243 *
## SquadAthletic_Club           0.53645    0.20638   2.599 0.009413 **
## PasProg                     -1.20918    0.47308  -2.556 0.010669 *
## Pas3rd                       1.25886    0.38941   3.233 0.001248 **
## CkOut                        1.39678    0.39451   3.541 0.000409 ***
## Fls                         -1.35753    0.32759  -4.144 3.57e-05 ***
## positionAttack               0.08228    0.06449   1.276 0.202160
## SquadLazio                  -0.53981    0.24727  -2.183 0.029154 *
## SquadLyon                    0.54201    0.20450   2.650 0.008108 **
## PasMedAtt                   -0.23124    0.99608  -0.232 0.816449
## Sw                           1.34275    0.43068   3.118 0.001850 **
## SquadBournemouth            -0.44142    0.20415  -2.162 0.030727 *
## TouDefPen                    1.32054    0.31898   4.140 3.63e-05 ***
## PasFK                       -0.19856    0.27035  -0.734 0.462771
## SquadWolfsburg               0.53956    0.24343   2.216 0.026780 *
## CrdY                         0.89260    0.42817   2.085 0.037233 *
## SquadVillarreal             -0.46369    0.22062  -2.102 0.035706 *
## SquadAlmería                -0.45383    0.23569  -1.926 0.054317 .
## PasOff                       1.07896    0.53953   2.000 0.045665 *
## SquadReal_Sociedad          -0.42374    0.23010  -1.842 0.065699 .
## ScaPassDead                 -0.88613    0.56419  -1.571 0.116439
## SquadMainz_05                0.43166    0.22341   1.932 0.053498 .
## SquadArsenal                -0.27971    0.24356  -1.148 0.250938
## Rk                           0.14440    0.07459   1.936 0.053021 .
## PasLonCmp.                   0.31776    0.12095   2.627 0.008682 **
## PasLonCmp                   -0.20183    0.76677  -0.263 0.792413
## SquadWest_Ham                0.47121    0.23994   1.964 0.049698 *
## SquadTroyes                 -0.44359    0.21814  -2.033 0.042147 *
## SquadAjaccio                -0.40306    0.20275  -1.988 0.046960 *
## Rec                          2.41587    0.93684   2.579 0.009993 **
## SquadToulouse               -0.33063    0.21913  -1.509 0.131511
## SquadAngers                 -0.40063    0.23047  -1.738 0.082327 .
## SquadAuxerre                -0.37634    0.21445  -1.755 0.079433 .
## PasTotDist                  -2.90374    1.76089  -1.649 0.099313 .
## SquadMilan                   0.39185    0.24870   1.576 0.115290
## SquadBarcelona               0.31096    0.23009   1.351 0.176711
## SquadTorino                  0.36702    0.22978   1.597 0.110373
## SquadMonaco                  0.28117    0.21320   1.319 0.187401
## GcaSh                       -1.29394    0.66232  -1.954 0.050893 .
## ScaSh                        1.31387    0.60697   2.165 0.030542 *
## SoT.                         0.28254    0.11696   2.416 0.015803 *
## SoT                         -0.64075    0.36409  -1.760 0.078599 .
## TklMid3rd                    0.61716    0.40484   1.524 0.127562
## G.SoT                       -0.13524    0.09298  -1.454 0.145977
## SquadMarseille               0.29970    0.21465   1.396 0.162810
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9252 on 1856 degrees of freedom
## Multiple R-squared:  0.6557, Adjusted R-squared:  0.6414
## F-statistic: 45.91 on 77 and 1856 DF,  p-value: < 2.2e-16
```

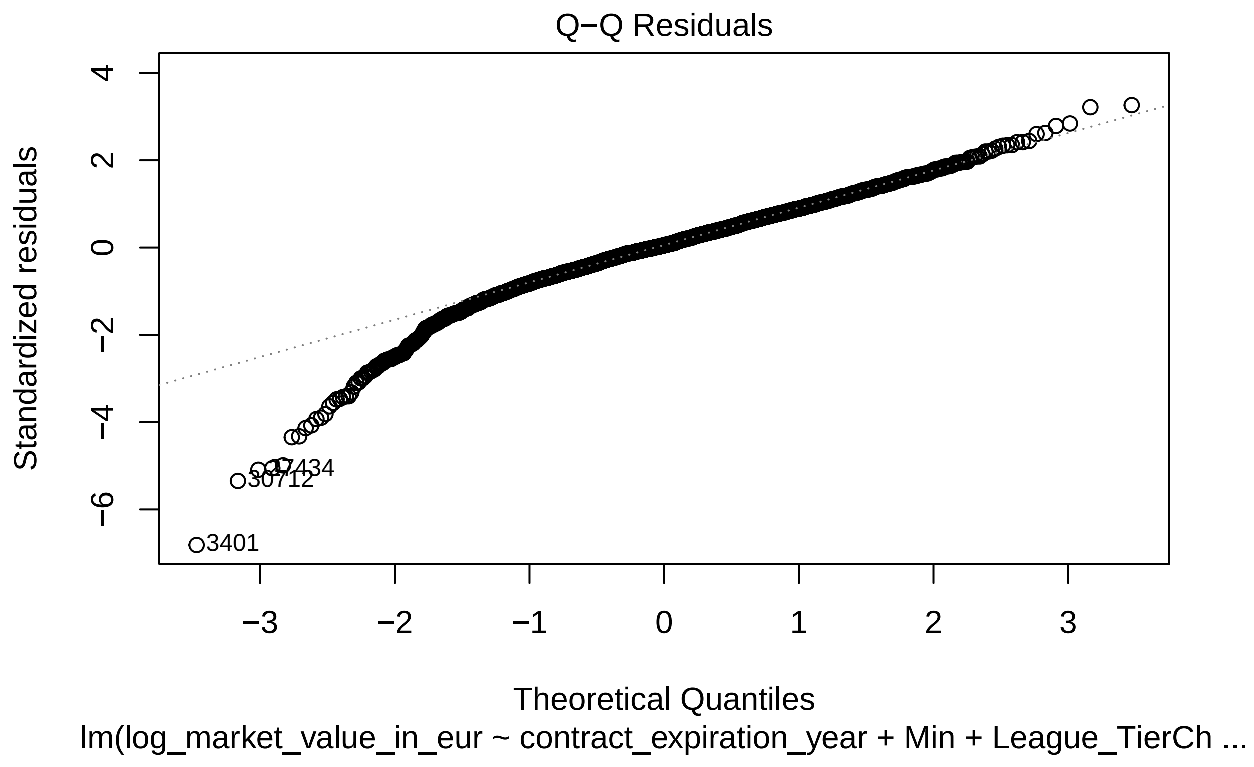
```
# stepwise selection with interaction effects
init_mod <- lm(stepwise_formula, data = train_data)

# include interaction effects (hidden due to high computational time)
# step(init_mod, scope = .~.^2, direction = "forward")

LM_Model <-  lm(log_market_value_in_eur ~ contract_expiration_year + Min + League_TierChampions_League
                  CompPremier_League + Age + footUnspecified + Goals +
                  League_TierRelegation + League_TierNot_Qualified + PasLonCmp. +
                  SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
                  SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                  SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                  SquadSevilla + SquadBochum + SquadBournemouth +
                  TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                  SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
```

```r
                   SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                   PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                   ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                   + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                   TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                   PasShoCmp.  + CompSerie_A + PKcon + MP + Age:MP +
                   contract_expiration_year:has_agent + CompPremier_League:Age +
                   height_in_cm:CompSerie_A + Age:PasMedCmp +
                   Age:SquadManchester_Utd + contract_expiration_year:Age +
                   contract_expiration_year:MP + League_TierChampions_League:MP,
                   data = train_data)

# summary output
summary(LM_Model)
```

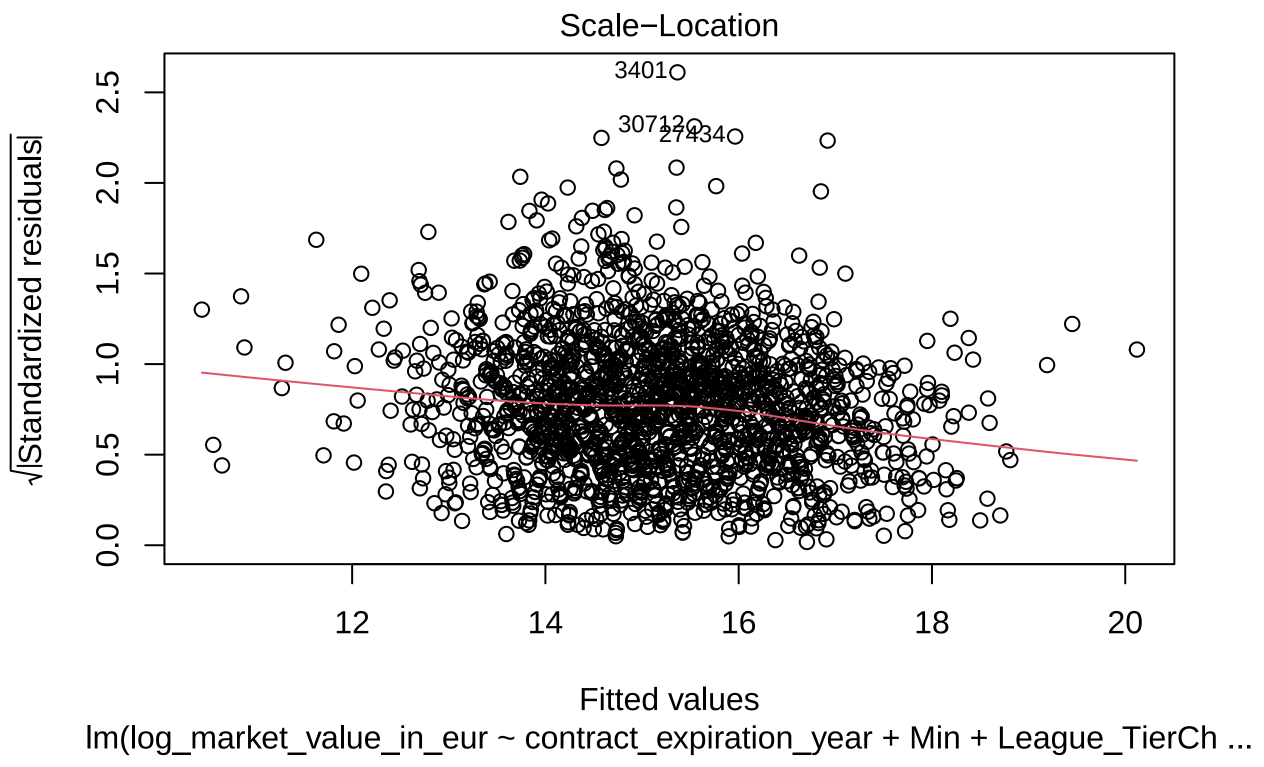
```
##
## Call:
## lm(formula = log_market_value_in_eur ~ contract_expiration_year +
##     Min + League_TierChampions_League + CompPremier_League +
##     Age + footUnspecified + Goals + League_TierRelegation + League_TierNot_Qualified +
##     PasLonCmp. + SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
##     SquadOsasuna + X2CrdY + SoT. + SquadVillarreal + SquadNewcastle_Utd +
##     SquadLens + SquadNice + SquadMonaco + SquadSevilla + SquadBochum +
##     SquadBournemouth + TI + SquadUnion_Berlin + SquadCádiz +
##     SquadLazio + SquadManchester_Utd + ToSuc + height_in_cm +
##     SquadValencia + SquadLille + SquadLyon + has_agent + SquadStuttgart +
##     TB + PasProg + Pas3rd + PasMedCmp. + CkOut + SquadBayern_Munich +
##     ShoPK + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería +
##     SquadSalernitana + SquadNottham_Forest + TklMid3rd + TklDriAtt +
##     TklAtt3rd + SquadArsenal + PasTotCmp. + PasShoCmp. + CompSerie_A +
##     PKcon + MP + Age:MP + contract_expiration_year:has_agent +
##     CompPremier_League:Age + height_in_cm:CompSerie_A + Age:PasMedCmp +
##     Age:SquadManchester_Utd + contract_expiration_year:Age +
##     contract_expiration_year:MP + League_TierChampions_League:MP,
##     data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1559 -0.4657  0.0472  0.5652  2.9019
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        12.81919    0.25744  49.795  < 2e-16 ***
## contract_expiration_year            3.81722    0.49162   7.764 1.34e-14 ***
## Min                                 1.43173    0.20977   6.825 1.18e-11 ***
## League_TierChampions_League         0.58204    0.14420   4.036 5.65e-05 ***
## CompPremier_League                  1.25064    0.14806   8.447  < 2e-16 ***
## Age                                -1.32206    0.35653  -3.708 0.000215 ***
## footUnspecified                    -1.67877    0.20442  -8.212 4.01e-16 ***
## Goals                               1.33575    0.34929   3.824 0.000136 ***
## League_TierRelegation              -0.97530    0.08263 -11.803  < 2e-16 ***
## League_TierNot_Qualified           -0.57886    0.07180  -8.062 1.33e-15 ***
## PasLonCmp.                          0.25675    0.10909   2.354 0.018699 *
## SCA                                0.79824    0.30021   2.659 0.007907 **
## SquadReal_Sociedad                -0.47653    0.22477  -2.120 0.034133 *
## Fls                               -0.75514    0.30014  -2.516 0.011955 *
## SquadClermont_Foot                -0.67493    0.24969  -2.703 0.006933 **
## SquadOsasuna                      -1.11914    0.23985  -4.666 3.29e-06 ***
## X2CrdY                             2.58483    0.88990   2.905 0.003720 **
## SoT.                               0.13935    0.09682   1.439 0.150238
## SquadVillarreal                   -0.28967    0.21818  -1.328 0.184457
## SquadNewcastle_Utd                -0.75911    0.21409  -3.546 0.000401 ***
## SquadLens                         -1.17422    0.22936  -5.120 3.38e-07 ***
## SquadNice                          0.76902    0.18836   4.083 4.64e-05 ***
## SquadMonaco                        0.24252    0.21062   1.151 0.249688
## SquadSevilla                       0.25311    0.17946   1.410 0.158594
## SquadBochum                       -0.68127    0.21573  -3.158 0.001614 **
## SquadBournemouth                  -0.58317    0.20218  -2.884 0.003966 **
## TI                                -0.27452    0.13759  -1.995 0.046165 *
## SquadUnion_Berlin                 -0.78245    0.22978  -3.405 0.000675 ***
## SquadCádiz                        -0.87693    0.20967  -4.182 3.02e-05 ***
## SquadLazio                        -0.75783    0.24908  -3.043 0.002379 **
## SquadManchester_Utd               -0.82087    0.55354  -1.483 0.138258
## ToSuc                              0.35346    0.41795   0.846 0.397822
## height_in_cm                       0.24785    0.15386   1.611 0.107370
## SquadValencia                      0.02482    0.24895   0.100 0.920601
## SquadLille                         0.71411    0.21049   3.393 0.000707 ***
## SquadLyon                          0.51140    0.20192   2.533 0.011402 *
## has_agent                          0.69478    0.11188   6.210 6.51e-10 ***
## SquadStuttgart                     0.43462    0.20482   2.122 0.033972 *
## TB                                 1.35117    0.63029   2.144 0.032184 *
## PasProg                           -1.34501    0.44286  -3.037 0.002422 **
## Pas3rd                             1.20623    0.36026   3.348 0.000830 ***
## PasMedCmp.                         0.47401    0.16224   2.922 0.003524 **
## CkOut                              1.17964    0.34128   3.457 0.000559 ***
## SquadBayern_Munich                 0.50288    0.22331   2.252 0.024441 *
## ShoPK                              0.20378    0.30541   0.667 0.504704
## positionGoalkeeper                -0.94411    0.17580  -5.370 8.84e-08 ***
## TouDefPen                          0.59659    0.28577   2.088 0.036964 *
## BlkSh                             -0.11121    0.47061  -0.236 0.813221
## SquadAlmería                      -0.37590    0.23284  -1.614 0.106610
## SquadSalernitana                  -0.28803    0.21205  -1.358 0.174528
## SquadNottham_Forest               -0.23885    0.17958  -1.330 0.183653
## TklMid3rd                          0.45503    0.40612   1.120 0.262672
## TklDriAtt                         -0.68820    0.48922  -1.407 0.159673
## TklAtt3rd                          0.88082    0.44767   1.968 0.049265 *
## SquadArsenal                      -0.49494    0.23973  -2.065 0.039100 *
## PasTotCmp.                        -0.23291    0.27676  -0.842 0.400145
## PasShoCmp.                         0.42348    0.20374   2.079 0.037799 *
## CompSerie_A                       -0.13949    0.18140  -0.769 0.441999
## PKcon                              0.70047    0.49916   1.403 0.160693
## MP                                 2.07526    0.32495   6.386 2.14e-10 ***
## Age:MP                            -3.85847    0.43466  -8.877  < 2e-16 ***
## contract_expiration_year:has_agent -1.50751    0.34289  -4.396 1.16e-05 ***
## CompPremier_League:Age            -1.04418    0.31596  -3.305 0.000968 ***
## height_in_cm:CompSerie_A           0.37746    0.32422   1.164 0.244483
## Age:PasMedCmp                     -3.65979    0.80079  -4.570 5.19e-06 ***
## Age:SquadManchester_Utd             3.17073    1.17998   2.687 0.007272 **
## contract_expiration_year:Age        5.89587    1.05297   5.599 2.47e-08 ***
## contract_expiration_year:MP        -2.05615    0.52736  -3.899 0.000100 ***
## League_TierChampions_League:MP      0.11153    0.19630   0.568 0.569978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9115 on 1865 degrees of freedom
## Multiple R-squared:  0.6642, Adjusted R-squared:  0.652
## F-statistic: 54.26 on 68 and 1865 DF,  p-value: < 2.2e-16
```

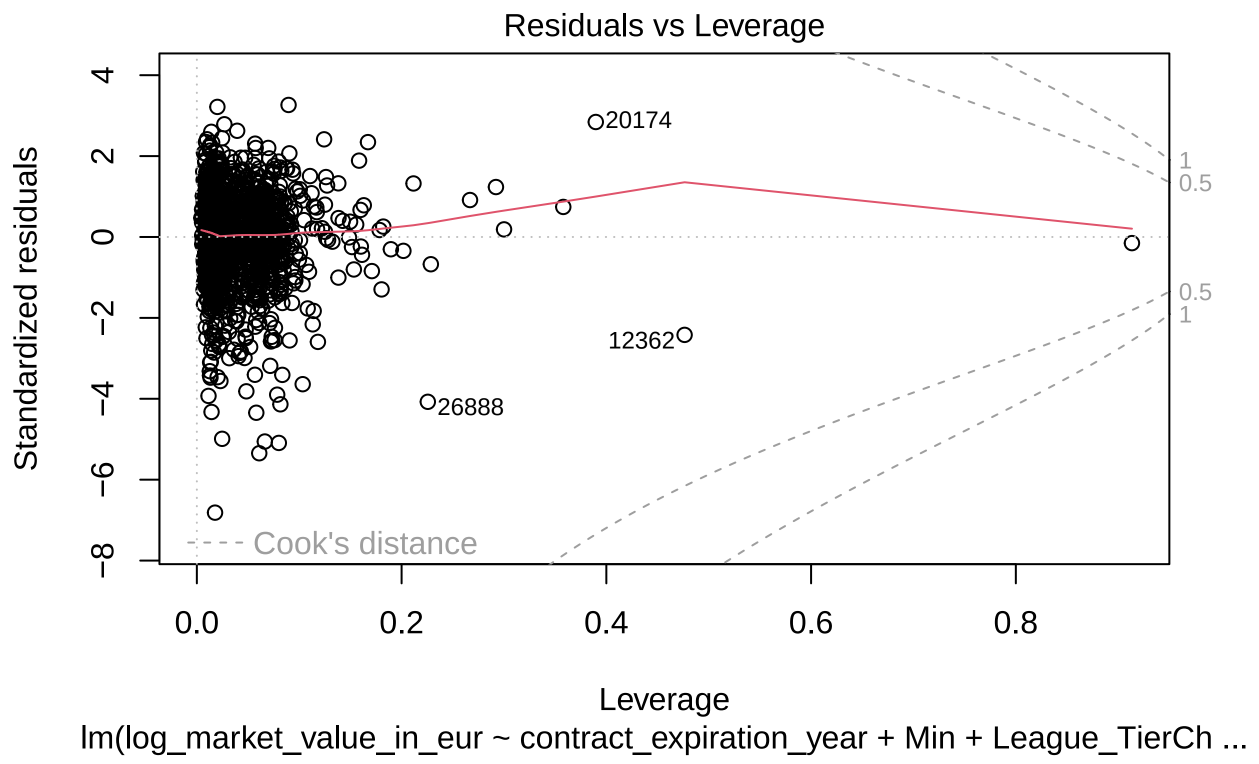
```
# diagnostic plots
plot(LM_Model)
```





Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(log_market_value_in_eur ~ contract_expiration_year + Min + League_TierCh ...
4
2
0
−2
−4
−6
−3
−2
−1
0
1
2
3
3401
30712
27434


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
12
14
16
18
20
3401
30712
27434
Fitted values
lm(log_market_value_in_eur ~ contract_expiration_year + Min + League_TierCh ...

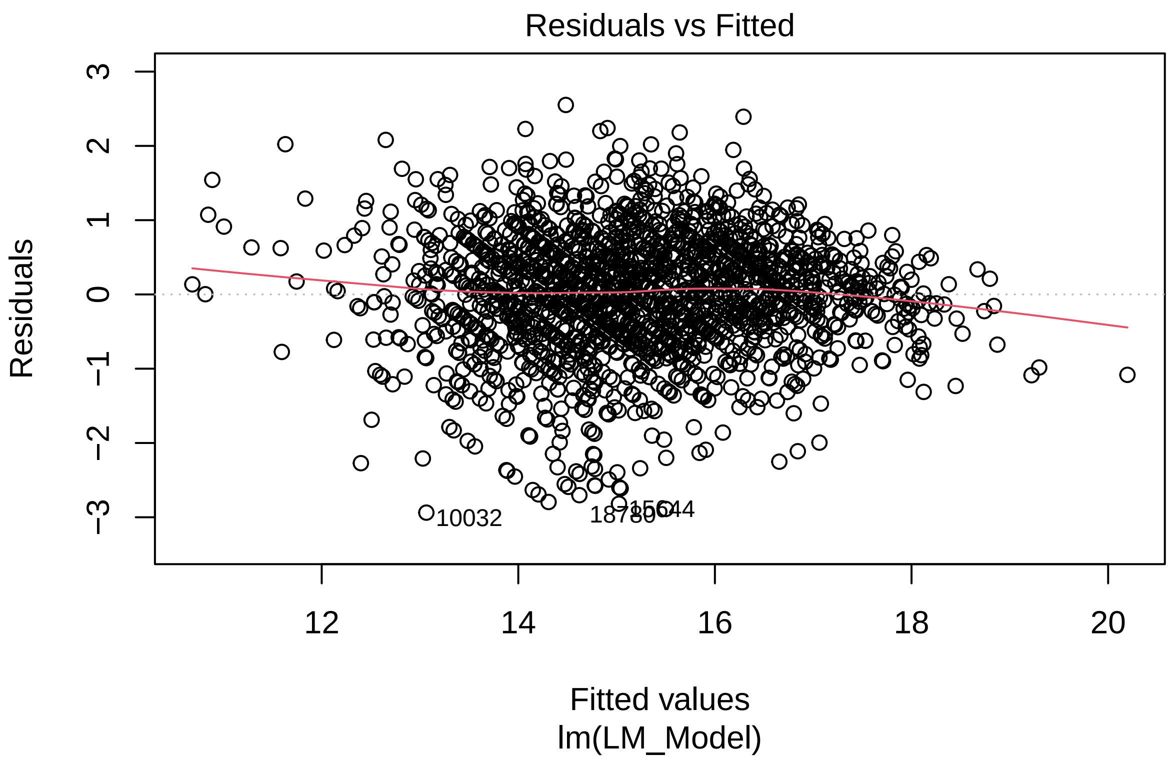


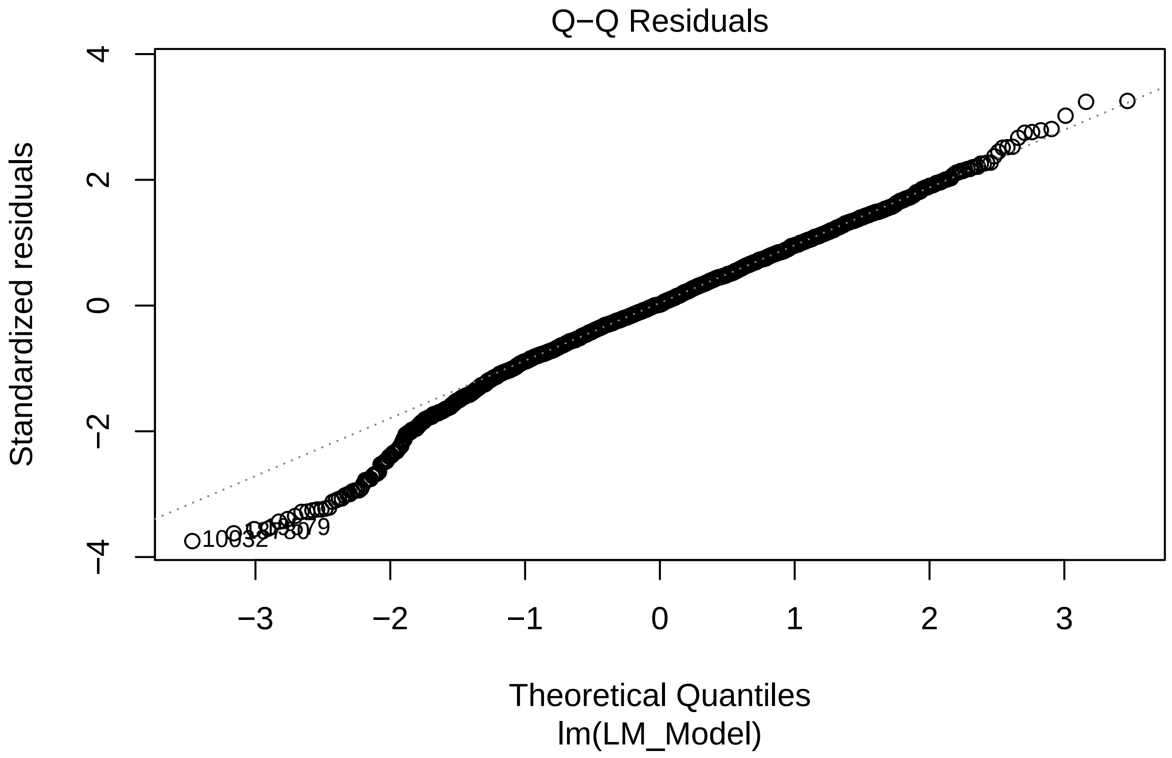
```
# extracting the formual from the first model
LM_MODEL1 <- formula(LM_Model)

# Compute the standardized residuals
residuals_standardized <- rstandard(LM_Model)

# Set a threshold for identifying outliers (e.g., standardized residuals > 3 or < -3)
outlier_threshold <- 3
outliers <- which(abs(residuals_standardized) > outlier_threshold)

# creating a subset of training data removing the outlier observations
train_data_clean <- train_data[-outliers, ]

# refitting the model with a cleaned data
LM_MODEL2 <- lm(LM_Model, data = train_data_clean)

# summary output
summary(LM_MODEL2)
```

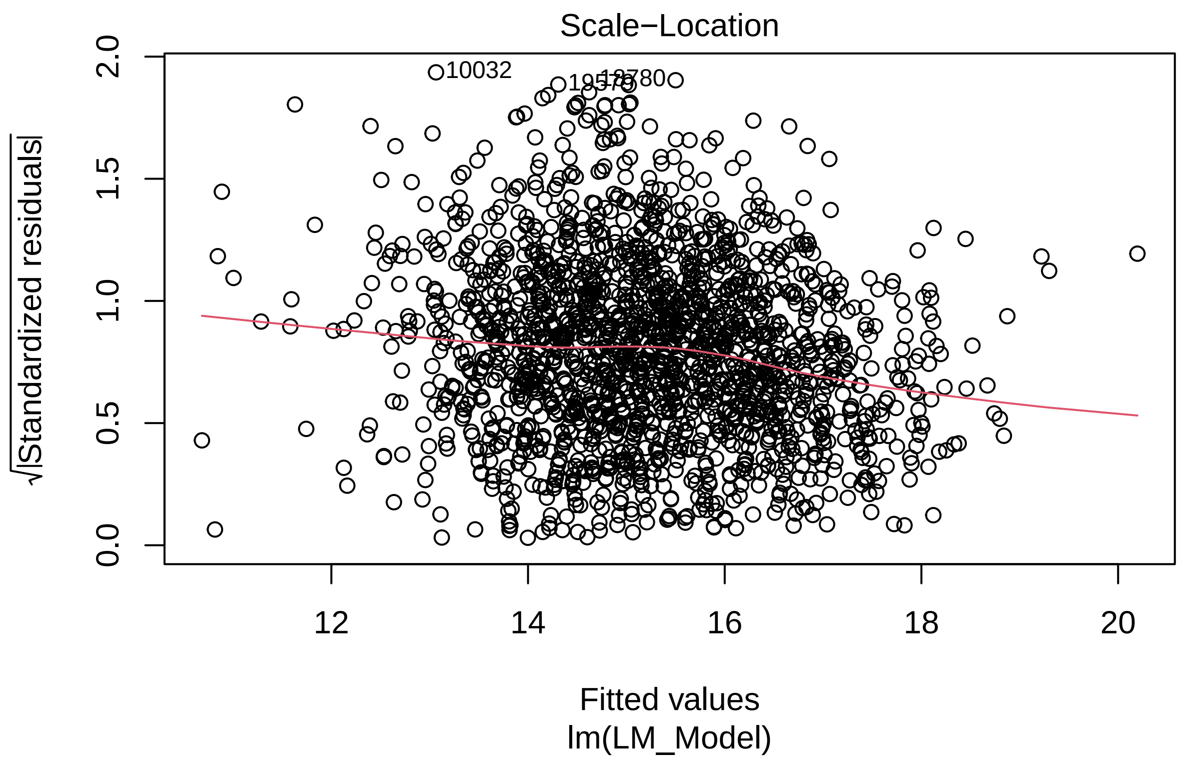
```
##
## Call:
## lm(formula = LM_Model, data = train_data_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.93790 -0.45872  0.01406  0.52553  2.55108
```

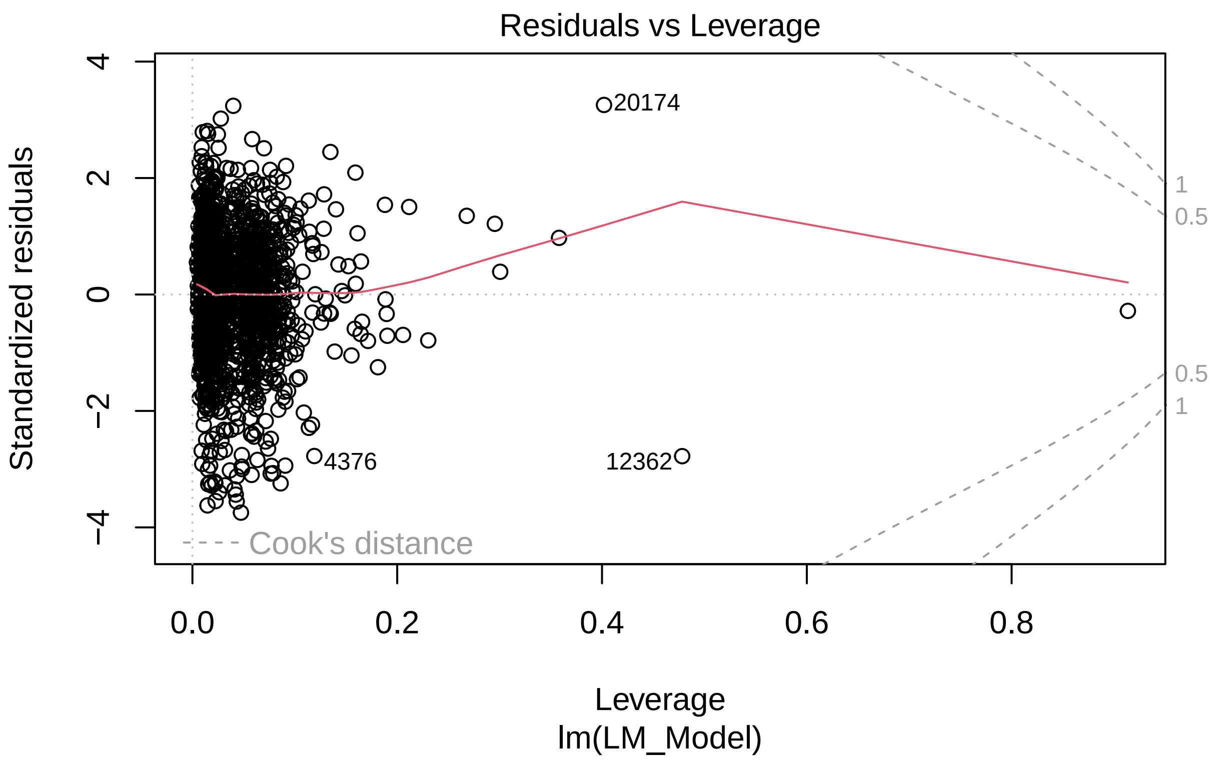
```
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      12.95309    0.22847  56.696  < 2e-16 ***
## contract_expiration_year          3.21594    0.43649   7.368 2.61e-13 ***
## Min                               1.58002    0.18565   8.511  < 2e-16 ***
## League_TierChampions_League       0.55274    0.12750   4.335 1.54e-05 ***
## CompPremier_League                1.27116    0.13207   9.625  < 2e-16 ***
## Age                              -1.23544    0.31546  -3.916 9.32e-05 ***
## footUnspecified                  -1.50130    0.18480  -8.124 8.18e-16 ***
## Goals                             1.10340    0.30934   3.567 0.000370 ***
## League_TierRelegation            -0.99776    0.07350 -13.574  < 2e-16 ***
## League_TierNot_Qualified         -0.61759    0.06381  -9.678  < 2e-16 ***
## PasLonCmp.                        0.23375    0.09666   2.418 0.015694 *
## SCA                               0.79483    0.26561   2.992 0.002805 **
## SquadReal_Sociedad               -0.45314    0.19822  -2.286 0.022367 *
## Fls                              -0.61870    0.26522  -2.333 0.019767 *
## SquadClermont_Foot               -0.71528    0.22014  -3.249 0.001178 **
## SquadOsasuna                     -0.93668    0.21759  -4.305 1.76e-05 ***
## X2CrdY                            2.64264    0.78457   3.368 0.000772 ***
## SoT.                              0.10747    0.08601   1.250 0.211596
## SquadVillarreal                  -0.27726    0.19247  -1.440 0.149899
## SquadNewcastle_Utd               -0.73956    0.19258  -3.840 0.000127 ***
## SquadLens                        -1.16222    0.20228  -5.746 1.07e-08 ***
## SquadNice                         0.76726    0.16613   4.619 4.13e-06 ***
## SquadMonaco                       0.76025    0.19502   3.898 0.000100 ***
## SquadSevilla                      0.38186    0.16074   2.376 0.017618 *
## SquadBochum                      -0.62782    0.19019  -3.301 0.000982 ***
## SquadBournemouth                 -0.53077    0.18178  -2.920 0.003545 **
## TI                               -0.40718    0.12229  -3.330 0.000886 ***
## SquadUnion_Berlin                -0.80298    0.20266  -3.962 7.71e-05 ***
## SquadCádiz                       -0.84455    0.18486  -4.569 5.24e-06 ***
## SquadLazio                       -0.74960    0.21987  -3.409 0.000666 ***
## SquadManchester_Utd              -0.79446    0.48829  -1.627 0.103904
## ToSuc                             1.12850    0.39563   2.852 0.004387 **
## height_in_cm                      0.25232    0.13659   1.847 0.064859 .
## SquadValencia                     0.15724    0.23670   0.664 0.506587
## SquadLille                        0.68078    0.18568   3.666 0.000253 ***
## SquadLyon                         0.49645    0.17805   2.788 0.005353 **
## has_agent                         0.39551    0.10073   3.926 8.94e-05 ***
## SquadStuttgart                    0.42649    0.18065   2.361 0.018334 *
## TB                                1.20818    0.55660   2.171 0.030086 *
## PasProg                          -1.36697    0.39263  -3.482 0.000510 ***
## Pas3rd                            1.26242    0.31875   3.960 7.76e-05 ***
## PasMedCmp.                        0.55179    0.14343   3.847 0.000124 ***
## CkOut                             0.97480    0.30114   3.237 0.001229 **
## SquadBayern_Munich                0.46497    0.19699   2.360 0.018358 *
## ShoPK                             0.53040    0.27105   1.957 0.050520 .
## positionGoalkeeper               -0.96765    0.16655  -5.810 7.35e-09 ***
## TouDefPen                         0.63116    0.26781   2.357 0.018541 *
## BlkSh                            -0.33429    0.41704  -0.802 0.422904
## SquadAlmería                     -0.37152    0.20631  -1.801 0.071904 .
## SquadSalernitana                 -0.29641    0.18698  -1.585 0.113095
## SquadNottham_Forest              -0.42838    0.15867  -2.700 0.007000 **
## TklMid3rd                          1.03122    0.36896   2.795 0.005245 **
## TklDriAtt                         -0.62451    0.43933  -1.421 0.155343
## TklAtt3rd                          0.89051    0.39573   2.250 0.024547 *
## SquadArsenal                      -0.42593    0.21630  -1.969 0.049080 *
## PasTotCmp.                        -0.35333    0.24530  -1.440 0.149930
## PasShoCmp.                         0.48442    0.17995   2.692 0.007168 **
## CompSerie_A                       -0.19151    0.16026  -1.195 0.232236
## PKcon                              0.67346    0.44017   1.530 0.126188
## MP                                 2.55945    0.28861   8.868  < 2e-16 ***
## Age:MP                            -4.53985    0.38621 -11.755  < 2e-16 ***
## contract_expiration_year:has_agent -0.84148    0.30591  -2.751 0.006004 **
## CompPremier_League:Age            -0.74330    0.28189  -2.637 0.008439 **
## height_in_cm:CompSerie_A           0.52242    0.28656   1.823 0.068454 .
## Age:PasMedCmp                     -3.53495    0.71298  -4.958 7.78e-07 ***
## Age:SquadManchester_Utd            2.85987    1.04097   2.747 0.006067 **
## contract_expiration_year:Age       6.05614    0.93148   6.502 1.02e-10 ***
## contract_expiration_year:MP       -2.78936    0.46737  -5.968 2.87e-09 ***
## League_TierChampions_League:MP     0.19595    0.17419   1.125 0.260763
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8034 on 1840 degrees of freedom
## Multiple R-squared:  0.7202, Adjusted R-squared:  0.7099
## F-statistic: 69.65 on 68 and 1840 DF,  p-value: < 2.2e-16
```

```
# diagnostic plots
plot(LM_MODEL2)
```


Residuals vs Fitted
Residuals
Fitted values
lm(LM_Model)
10032
18780
15644
3
2
1
0
−1
−2
−3
12
14
16
18
20


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(LM_Model)
10032
18780
5579


Scale−Location
10032
19578
12780
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
12
14
16
18
20
Fitted values
lm(LM_Model)



```r
# checking for multicollinearity
vif(LM_MODEL2, "predictor")
```

```
## GVIFs computed for predictors

##                                     GVIF Df GVIF^(1/(2*Df))
## contract_expiration_year       25.664498  8        1.224851
## Min                             8.354332  1        2.890386
## League_TierChampions_League    74.108823  3        2.049486
## CompPremier_League             15.352123  3        1.576503
## Age                           187.093712 11        1.268456
## footUnspecified                 1.098808  1        1.048241
## Goals                           1.945127  1        1.394678
## League_TierRelegation           2.355085  1        1.534628
## League_TierNot_Qualified        2.997939  1        1.731456
## PasLonCmp.                      1.552008  1        1.245796
## SCA                             1.687198  1        1.298922
## SquadReal_Sociedad              1.145071  1        1.070080
## Fls                             1.310533  1        1.144785
## SquadClermont_Foot              1.043430  1        1.021484
## SquadOsasuna                    1.091551  1        1.044773
## X2CrdY                          1.044392  1        1.021955
## SoT.                            1.285857  1        1.133956
## SquadVillarreal                 1.135843  1        1.065759
## SquadNewcastle_Utd              1.249466  1        1.117795
## SquadLens                     1.130249  1        1.063132
## SquadNice                     1.054893  1        1.027080
## SquadMonaco                   1.050539  1        1.024958
## SquadSevilla                  1.065437  1        1.032200
## SquadBochum                   1.054201  1        1.026743
## SquadBournemouth              1.113291  1        1.055126
## TI                            1.238723  1        1.112979
## SquadUnion_Berlin             1.134470  1        1.065115
## SquadCádiz                    1.047792  1        1.023617
## SquadLazio                    1.188316  1        1.090099
## SquadManchester_Utd          10.208135  3        1.472847
## ToSuc                         1.296417  1        1.138603
## height_in_cm                  1.938500  3        1.116634
## SquadValencia                 1.035036  1        1.017367
## SquadLille                    1.057118  1        1.028162
## SquadLyon                     1.068020  1        1.033451
## has_agent                     8.679389  3        1.433557
## SquadStuttgart                1.099428  1        1.048536
## TB                            1.169516  1        1.081441
## PasProg                       2.624877  1        1.620147
## Pas3rd                        2.542989  1        1.594675
## PasMedCmp.                    2.102769  1        1.450093
## CkOut                         1.216103  1        1.102771
## SquadBayern_Munich            1.130854  1        1.063416
## ShoPK                         1.273329  1        1.128419
## positionGoalkeeper            4.492962  1        2.119661
## TouDefPen                     5.842894  1        2.417208
## BlkSh                         1.381167  1        1.175231
## SquadAlmería                  1.111055  1        1.054066
## SquadSalernitana              1.071977  1        1.035363
## SquadNottham_Forest           1.151664  1        1.073156
## TklMid3rd                     1.229887  1        1.109003
## TklDriAtt                     1.510371  1        1.228971
## TklAtt3rd                     1.183518  1        1.087896
## SquadArsenal                  1.221187  1        1.105073
## PasTotCmp.                    2.923175  1        1.709730
## PasShoCmp.                    2.207129  1        1.485641
## CompSerie_A                   1.938500  3        1.116634
## PKcon                         1.071756  1        1.035256
## MP                          238.664026  8        1.408029
## PasMedCmp                    21.484783  2        2.152944
##                                                                                                    Interac
## contract_expiration_year                                                                       has_agent,
## Min
## League_TierChampions_League
## CompPremier_League
## Age                         MP, CompPremier_League, PasMedCmp, SquadManchester_Utd, contract_expirati
## footUnspecified
## Goals
## League_TierRelegation
## League_TierNot_Qualified
## PasLonCmp.
## SCA
## SquadReal_Sociedad
## Fls
## SquadClermont_Foot
## SquadOsasuna
## X2CrdY
## SoT.
## SquadVillarreal
## SquadNewcastle_Utd
## SquadLens
## SquadNice
## SquadMonaco
## SquadSevilla
## SquadBochum
## SquadBournemouth
## TI
## SquadUnion_Berlin
## SquadCádiz
## SquadLazio
## SquadManchester_Utd
## ToSuc
## height_in_cm                                                                                     Comp
## SquadValencia
## SquadLille
## SquadLyon
## has_agent                                                                              contract_expirati
## SquadStuttgart
## TB
## PasProg
## Pas3rd
## PasMedCmp.
## CkOut
## SquadBayern_Munich
## ShoPK
## positionGoalkeeper
## TouDefPen
## BlkSh
## SquadAlmería
## SquadSalernitana
## SquadNottham_Forest
## TklMid3rd
## TklDriAtt
## TklAtt3rd
## SquadArsenal
## PasTotCmp.
## PasShoCmp.
## CompSerie_A                                                                                      heig
## PKcon
## MP                                                   Age, contract_expiration_year, League_TierChampions
## PasMedCmp                                                                                             Pa
##
## contract_expiration_year                                                 Min, League_TierChampions_Leag
## Min                              contract_expiration_year, League_TierChampions_League, CompPremier_Leag
## League_TierChampions_League                                  contract_expiration_year, Min, CompPremier_
## CompPremier_League                                   contract_expiration_year, Min, League_TierChampions_
## Age
## footUnspecified                          contract_expiration_year, Min, League_TierChampions_League,
## Goals                          contract_expiration_year, Min, League_TierChampions_League, CompPremi
## League_TierRelegation                          contract_expiration_year, Min, League_TierChampions_Le
## League_TierNot_Qualified                          contract_expiration_year, Min, League_TierChampions
## PasLonCmp.                          contract_expiration_year, Min, League_TierChampions_League, CompP
## SCA                          contract_expiration_year, Min, League_TierChampions_League, CompPremier_
## SquadReal_Sociedad                          contract_expiration_year, Min, League_TierChampions_Leag
## Fls                          contract_expiration_year, Min, League_TierChampions_League, CompPremier_
## SquadClermont_Foot                          contract_expiration_year, Min, League_TierChampions_Leag
## SquadOsasuna                          contract_expiration_year, Min, League_TierChampions_League, Co
## X2CrdY                          contract_expiration_year, Min, League_TierChampions_League, CompPrem
## SoT.                          contract_expiration_year, Min, League_TierChampions_League, CompPremie
## SquadVillarreal                          contract_expiration_year, Min, League_TierChampions_League,
## SquadNewcastle_Utd                          contract_expiration_year, Min, League_TierChampions_Leag
## SquadLens                          contract_expiration_year, Min, League_TierChampions_League, CompPr
## SquadNice                          contract_expiration_year, Min, League_TierChampions_League, CompPr
## SquadMonaco                          contract_expiration_year, Min, League_TierChampions_League, Comp
## SquadSevilla                          contract_expiration_year, Min, League_TierChampions_League, Co
## SquadBochum                          contract_expiration_year, Min, League_TierChampions_League, Comp
## SquadBournemouth                          contract_expiration_year, Min, League_TierChampions_League
## TI                          contract_expiration_year, Min, League_TierChampions_League, CompPremier_
## SquadUnion_Berlin                          contract_expiration_year, Min, League_TierChampions_League
## SquadCádiz                          contract_expiration_year, Min, League_TierChampions_League, CompP
## SquadLazio                          contract_expiration_year, Min, League_TierChampions_League, CompP
## SquadManchester_Utd                               contract_expiration_year, Min, League_TierChampions
## ToSuc                          contract_expiration_year, Min, League_TierChampions_League, CompPremi
## height_in_cm                                       contract_expiration_year, Min, League_TierChampio
## SquadValencia                          contract_expiration_year, Min, League_TierChampions_League, C
## SquadLille                          contract_expiration_year, Min, League_TierChampions_League, CompP
## SquadLyon                          contract_expiration_year, Min, League_TierChampions_League, CompPr
## has_agent                                                    Min, League_TierChampions_League, CompPr
## SquadStuttgart                          contract_expiration_year, Min, League_TierChampions_League,
## TB                          contract_expiration_year, Min, League_TierChampions_League, CompPremier_
## PasProg                          contract_expiration_year, Min, League_TierChampions_League, CompPrem
## Pas3rd                          contract_expiration_year, Min, League_TierChampions_League, CompPrem
## PasMedCmp.                          contract_expiration_year, Min, League_TierChampions_League, CompP
## CkOut                          contract_expiration_year, Min, League_TierChampions_League, CompPremi
## SquadBayern_Munich                          contract_expiration_year, Min, League_TierChampions_Leag
## ShoPK                          contract_expiration_year, Min, League_TierChampions_League, CompPremi
## positionGoalkeeper                          contract_expiration_year, Min, League_TierChampions_Leag
## TouDefPen                          contract_expiration_year, Min, League_TierChampions_League, CompPr
## BlkSh                          contract_expiration_year, Min, League_TierChampions_League, CompPremi
## SquadAlmería                          contract_expiration_year, Min, League_TierChampions_League, Co
## SquadSalernitana                          contract_expiration_year, Min, League_TierChampions_League
## SquadNottham_Forest                          contract_expiration_year, Min, League_TierChampions_Lea
## TklMid3rd                          contract_expiration_year, Min, League_TierChampions_League, CompPr
## TklDriAtt                          contract_expiration_year, Min, League_TierChampions_League, CompPr
## TklAtt3rd                          contract_expiration_year, Min, League_TierChampions_League, CompPr
## SquadArsenal                          contract_expiration_year, Min, League_TierChampions_League, Co
## PasTotCmp.                          contract_expiration_year, Min, League_TierChampions_League, CompP
## PasShoCmp.                          contract_expiration_year, Min, League_TierChampions_League, CompP
## CompSerie_A                                        contract_expiration_year, Min, League_TierChampio
## PKcon                          contract_expiration_year, Min, League_TierChampions_League, CompPremi
## MP                                                                                       Min, CompPre
## PasMedCmp                               contract_expiration_year, Min, League_TierChampions_League,
```

## Model Selection & Building

**Linear Regression** Our first model in the model building process of this analysis is an extension of the section discussed earlier where we explored different linear models fit with variables found in the step-wise selection process. Using the LM Model outlined in the previous section, we found a training error rate of 0.791 and a test error rate of 0.718.

For curiosities sake, we decided to run the model again on the training data with outliers remove, resulting in LM 2 Model (LM2). Our MSE using LM2 on the training data without outliers included resulted in a train error of 0.6082, additionally running LM2 to predict the test data set resulted in an MSE of 0.6955 which is higher then the train error rate. We noted that both metrics are markedly lower than the original LM Model.

```
# -----------------------------------------------
# 1. Fit a Linear Model (LM)
# -----------------------------------------------
# Predict using training data with outliers
LM_MODEL_train_preds <- predict(LM_Model, train_data)
# Predict using testing data with outliers
LM_MODEL_test_preds <- predict(LM_Model, test_data)

# training error on the LM1 model with outliers
LM1_train_error <-  mean((LM_MODEL_train_preds - train_data$log_market_value_in_eur)^2)
# testing error on the LM1 model with outliers
LM1_test_error <-  mean((LM_MODEL_test_preds - test_data$log_market_value_in_eur)^2)

cat("Linear Regression Train MSE: ", LM1_train_error, "\n")
```

```
## Linear Regression Train MSE:  0.8011134
```

```
cat("Linear Regression Train MSE: ", LM1_test_error, "\n")
```

```
## Linear Regression Train MSE:  0.6936898
```

```
# Predict using training data without outliers
LM_MODEL2_train_preds <- predict(LM_MODEL2, train_data_clean)
# Predict using testing data without outliers
LM_MODEL2_test_preds <- predict(LM_MODEL2, test_data)

# training error on the LM2 model without outliers
LM2_train_error <- mean((LM_MODEL2_train_preds - train_data_clean$log_market_value_in_eur)^2)
# testing error on the LM2 model without outliers
LM2_test_error <- mean((LM_MODEL2_test_preds - test_data$log_market_value_in_eur)^2)

cat("Linear Regression (cleaned data) Train MSE: ", LM2_train_error, "\n")
```

```
## Linear Regression (cleaned data) Train MSE:  0.6221485
```

```r
cat("Linear Regression (cleaned data) Test MSE: ", LM2_test_error, "\n")
```

```
## Linear Regression (cleaned data) Test MSE:  0.6700709
```

**Tree Regression** We next decided to fit the data with a regression tree model. These types of models are generally easier to interpret then the linear models we fit previously. A regression tree model was fit with on the training data using the predictors found using the original step wise regression.

A summary of our tree regression model (tree model) is outlined below. The outputted model includes 12 terminal nodes and has a residual mean deviance of 1.295. Furthermore, the MSE on the training data was 1.286 and the MSE on the testing data was 1.3937.

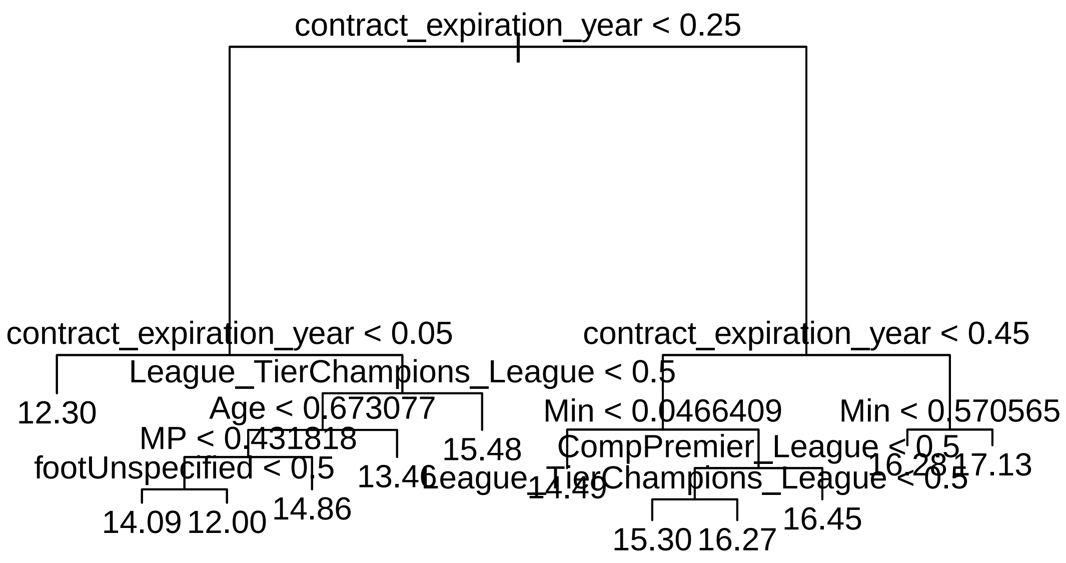
It is generally wise to perform cross validation procedures on regression tree models in a process called "pruning". Specifically, we are trying to determine the optimal level of tree complexity. Doing this, we found a tree with 12 terminal nodes results in the lowest cross validation error rate.

Interestingly, after pruning, we are left with the same exact regression tree model found prior to any cross validation procedures. As such, our MSE on the training and test data was identical after cross validation: 1.286 and 1.3937 on training and test data respectively.

```r
# ------------------------------------------------
# 2. Fit a Regression Tree
# ------------------------------------------------
# Fit the regression tree model

tree_model <- tree(log_market_value_in_eur ~ contract_expiration_year + Min +
                  League_TierChampions_League + CompPremier_League + Age +
                  footUnspecified + Goals + League_TierRelegation +
                  League_TierNot_Qualified + PasLonCmp. + SCA +
                  SquadReal_Sociedad + Fls + SquadClermont_Foot +
                  SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                  SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                  SquadSevilla + SquadBochum + SquadBournemouth +
                  TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                  SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                  SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                  PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                  ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                  + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                  TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                  PasShoCmp.  + CompSerie_A + PKcon + MP, data = train_data)

# Summary of the fitted tree model
summary(tree_model)
```

```
##
## Regression tree:
## tree(formula = log_market_value_in_eur ~ contract_expiration_year +
##     Min + League_TierChampions_League + CompPremier_League +
##     Age + footUnspecified + Goals + League_TierRelegation + League_TierNot_Qualified +
##     PasLonCmp. + SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
##     SquadOsasuna + X2CrdY + SoT. + SquadVillarreal + SquadNewcastle_Utd +
##     SquadLens + SquadNice + SquadMonaco + SquadSevilla + SquadBochum +
##     SquadBournemouth + TI + SquadUnion_Berlin + SquadCádiz +
##     SquadLazio + SquadManchester_Utd + ToSuc + height_in_cm +
##     SquadValencia + SquadLille + SquadLyon + has_agent + SquadStuttgart +
##     TB + PasProg + Pas3rd + PasMedCmp. + CkOut + SquadBayern_Munich +
##     ShoPK + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería +
##     SquadSalernitana + SquadNottham_Forest + TklMid3rd + TklDriAtt +
##     TklAtt3rd + SquadArsenal + PasTotCmp. + PasShoCmp. + CompSerie_A +
##     PKcon + MP, data = train_data)
## Variables actually used in tree construction:
## [1] "contract_expiration_year"    "League_TierChampions_League"
## [3] "Age"                         "MP"
## [5] "footUnspecified"             "Min"
## [7] "CompPremier_League"
## Number of terminal nodes:  12
## Residual mean deviance:  1.261 = 2423 / 1922
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -5.6480 -0.6961  0.0489  0.0000  0.7632  3.4240
```

```r
# Plot the tree structure
plot(tree_model)
text(tree_model, pretty = 0)
```



```r
# Predictions and Errors for Regression Tree
tree_train_preds <- predict(tree_model, train_data)
```

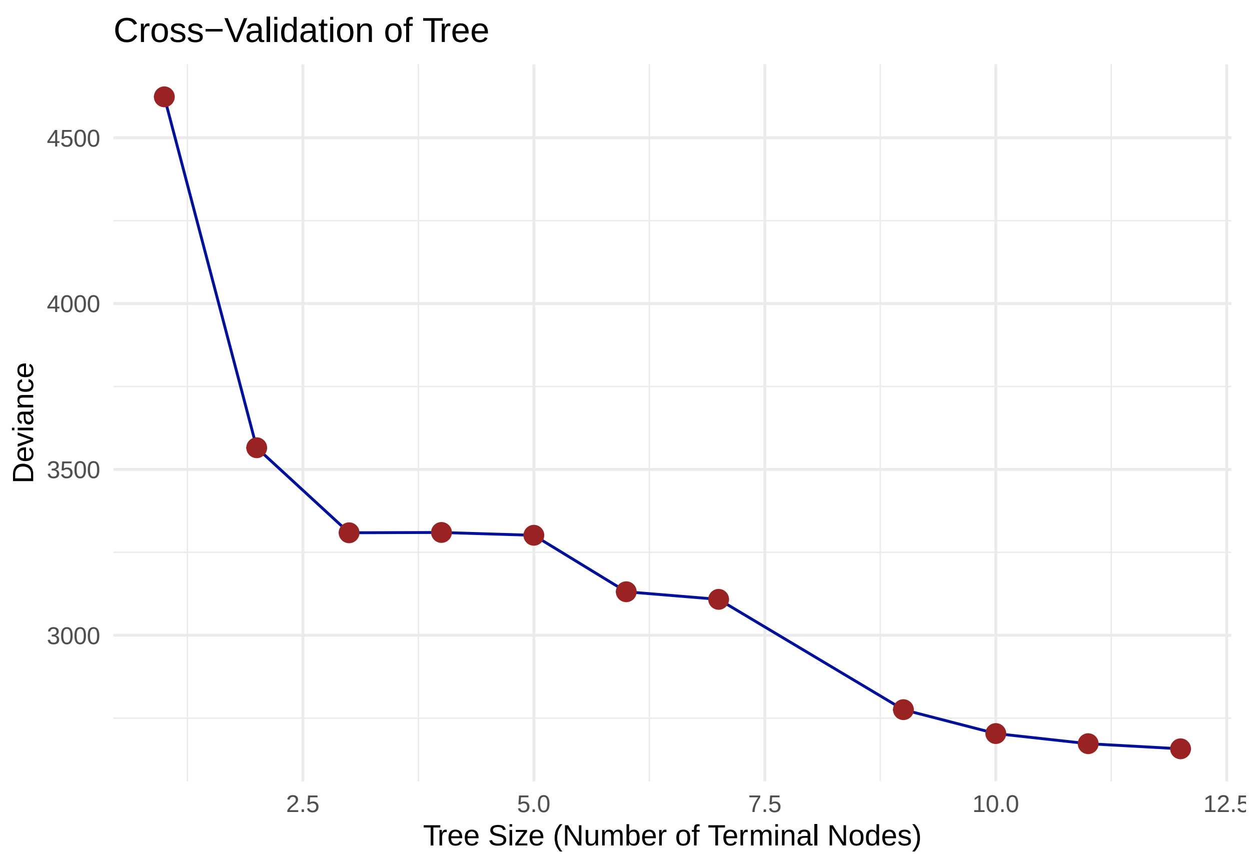
```r
tree_test_preds <- predict(tree_model, test_data)

# computing training and testing error
tree_train_error <- mean((tree_train_preds - train_data$log_market_value_in_eur)^2)
tree_test_error <- mean((tree_test_preds - test_data$log_market_value_in_eur)^2)

cat("Tree Regression Train MSE: ", tree_train_error, "\n")
```

```
## Tree Regression Train MSE:  1.253034
```

```r
cat("Tree Regression Test MSE: ", tree_test_error, "\n")
```

```
## Tree Regression Test MSE:  1.156619
```

```r
# ------------------------------------------------
# 3. Pruning the Tree
# ------------------------------------------------
set.seed(209)
CV1 <- cv.tree(tree_model)
cv_data <- data.frame(Size = CV1$size, Deviance = CV1$dev)

# Generate the plot
ggplot(cv_data, aes(x = Size, y = Deviance)) +
  geom_line(color = "#019") +
  geom_point(size = 3, color = "#922") +
  labs(
    title = "Cross-Validation of Tree",
    x = "Tree Size (Number of Terminal Nodes)",
    y = "Deviance"
  ) +
  theme_minimal()
```



```r
# Perform cross-validation
# pruning the tree based on the optimal level of trees, 7-node tree.
pruned_tree = prune.tree(tree_model, best = 13)
```

```
## Warning in prune.tree(tree_model, best = 13): best is bigger than tree size
```

```r
# Summary of the pruned tree
summary(pruned_tree)
```

```
##
## Regression tree:
## tree(formula = log_market_value_in_eur ~ contract_expiration_year +
##     Min + League_TierChampions_League + CompPremier_League +
##     Age + footUnspecified + Goals + League_TierRelegation + League_TierNot_Qualified +
##     PasLonCmp. + SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
##     SquadOsasuna + X2CrdY + SoT. + SquadVillarreal + SquadNewcastle_Utd +
##     SquadLens + SquadNice + SquadMonaco + SquadSevilla + SquadBochum +
##     SquadBournemouth + TI + SquadUnion_Berlin + SquadCádiz +
##     SquadLazio + SquadManchester_Utd + ToSuc + height_in_cm +
##     SquadValencia + SquadLille + SquadLyon + has_agent + SquadStuttgart +
##     TB + PasProg + Pas3rd + PasMedCmp. + CkOut + SquadBayern_Munich +
##     ShoPK + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería +
##     SquadSalernitana + SquadNottham_Forest + TklMid3rd + TklDriAtt +
##     TklAtt3rd + SquadArsenal + PasTotCmp. + PasShoCmp. + CompSerie_A +
##     PKcon + MP, data = train_data)
## Variables actually used in tree construction:
## [1] "contract_expiration_year"    "League_TierChampions_League"
## [3] "Age"                         "MP"
## [5] "footUnspecified"             "Min"
## [7] "CompPremier_League"
## Number of terminal nodes:  12
## Residual mean deviance:  1.261 = 2423 / 1922
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -5.6480 -0.6961  0.0489  0.0000  0.7632  3.4240
```

```r
# Predictions on training data
pruned_train_preds <- predict(pruned_tree, train_data)
pruned_test_preds <- predict(pruned_tree, test_data)


# Predictions on test data
pruned_train_error <- mean((pruned_train_preds - train_data$log_market_value_in_eur)^2)
pruned_test_error <- mean((pruned_test_preds - test_data$log_market_value_in_eur)^2)

# print training error
cat("Pruned Tree Train MSE: ", pruned_train_error, "\n")
```

```
## Pruned Tree Train MSE:  1.253034
```

```r
# print testing error
cat("Pruned Tree Test MSE: ", pruned_test_error, "\n")
```

```
## Pruned Tree Test MSE:  1.156619
```

**Neural Network** In this section, we explore fitting a neural network using the h2o.deeplearning function (h2o package). Using this package provides a more efficient method training a neural network, especially in handling a large set of predictors, in comparison to using the (neuralnet) package. We begin by initializing the H2O environment and convert our data sets to h2o data frame, as this is the only method of doing so.

The next step involves defining our predictors and response, in this case are we using the predictors that we had extracted from the forward selection step wise function, this narrows down the number of predictors to use. Furthermore to using this package, we are able to further configure beyond just number of hidden layers. In this case, the neural network is configured with three hidden layers, each containing five neurons, a learning rate of 0.0003, and 65 epochs. The activation function used in this case is Rectifier, and defined an automatic loss function, allowing the model to select best loss function based on the data we trained it on.

Furthermore, before training, we performed 10-cross validation to better assess the models performance, to evaluate the model on different subsets of the data. In the plot, we can observe the training record history across different epochs, in which the training MSE decreases in a smooth gradual slope as the number of epochs increases. This reflects on the chosen hyperparameters, tuned to specifically achieve this result, such as the learning rate, number of hidden layers and neurons. Unfortunately, in this package there isn't a function to plot the neural network architecture.

```r
# ------------------------------------------------
# 4. Fit a Neural Network
# ------------------------------------------------
# Initialize and Connect to H2O
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         5 days 16 hours
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.44.0.3
##     H2O cluster version age:    11 months and 12 days
##     H2O cluster name:           H2O_started_from_R_firasxcx_njo763
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.11 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.4.1 (2024-06-14)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is (11 months and 12 days) old. There may be a newer version available.
## Please download and install the latest version from: https://h2o-release.s3.amazonaws.com/h2o/latest
```

```r
# Converting training and testing data to an H2O frame
Train_h2o_data <- as.h2o(train_data)
```

```
##   |                                                                              |
```

```r
Test_h2o_data <- as.h2o(test_data)
```

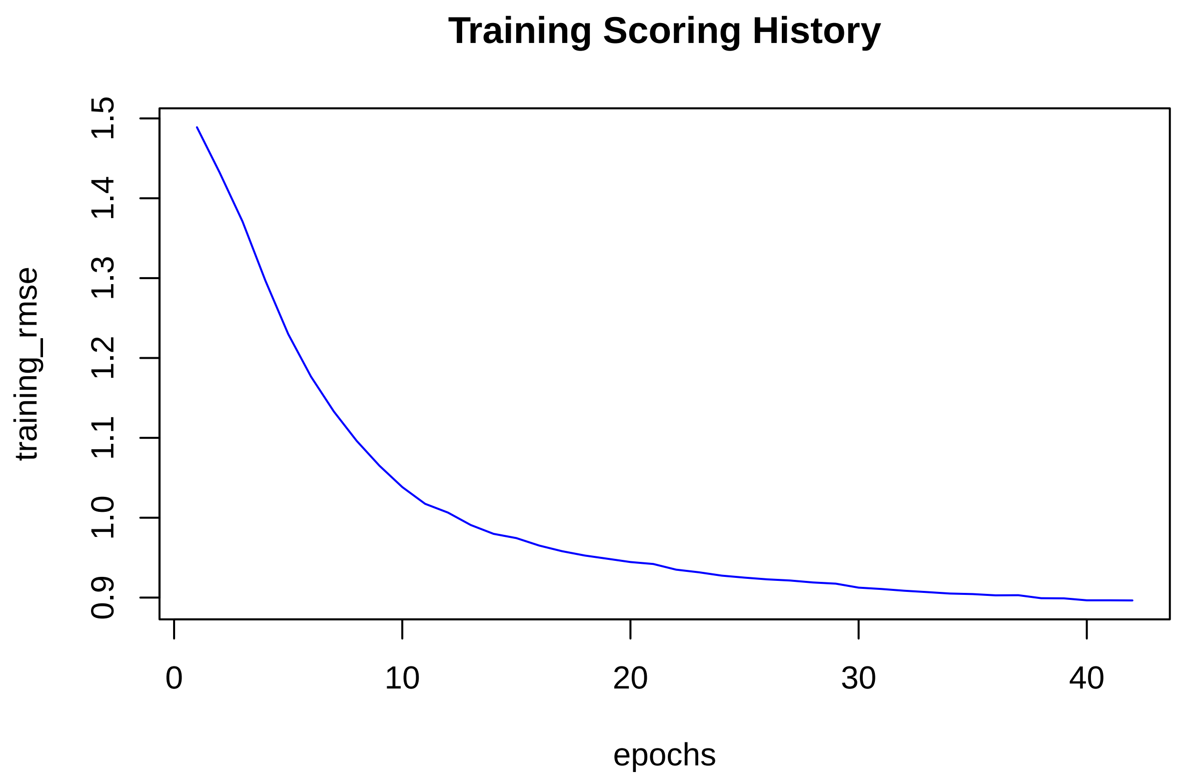
```
##   |                                                                              |
```

```r
# Defining our predictors to input into our NN model
predictors <- c("contract_expiration_year", "Min", "League_TierChampions_League",
                "CompPremier_League", "Age", "footUnspecified", "Goals",
                "League_TierRelegation", "League_TierNot_Qualified", "PasLonCmp.",
                "SCA", "SquadReal_Sociedad", "Fls", "SquadClermont_Foot",
                "SquadOsasuna", "X2CrdY", "SoT.", "SquadVillarreal", "SquadNewcastle_Utd",
                "SquadLens", "SquadNice", "SquadMonaco", "SquadSevilla", "SquadBochum",
                "SquadBournemouth", "TI", "SquadUnion_Berlin", "SquadCádiz", "SquadLazio",
                "SquadManchester_Utd", "ToSuc", "height_in_cm", "SquadValencia", "SquadLille",
                "SquadLyon","has_agent","SquadStuttgart", "TB", "PasProg", "Pas3rd", "PasMedCmp.",
                "CkOut", "SquadBayern_Munich", "ShoPK", "positionGoalkeeper", "TouDefPen",
                "BlkSh", "SquadAlmería", "SquadSalernitana", "SquadNottham_Forest",
                "TklMid3rd", "TklDriAtt", "TklAtt3rd", "SquadArsenal", "PasTotCmp.",
                "PasShoCmp.", "CompSerie_A", "PKcon", "MP")
```

```r
# defining our response variable
response <- "log_market_value_in_eur"

# Defining our model to train with 3 layers of 5 neurons, learning rate = 0.0003, and epochs = 65
NeuralNet_Model <- h2o.deeplearning(x = predictors, y = response,
                                    training_frame = Train_h2o_data,
                                    hidden = c(5, 5, 5), epochs = 65,
                                    activation = "Rectifier",
                                    loss = "Automatic", rate = 0.0003,
                                    adaptive_rate = FALSE,
                                    reproducible = TRUE, seed = 209,
                                    nfolds = 10, fold_assignment = "AUTO")
```

```
##   |                                                                              |
```

```r
# Training Score History plot
plot(NeuralNet_Model)
```



```r
# Make predictions on the training and testing data
nn_train_preds <- h2o.predict(NeuralNet_Model, Train_h2o_data)
```

```
##   |                                                                              |
```

```r
nn_test_preds <- h2o.predict(NeuralNet_Model, Test_h2o_data)
```

```
##   |                                                                              |
```

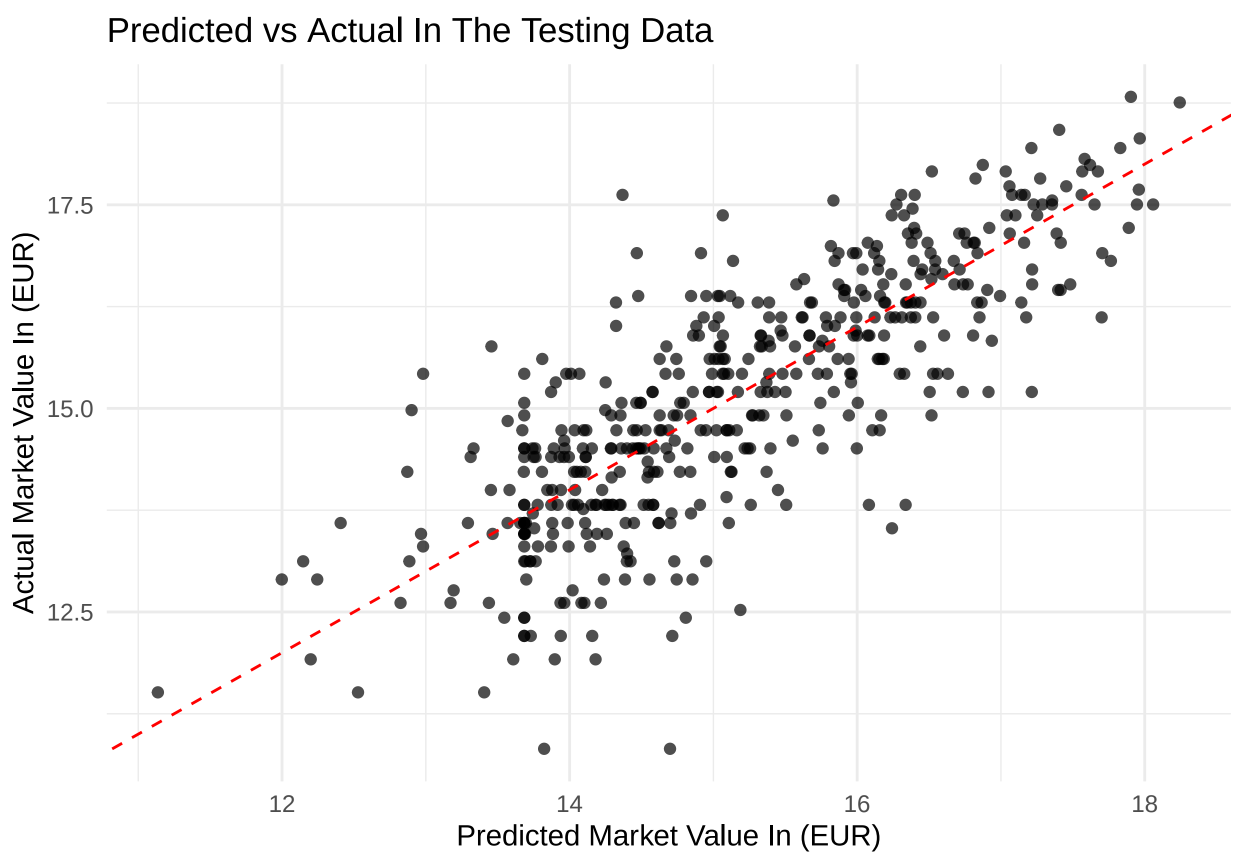
```r
# Converting to h2o data frames
nn_train_preds <- as.data.frame(nn_train_preds)
nn_test_preds <- as.data.frame(nn_test_preds)


# Compute MSE of the training and testing data
nn_train_error <- mean((nn_train_preds$predict - train_data$log_market_value_in_eur)^2)
nn_test_error <- mean((nn_test_preds$predict - test_data$log_market_value_in_eur)^2)
cat("Neural Net Training MSE: ", nn_train_error, "\n")
```

```
## Neural Net Training MSE:  0.8036039
```

```r
cat("Neural Net Testing MSE: ", nn_test_error, "\n")
```

```
## Neural Net Testing MSE:  0.8223758
```

```r
# Plotting predicted vs actual
ggplot(data = nn_test_preds, aes(x = predict, y = test_data$log_market_value_in_eur)) +
  geom_point(color = "black", alpha = 0.7) +
  labs(title = "Predicted vs Actual In The Testing Data",
       x = "Predicted Market Value In (EUR)",
       y = "Actual Market Value In (EUR)") +
  theme_minimal() +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed")
```



```r
dim(train_data)
```

```
## [1] 1934  240
```

```r
nrow(Train_h2o_data)
```

```
## [1] 1934
```

**Random Forest** The next objective in model building leads us to fitting a Random Forest model to predict the market value, by using the same selected predictors from the forward step wise selection function. Through the training progress, we fine tune the model in order to avoid underfitting and overfitting.

In this case, we find the best possible optimal tuning parameters, as mtry (53), ntree (100), and nodesize (108), this was decided based on another of tuning made. Initially, when assessing the training and test MSE, it resulted in 0.1658115, and 0.7310265, respectively. This was a concerning case of a model underfitting, which lead us increases the node size to 108, as before it wasn't defined.

When using the importance() function, we found that contract_expiration_year, Min, Comp-Premier_League yield the highest %IncMSE of 55.34932948, 21.03321502, 27.83448095, indicating to us the importance of these predictors in our model to predict the market value. While IncNodePurity values for contract_expiration_year, Min, League_TierChampions_League, yield the highest at 1354.2403852, 248.5828831, 173.4051049. This provides us significant these predictors are in reducing the node impurity / variance in our model.

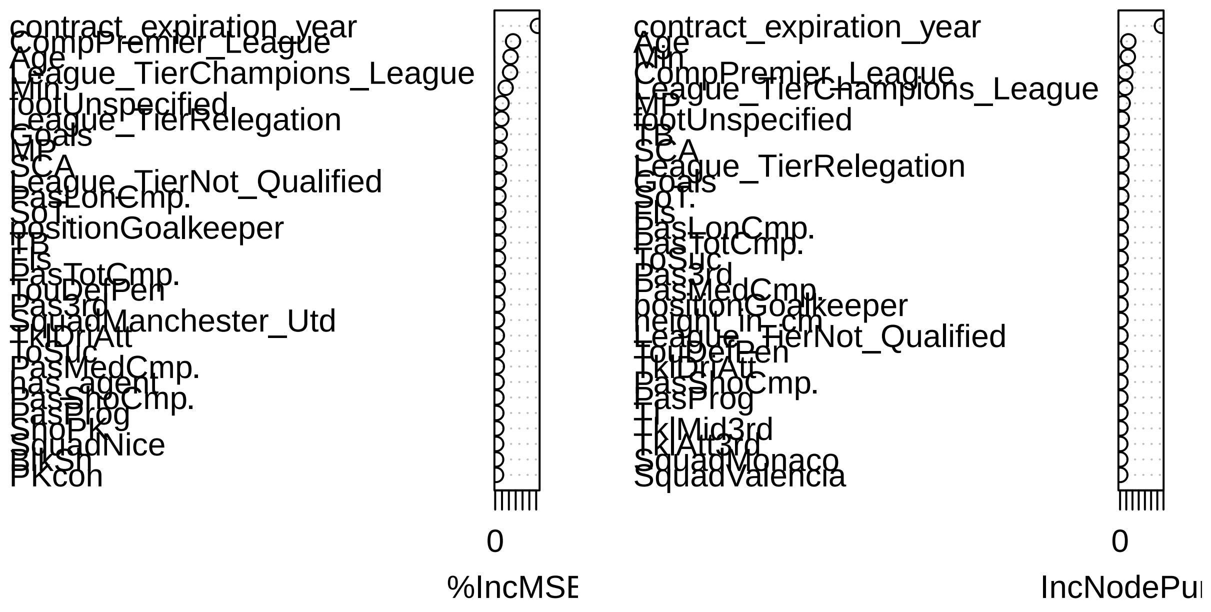
After training, we assessed the performance by computing the MSE of both the training and testing set, which as a result 0.7988766, and 0.8334457, respectively.

```
# ------------------------------------------------
# 5. Fit a Random Forest
# ------------------------------------------------
RF_M11 <- randomForest(log_market_value_in_eur ~ contract_expiration_year + Min + League_TierChampions_L
                  CompPremier_League + Age + footUnspecified + Goals +
                  League_TierRelegation + League_TierNot_Qualified + PasLonCmp. +
                  SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
                  SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                  SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                  SquadSevilla + SquadBochum + SquadBournemouth +
                  TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                  SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                  SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                  PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                  ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                  + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                  TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                  PasShoCmp.  + CompSerie_A + PKcon + MP, data = train_data,
                      mtry = 50,
                      importance = TRUE,
                      ntree = 100, nodesize = 108)
# Feature Importance
importance(RF_M11)
```

```
##                                 %IncMSE IncNodePurity
## contract_expiration_year    62.5382199  1349.2289938
## Min                         15.3024586   249.8179660
## League_TierChampions_League 21.8240585   165.0624246
## CompPremier_League          26.1607809   173.7305995
## Age                         22.5016299   266.4703301
## footUnspecified              9.2430311    56.8762471
## Goals                        6.6893138    42.8881332
## League_TierRelegation        9.1291783    43.0189607
## League_TierNot_Qualified     5.3034762    14.5803784
## PasLonCmp.                   4.7181455    21.5540396
## SCA                          5.9141940    45.7968779
## SquadReal_Sociedad           0.0000000     0.0000000
## Fls                          4.0816759    25.9141213
## SquadClermont_Foot           0.1683618     0.1755637
## SquadOsasuna                -1.2033857     2.8050971
## X2CrdY                       1.0050378     0.8239221
## SoT.                         4.5723524    40.3797244
## SquadVillarreal              0.0000000     0.0000000
## SquadNewcastle_Utd           0.0000000     0.0000000
## SquadLens                   -1.0050378     0.1479170
## SquadNice                    1.4845865     0.4967734
## SquadMonaco                 -2.7880504     7.1324589
## SquadSevilla                -1.7743166     0.9757043
## SquadBochum                  0.8147764     1.0331870
## SquadBournemouth            -1.4071726     0.9434905
## TI                           0.3308674    11.0398603
## SquadUnion_Berlin            0.0000000     0.0000000
## SquadCádiz                   0.9880111     0.6285657
## SquadLazio                 0.0000000   0.0000000
## SquadManchester_Utd        2.5865640   2.9704814
## ToSuc                      2.3807961  19.0558598
## height_in_cm               1.2884031  14.7066316
## SquadValencia             -1.2198985   6.3523388
## SquadLille                -0.4594166   0.4369492
## SquadLyon                  0.2104803   0.4303639
## has_agent                  2.1368363   5.2448159
## SquadStuttgart             0.0000000   0.0000000
## TB                         4.3407570  49.8695994
## PasProg                    1.6788400  11.2907737
## Pas3rd                     3.0071180  17.5414200
## PasMedCmp.                 2.3693410  16.8325459
## CkOut                     -1.0050378   0.1162074
## SquadBayern_Munich         0.0000000   0.0000000
## ShoPK                      1.6219884   1.8633440
## positionGoalkeeper         4.4842623  15.8698154
## TouDefPen                  3.6168646  13.8550918
## BlkSh                      1.3563194   4.7756670
## SquadAlmería               0.0000000   0.0000000
## SquadSalernitana           0.0000000   0.0000000
## SquadNottham_Forest        0.0000000   0.0000000
## TklMid3rd                  0.7327949  10.4899108
## TklDriAtt                  2.4452250  13.5884304
## TklAtt3rd                  1.1636275   7.5999700
## SquadArsenal              -1.0050378   0.1450186
## PasTotCmp.                 3.8054943  21.0574715
## PasShoCmp.                 1.8061213  12.3695439
## CompSerie_A                0.3659867   0.4108770
## PKcon                      1.2891938   0.7106361
## MP                         6.2433806  85.0529461
```

```r
varImpPlot(RF_M11)
```

RF_M11



```
# Use the best Random Forest model to predict on the training data
rf_train_preds <- predict(RF_M11, train_data)
rf_test_preds <- predict(RF_M11, test_data)

# Calculate the training error (mean squared error)
rf_train_error <- mean((rf_train_preds - train_data$log_market_value_in_eur)^2)
# Calculate the test error (mean squared error)
rf_test_error <- mean((rf_test_preds - test_data$log_market_value_in_eur)^2)
cat("Random Forest Train MSE: ", rf_train_error, "\n")
```

```
## Random Forest Train MSE:  0.7982987
```

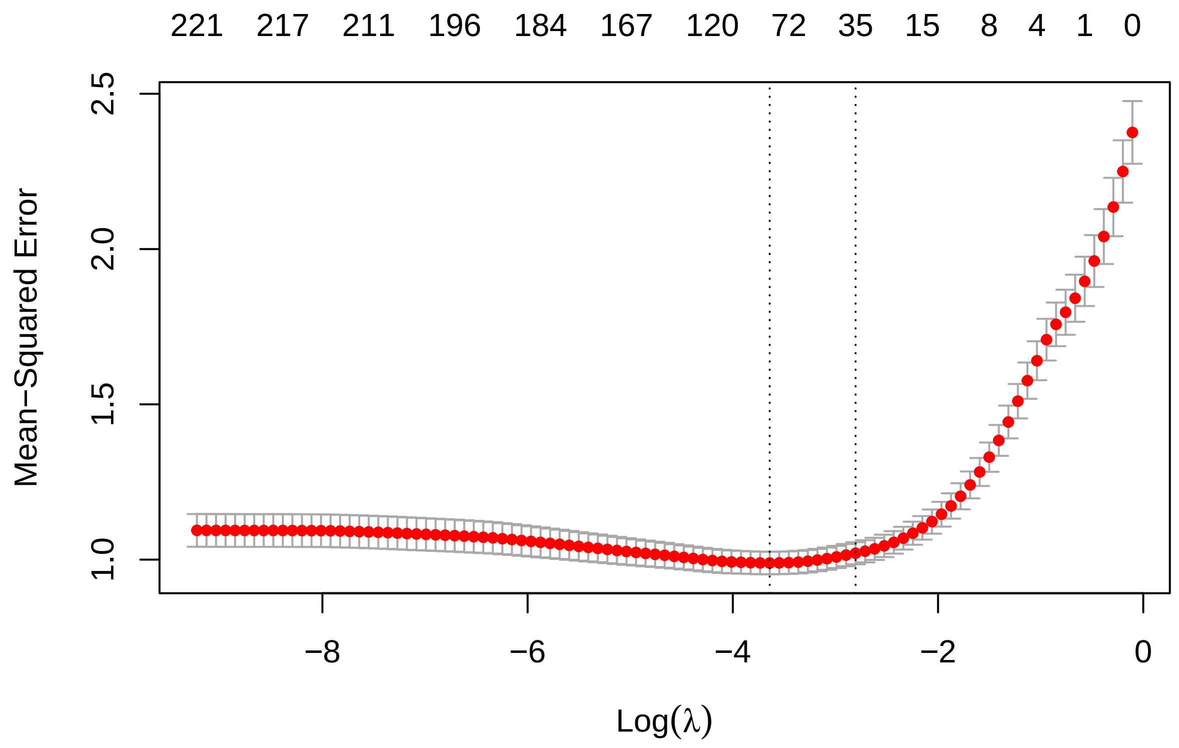
```
cat("Random Forest Test MSE: ", rf_test_error, "\n")
```

```
## Random Forest Test MSE:  0.8258373
```

**LASSO & Ridge Regression** Next we elected to construct both a LASSO model and Ridge Regression model to fit our data. These methods involve their own predictor selection as well, so we will not be utilizing the predictors outlined in our step wise selection section. The advantage of lasso over ridge is that lasso will actually force some predictor coefficients to be exactly equal to 0 in practice. This makes LASSO easier to interpret over ridge in some cases as ridge regression will generate a model with all predictors included even if the weights of some are extremely negligible.

We then fit a LASSO model, and after cross validation, we found the optimal lambda value to be 0.02623263 in reducing our MSE, and in this case came out to be 0.8684051 (training) and 0.9331179 (testing). We subsequently went on to fit a ridge regression using the same cross validation technique, with an optimal lambda value of 0.4375866. In this case our training MSE came out to be 0.8190284, and the testing was 0.8831628, this was a marginal difference, indicating a good fit for both models.

```
# ----------------------------------------------
# 6. Fit a Lasso model
# ----------------------------------------------
x_train <- model.matrix(log_market_value_in_eur ~ ., data = train_data1) # Remove intercept
y_train <- train_data1$log_market_value_in_eur

x_test <- model.matrix(log_market_value_in_eur ~ ., data = test_data1) # Remove intercept
y_test <- test_data1$log_market_value_in_eur

# Fit LASSO with cross-validation
lasso_model <- cv.glmnet(x_train, y_train, alpha = 1, nfolds = 10) # alpha = 1 for LASSO
plot(lasso_model)
```



```
# Optimal lambda
optimal_lambda1 <- lasso_model$lambda.min
print(optimal_lambda1)
```

```
## [1] 0.02623263
```

```r
# Predict on training and testing data
LSO_train_preds <- predict(lasso_model, s = optimal_lambda1, newx = x_train)
LSO_test_preds <- predict(lasso_model, s = optimal_lambda1, newx = x_test)

# Calculate MSE for train and test
LSO_train_mse <- mean((LSO_train_preds - y_train)^2)
LSO_test_mse <- mean((LSO_test_preds - y_test)^2)

cat("Lasso Train MSE: ", LSO_train_mse, "\n")
```

```
## Lasso Train MSE:  0.8684051
```

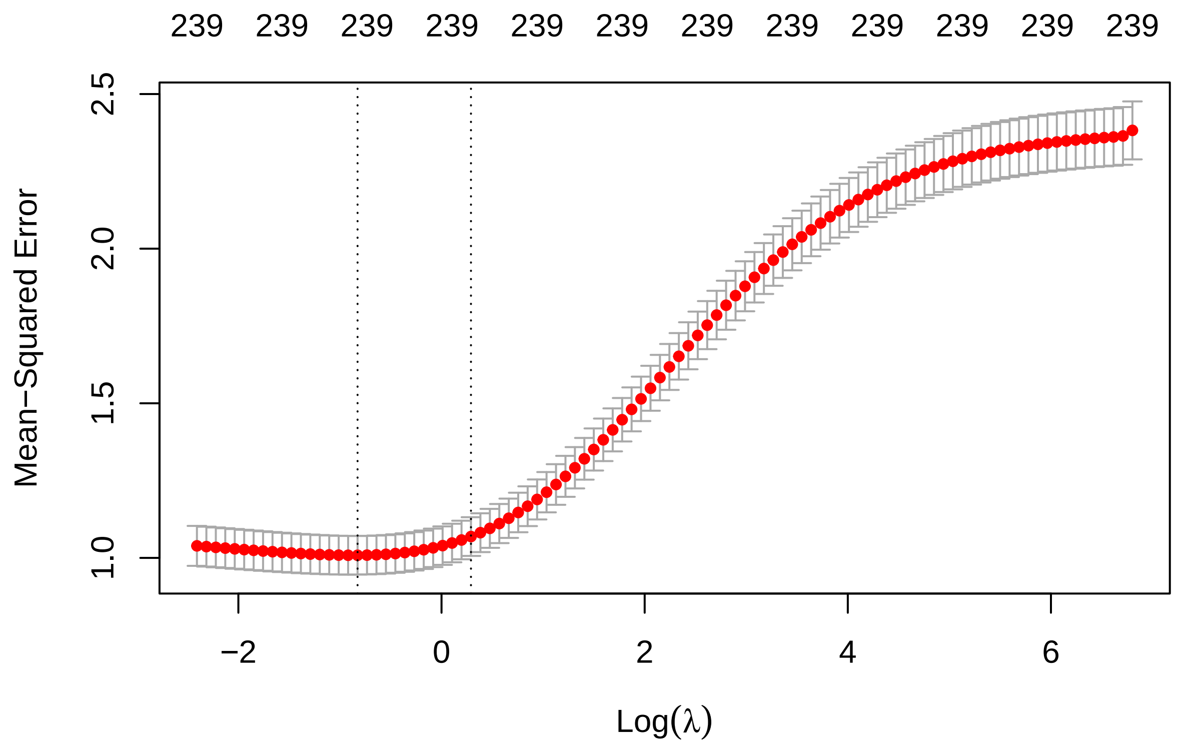
```r
cat("Lasso Test MSE: ", LSO_test_mse, "\n")
```

```
## Lasso Test MSE:  0.9331179
```

```r
# ------------------------------------------------
# 7. Fit a ridge regression
# ------------------------------------------------
# Fit ridge regression
ridge_model <- glmnet(x_train, y_train, alpha = 0)


# Cross-validation to find the best lambda
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0)
best_lambda1 <- cv_ridge$lambda.min
print(best_lambda1)
```

```
## [1] 0.4375866
```

```r
# Plot
plot(cv_ridge)
```



```r
# Predict on training data
RDG_train_preds <- predict(ridge_model, s = best_lambda1, newx = x_train)
RDG_test_preds <- predict(ridge_model, s = best_lambda1, newx = x_test)

# Predict on test data
RDG_train_error <- mean((RDG_train_preds - y_train)^2)
RDG_test_error <- mean((RDG_test_preds - y_test)^2)

cat("Ridge Regression Train MSE: ", RDG_train_error, "\n")
```

```
## Ridge Regression Train MSE:  0.8190284
```

```r
cat("Ridge Regression Test MSE: ", RDG_test_error, "\n")
```

```
## Ridge Regression Test MSE:  0.8831628
```

**Generalized Additive Models (GAM)** In this section, we explored fitting a Generalized Additive Model, in order to capture the non-linear relationships between our predictors and the response variable. In our case initially fitted the the predictors from our step-wise forward selection and used the smoothing terms for each of the numerical predictors.

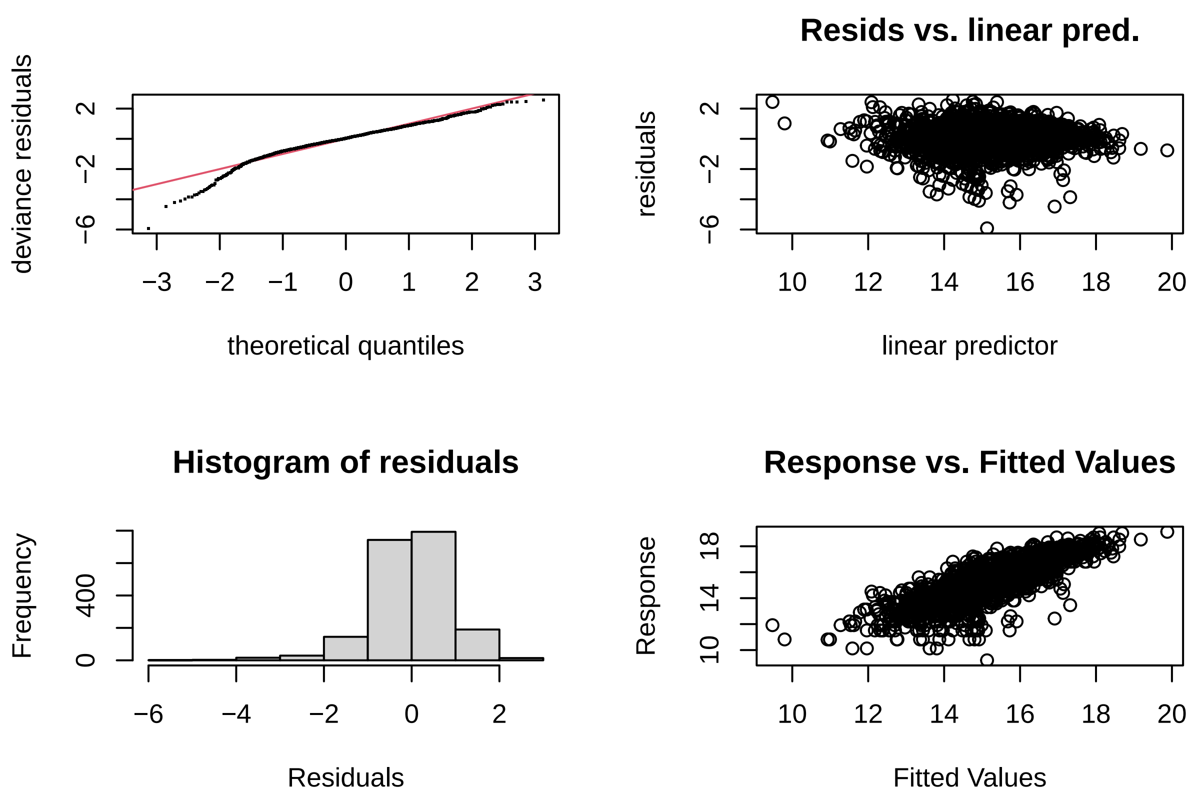
We verified that through looking at that summary output and gam.check diagnostics that these were the only predictors that required a smooth term; contract_expiration_year, Min, Age, SCA, TB, TouDefPen, TklDriAtt. While certain variable did seem to require a smooth term, however they proved to be more difficult to, and other predictors that were not needed due to their lack of importance in the model, were dropped.

As a result, we verified the model by computing the MSE on the training and testing sets, which in this case came out to be 0.7009415 (train MSE), 0.7835914 (test MSE).

```r
# ------------------------------------------------
# 8. Fitting a Gam Model
# ------------------------------------------------
gam_model <- gam(log_market_value_in_eur ~ s(contract_expiration_year) + s(Min) +
                   League_TierChampions_League + CompPremier_League + s(Age) +
                   footUnspecified + Goals + League_TierRelegation +
                   League_TierNot_Qualified + PasLonCmp. + s(SCA) + SquadReal_Sociedad
                  + SquadClermont_Foot + SquadOsasuna + X2CrdY + SquadNewcastle_Utd
                 + SquadLens + SquadNice  + SquadBochum + SquadBournemouth  +
                   SquadUnion_Berlin + SquadCádiz + SquadLazio + SquadManchester_Utd +
                   height_in_cm + SquadLille + SquadLyon + has_agent + s(TB) + PasProg
                 + Pas3rd + CkOut + SquadBayern_Munich + positionGoalkeeper +
                   s(TouDefPen) + SquadAlmería + s(TklDriAtt) + PasShoCmp. +
                   MP, data = train_data)

# output the summary
summary(gam_model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log_market_value_in_eur ~ s(contract_expiration_year) + s(Min) +
##     League_TierChampions_League + CompPremier_League + s(Age) +
##     footUnspecified + Goals + League_TierRelegation + League_TierNot_Qualified +
##     PasLonCmp. + s(SCA) + SquadReal_Sociedad + SquadClermont_Foot +
##     SquadOsasuna + X2CrdY + SquadNewcastle_Utd + SquadLens +
##     SquadNice + SquadBochum + SquadBournemouth + SquadUnion_Berlin +
##     SquadCádiz + SquadLazio + SquadManchester_Utd + height_in_cm +
##     SquadLille + SquadLyon + has_agent + s(TB) + PasProg + Pas3rd +
##     CkOut + SquadBayern_Munich + positionGoalkeeper + s(TouDefPen) +
##     SquadAlmería + s(TklDriAtt) + PasShoCmp. + MP
##
## Parametric coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  14.90781    0.21312  69.951  < 2e-16 ***
## League_TierChampions_League   0.56106    0.08944   6.273 4.40e-10 ***
## CompPremier_League            0.76941    0.05587  13.772  < 2e-16 ***
## footUnspecified              -1.64289    0.20159  -8.150 6.62e-16 ***
## Goals                         1.52177    0.31588   4.818 1.57e-06 ***
## League_TierRelegation        -0.90178    0.07885 -11.437  < 2e-16 ***
## League_TierNot_Qualified     -0.52537    0.06769  -7.761 1.38e-14 ***
## PasLonCmp.                    0.21293    0.10529   2.022 0.043284 *
## SquadReal_Sociedad           -0.48385    0.22053  -2.194 0.028359 *
## SquadClermont_Foot           -0.77284    0.24605  -3.141 0.001710 **
## SquadOsasuna                 -1.20035    0.23951  -5.012 5.90e-07 ***
## X2CrdY                        2.33986    0.87182   2.684 0.007342 **
## SquadNewcastle_Utd          -0.60252    0.20713  -2.909 0.003670 **
## SquadLens                   -1.02347    0.22626  -4.523 6.47e-06 ***
## SquadNice                    0.62599    0.18561   3.373 0.000760 ***
## SquadBochum                 -0.69811    0.21163  -3.299 0.000989 ***
## SquadBournemouth            -0.52055    0.19808  -2.628 0.008660 **
## SquadUnion_Berlin           -0.75886    0.22557  -3.364 0.000783 ***
## SquadCádiz                  -0.82134    0.20726  -3.963 7.69e-05 ***
## SquadLazio                  -0.59424    0.23923  -2.484 0.013080 *
## SquadManchester_Utd          0.50924    0.19356   2.631 0.008586 **
## height_in_cm                 0.34229    0.14280   2.397 0.016628 *
## SquadLille                   0.58112    0.20771   2.798 0.005199 **
## SquadLyon                    0.49265    0.19837   2.483 0.013097 *
## has_agent                    0.20326    0.04808   4.228 2.47e-05 ***
## PasProg                     -1.39730    0.42428  -3.293 0.001009 **
## Pas3rd                       0.78194    0.34501   2.266 0.023539 *
## CkOut                        0.90729    0.34225   2.651 0.008094 **
## SquadBayern_Munich           0.61578    0.22059   2.791 0.005301 **
## positionGoalkeeper          -1.12560    0.19773  -5.692 1.45e-08 ***
## SquadAlmería                -0.38942    0.23317  -1.670 0.095061 .
## PasShoCmp.                   0.38915    0.15902   2.447 0.014489 *
## MP                          -0.43638    0.21001  -2.078 0.037857 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                               edf Ref.df      F  p-value
## s(contract_expiration_year) 7.847  8.463 76.315  < 2e-16 ***
## s(Min)                      8.155  8.801  7.969  < 2e-16 ***
## s(Age)                      7.550  8.383 39.981  < 2e-16 ***
## s(SCA)                      2.907  3.640  4.715 0.001938 **
## s(TB)                       4.147  4.919  4.643 0.000294 ***
## s(TouDefPen)                5.070  6.204  3.573 0.001293 **
## s(TklDriAtt)                2.117  2.638  2.116 0.087928 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.659   Deviance explained = 67.2%
## GCV = 0.84427  Scale est. = 0.81336   n = 1934
```

```r
# diagnostic of the model
gam.check(gam_model)
```



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 12 iterations.
## The RMS GCV score gradient at convergence was 2.804301e-06 .
## The Hessian was positive definite.
## Model rank =  96 / 96
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                                k'  edf k-index p-value
## s(contract_expiration_year) 9.00 7.85    0.97    0.12
## s(Min)                      9.00 8.16    1.03    0.88
## s(Age)                      9.00 7.55    1.04    0.96
## s(SCA)                      9.00 2.91    1.03    0.92
## s(TB)                       9.00 4.15    0.99    0.32
## s(TouDefPen)                9.00 5.07    1.03    0.84
## s(TklDriAtt)                9.00 2.12    1.00    0.49
```

```
# Predictions and Errors for Gam Model
gam_train_preds <- predict(gam_model, train_data)
gam_test_preds <- predict(gam_model, test_data)

# computing the MSE on both training and testing set
gam_train_error <- mean((gam_train_preds - train_data$log_market_value_in_eur)^2)
```

```
gam_test_error <- mean((gam_test_preds - test_data$log_market_value_in_eur)^2)

cat("GAM Train MSE: ", gam_train_error, "\n")
```

```
## GAM Train MSE:  0.7835914
```

```
cat("GAM Test MSE: ", gam_train_error, "\n")
```

```
## GAM Test MSE:  0.7835914
```

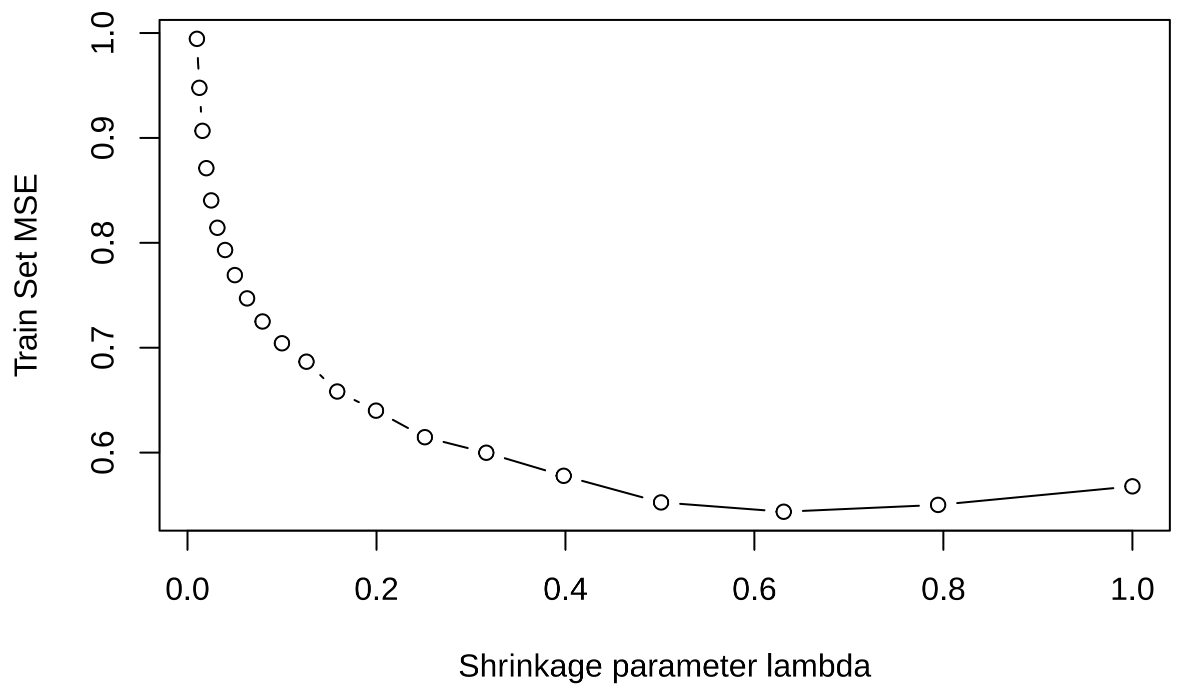
**Gradient Boosting Machine (GBM)** In the next section, we had explored fitting a GBM (Gradient Boosting Model), by fitting the model with our selected predictors to predict the market value. We evaluated the effect of our defined shrinkage parameter (lambda) to control the learning rate, and determine how the changes impact the model. We defined a sequence of shrinkage values ranging from 10^-2 to 10^0.1, and fitted into our model to determine the best optimal shrinkage value.

After looping over the sequence of shrinkage parameters (lambda) values, it is often an added value to visualize how the changes in these values correlate with the changes in the MSE. To achieve this, we first plotted the using training set, as expected we observed a gradual decline in the training set MSE as the shrinkage lambda value increase. However, when observing the testing set, we notice a steady incline as the shrinkage value lambda increased, this tells use that a larger shrinkage value might be leading to overftting our model, which will most likely provide inaccurate predictions due to poor generalization.

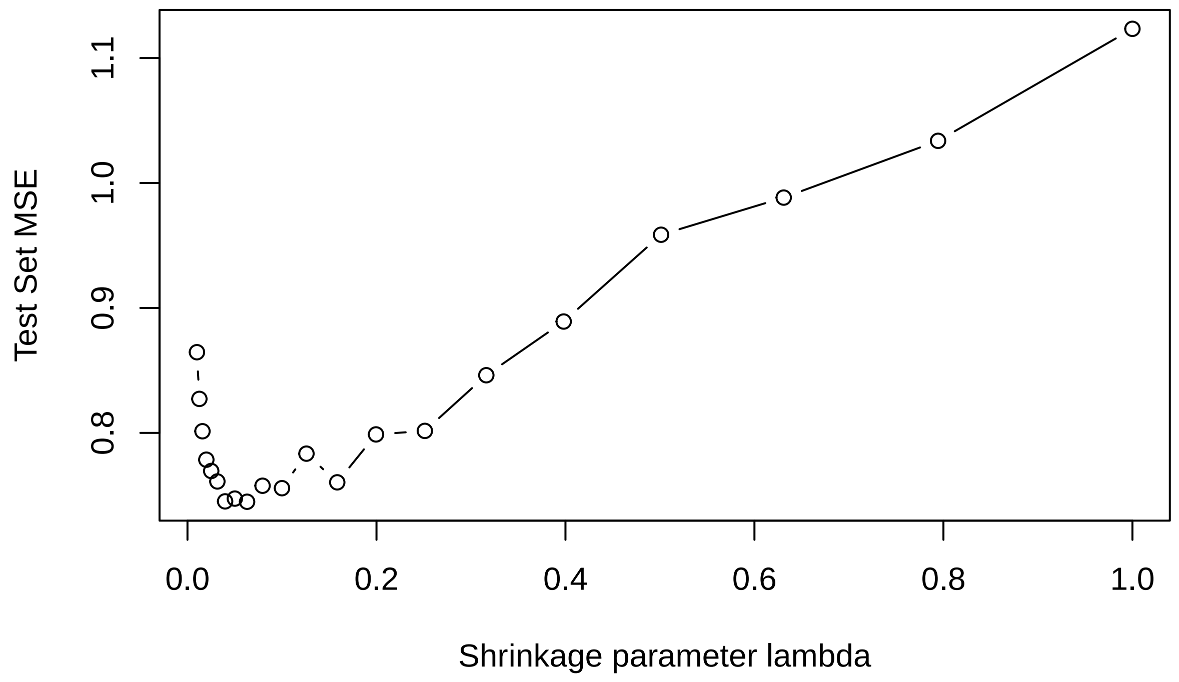
Before computing the MSE, we wanted to observe the output of our final model with the optimal shrinkage value (lambda), to determine what how our model valued the importance of the predictors. In this case, the top 5 predictors, in terms of relative importance, were:contract_expiration_year [13.05661592], Min [9.25264829], TklDriAtt. [7.74001819], PasTotCmp. [6.48737908] and ToSuc [5.50363204]

In this case, after plotting the shrinkage values when evaluated on both the training and testing set, we found the best possible value to be 0.06309573 This gave us a training error of 0.7442913, and a testing error of 0.7563976, which is indicating to us that our model is generalizing well, as there only a slight difference of 0.0121063, which is expected.

```
# ------------------------------------------------
# 9. Fit a GBM
# ------------------------------------------------
# Setting the seed
set.seed(209)

# Defining the shrinkage values
Seq <- seq(-2, 0, 0.1)
Lambdas <- 10^Seq

# Initialize to store train error
B_Train_Error <- rep(NA, length(Lambdas))

# Loop function to loop over Lambdas on the training set with 1,000 trees
for (i in 1:length(Lambdas)) {
  Boosted_ML = gbm(log_market_value_in_eur ~ contract_expiration_year + Min +
                     League_TierChampions_League +
                     CompPremier_League + Age + footUnspecified + Goals +
                     League_TierRelegation + League_TierNot_Qualified + PasLonCmp. +
                     SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
                     SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                     SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                     SquadSevilla + SquadBochum + SquadBournemouth +
                     TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                     SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                     SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                     PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                     ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería +
                     SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                     TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                     PasShoCmp.  + CompSerie_A + PKcon + MP, data = train_data,
                     distribution = "gaussian", n.trees = 1000, shrinkage = Lambdas[i])

  # Predict with training data
  Boosted_ML_Pred = predict(Boosted_ML, train_data, n.trees = 1000)

  # Computing training error (use log_market_value_in_eur instead of Salary)
  B_Train_Error[i] = mean((Boosted_ML_Pred - train_data$log_market_value_in_eur)^2)}


# Initialize to store test error
B_Test_Error <- rep(NA, length(Lambdas))

# Loop function to loop over Lambdas on the training set with 1,000 trees
for (i in 1:length(Lambdas)) {
  # Fit the model on the training data
  Boosted_ML = gbm(log_market_value_in_eur ~ contract_expiration_year + Min +
                     League_TierChampions_League + CompPremier_League + Age +
                     footUnspecified + Goals + League_TierRelegation +
                     League_TierNot_Qualified + PasLonCmp. + SCA +
                     SquadReal_Sociedad + Fls + SquadClermont_Foot +
                     SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                     SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                     SquadSevilla + SquadBochum + SquadBournemouth +
                     TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                     SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                     SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                     PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                     ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                   + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                     TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                     PasShoCmp.  + CompSerie_A + PKcon + MP, data = train_data,
                   distribution = "gaussian", n.trees = 1000, shrinkage = Lambdas[i])

  # Predict on the test data
  Boosted_ML_Pred = predict(Boosted_ML, test_data, n.trees = 1000)

  # Compute test error (MSE)
  B_Test_Error[i] = mean((Boosted_ML_Pred - test_data$log_market_value_in_eur)^2)}

# Plotting train error against Lambdas
plot(Lambdas, B_Train_Error, type = "b", xlab = "Shrinkage parameter lambda", ylab =
       "Train Set MSE")
```



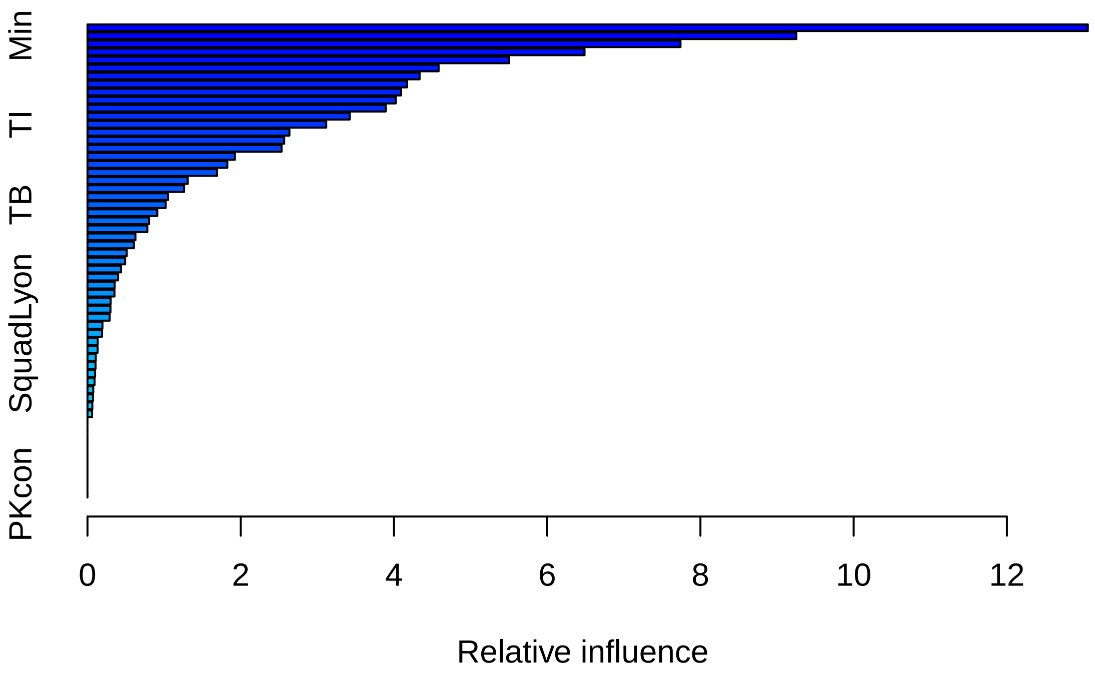
```r
# Plotting test error against Lambdas
plot(Lambdas, B_Test_Error, type = "b", xlab = "Shrinkage parameter lambda", ylab =
       "Test Set MSE")
```



```r
# Finding the optimal shrinkage value (lambda) with the minimum test MSE
optimal_lambda <- Lambdas[which.min(B_Test_Error)]
cat("Optimal shrinkage parameter (lambda): ", optimal_lambda, "\n")
```

```
## Optimal shrinkage parameter (lambda):  0.06309573
```

```r
# summary output
summary(Boosted_ML)
```



```
##                                                             var     rel.inf
## contract_expiration_year       contract_expiration_year 13.05661592
## Min                                                         Min  9.25264829
## TklDriAtt                                             TklDriAtt  7.74001819
## PasTotCmp.                                           PasTotCmp.  6.48737908
## ToSuc                                                     ToSuc  5.50363204
## Pas3rd                                                   Pas3rd  4.58299443
## PasProg                                                 PasProg  4.33525280
## Fls                                                         Fls  4.17292882
## TouDefPen                                             TouDefPen  4.09356455
## TklMid3rd                                             TklMid3rd  4.02317394
## Age                                                         Age  3.89267286
## SCA                                                         SCA  3.42244919
## TI                                                           TI  3.11643563
## PasMedCmp.                                           PasMedCmp.  2.63606530
## PasShoCmp.                                           PasShoCmp.  2.56810100
## height_in_cm                                       height_in_cm  2.53303450
## CompPremier_League                           CompPremier_League  1.92468411
## PasLonCmp.                                           PasLonCmp.  1.82518559
## League_TierChampions_League League_TierChampions_League  1.69109167
## BlkSh                                                     BlkSh  1.30872607
## TklAtt3rd                                             TklAtt3rd  1.26157310
## MP                                                           MP  1.05230977
## TB                                                           TB  1.02067593
## SquadSevilla                                       SquadSevilla  0.91098541
## CkOut                                                     CkOut  0.80442546
## League_TierRelegation              League_TierRelegation  0.78140230
## Goals                                              Goals  0.62602234
## SoT.                                                SoT.  0.60633999
## has_agent                                      has_agent  0.51267656
## ShoPK                                              ShoPK  0.49121343
## footUnspecified                          footUnspecified  0.43760784
## League_TierNot_Qualified        League_TierNot_Qualified  0.39969080
## positionGoalkeeper                    positionGoalkeeper  0.35482106
## SquadCádiz                                    SquadCádiz  0.35237928
## SquadBournemouth                        SquadBournemouth  0.30248104
## SquadNottham_Forest                  SquadNottham_Forest  0.30008895
## SquadOsasuna                                SquadOsasuna  0.28958823
## SquadNice                                      SquadNice  0.19576602
## SquadLyon                                      SquadLyon  0.19050707
## SquadStuttgart                            SquadStuttgart  0.13338805
## SquadSalernitana                        SquadSalernitana  0.13273123
## SquadReal_Sociedad                    SquadReal_Sociedad  0.10867066
## SquadNewcastle_Utd                    SquadNewcastle_Utd  0.10554506
## SquadLens                                      SquadLens  0.09911537
## SquadBochum                                  SquadBochum  0.09269474
## SquadUnion_Berlin                      SquadUnion_Berlin  0.07527289
## SquadLazio                                    SquadLazio  0.07152785
## SquadManchester_Utd                  SquadManchester_Utd  0.06375933
## SquadBayern_Munich                    SquadBayern_Munich  0.06008626
## SquadClermont_Foot                    SquadClermont_Foot  0.00000000
## X2CrdY                                            X2CrdY  0.00000000
## SquadVillarreal                          SquadVillarreal  0.00000000
## SquadMonaco                                  SquadMonaco  0.00000000
## SquadValencia                              SquadValencia  0.00000000
## SquadLille                                    SquadLille  0.00000000
## SquadAlmería                                SquadAlmería  0.00000000
## SquadArsenal                                SquadArsenal  0.00000000
## CompSerie_A                                  CompSerie_A  0.00000000
## PKcon                                              PKcon  0.00000000
```

```
# 'best_lambda optimal value of lambda obtained from the testing loop
best_lambda <- Lambdas[which.min(B_Test_Error)]
best_lambda
```

```
## [1] 0.06309573
```

```
# Train the final model with the best lambda
Final_Model <- gbm(log_market_value_in_eur ~ contract_expiration_year + Min + League_TierChampions_Leagu
                  CompPremier_League + Age + footUnspecified + Goals +
                  League_TierRelegation + League_TierNot_Qualified + PasLonCmp. +
                  SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
                  SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                  SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                  SquadSevilla + SquadBochum + SquadBournemouth +
                  TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                  SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                  SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                  PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                  ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                  + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                  TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                  PasShoCmp.  + CompSerie_A + PKcon + MP, data = train_data,
                   distribution = "gaussian",
                   n.trees = 1000,
                   shrinkage = best_lambda)

# Predicting Training and testing
gbm_train_preds <- predict(Final_Model, train_data, n.trees = 1000)
gbm_test_preds <- predict(Final_Model, test_data, n.trees = 1000)


# Compute the training and testing error
gbm_train_error <- mean((gbm_train_preds - train_data$log_market_value_in_eur)^2)
gbm_test_error <- mean((gbm_test_preds - test_data$log_market_value_in_eur)^2)


# output the training error
cat("GBM Train MSE: ", gbm_train_error, "\n")
```

```
## GBM Train MSE:  0.7442913
```

```
# output the testing error
cat("GBM Train MSE: ", gbm_test_error, "\n")
```

```
## GBM Train MSE:  0.7563976
```

**Support Vector Machine (SVM)** In our next model we wanted to explore and training was a SVM (Support Vector Machine). The model was tuned in order to perform better then the base model, in which the cost, Kernal, and cross where not specified. In which the performance was extremely poor with a training MSE of 0.585998, and testing MSE of 0.8305746, this indicated to use that our trained SVM was overfitting, and poor generalizing new unseen data.

In this case, the parameters that we tuned include the cost parameter (C) which we set at 0.31, and specified the kernel (radial base function). The model was then trained with the same predictors as some of the other models, and evaluated the model by computing the MSE on both the training and testing. As result, we were able to achieve a training error of 0.8529843, and testing error of 0.9325839, this is higher then the initial model. While these error are higher than those of the initial model, they suggest that the tuned SVM is not underfitting and has better generalization than the baseline model

```
# -------------------------------------------------
# 10. Fit an SVM
# -------------------------------------------------
# Fit SVM model
# Fit SVM model
svm_model <- svm(log_market_value_in_eur ~ contract_expiration_year + Min + League_TierChampions_League
                  CompPremier_League + Age + footUnspecified + Goals +
                  League_TierRelegation + League_TierNot_Qualified + PasLonCmp. +
                  SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
                  SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                  SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                  SquadSevilla + SquadBochum + SquadBournemouth +
                  TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                  SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                  SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                  PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                  ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                  + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                  TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                  PasShoCmp.  + CompSerie_A + PKcon + MP,
                  data = train_data)


# Predicting Training Error
svm_train_preds1 <- predict(svm_model, train_data)
svm_test_preds1 <- predict(svm_model, test_data)


# Compute  Error
svm_train_error1 <- mean((svm_train_preds1 - train_data$log_market_value_in_eur)^2)
svm_test_error1 <- mean((svm_test_preds1 - test_data$log_market_value_in_eur)^2)
cat("SVM Train MSE: ", svm_train_error1, "\n")
```

```
## SVM Train MSE:  0.585998
```

```
cat("SVM Test MSE: ", svm_test_error1, "\n")
```

```
## SVM Test MSE:  0.8305746
```

```
# Tuning the SVM
svm_model_tuned <- svm(log_market_value_in_eur ~ contract_expiration_year + Min +
                          CompPremier_League + Age + footUnspecified + Goals +
                          League_TierRelegation + League_TierNot_Qualified + PasLonCmp. +
                          SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
                          SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                          SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                          SquadSevilla + SquadBochum + SquadBournemouth +
                          TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                          SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                          SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                          PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                          ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                        + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                          TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                          PasShoCmp.  + CompSerie_A + PKcon + MP,
                        data = train_data,kernel = "radial", cost = 0.31)

# Predicting Training Error
svm_train_preds <- predict(svm_model_tuned, train_data)
svm_test_preds <- predict(svm_model_tuned, test_data)

# Compute Error
svm_train_error <- mean((svm_train_preds - train_data$log_market_value_in_eur)^2)
```

```r
svm_test_error <- mean((svm_test_preds - test_data$log_market_value_in_eur)^2)

# output the training error of the tuned model
cat("SVM Tuned Train MSE: ", svm_train_error, "\n")
```

```
## SVM Tuned Train MSE:  0.8529843
```

```r
# output the testing error of the tuned model
cat("SVM Tuned Test MSE: ", svm_test_error, "\n")
```

```
## SVM Tuned Test MSE:  0.9325839
```

**K-Nearest Neighbors (KNN)** In this section we elected to create and fit a K-Nearest Neighbors (KNN) model to the data. First a k-fold cross validation was fit to the training data with a k-value of 10. Thus our 10-fold cross validation will be the average of the 10 MSE's computed using the 10 folds. Training the model with this cross-validation technique, we found the optimal number for k to use was k = 7. The training and test MSE found using this method was 1.11 for the training MSE and 1.19 for the testing MSE.

```r
# ------------------------------------------------
# 11. Fit an KNN
# ------------------------------------------------

# Train control for cross-validation
train_control <- trainControl(method = "cv", number = 10)  # 10-fold cross-validation

# Train the KNN model with cross-validation
knn_cv_model <- train(
  log_market_value_in_eur ~ contract_expiration_year + Min + League_TierChampions_League +
                CompPremier_League + Age + footUnspecified + Goals +
                League_TierRelegation + League_TierNot_Qualified + PasLonCmp. +
                SCA + SquadReal_Sociedad + Fls + SquadClermont_Foot +
                SquadOsasuna + X2CrdY + SoT. + SquadVillarreal +
                SquadNewcastle_Utd + SquadLens + SquadNice + SquadMonaco +
                SquadSevilla + SquadBochum + SquadBournemouth +
                TI + SquadUnion_Berlin + SquadCádiz + SquadLazio +
                SquadManchester_Utd  + ToSuc + height_in_cm + SquadValencia +
                SquadLille + SquadLyon + has_agent + SquadStuttgart + TB +
                PasProg + Pas3rd  + PasMedCmp. + CkOut + SquadBayern_Munich +
                ShoPK  + positionGoalkeeper + TouDefPen + BlkSh + SquadAlmería
                + SquadSalernitana  + SquadNottham_Forest + TklMid3rd +
                TklDriAtt + TklAtt3rd + SquadArsenal + PasTotCmp. +
                PasShoCmp.  + CompSerie_A + PKcon + MP, data = train_data, method = "knn",
                trControl = train_control, tuneLength = 13)

# Output the best k value from cross-validation
best_k <- knn_cv_model$bestTune$k
print(paste("Best K value from cross-validation:", best_k))
```

```
## [1] "Best K value from cross-validation: 7"
```

```r
# Predicting Training Error
knn_train_preds <- predict(knn_cv_model, train_data)
knn_test_preds <- predict(knn_cv_model, test_data)

# Compute  Error
knn_train_error <- mean((knn_train_preds - train_data$log_market_value_in_eur)^2)
knn_test_error <- mean((knn_test_preds - test_data$log_market_value_in_eur)^2)

# output the training error of the tuned model
cat("KNN Train MSE: ", knn_train_error, "\n")
```

```
## KNN Train MSE:  1.024996
```

```r
# output the training error of the tuned model
cat("KNN Test MSE: ", knn_test_error, "\n")
```

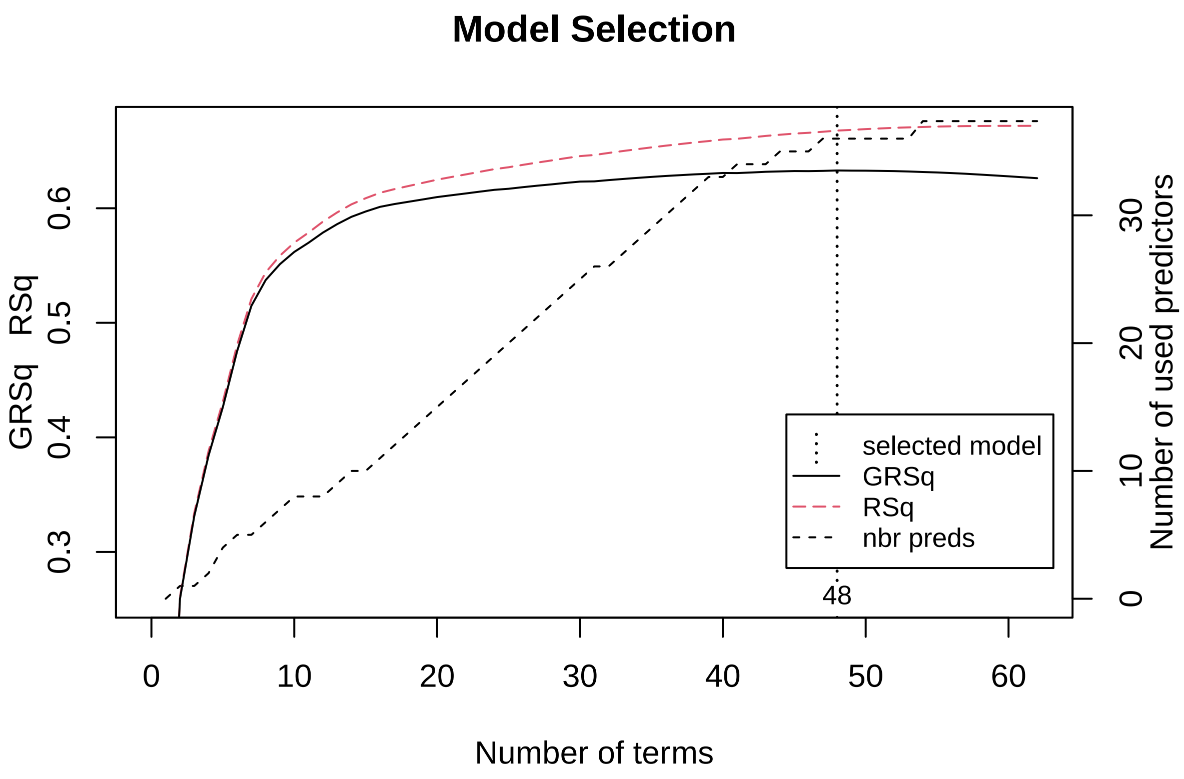
```
## KNN Test MSE:  1.187563
```

**Multivariate Adaptive Regression Splines (MARS)** Below is a model we fit using Multivariate Adaptive Regression Splines. The original model took quite of bit of time to generate. We can see from the original summary of the first model fit (Mars Model) that it achieved an R^2 of 0.69 on the training data. Additionally a plot of the MarsModel is included. We can see that 63 of 88 terms were used from the 47 of the 239 in our model. Our procedure dummy codes all categorical predictors. The vertical dashed line from the plot tells us the optimal number of non-intercept terms retained when the marginal increase in an additional predictor to GCV R^2 is less than 0.001. This occurs at 63 terms. Cross validation of the model is necessary. This process involves deciding what order of interaction effects to include and the number of terms to retain in the model. In our cross validation procedures we found that the ideal number of parameters was 23 and to only fit a degree of 1. Below a plot of our generalized cross validation procedures outlines the models most important predictors. Running the cross validated final MARS model (MarsModelFinal), we achieved a training MSE of 0.842 and a test MSE of 0.832.

```r
# -----------------------------------------------------------
# 12. Multivariate Adapative Regression Splines (MARS) Model
# -----------------------------------------------------------
MarsModel <- earth(log_market_value_in_eur~., data = train_data)

print(MarsModel)
```

```
## Selected 48 of 62 terms, and 36 of 239 predictors
## Termination condition: RSq changed by less than 0.001 at 62 terms
## Importance: contract_expiration_year, Min, Age, ...
## Number of terms at each degree of interaction: 1 47 (additive model)
## GCV 0.8766385    RSS 1532.948    GRSq 0.6329557    RSq 0.6677858
```

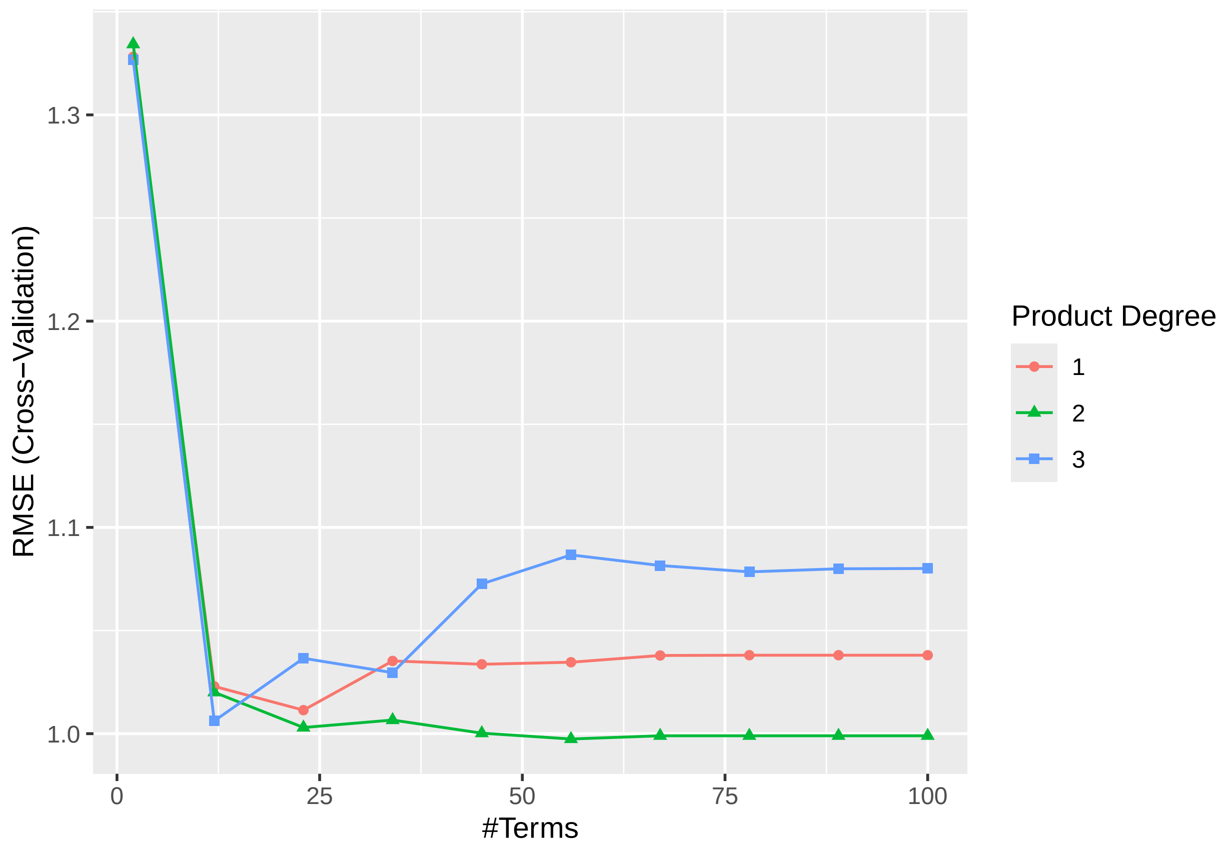
```r
plot(MarsModel, which = 1)
```



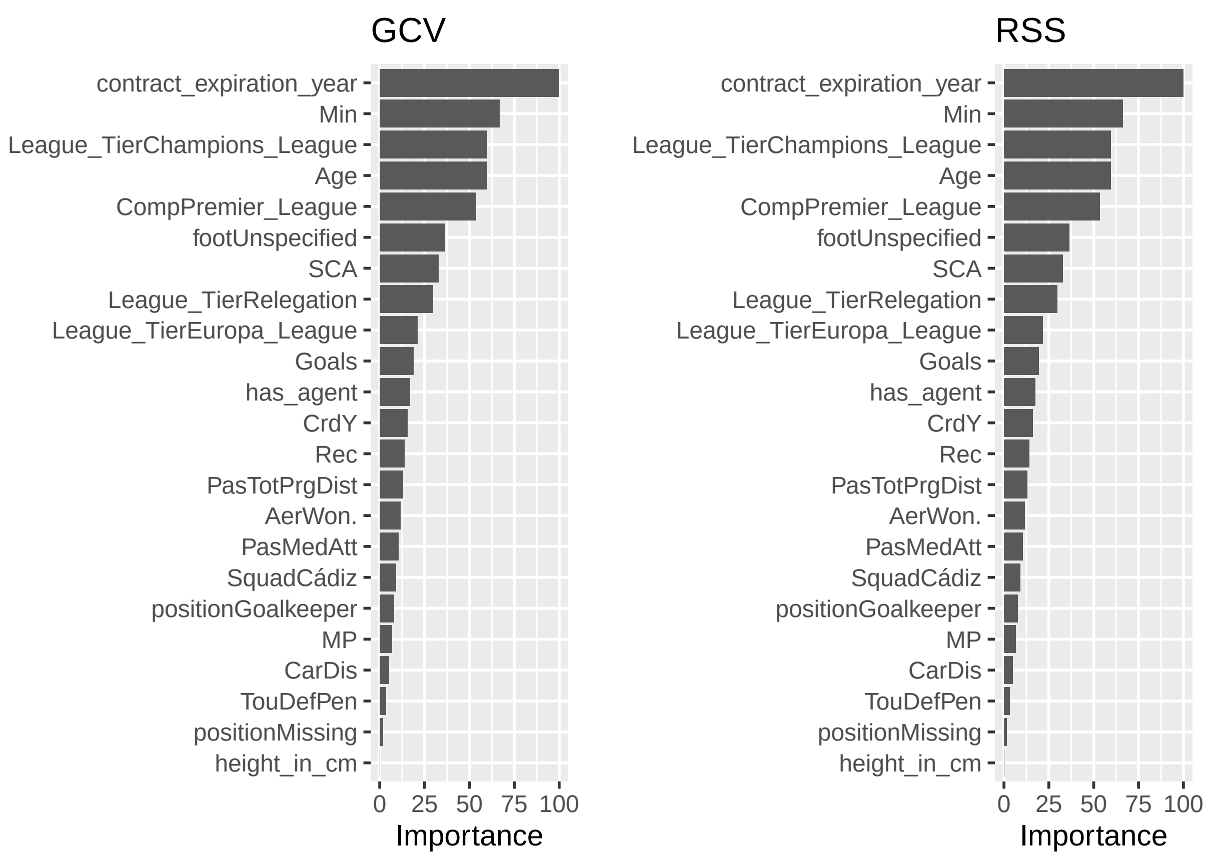
```
#We can see that 63 of 88 terms were used from the 47 of the 239 in our model. Our procedure dummy code

#Tuning (Cross Validation) of the MARS Model:

hyper_grid <- expand.grid(
  degree = 1:3,
  nprune = seq(2,100, length.out = 10) %>% floor()
)

set.seed(1)

tuner <- train(
  x = subset(train_data, select = -log_market_value_in_eur),
  y = train_data$log_market_value_in_eur,
  method = "earth",
  metric = "RMSE",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = hyper_grid
)

tuner$bestTune
```

```
##    nprune degree
## 16     56      2
```

```
#Our idea paramters are: nprune = 23, and degree = 1

ggplot(tuner)
```



```
p1 <- vip(tuner, num_features = 40, bar = FALSE, value = "gcv") + ggtitle("GCV")
p2 <- vip(tuner, num_features = 40, bar = FALSE, value = "rss") + ggtitle("RSS")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```



```r
#Final Model

MarsModelFinal <- earth(log_market_value_in_eur~., data = train_data, degree = 1, nprune = 23)

summary(MarsModelFinal)
```

```
## Call: earth(formula=log_market_value_in_eur~., data=train_data, degree=1,
##             nprune=23)
##
##                                    coefficients
## (Intercept)                          18.5184953
## positionMissing                      -3.8022660
## footUnspecified                      -1.8160169
## SquadBayern_Munich                    0.7849374
## SquadCádiz                           -0.8529773
## SquadClermont_Foot                   -0.9936160
## CompPremier_League                    0.7096522
## League_TierChampions_League           0.7170260
## League_TierEuropa_League              0.4379407
## League_TierRelegation                -0.3772336
## h(0.461538-Age)                       1.1916348
## h(Age-0.461538)                      -3.8311794
## h(0.110681-Min)                      -6.1218872
## h(Min-0.110681)                       0.9070493
## h(Goals-0.04)                         1.9462718
## h(0.0777236-PasTotPrgDist)            5.1864538
## h(PasMedCmp.-0.333)                0.7466832
## h(0.111-TB)                       -6.5960528
## h(0.054-TouAttPen)                -4.8255433
## h(0.369565-Rec)                   -1.7407891
## h(contract_expiration_year-0.1)  -14.1610121
## h(0.3-contract_expiration_year)  -18.3560049
## h(contract_expiration_year-0.3)   17.4459405
##
## Selected 23 of 62 terms, and 18 of 239 predictors (nprune=23)
## Termination condition: RSq changed by less than 0.001 at 62 terms
## Importance: contract_expiration_year, Min, Age, ...
## Number of terms at each degree of interaction: 1 22 (additive model)
## GCV 0.9207366    RSS 1698.802    GRSq 0.6144921    RSq 0.6318426
```

```r
predictionMars_train = predict(MarsModelFinal, train_data)

predictionMars_test = predict(MarsModelFinal, test_data)

(mean((train_data$log_market_value_in_eur - predictionMars_train)^2))
```

```
## [1] 0.878388
```

```r
(mean((test_data$log_market_value_in_eur - predictionMars_test)^2))
```

```
## [1] 0.7286166
```

```r
Mars_train_error <- mean((predictionMars_train - train_data$log_market_value_in_eur)^2)
Mars_test_error <- mean((predictionMars_test - test_data$log_market_value_in_eur)^2)

cat("MARS Model Train MSE: ", Mars_train_error, "\n")
```

```
## MARS Model Train MSE:  0.878388
```

```r
cat("MARS Model Test MSE: ", Mars_test_error, "\n")
```

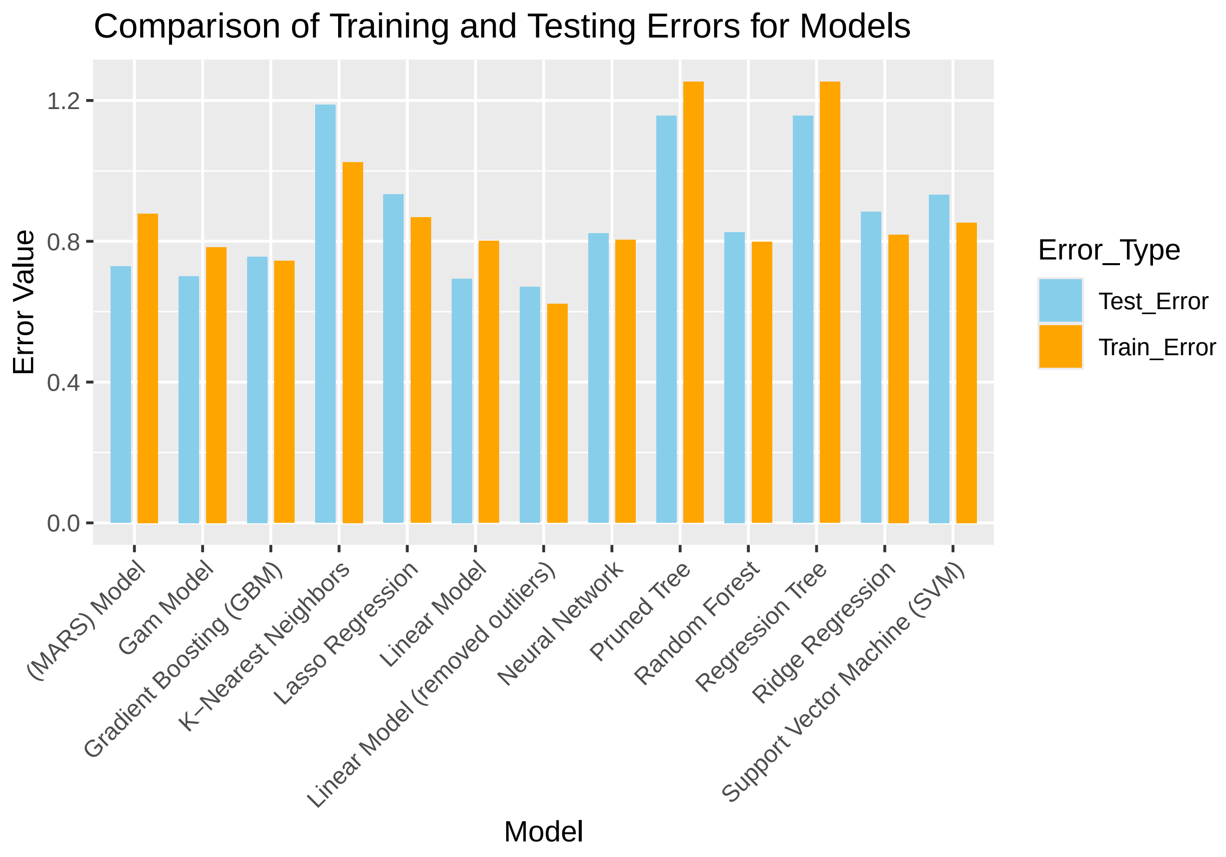
```
## MARS Model Test MSE:  0.7286166
```

## Model Comparisons

```r
# ------------------------------------------------
# Combine Results in a Table
# ------------------------------------------------
results <- data.frame(
  Model = c("Linear Model","Linear Model (removed outliers)","Regression Tree","Pruned Tree","Neural Ne
            "Random Forest","Lasso Regression","Ridge Regression","Gam Model",
            "Gradient Boosting (GBM)", "Support Vector Machine (SVM)","K-Nearest Neighbors","(MARS) Mode
  Train_Error = c(LM1_train_error, LM2_train_error ,tree_train_error, pruned_train_error,
                  nn_train_error,rf_train_error,LSO_train_mse, RDG_train_error,
                  gam_train_error,gbm_train_error,svm_train_error,knn_train_error,Mars_train_error),
  Test_Error = c(LM1_test_error, LM2_test_error ,tree_test_error, pruned_test_error, nn_test_error,
                 rf_test_error,LSO_test_mse, RDG_test_error,
                 gam_test_error,gbm_test_error, svm_test_error, knn_test_error,Mars_test_error))

print(results)
```

```
##                                Model Train_Error Test_Error
## 1                       Linear Model   0.8011134  0.6936898
## 2  Linear Model (removed outliers)   0.6221485  0.6700709
## 3                    Regression Tree   1.2530340  1.1566189
## 4                        Pruned Tree   1.2530340  1.1566189
## 5                     Neural Network   0.8036039  0.8223758
## 6                      Random Forest   0.7982987  0.8258373
## 7                   Lasso Regression   0.8684051  0.9331179
## 8                   Ridge Regression   0.8190284  0.8831628
## 9                          Gam Model   0.7835914  0.7009415
## 10           Gradient Boosting (GBM)   0.7442913  0.7563976
## 11      Support Vector Machine (SVM)   0.8529843  0.9325839
## 12               K-Nearest Neighbors   1.0249959  1.1875629
## 13                      (MARS) Model   0.8783880  0.7286166
```

```r
results_long <- tidyr::pivot_longer(results, cols = c("Train_Error", "Test_Error"),
                                    names_to = "Error_Type", values_to = "Error_Value")

# Plot the comparison of Train vs Test errors
ggplot(results_long, aes(x = Model, y = Error_Value, fill = Error_Type)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), width = 0.6) +
  labs(title = "Comparison of Training and Testing Errors for Models",
       x = "Model",
       y = "Error Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("skyblue", "orange"))
```



## Results and Findings

Based on the comparison table of our models, the performance was evaluated in terms of Mean Squared Error (MSE) on both the training and testing data sets. Thus ultimately allowing us to pin point specific models that are capable in accurately predicting the market value of a football player. As a result, our top five performing models based on the MSE comparison;

1. Linear Model (removed outliers): This essentially captures how well our our model is capable to standout in terms of performance after isolating the outliers that were observed during the diagnostic stage, as this is the only model that has the lowest training and testing error. This result highlights the critical role of pre-processing and diagnostics in enhancing a model's performance, by addressing outliers, and ensuring that all the assumptions of linear regression are kept true.

2. Gam Model: The Generalized Additive Model (GAM) demonstrated a significantly better performance in comparison to our initial linear regression model that contained outliers. As the training and testing (MSE) indicates to us that is capable of capturing the underlying non-linear relationships between the defined predictors and the response variable. The improvement highlights the GAM's capabilities in modeling a complex non-linear trends that a traditional linear model might not be able to pick up on.

3. Gradient Boosting (GBM): When a training a GMB, the model performed well, in achieving a training MSE of 0.7443 and a test MSE of 0.7564, a marginal difference of 0.0121, indicating a strong balance between variance and bias. This performance highlights the capabilities of a GBM ability to be reliable as predictive model, as the bias-variance trade off is effective in our case. This tells us that a GBM may be a strong candidate as the optimal choice in capturing the complex patterns within our data without sacrificing computational cost and interpretability.

4. Neural Network (NN): In terms of marginal differences, this model was able to exhibit a notable balance, as a result achieving a training MSE of 0.8036039, and a testing MSE of 0.8223758. The marginal difference of 0.0187719, highlighting the model's ability to generalize well to unseen data without overfitting, which is often a challenge we had faced when tuning and training our model, as the complexity for learning the underlying patterns is often challenging in terms of computational costs and time. Furthermore, it is worth noting, that this model utilized only a subset of our predictors. As the added option of using a step-wise selection function, we were able to reduce the number of predictors, make the model more manageable and computationally efficient, thus this approach was necessary to accommodate the limitations of our current computational resources. If we had access to greater computational power, we could expand the network to include additional predictors and a more complex architecture, this added benefit could potentially enable the neural network to outperform other models in this project.

5. Random Forest: The Random Forest model (RF) was able to achieve a training MSE of 0.8006667, and testing MSE of 0.8382549, this is again are marginal difference of 0.0375882. This would indicate to us that the model generalizing well to new unseen data, without overfitting on the training data. The relatively consistent performance highlights the effectiveness of the tune parameters, such as mtry, ntree and nodesize, which significantly impacted the overall performance. By fine-tuning these parameters, we were able to ensure that the model strikes a good balance between bias and variance. In comparison to the other models, the Random Forest (RF) performed quite well, by capturing complex relationships between the variables, while it didn't outperform the four models listed above, it still remains an optimal choice in handling a complex dataset.

## Conclusion

In our project, we aimed to predict the market value of football/soccer players based on a wide variety of predictors, such player-performance metrics, team affiliation, contract length details, player demographic details and other variables. Throughout this process, we had carefully managed and processed the data, ensuring its quality through appropriate data cleaning and feature engineering, all of which we determined was essential in enhancing the predictive power of our models.

By doing, we were able to avoid potential issues that could arise when fitting our models, particularly with complex models like neural networks. As neural networks often require the variables to be scaled, and they typically do not handle categorical variables well. By transforming categorical variables into dummy variables, we were able to include them in our models without losing valuable information. This allowed us to maintain a comprehensive set of predictors while ensuring the data was set up in suitable format for such models.

While in this project, we explored and trained a wide variety of models, in which most demonstrated varying levels of performance such as Linear Regression, Generalized Additive Model (GAM), Gradient Boosting (GBM), Neural Network (NN), and Random Forest. These model often struggled when handling of the data was not properly handled. This included issues with improper feature engineering, lack of scaling, and inadequate treatment of categorical variables hindered their performance. These factors significantly motivated the need for proper handling of the data processing, which in our case significantly improved the accuracy and reliability, allowing the models to perform optimally.

**Unexplained Variability** Despite the careful data processing and fine model tuning, there remains some unexplained variability in our predictions. One significant factor contributing to this is the tendency of certain clubs within these leagues to pay disproportionately high sums of money for players, often driven by factor beyond on-field performance metrics. As in most cases, some players are positioned significantly high in terms of their value in the football/soccer transfer market. As some are not just the reflection of a player's talent but also as a result of the football club's strategic investment, often linked to the potential for future value appreciation or the media-driven hyper around certain players.

For instance, a clubs may value a player due to anticipated marketability, brand value or the belief that the player's future performance could exceed expectations, even if their current performance metrics do not entirely justify the high value. Furthermore, media attention and fan perception often play a pivotal role in shaping these decisions, further contributing to the unpredictability of certain players value in the football transfer market. This highlights that player value is not solely determined by on-field performance but is also influenced by intangible elements such as marketability, fan engagement and media narratives.

By acknowledging these factors, we can recognize the limitations of predictive models in fully capturing the complexities of a player value in the market. Ultimately, our findings emphasize the interplay between performance metrics and league specific dynamics, such as varying levels of competition, and purchasing powers of certain teams. To counter these challenges, future work would require the incorporation of alternative data sources, such as sentiment analysis, player social media metrics, and club specific purchasing power, to better capture these influences. Recognizing these limitations not only refines the interpretation of our model outputs but also underscores the need for a deeper approach when analyzing the intricate economics of the football/soccer transfer market.