Fifa Players Project

Dan Reeves

25/09/2020

Introduction

Football has always been a hugely popular sport, attracting supporters from all over the globe to tune in and watch their favoured teams regardless of the time of day. Due to the huge amount of support and interest in the sport, football clubs and players are worth a huge amount of money and the competitions each clubs plays in can offer huge financial bonuses. With sports technology being at the cutting edge of innovation in todays world, all clubs are looking to find an edge over their rivals. This is where data analysis and machine learning can help. From in depth player performance analysis to aiding with business strategy and player valuation. It is with this in mind that I aim to build a model capable of predicting a players market value based on variables such as where they play and who they play for.

In the following report we shall look closer at the data set of players in todays footballing competitions and investigate the process of exploring and visualizing the data before analysing and drawing conclusions and making predictions.

The data set in question is found on kaggle.com but has been downloaded from my github repo and comprises of data scraped from the latest FIFA video game. The majority of this data consists of skill based attributes which are given a rating out of 100. These variables are excluded from my analysis as they are subjective. The features we will be looking to use in our model are for example; club, league, nationality, contract, age, european competition and international reputation. These variables are all factual and proven, and do not rely on the accuracy of a video game developers footballing knowledge.

The full data set can be found at https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset/download.

My Method

Once our dataset is downloaded into R, our first key step is to clean the data. This is to say we need to transform the data into a format in which we can use it in our analysis. Following this we explore the relationships between features and try to visulalize these in the most informative ways possible. Using the conclusions we draw from our findings, we can build the most appropriate model to minimize our loss function.

Preparation

There are various packages required in my analysis which may not be present in your current library. It is therefore necessary to prepare for the analysis with the following code.

```
if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table))
  install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(recosystem))
  install.packages("recosystem", repos = "http://cran.us.r-project.org")
if(!require(anytime))
  install.packages("anytime", repos = "http://cran.us.r-project.org")
if(!require(gridExtra))
  install.packages("gridExtra", repos = "http://cran.us.r-project.org")
if(!require(broom))
  install.packages("gridExtra", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(tinytex)
library(dbplyr)
library(dplyr)
library(caret)
library(dslabs)
library(ggplot2)
library(knitr)
library(latexpdf)
library(rmarkdown)
library(recosystem)
library(tidyverse)
library(tidyr)
library(gridExtra)
library(broom)
library(rpart)
library(rpart.plot)
```

The Data

Now we can begin cleaning our data. The first step is to remove the variables we are not interested in.

```
players_data <- data_import %>% select(sofifa_id, short_name, value_eur,
                                       age, height_cm, weight_kg, nationality, club,
                                       player_positions, international_reputation,
                                       team_position, contract_valid_until)
```

We are now only dealing with the useful features which we intend to use in our analysis. So let's inspect the dataset before we go any further.

```
##   sofifa_id        short_name value_eur age height_cm weight_kg nationality
## 1    158023          L. Messi  95500000  32       170        72   Argentina
## 2     20801 Cristiano Ronaldo  58500000  34       187        83    Portugal
## 3    190871         Neymar Jr 105500000  27       175        68      Brazil
## 4    200389          J. Oblak  77500000  26       188        87    Slovenia
```

```
## 5    183277         E. Hazard  90000000  28       175        74     Belgium
## 6    192985      K. De Bruyne  90000000  28       181        70     Belgium
##                  club player_positions international_reputation team_position
## 1        FC Barcelona       RW, CF, ST                         5            RW
## 2            Juventus           ST, LW                         5            LW
## 3 Paris Saint-Germain          LW, CAM                         5           CAM
## 4    AtlÃ©tico Madrid               GK                         3            GK
## 5         Real Madrid           LW, CF                         4            LW
## 6     Manchester City          CAM, CM                         4           RCM
##   contract_valid_until
## 1                 2021
## 2                 2022
## 3                 2022
## 4                 2023
## 5                 2024
## 6                 2023
```

We have 18278 observations of 12 variables in our dataset at this point, with variables saved as factors, characters and integers. We begin by altering the column concerning players contracts. We remove players who are not tied to a club, as their value is zero regardless and therefore not interesting to us. We also mutate the column to tell us how long is left on the current contract of each player rather than the year the deal expires.

```
## Removing players who are not contracted to a club,
# otherwise known as free agents.
contracted_players <- players_data %>%
  filter(!is.na(contract_valid_until))
## Now we alter the contract column to tell us how many years
# are left on the contract rather than the year it expires.
altered_contract_players <- contracted_players %>%
  mutate(yrs_left_on_contract = contract_valid_until-2019)
```

We now attempt to create a variable which categorises each player as either an attacker, midfielder, defender or goalkeeper. This is achieved through a simple mutate function, ifelse statements and regular expressions. We then use the same techniques to classify each players squad status, the league in which they play and whether or not they participate in European competition.

```
## Now we introduce a column which determines the class of the player,
# meaning whether they are an attacker or defender etc
# Position Class
attacker = c("ST", "LW", "RW", "LF", "RF", "CF")
midfielder = c("CM", "CDM", "CAM", "RM", "LM")
position_classes <- altered_contract_players %>%
  mutate(strongest_position = str_extract(player_positions, "^[A-Z]{2,3}")) %>%
  mutate(position_class = as.factor(
    if_else(strongest_position %in% attacker, "Attacker",
          if_else(strongest_position %in% midfielder, "Midfielder",
                  if_else(strongest_position == "GK", "Goal_Keeper", "Defender")))))
```

We must also defines the leagues and cups as vectors containing the clubs which participate in order generate these features. Once, we have defined all our features we can apply our own player ID system with our final dataset. This is done with simple arrange and mutate manipulations.

```r
## My next task is to sort an official player ID system.
add_player_id <- added_leagues %>% arrange(sofifa_id) %>%
  mutate(Player_ID = row_number())
```

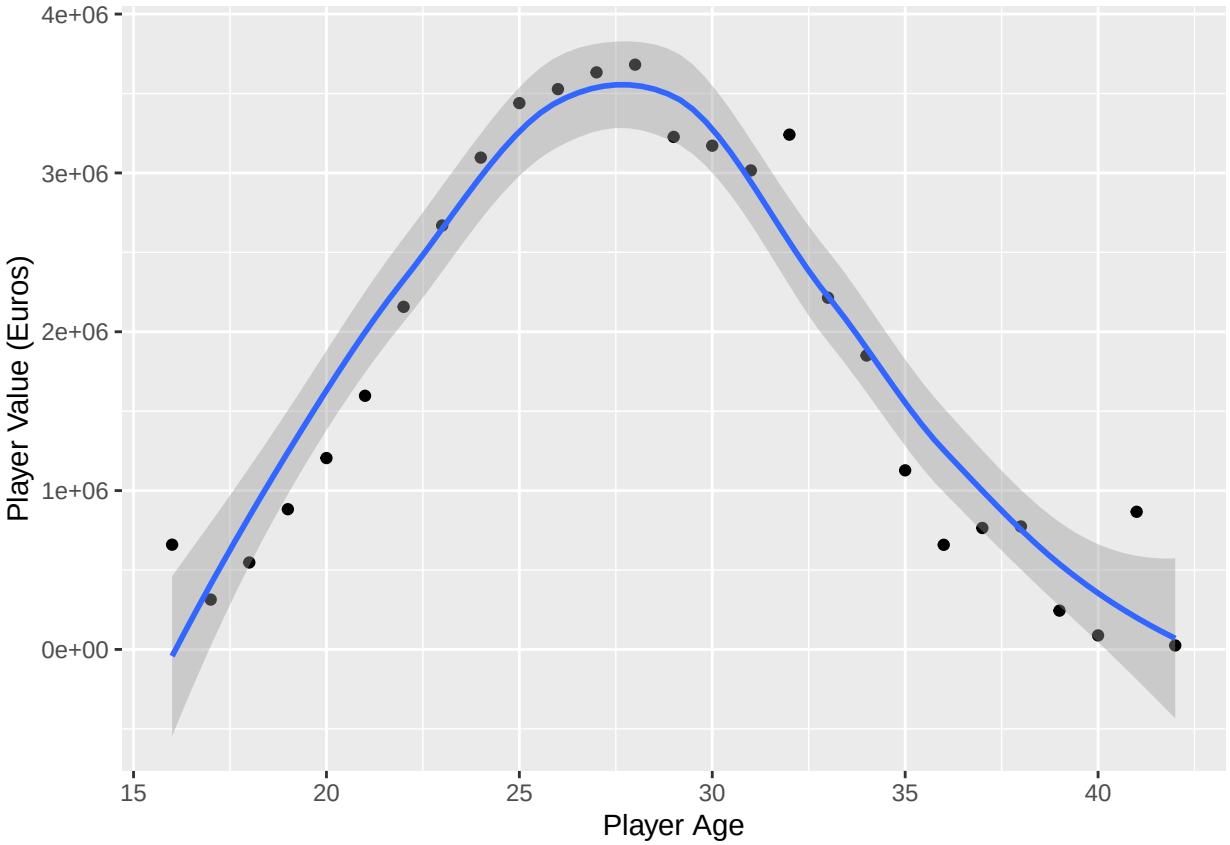
We now select the items we wish to continue with and we are ready to explore the relationships in our data. Let's take one final look at our clean data before we move on.

```
##   Player_ID      short_name value_eur age       Club         league nationality
## 1         1          Felipe   2100000  34       SPAL        Serie A      Brazil
## 2         2       G. Buffon   2600000  41   Juventus        Serie A       Italy
## 3         3 M. Stekelenburg    675000  36    Everton Premier League Netherlands
## 4         4    A. Wilbraham     90000  39   Rochdale          other     England
## 5         5      K. Ellison         0  40  Morecambe          other     England
## 6         6       Tarantini   2400000  35 Rio Ave FC          other    Portugal
##   strongest_position position_class squad_rank european_competition
## 1                 CB       Defender First_Team                none
## 2                 GK    Goal_Keeper      Bench     Champions League
## 3                 GK    Goal_Keeper   Reserves                none
## 4                 ST       Attacker      Bench                none
## 5                 LM     Midfielder First_Team                none
## 6                 CM     Midfielder First_Team                none
##   international_reputation yrs_left_on_contract
## 1                        2                    1
## 2                        4                    5
## 3                        2                    1
## 4                        1                    1
## 5                        1                    1
## 6                        1                    1
```

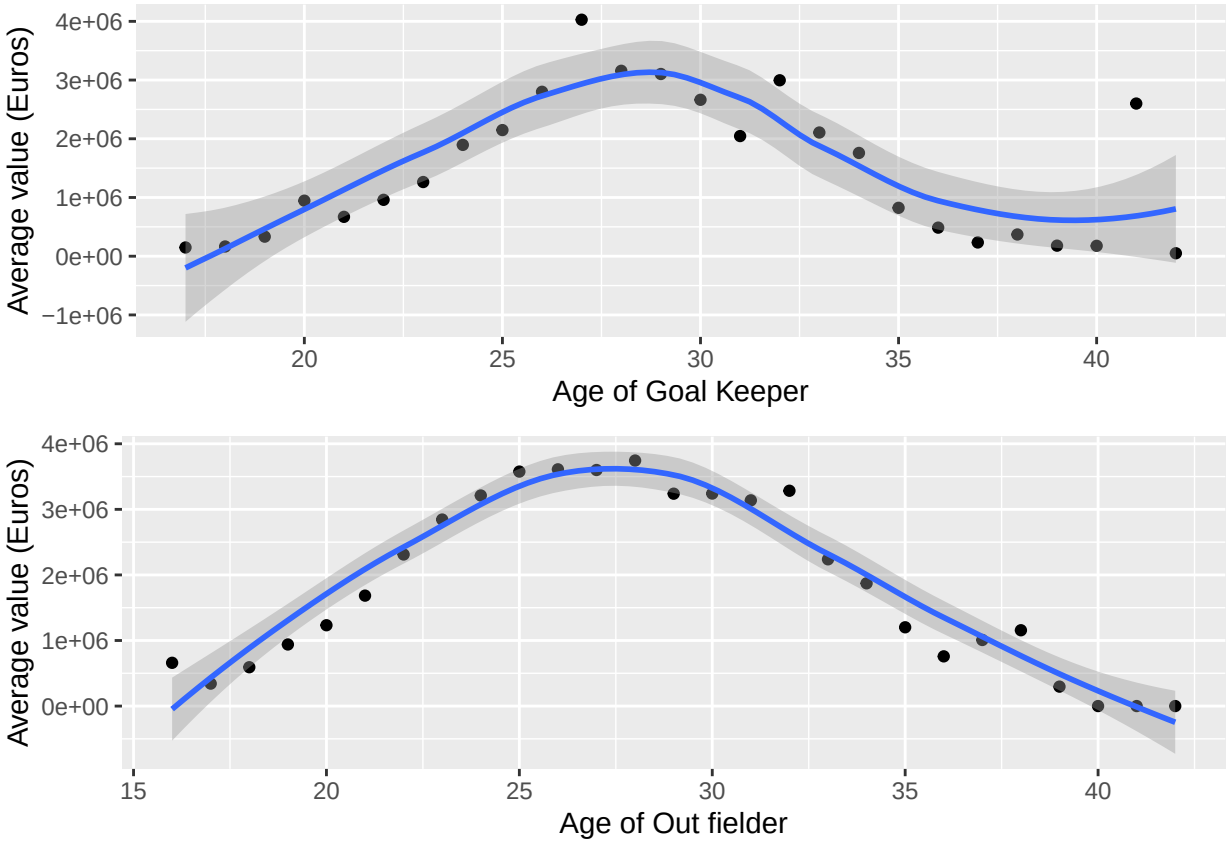
Visualisation

We can now begin to explore how player values behave with respect to other variables in our data set. We begin by creating a partition so that we only investigate our training set. We save our validation set for the final key step as to avoid overtraining.

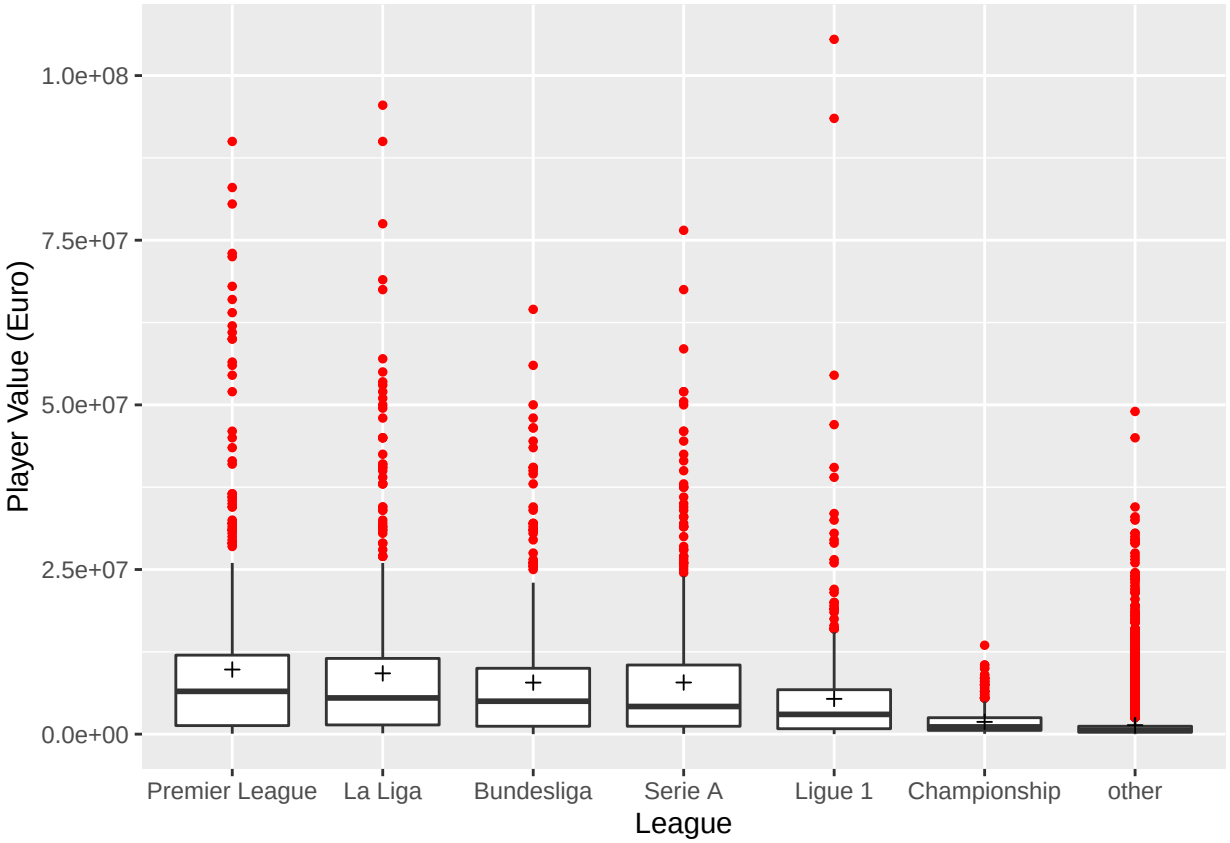
First we inspect the effect a players age has on their value.



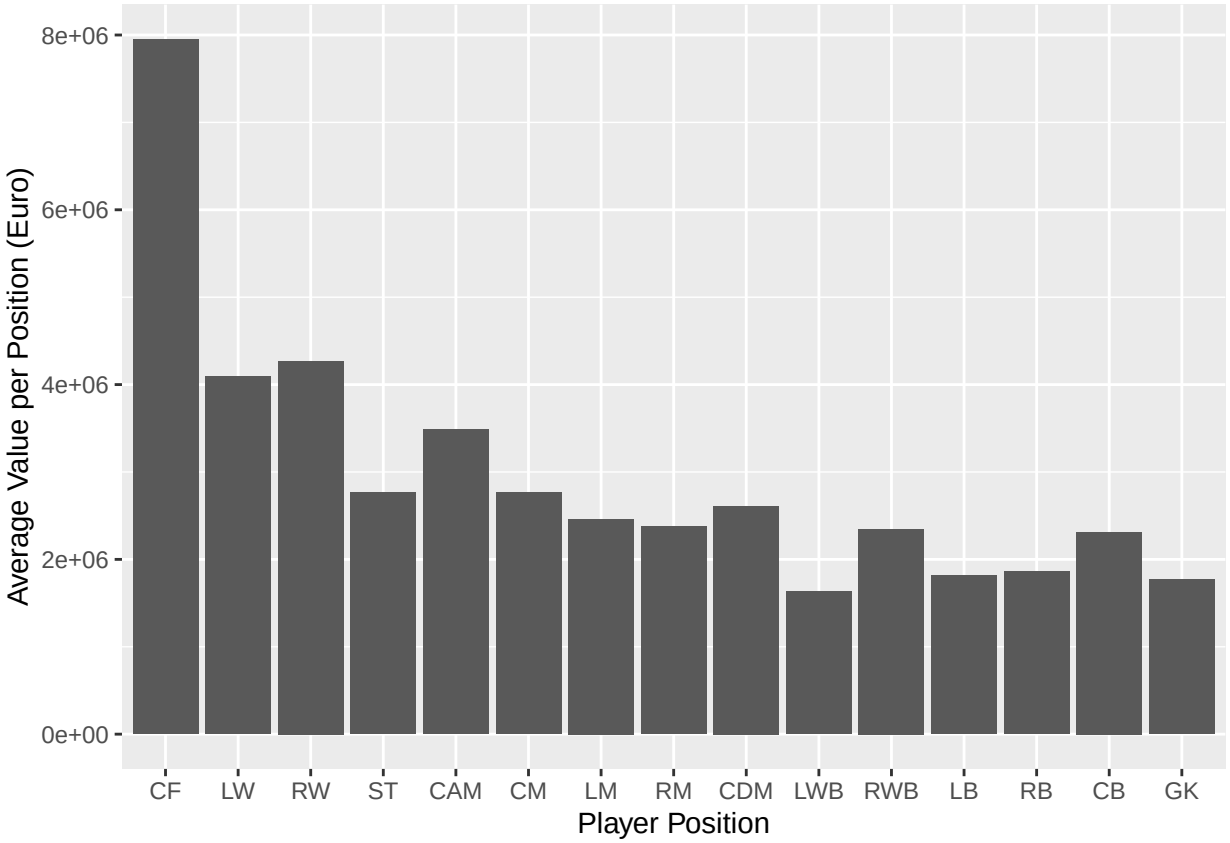
As we can see, there is a clear relationship between the two, with the players value peaking when the player is in their late twenties. It is worth considering how the players position can combine with a players age to create different effects. For example a goal keeper is considered to have a longer shelf life than an outfielder due to the lower physical demands of the role. This can be seen below.



Next we can consider how the league a player competes in can effect the market value of said player. Our focus is primarily on the top 5 European leagues and English leagues due to the increased sponsorship and promotion of these leagues. We can see from the box plot below how the elite leagues such as the Premier League and La Liga have a higher median, mean and overall range. We can also see that Ligue 1 has significantly smaller mean and interquartile ranges than the other elite European leagues despite having two of the highest value players. This is to be expected as the French League is considered by some to be less competitive. We can see that the category defined as other, despite having outliers with significant value, has a much lower value of player by comparison. Noteably the Championship has a slightly higher mean player value as we predicted due to the league being English.

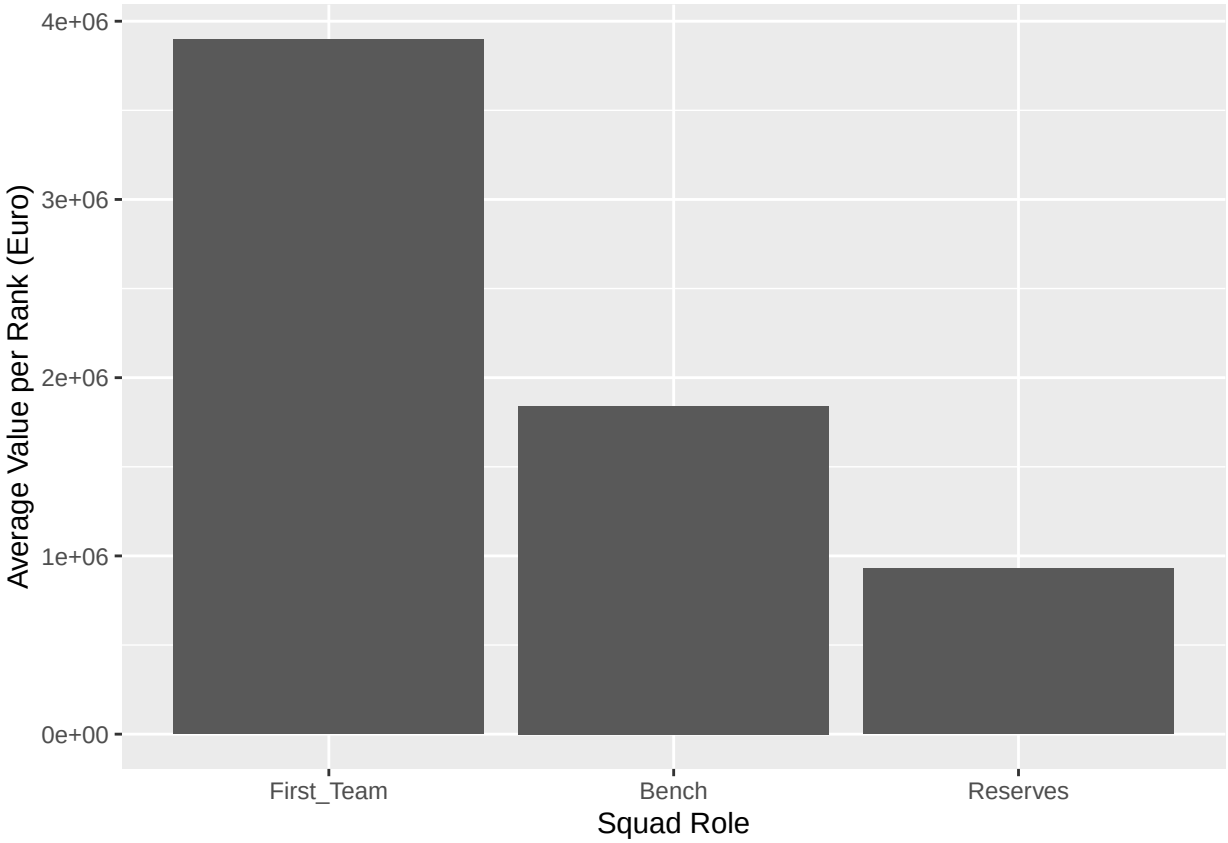


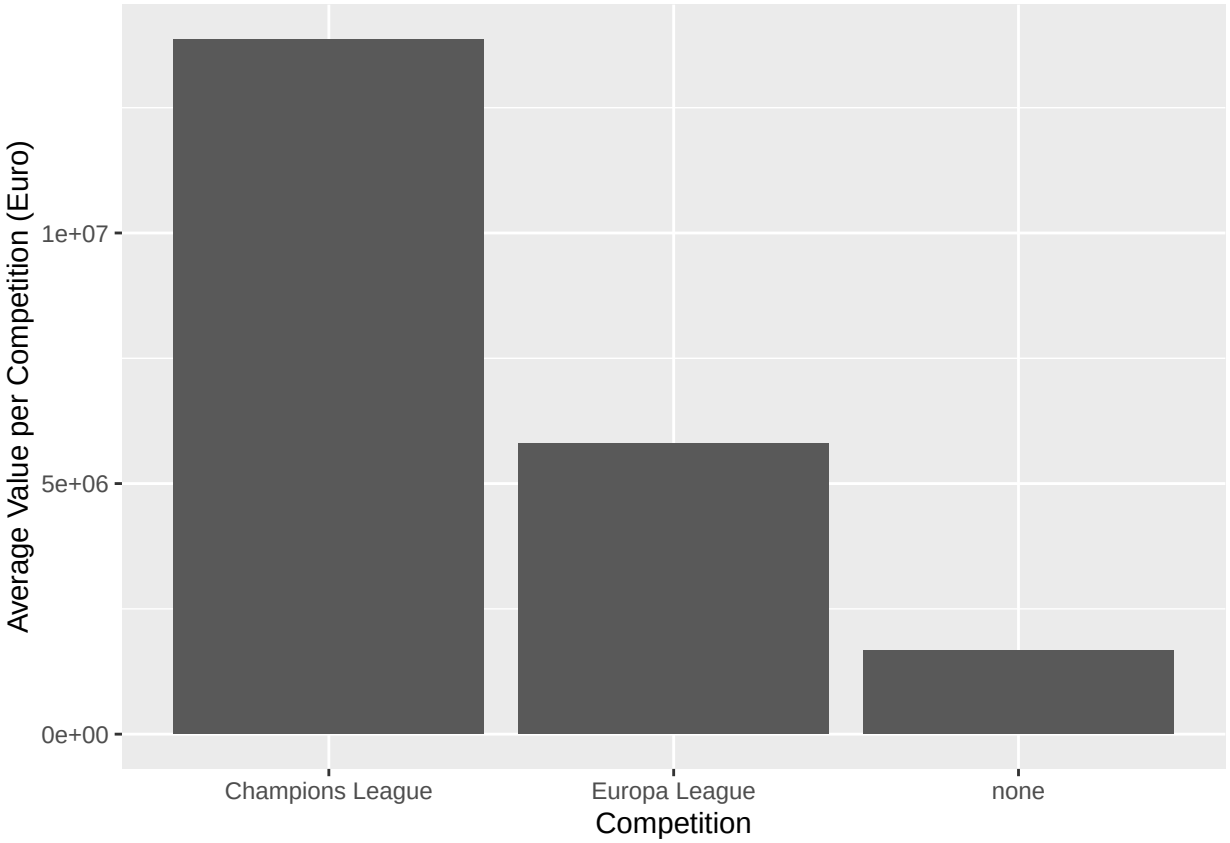
Another key feature to consider is the position a player takes on the pitch. With attacking players considered as the more marquee signing due to the added attraction to a clubs fan base. We would expect the attacking players to be of a higher value as they have other benefits off the field such as an increase in commercial revenue from selling player shirts and a greater turnout of fans at the stadium for matches. Fan are more likely to purchase shirts with the name of their favourite attacker than a defender.

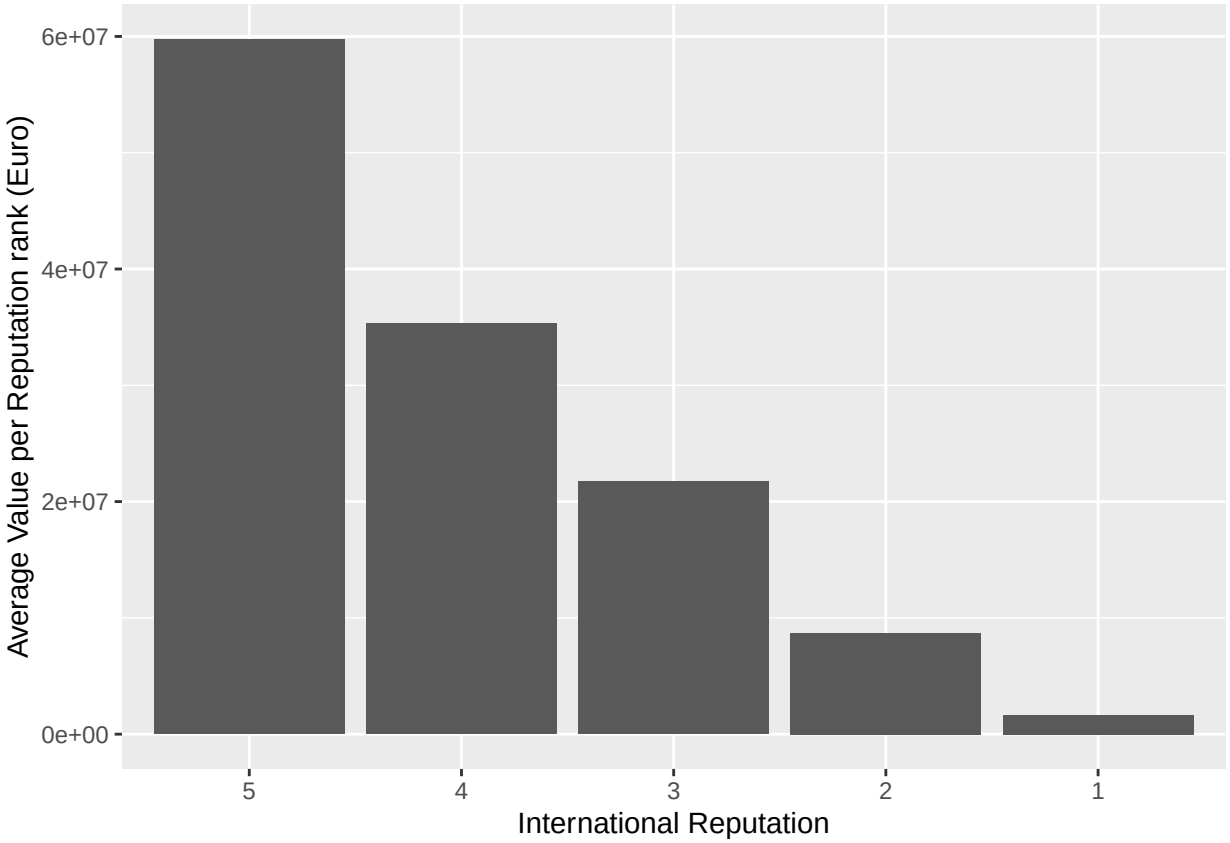


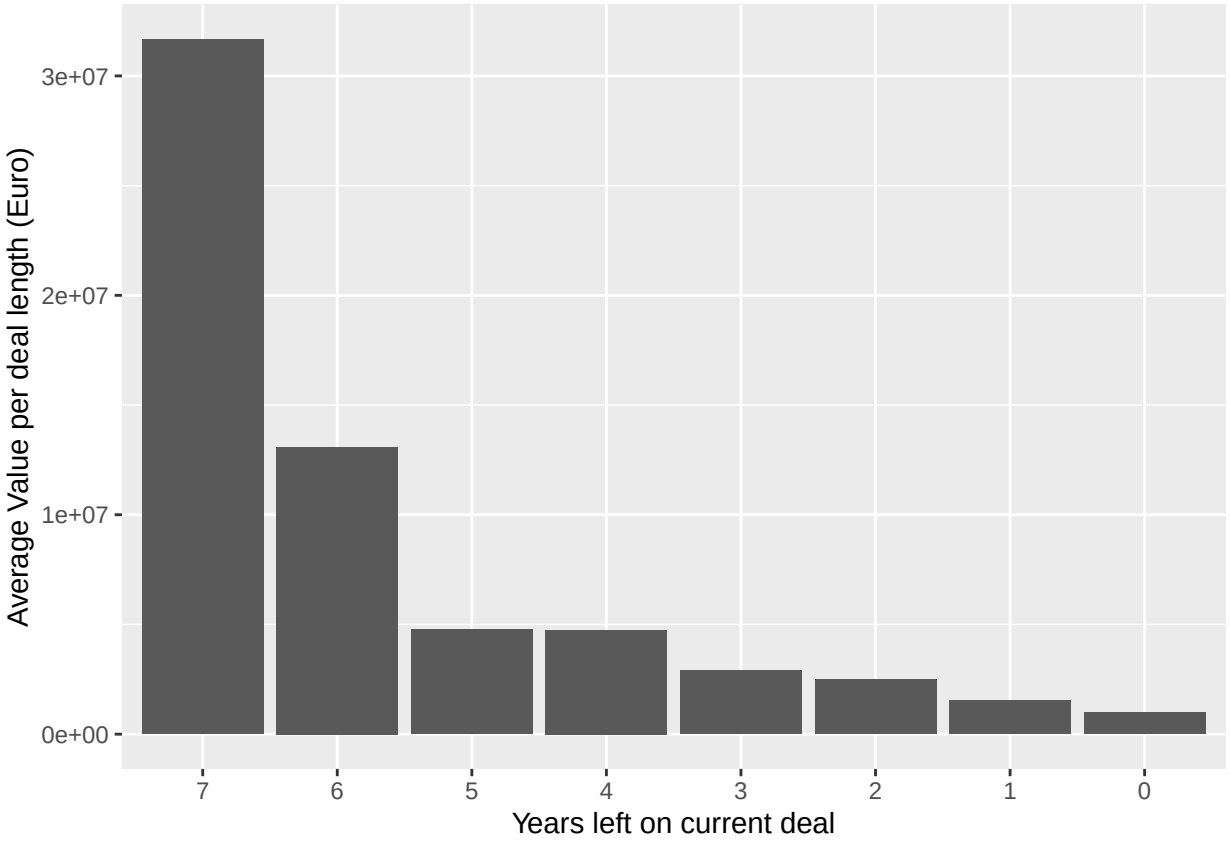
As expected, the CF position is of considerably higher value than other positions and there is a clear negative correlation between value and position as you move from the offensive to defensive positions. There are also noticeable spikes in central positions, as CF, CAM, CM, CDM, and CB are all slightly more valuable on average than those positions around them.

There are further key features which could possibly attribute to the differences in values between players. Such as how often a player actually plays for their club, how they perform for their country (or if they even play for their country at all), European club competitions participated in and how long they are contracted to their club for. Each of these have been explored and we can see the outcomes below.


4e+06
3e+06
2e+06
1e+06
0e+00
Average Value per Rank (Euro)
First_Team
Bench
Reserves
Squad Role


Average Value per Competition (Euro)
1e+07
5e+06
0e+00
Champions League
Europa League
none
Competition


6e+07
4e+07
2e+07
0e+00
Average Value per Reputation rank (Euro)
5
4
3
2
1
International Reputation



It is clear that there is an effect on value with each of these variables. Whether a player is chosen for the first team reguarly is obviously going to have an effect on the average value of players as a club will choose their best players to start most matches. It is also clear that the better and therefore more valuable players are likely to be playing in the more elite competitions such as the Champions League. Thus the effect of European club competitions is to be expected. The effect of international reputation which has been characterised in a five star system, so those players awarded five stars are considered the most reputable, is clear to see. Those which perform at the highest level on the international stage are much sought after and are therefore very highly valued. The final variable we discuss is the length of the contract a player is attached to. It would make sense that a club would like to have their best players sign long term contracts to keep them at the club. It is also note worthy that a player with little or no time left on their contract usually does not demand a high transfer fee, as a club selling such a player has little negotiating power in this scenario. They must either accept a smaller fee or risk the player running down the remainder of their contract and leaving the club for free further down the road. This is obviously a much less attractive outcome for the club.

Building the Model

Having explored and discussed the key variables, and the relationships to a players value we can now begin to build a model which utilizes these features to the best of its ability. It is worth noting first that we shall use the RMSE as our loss function to determine the accuracy of our models.

Root Mean Square Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}(\hat{y}_i - y_i)^2}$$

The root mean square error metric compares our predictions ($\hat{y}_i$) with our true values (y_i). By squaring the difference between them all values are positive and we can analyse the magnitude of all errors without them being cancelled out due to the varying signs. Thus the RMSE is always positive and values closer to zero indicate a better estimate.

```
## We define our RMSE function as
RMSE <- function(true_value, predicted_value){
  sqrt(mean((true_value - predicted_value)^2))
}
```

The Model

We begin once again by creating a partition in our training set to create a smaller training set and test set. We can now train and test models without using our validation set in order to avoid over training.

Our first attempt at a model to predict the players values is simply predicting the average of all players for every single player. We shall create a small table to add each model to in order to determine if the changes we a re making to our models are useful.

```
mu_hat <- mean(train$value_eur)
naive_rmse <- RMSE(test$value_eur, mu_hat)
rmse_results <- data.frame(method = "Just the average", RMSE = naive_rmse)
rmse_results
```

```
##             method    RMSE
## 1 Just the average 5788860
```

As we can see, the achieved RMSE is 5,788,860. When we consider the average player is valued at around 2.5 million Euros this is not a good enough prediction, as we would expect with such a basic model. Therefore we begin by creating a model based on assuming a linear relationship between the players value and the interesting variables. We define the effects each feature has by first defining the residuals of each variable. We then combine these to calculate a prediction generated by a linear combination of the effects of these residuals.

```
age_effect <- train %>% group_by(age) %>%
  summarise(b_age = mean(value_eur - mu_hat))

league_effect <- train %>%
  left_join(age_effect, by = "age") %>%
  group_by(league) %>%
  summarise(b_league = mean(value_eur - mu_hat - b_age))

position_effect <- train %>%
  left_join(age_effect, by = "age") %>%
  left_join(league_effect, by = "league") %>%
  group_by(strongest_position) %>%
  summarise(b_pos = mean(value_eur - mu_hat - b_age - b_league))

role_effect <- train %>%
  left_join(age_effect, by = "age") %>%
  left_join(league_effect, by = "league") %>%
  left_join(position_effect, by = "strongest_position") %>%
  group_by(squad_rank) %>%
```

```
  summarise(b_role = mean(value_eur - mu_hat - b_age - b_league - b_pos))

euro_effect <- train %>%
  left_join(age_effect, by = "age") %>%
  left_join(league_effect, by = "league") %>%
  left_join(position_effect, by = "strongest_position") %>%
  left_join(role_effect, by = "squad_rank") %>%
  group_by(european_competition) %>%
  summarise(b_euro = mean(value_eur - mu_hat - b_age - b_league - b_pos - b_role))

international_effect <- train %>%
  left_join(age_effect, by = "age") %>%
  left_join(league_effect, by = "league") %>%
  left_join(position_effect, by = "strongest_position") %>%
  left_join(role_effect, by = "squad_rank") %>%
  left_join(euro_effect, by = "european_competition") %>%
  group_by(international_reputation) %>%
  summarise(b_int_rep = mean(value_eur - mu_hat - b_age - b_league
                             - b_pos - b_role - b_euro))

contract_effect <- train %>%
  left_join(age_effect, by = "age") %>%
  left_join(league_effect, by = "league") %>%
  left_join(position_effect, by = "strongest_position") %>%
  left_join(role_effect, by = "squad_rank") %>%
  left_join(euro_effect, by = "european_competition") %>%
  left_join(international_effect, by = "international_reputation") %>%
  group_by(yrs_left_on_contract) %>%
  summarise(b_contract = mean(value_eur - mu_hat - b_age - b_league
                              - b_pos - b_role - b_euro - b_int_rep))


pred_values <- test %>%
  left_join(age_effect, by = "age") %>%
  left_join(league_effect, by = "league") %>%
  left_join(position_effect, by = "strongest_position") %>%
  left_join(role_effect, by = "squad_rank") %>%
  left_join(euro_effect, by = "european_competition") %>%
  left_join(international_effect, by = "international_reputation") %>%
  left_join(contract_effect, by = "yrs_left_on_contract") %>%
  mutate(pred = mu_hat + b_age + b_league + b_pos
         + b_role + b_euro + b_int_rep + b_contract) %>%
  .$pred
pred_values <- data.frame(pred_values) %>%
  mutate(adj = case_when(pred_values<= 0 ~ 0, pred_values > 0 ~ pred_values)) %>%
  .$adj
```

We find that some predictions fall outside the boundary of possibilty, that is to say that some players are predicted negative values from our model so we include the final line of code to adjust all negative predictions to zero. We can now observe the results from this model. We add the results to our table and notice the improvement on our previous prediction.

```
rmse_results <- bind_rows(rmse_results, data_frame(method="Effects Model",
                                                   RMSE = effects_model))
rmse_results
```

```
##             method    RMSE
## 1 Just the average 5788860
## 2    Effects Model 3819702
```

Our latest model achieves an RMSE of 3,819,702 which is a clear improvement on the previous estimate. However we still have plenty of room for improvement as our model is still rather basic. Our next step is to introduce regularization.

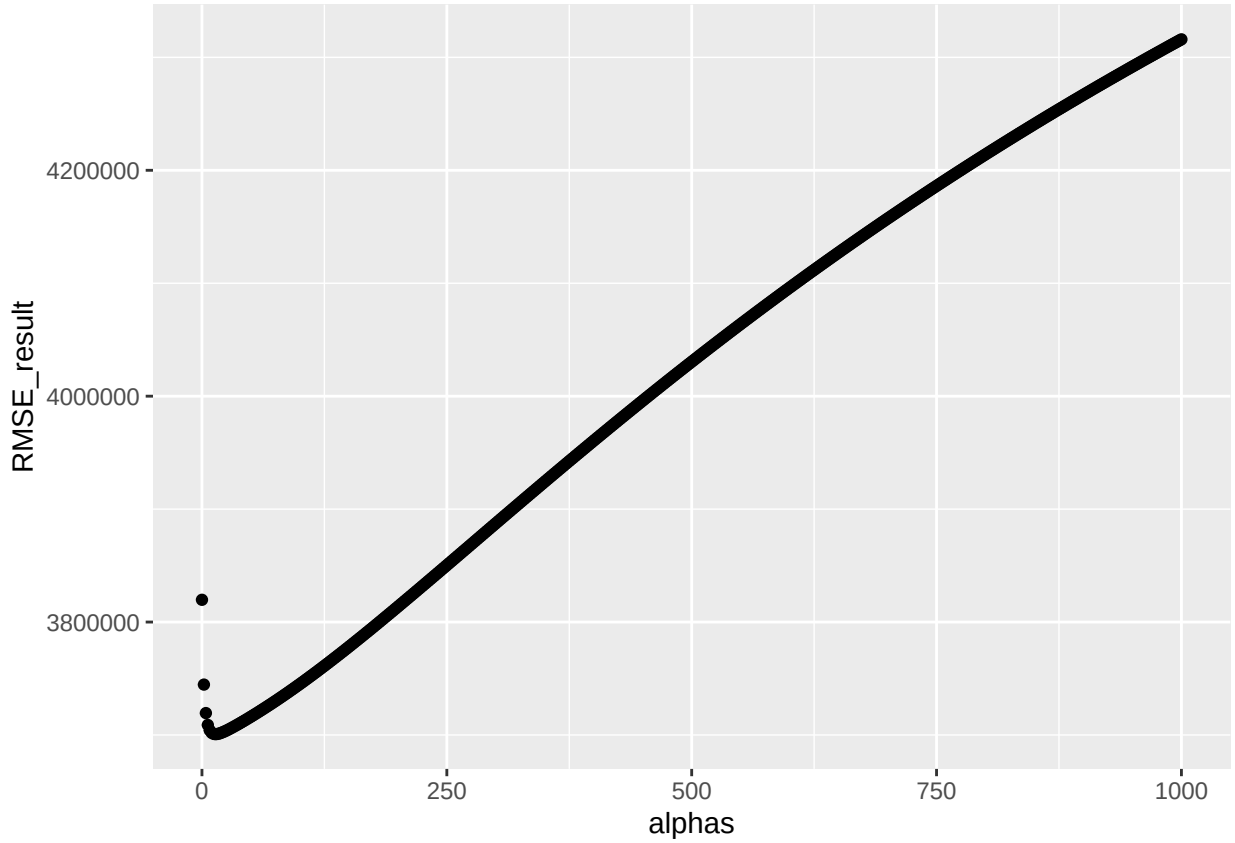
We use regularization to try to minimize the effect of noise in the data, and we use a tuning parameter to ensure we do not overfit. Our tuning parameter, α, will be selected by running a many α_j through our adjusted model from before, and finding the value which produces the lowest RMSE.

```
alphas <- seq(0, 1000, 2) # Here we define a vector
# of possible alpha values to be passed through the function.
RMSE_result <- sapply(alphas, function(A){
  mu = mean(train$value_eur)
  age_effect <- train %>% group_by(age) %>%
    summarise(b_age = sum(value_eur - mu)/(n() + A))
  league_effect <- train %>%
    left_join(age_effect, by = "age") %>%
    group_by(league) %>%
    summarise(b_league = sum(value_eur - mu - b_age)/(n() + A))
  position_effect <- train %>%
    left_join(age_effect, by = "age") %>%
    left_join(league_effect, by = "league") %>%
    group_by(strongest_position) %>%
    summarise(b_pos = sum(value_eur - mu - b_age - b_league)/(n() + A))
  role_effect <- train %>%
    left_join(age_effect, by = "age") %>%
    left_join(league_effect, by = "league") %>%
    left_join(position_effect, by = "strongest_position") %>%
    group_by(squad_rank) %>%
    summarise(b_role = sum(value_eur - mu - b_age - b_league - b_pos)/(n() + A))
  euro_effect <- train %>%
    left_join(age_effect, by = "age") %>%
    left_join(league_effect, by = "league") %>%
    left_join(position_effect, by = "strongest_position") %>%
    left_join(role_effect, by = "squad_rank") %>%
    group_by(european_competition) %>%
    summarise(b_euro = sum(value_eur - mu - b_age - b_league - b_pos
                           - b_role)/(n() + A))
  international_effect <- train %>%
    left_join(age_effect, by = "age") %>%
    left_join(league_effect, by = "league") %>%
    left_join(position_effect, by = "strongest_position") %>%
    left_join(role_effect, by = "squad_rank") %>%
    left_join(euro_effect, by = "european_competition") %>%
    group_by(international_reputation) %>%
    summarise(b_int_rep = sum(value_eur - mu - b_age - b_league - b_pos
```

```
                              - b_role - b_euro)/(n() + A))
  contract_effect <- train %>%
    left_join(age_effect, by = "age") %>%
    left_join(league_effect, by = "league") %>%
    left_join(position_effect, by = "strongest_position") %>%
    left_join(role_effect, by = "squad_rank") %>%
    left_join(euro_effect, by = "european_competition") %>%
    left_join(international_effect, by = "international_reputation") %>%
    group_by(yrs_left_on_contract) %>%
    summarise(b_contract = sum(value_eur - mu - b_age - b_league
                               - b_pos - b_role - b_euro - b_int_rep)/(n() + A))
  pred_values <- test %>%
    left_join(age_effect, by = "age") %>%
    left_join(league_effect, by = "league") %>%
    left_join(position_effect, by = "strongest_position") %>%
    left_join(role_effect, by = "squad_rank") %>%
    left_join(euro_effect, by = "european_competition") %>%
    left_join(international_effect, by = "international_reputation") %>%
    left_join(contract_effect, by = "yrs_left_on_contract") %>%
    mutate(pred = mu_hat + b_age + b_league + b_pos + b_role
           + b_euro + b_int_rep + b_contract) %>%
    .$pred
  pred_values <- data.frame(pred_values) %>%
    mutate(adj = case_when(pred_values<= 0 ~ 0, pred_values > 0 ~ pred_values)) %>%
    .$adj
  return(RMSE(test$value_eur, pred_values))
  ## Each alpha is passed through and the resulting RMSE is stored in each scenario.
})
```

From this code we are able to see how the RMSE is dependent on the value of the tuning parameter. We can visualize this and find the minimum using the following code.

```
qplot(alphas, RMSE_result)
```



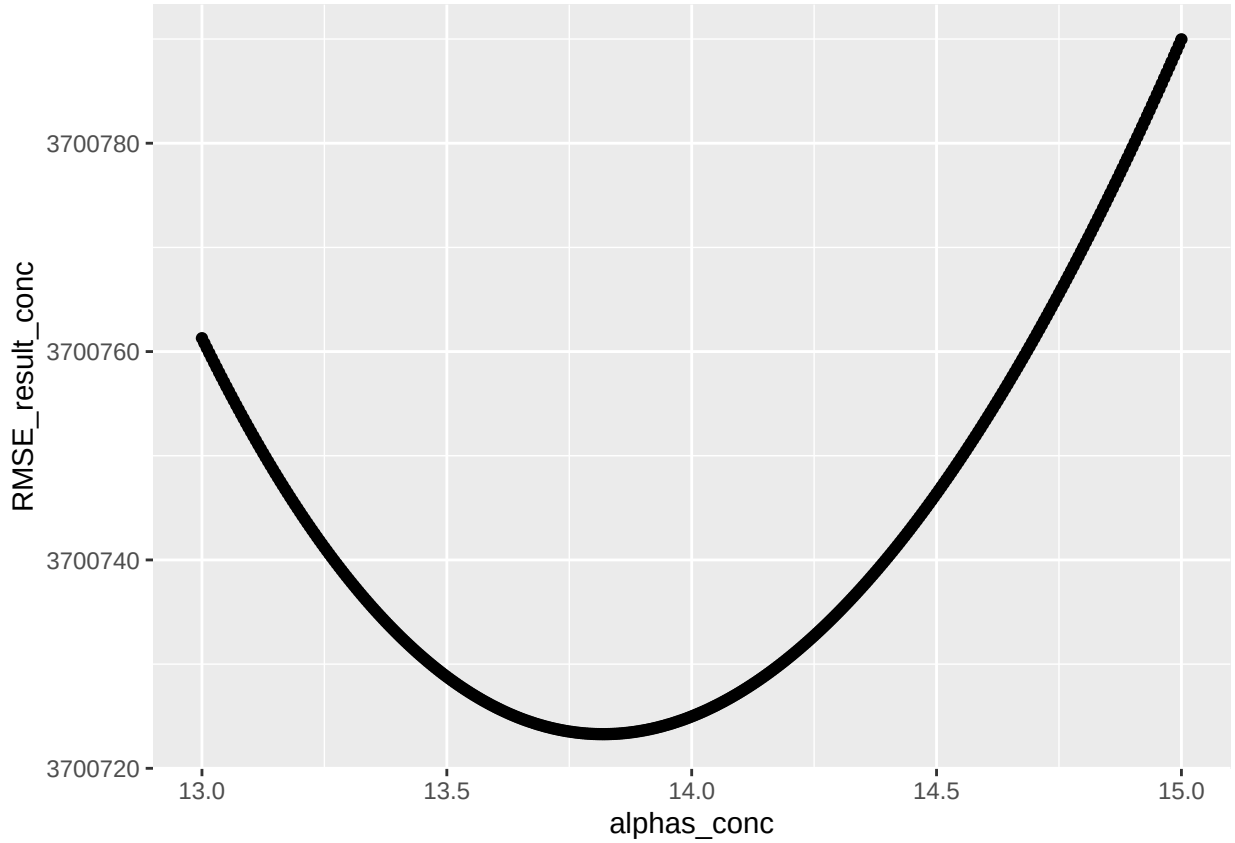

```
opt_alpha <- alphas[which.min(RMSE_result)]
opt_alpha
```

```
## [1] 14
```

Now we know that to the nearest even number, 14 gives the best RMSE. So now we can run the same function again with a much more refined range.

```
alphas_conc <- seq(13, 15, 0.005)
```

```
qplot(alphas_conc, RMSE_result_conc)
```




```
opt_alpha_conc <- alphas_conc[which.min(RMSE_result_conc)]
opt_alpha_conc
```

```
## [1] 13.82
```

We now see the optimal value of $\alpha = 13.82$. Thus we use this in our regularized model and see our obtained RMSE in our updated results table.

```
effect_with_reg <- RMSE_result_conc[which.min(RMSE_result_conc)]
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Effects Model with Regularization",
                                     RMSE = effect_with_reg))
rmse_results
```

```
##                               method    RMSE
## 1                  Just the average 5788860
## 2                     Effects Model 3819702
## 3 Effects Model with Regularization 3700723
```

We have made a very small improvement but it is still a step in the right direction. In order to obtain an even smaller RMSE we now make use of the regression packages.

First we attempt a linear regression model. This should not be too dissimilar in result to our previous regularized model. Each assumes a linear relationship between variables. We fit our model using the simple line of code seen below.

```
fit <- train %>% lm(value_eur ~ age + league + strongest_position + squad_rank +
                      european_competition + international_reputation +
                      yrs_left_on_contract, data = .)
```

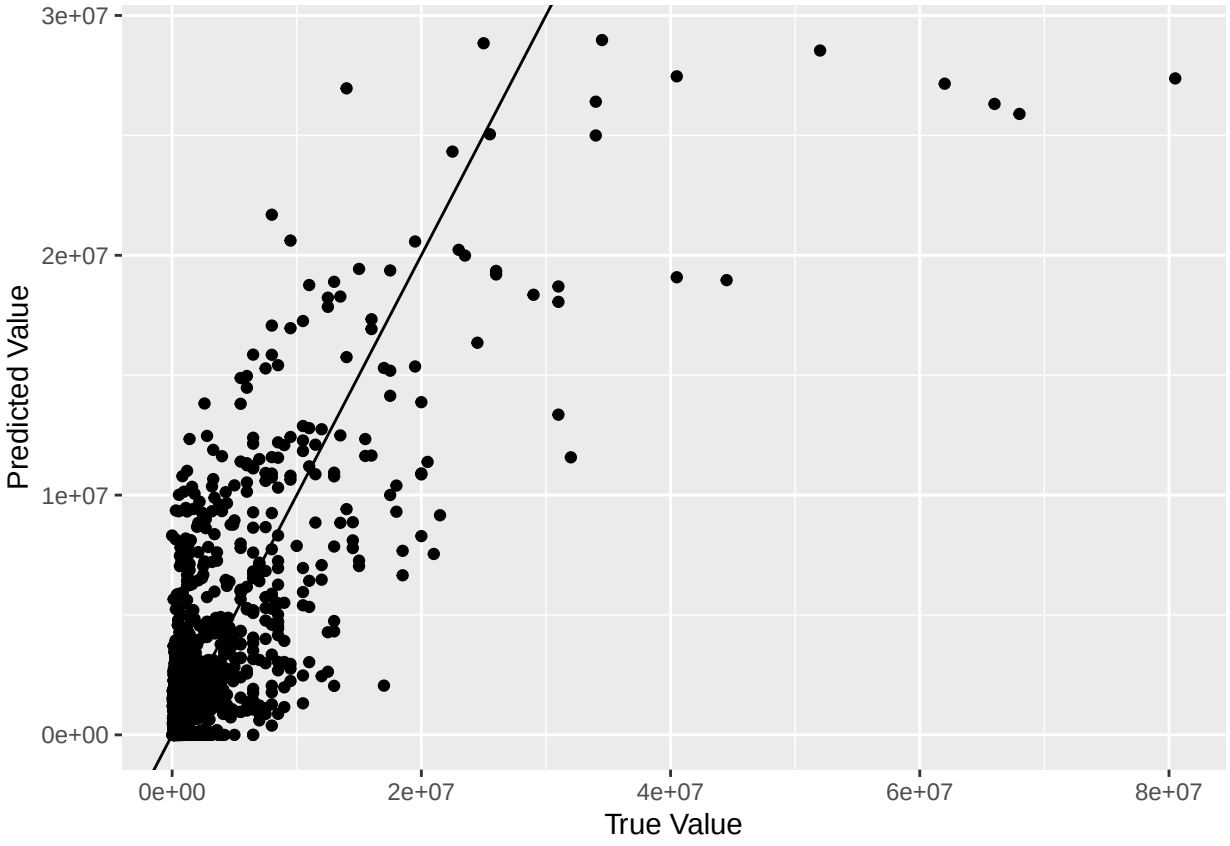
We are also able to investigate which features are not valuable in the modelling process. We can observe which variables have a p value greater than 0.01 and therefore we should reject the null hypothesis that the coefficient is equal to zero.

```
lm_coefs <- as.data.frame(tidy(fit))
lm_coefs$term[which(lm_coefs$p.value>0.01)]
```

Now we are ready to make predictions from our model and observe the resulting RMSE.

```
model_predictions <- test %>% predict(fit, newdata = .)
adj_model_predict <- case_when(model_predictions <= 0 ~ 0,
                               model_predictions > 0 ~ model_predictions)
linear_regression_model <- RMSE(test$value_eur, adj_model_predict)
```

We can investigate how our predictions are performing in great detail. We can see how each prediction performs against the related true value in the plot below.



It is clear that for lesser valued players, the model performs adequately but appears to undervalue the more premium players. We need to think of a way to combat this short coming. The RMSE of the model can be added to the table of results.

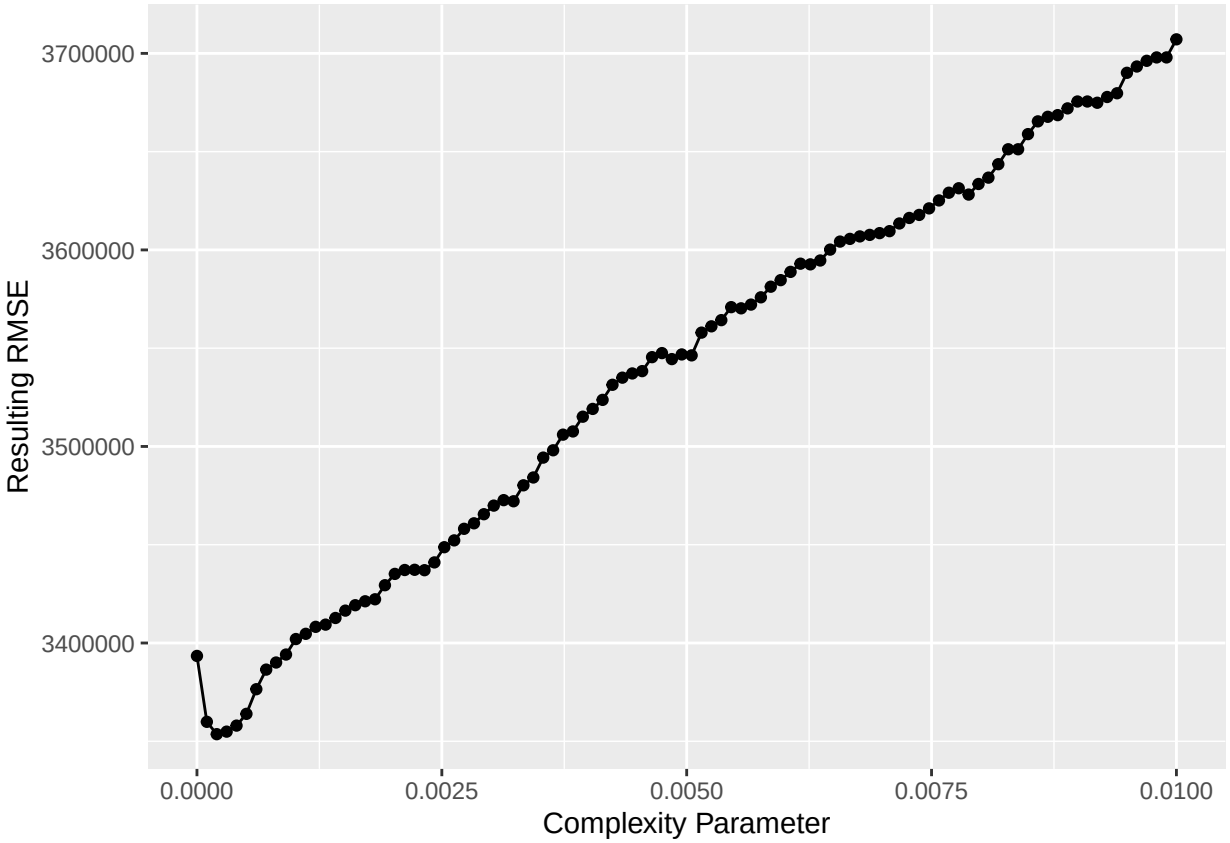
```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Linear Regression Model",
                                     RMSE = linear_regression_model))
rmse_results
```

```
##                              method    RMSE
## 1                  Just the average 5788860
## 2                     Effects Model 3819702
## 3 Effects Model with Regularization 3700723
## 4           Linear Regression Model 3700597
```

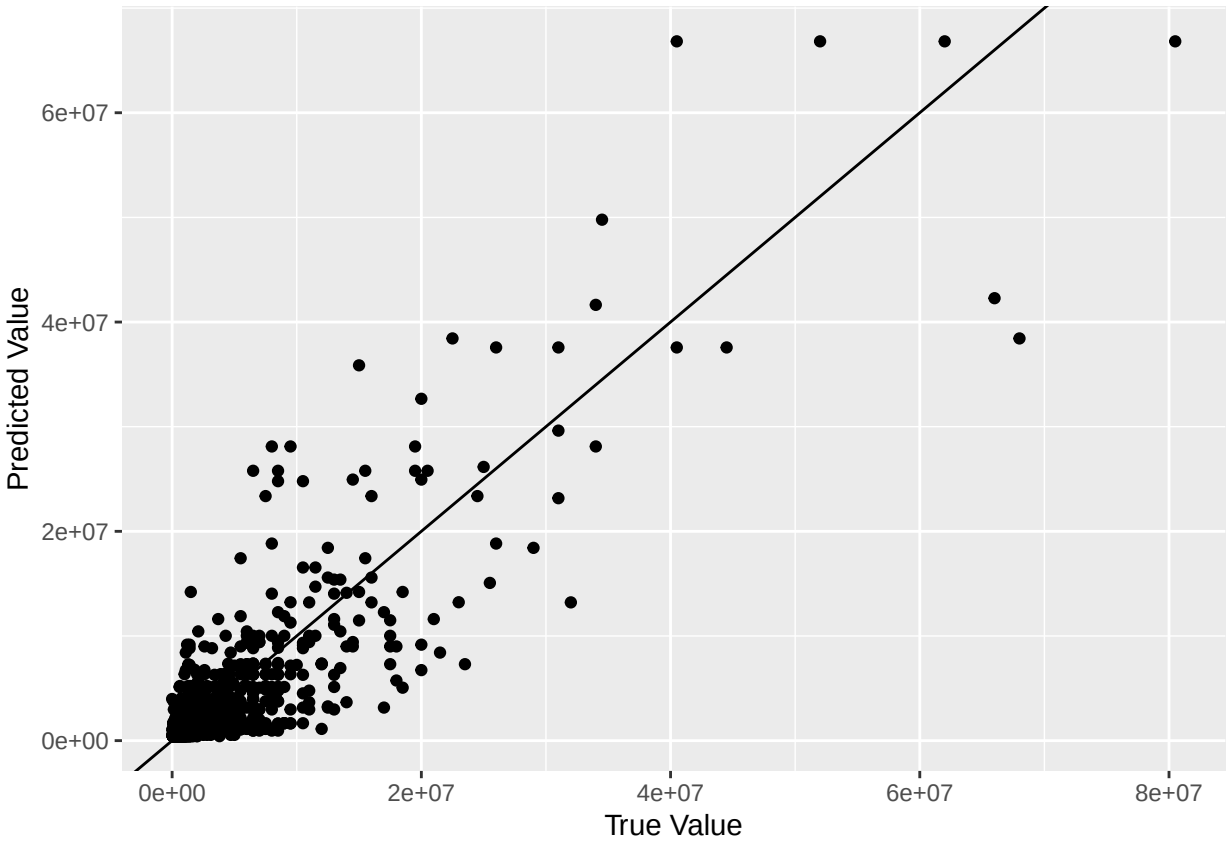
We observe a very similar result from previous models. This may suggest that a new strategy is required in order to obtain a further improvement on our models result. Therefore, for our next approach we shall utilize a regression tree model rather than a linear method.

Using the rpart package, we are able to construct a regression tree model with a prebuilt function in R. Our first task is to tune the model to our data. We need to obtain the optimal tuning parameter and to do that we use the train function with the tuneGrid input. Similar to previous techniques, this allows us to run a sequence of values through the function and select the value which relates to the best output. We save this value for later use.

```
RT_tune <- train(value_eur ~ age + league + strongest_position +
                   squad_rank + european_competition +
                   international_reputation + yrs_left_on_contract,
                           method = "rpart",
                 tuneGrid = data.frame(cp = seq(0, 0.01, len = 100)),
                 data = train)
RT_best_cp <- RT_tune$bestTune
```



We can now use the train function with method equal to rpart again but this time using rpart.control of our best value we saved earlier. We fit our model and can immediately make predictions using our smaller test set.



We can already see this model produces predictions much more consistent with the true values across the full range of player prices. This model also does not produce any negative predictions so we are not forced to adjust such values after the fact. If we take a look at the resulting RMSE we see another large improvement in the outcome of the predictions.

```
##                              method    RMSE
## 1                  Just the average 5788860
## 2                     Effects Model 3819702
## 3 Effects Model with Regularization 3700723
## 4           Linear Regression Model 3700597
## 5                    Regression Tree 3043593
```

Validation

We shall therefore use this modelling approach as our final model to be used on our validation set. We now train the model in the same way on our full training set, tuning to obtain the optimal complexity parameter and then fitting the model using this cp. We can then immediately move on to making predictions with this model with our chosen loss function, the RMSE. We save this result to add to our table.

```
FM_tune <- train(value_eur ~ age + league + strongest_position + squad_rank +
                   european_competition + international_reputation + yrs_left_on_contract,
                 method = "rpart", tuneGrid = data.frame(cp = seq(0, 0.01, len = 100)),
                 data = training_data)
FM_best_cp <- FM_tune$bestTune # Save our optimal choice of tuning parameter cp.
```

```r
## We fit our model using our chosen cp and relevant features using rpart.
FM_fit <- training_data %>% rpart(value_eur ~ age + league + strongest_position +
                                    squad_rank + european_competition +
                                    international_reputation + yrs_left_on_contract,
                            data =., control = rpart.control(cp = FM_best_cp))
FM_predict <- validation_set %>% predict(FM_fit, newdata = .) # We make our predictions.
FM_result <- RMSE(validation_set$value_eur, FM_predict) # Save the resulting RMSE
```

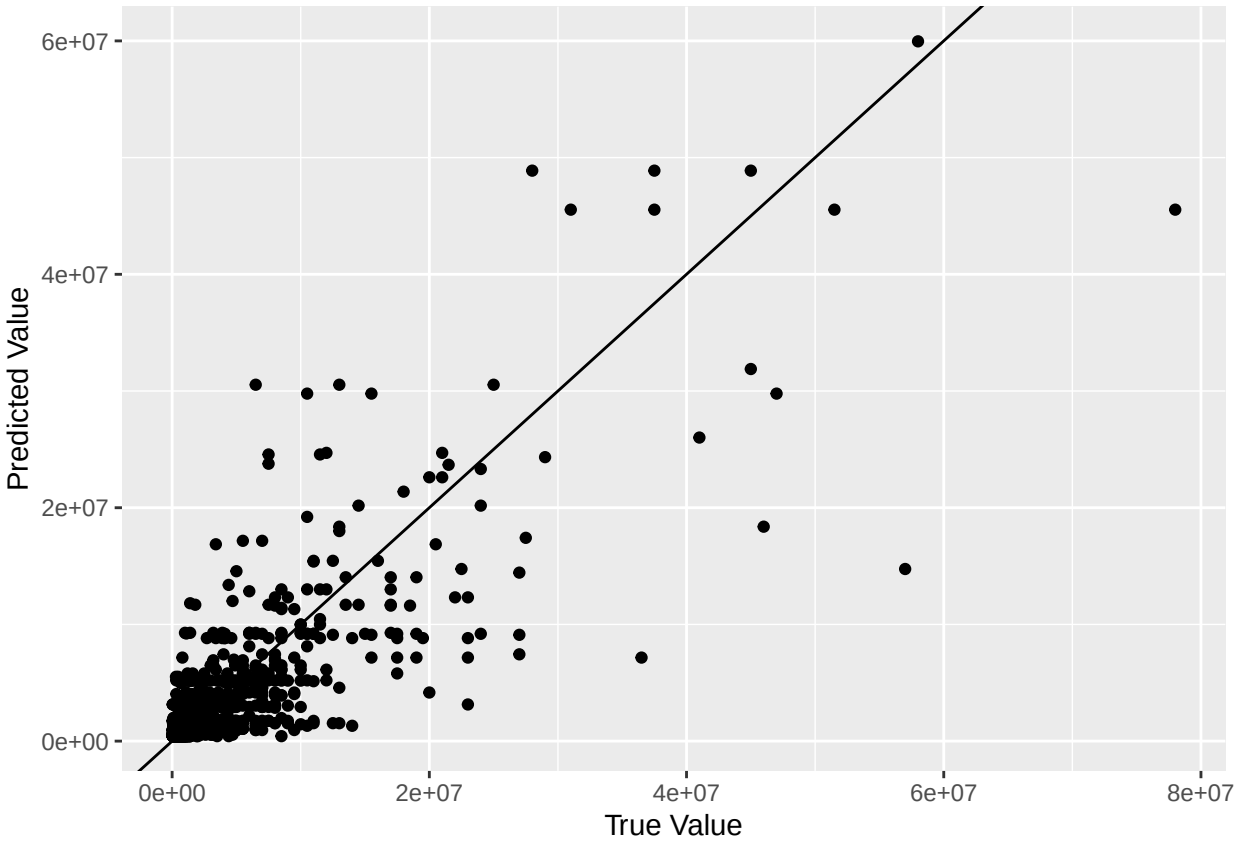
```
##                               method    RMSE
## 1                  Just the average 5788860
## 2                     Effects Model 3819702
## 3 Effects Model with Regularization 3700723
## 4           Linear Regression Model 3700597
## 5                    Regression Tree 3043593
## 6                       Final Model 3209748
```

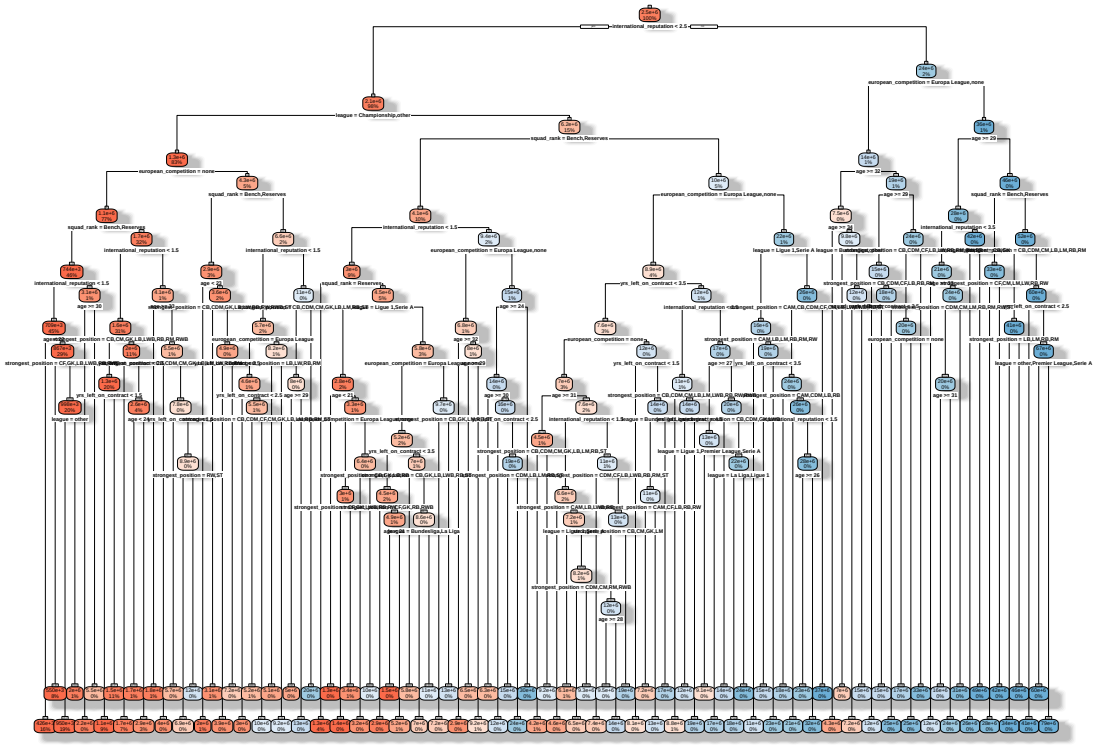
We can see that our final result is 3,209,748, meaning we typically have an error of magnitude 3,209,748 Euros when predicting the value of a player.

Results

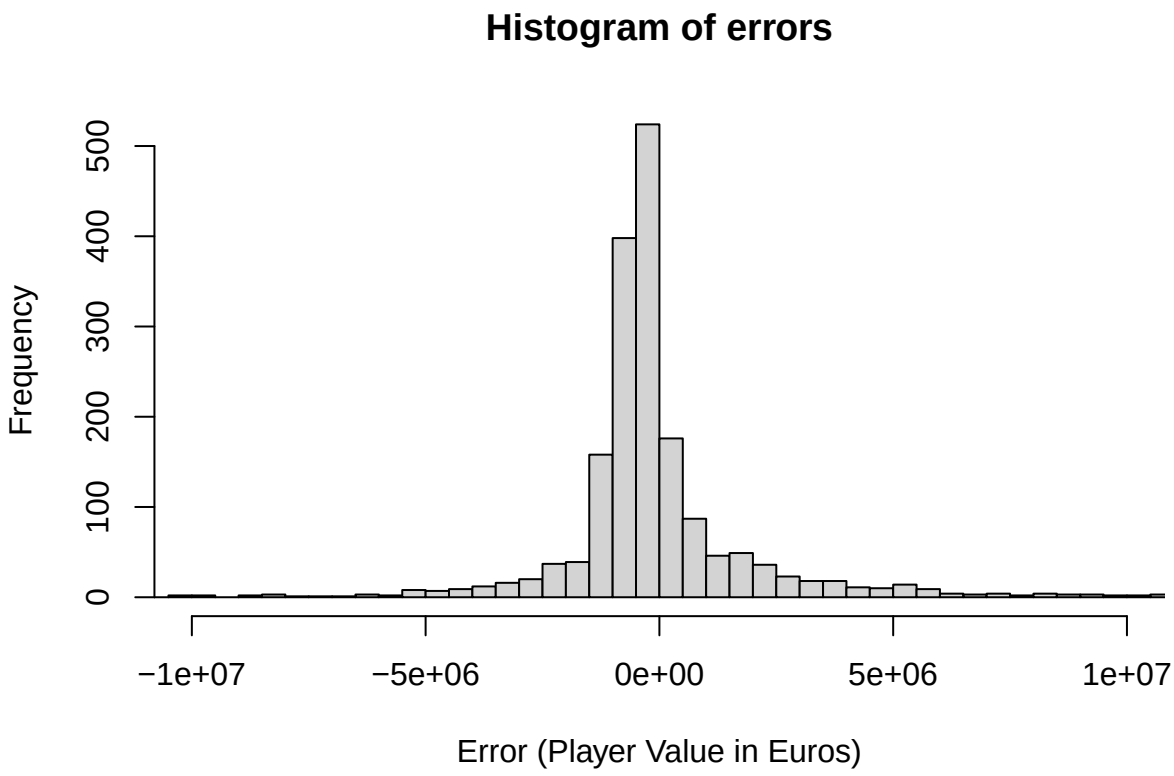
```
##                               method    RMSE
## 1                  Just the average 5788860
## 2                     Effects Model 3819702
## 3 Effects Model with Regularization 3700723
## 4           Linear Regression Model 3700597
## 5                    Regression Tree 3043593
## 6                       Final Model 3209748
```

Our most effective model was the regression tree approach, allowing us to achieve a final RMSE result of 3,209,748. We can take a closer look at how the model performed.


6e+07
4e+07
2e+07
0e+00
Predicted Value
0e+00
2e+07
4e+07
6e+07
8e+07
True Value



```
## international_reputation                 league      european_competition
##             1.841764e+17           7.851837e+16             7.437365e+16
##                      age             squad_rank       strongest_position
##             4.318633e+16           3.826137e+16             3.605090e+16
##     yrs_left_on_contract
##             2.550335e+16
```




```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -24038462   -810784   -337143    -18807    207074  42250000
```

```
sd(errors) ## Standard Deviation in errors:
```

```
## [1] 3210583
```

We can see from the tree plot just how many decisions were made in this model. Although there are too many to label each node and be able to read them, we can see from the list produced from the variable.importance function the first nodes are decisions based on international reputation, league then european competition. This is not that much of a surprise, we expect to see the better players performing in the biggest competitions with very few exceptions. We can also see from the histogra of errors in the predictions that they appear to approximate to a normal distribution which can be expected. They are centered around a mean of $-18,807$ which is very small, obviously we expect $\mu = 0$ so this is a good result. We also have $\sigma = 3210583$ which is very close to our RMSE.

There is strong evidence to say the model performed well agaisnt it's initial aim to form predictions of a players market value. Considering only variables which described the players non-performance based characteristics were used, it has successfully distinguished between higher and lower value players.

Conclusion

This project has seen us drastically improve our estimates of football player valuations from our initial naive method of using the mean average to developing a much more sophistcated regression tree with an optimal complexity parameter to obtain an RMSE of 3,209,748.

This model is of course limited by the fact that no variables were used which describe each players individual performance attributes. Although such a variable was available in the original data set which assigned each player and "overall" rating out of 100, this value is subjective and not calculated through analysis by experts of the game but rather through analysis by the game makers themselves. To further the model, I would have liked to use variables which describe for example, the number of games a player had participated in during the season, or the number of goals scored, number of passes attempted/ completed etc. Through these variables I might have been able to create my own basis for a player rating system to use in turn to value each player.