PGA Tour Analysis

2024-10-02

Source for the dataset: https://www.kaggle.com/datasets/jmpark746/pga-tour-data-2010-2018?resource=download (https://www.kaggle.com/datasets/jmpark746/pga-tour-data-2010-2018?resource=download)

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

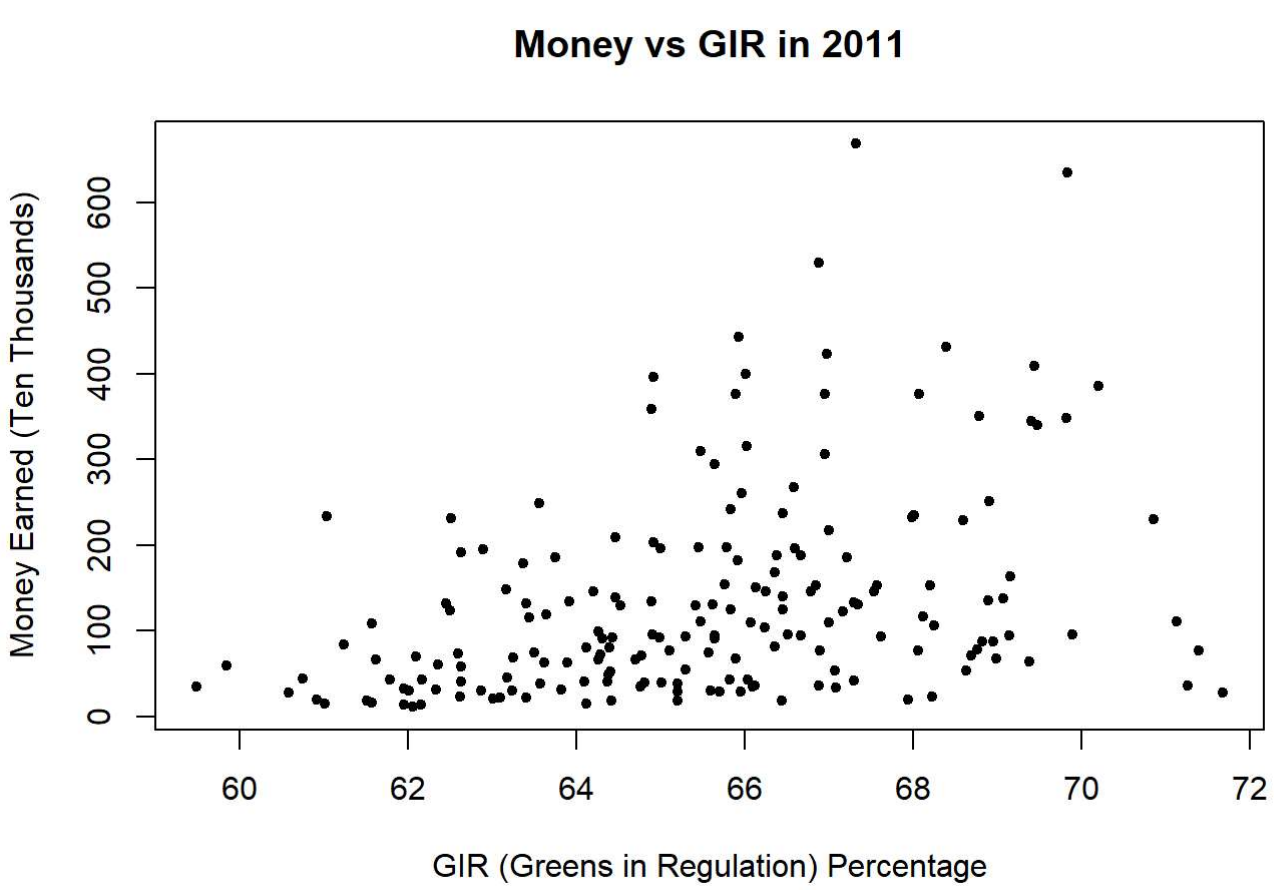
```
##
## Attaching package: 'ggplot2'
```

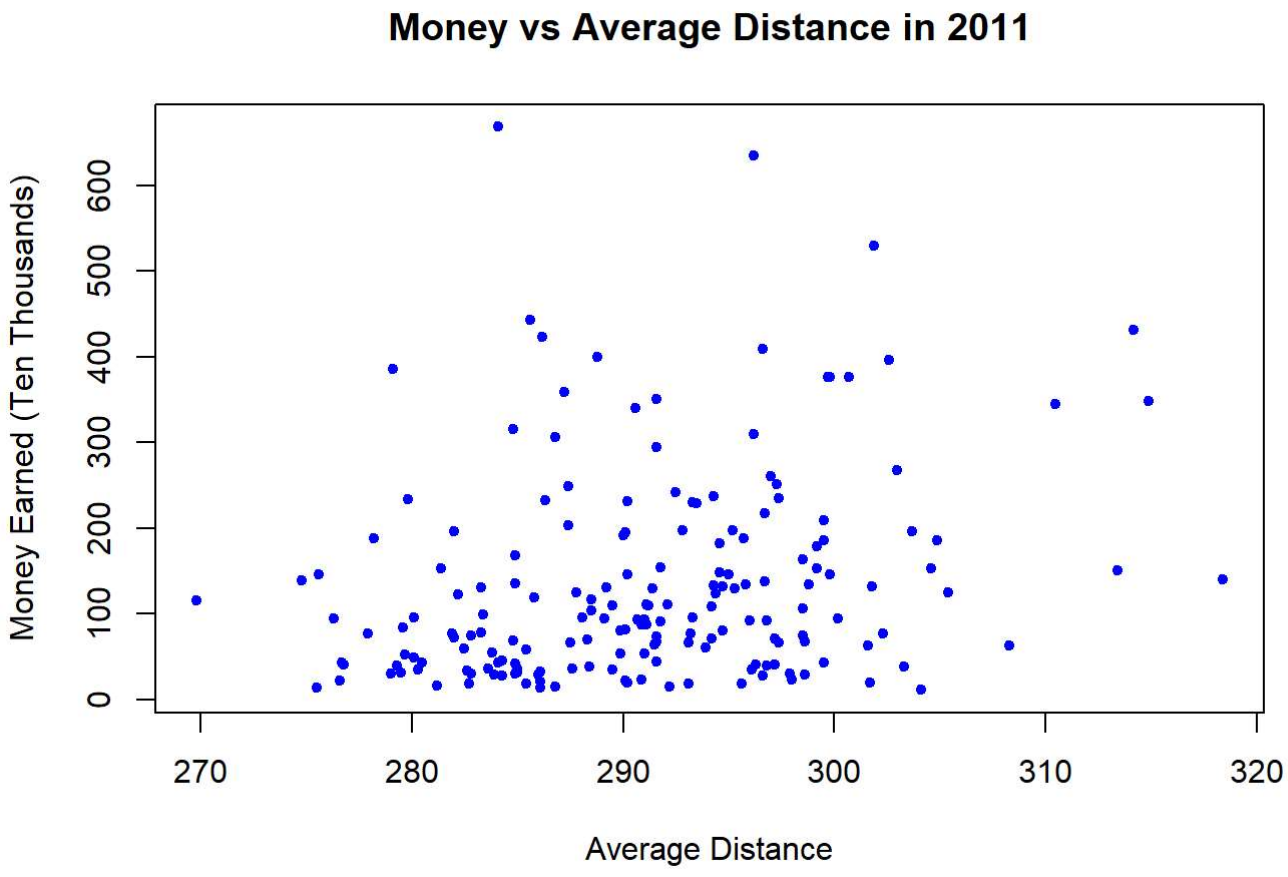
```
## The following object is masked from 'package:randomForest':
##
##     margin
```

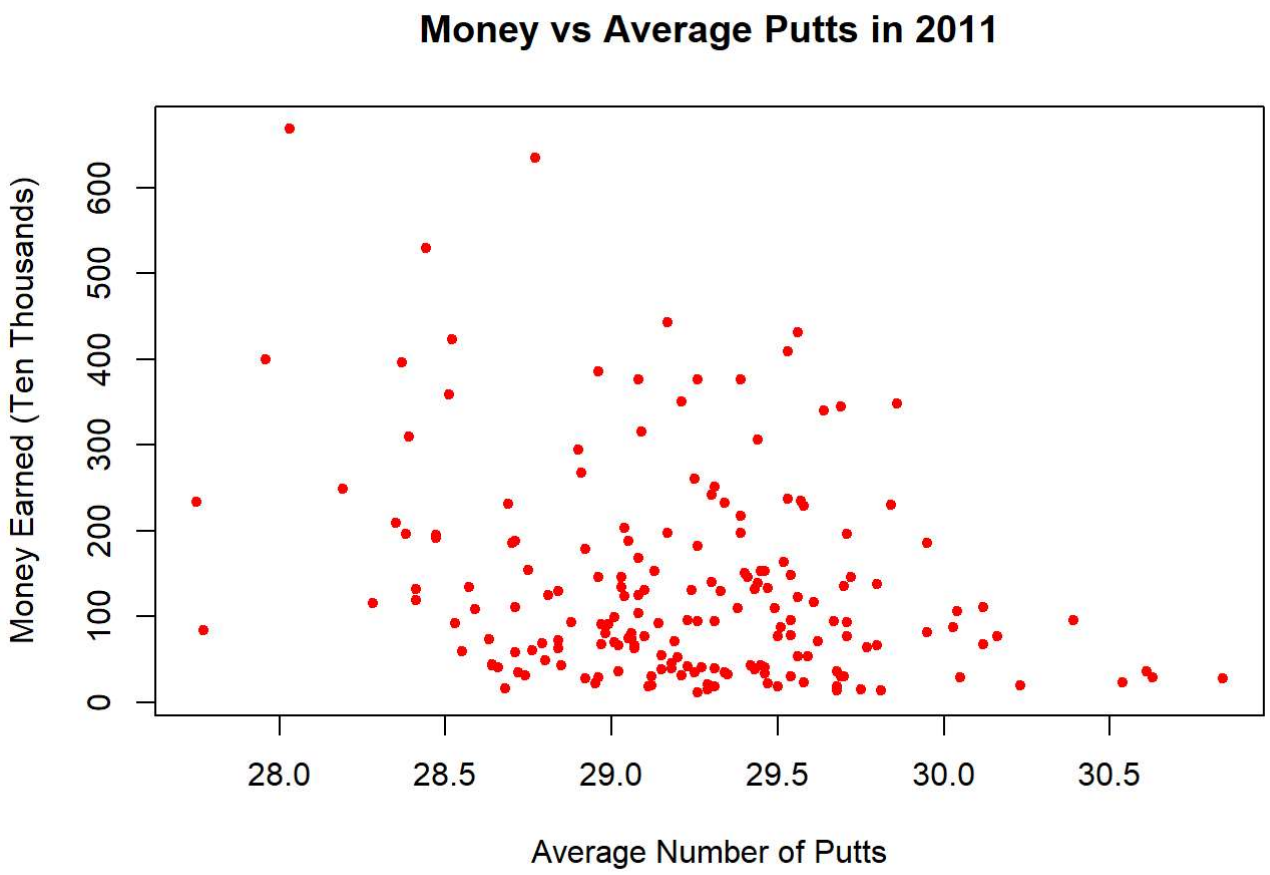
```
## Loading required package: lattice
```

```
library(groupdata2)
library(dplyr)


pga_tour = read.csv("C:/Users/ryne0/Downloads/pgaTourData.csv")
pga_tour
```

```
#Plot of money vs other variables thought to influence annual total prize winnings.
#Plot the data to examine possibe relationship between the variables.
plot_of_money_vs = function(pga_dataset, year){


  yearly_dataset = subset(pga_dataset, pga_dataset$Year == year)


  green_ir = yearly_dataset$gir
  money = yearly_dataset$Money
  plot(green_ir, money , pch = 20, xlab = "GIR (Greens in Regulation) Percentage", ylab = "Money
Earned (Ten Thousands)",  main = paste("Money vs GIR in", year))



  avg_distance = yearly_dataset$Avg.Distance
  plot(avg_distance, money , pch = 20, xlab = "Average Distance", ylab = "Money Earned (Ten Thou
sands)",  main = paste("Money vs Average Distance in", year), col="blue")


  avg_putts = yearly_dataset$Average.Putts
  plot(avg_putts, money , pch = 20, xlab = "Average Number of Putts", ylab = "Money Earned (Ten
Thousands)",  main = paste("Money vs Average Putts in", year), col="red")
}

plot_of_money_vs(pga_tour_cleaned, 2011)
```


Money vs GIR in 2011
Money Earned (Ten Thousands)
0
100
200
300
400
500
600
60
62
64
66
68
70
72
GIR (Greens in Regulation) Percentage


Money vs Average Distance in 2011
Money Earned (Ten Thousands)
0
100
200
300
400
500
600
270
280
290
300
310
320
Average Distance



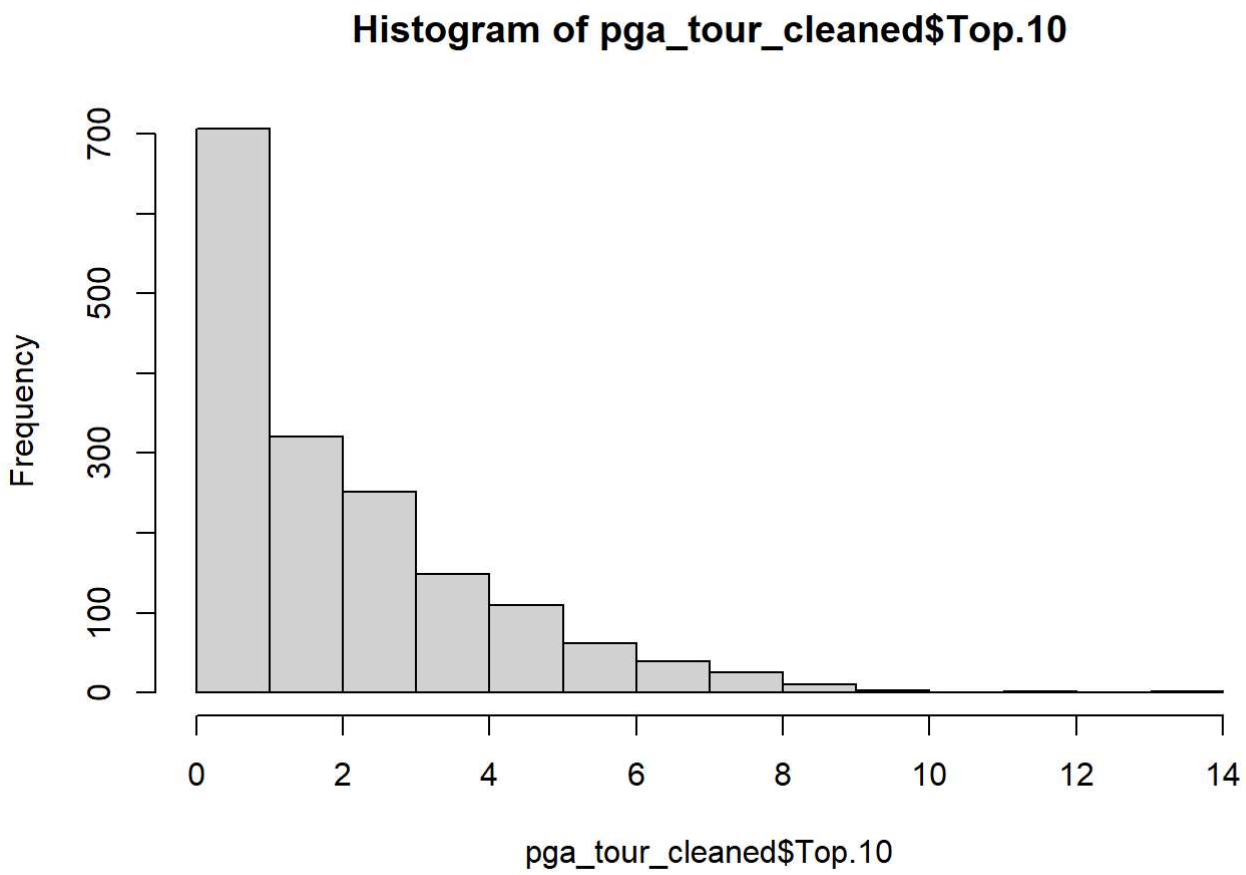
Can predict for how many top 10 wins a player has for the year. make it into a binomial or trinomial category

Preparing the dataset for machine learning

```
#Sorting the dataset
pga_tour_cleaned = pga_tour_cleaned[order(pga_tour_cleaned$Year, pga_tour_cleaned$Player.Name),
]



pga_tour_money_per_year = pga_tour_cleaned %>%
  group_by(Year) %>%
  summarise(money_per_year = sum(Money, na.rm = TRUE))
pga_tour_money_per_year
```

```
## # A tibble: 9 × 2
##    Year money_per_year
##   <int>          <dbl>
## 1  2010         24117.
## 2  2011         24898.
## 3  2012         25968.
## 4  2013         24252.
## 5  2014         27756.
## 6  2015         28477.
## 7  2016         28794.
## 8  2017         31104.
## 9  2018         33839.
```

```
#From the table we can see that total prize winnings increase over time. We will convert year in to a trinomial category to address this lurking variable. The categories will go from 2010-2013, 2014-2016, and 2017-2018


pga_tour_cleaned$Year[pga_tour_cleaned$Year<2014] = 0
pga_tour_cleaned$Year[pga_tour_cleaned$Year>2013 & pga_tour_cleaned$Year<2017] = 1
pga_tour_cleaned$Year[pga_tour_cleaned$Year>2016] = 2



#Separates the data into 80% training and 20% learning, and groups players together by name such that no player can be used for both training and testing
pga_tour_split = partition(pga_tour_cleaned, p = 0.8, cat_col = "Player.Name")
pga_tour_train = pga_tour_split[[1]]
pga_tour_test = pga_tour_split[[2]]



#From this histogram, it can be observed that roughly half of the players in the dataset did not finish in the top 10 for a given year. Because of this, conversion of top 10 into a binomial category is optional, but will not be performed to avoid loss of information.
hist(pga_tour_cleaned$Top.10)
```



rf_model_tuning_cv performs cross-validation on random forest

```
rf_model_tuning_cv = function(train_dataset, ntrees, predictor_columns, k_value, response_colum
n) {

  #Create k folds
  folds <- createFolds(train_dataset[[response_column]], k = k_value, list = TRUE)

  best_model_error = Inf

  for (i in 1:length(ntrees)) {
    model_errors = c()


    #Cross-validation loop
    for (j in 1:k_value) {

      #Split the data into training and validation
      train_fold = train_dataset[-folds[[j]], ]
      validation_fold = train_dataset[folds[[j]], ]


      rf_model = randomForest(y = train_fold[[response_column]], x = train_fold[,predictor_colum
ns], ntree = ntrees[i])


      #Predict on the validation fold
      predictions = predict(rf_model, validation_fold[, predictor_columns])


      model_error = mean((predictions - validation_fold[[response_column]])^2)
      model_errors = c(model_errors, model_error)
    }


    #Calculate the average cross-validation error
    final_model_error = mean(model_errors)

    if (final_model_error < best_model_error) {
      best_model_error = final_model_error
      best_model = rf_model
    }
  }

  return(best_model)
}
```

Running the model

```
set.seed(300)
rf_pga_model = rf_model_tuning_cv(pga_tour_train, c(125:150), c(3:17), 5, "Money")


#Getting the error of the model
tail(rf_pga_model$mse, 1)
```

```
## [1] 1936.001
```

```
tail(rf_pga_model$rsq, 1)
```

```
## [1] 0.913193
```

```
#Traditionally, a player's average shooting distance as well as performance on tee and approach shots

#We will remove any variable which has its IncNodePurity score being less than 1,000,000. We wil l rerun random forest to determine if the importance of our predictors has changed, and to see i f our model has improved by examining its r squared value and mean squared error value.



rf_pga_model$importance
```

```
##                    IncNodePurity
## Fairway.Percentage     128690.69
## Year                    48789.97
## Avg.Distance           315898.17
## gir                    193363.95
## Average.Putts          184509.56
## Average.Scrambling     150489.50
## Average.Score         3352927.26
## Points                7332206.64
## Wins                  1470284.65
## Top.10                2502727.40
## Average.SG.Putts       116040.42
## Average.SG.Total      3115465.83
## SG.OTT                 271363.93
## SG.APR                 285974.21
## SG.ARG                 132580.33
```

Rerunning the model with less predictors

```
rf_pga_model_new = rf_model_tuning_cv(pga_tour_train, c(125:150), c(9:12, 14), 5, "Money")

#We observe that the mean squared error (MSE) is higher and the r squared value is lower, implyi
ng that this model is worse than the previous one. But the previous model's values may indicate
that overfitting occurred. To determine which is the best model, we shall run them on our testin
g dataset.
tail(rf_pga_model_new$mse, 1)
```

```
## [1] 2118.121
```

```
tail(rf_pga_model_new$rsq, 1)
```

```
## [1] 0.907632
```

```
rf_pga_model_new$importance
```

```
##                   IncNodePurity
## Average.Score           3221439
## Points                  6055263
## Wins                    2856224
## Top.10                  3326944
## Average.SG.Total        3304608
```

Testing the models

```
set.seed(300)
rf_model_predictions = predict(rf_pga_model, pga_tour_test[,3:17])
rf_model_predictions[1:20]
```

```
##         1         2         3         4         5         6         7         8
##  75.95413  70.15600  36.49148 289.58959 135.13052 340.95253  51.83836 357.40959
##         9        10        11        12        13        14        15        16
## 207.40100 159.57651  50.47717 275.33863  82.91535  34.90160  20.38052  22.67785
##        17        18        19        20
##  78.32852  63.34395  47.57710  73.04586
```

```
pga_tour_test$Money[1:20]
```

```
##  [1]  72.1024  75.5356  21.7495 348.6407 167.6695 345.5012  47.4923 376.4797
##  [9] 169.5144 122.7196  61.8171 269.9150  63.6373  61.4658  16.3134  18.6847
## [17]  71.8507  61.7253  37.3980  56.3121
```

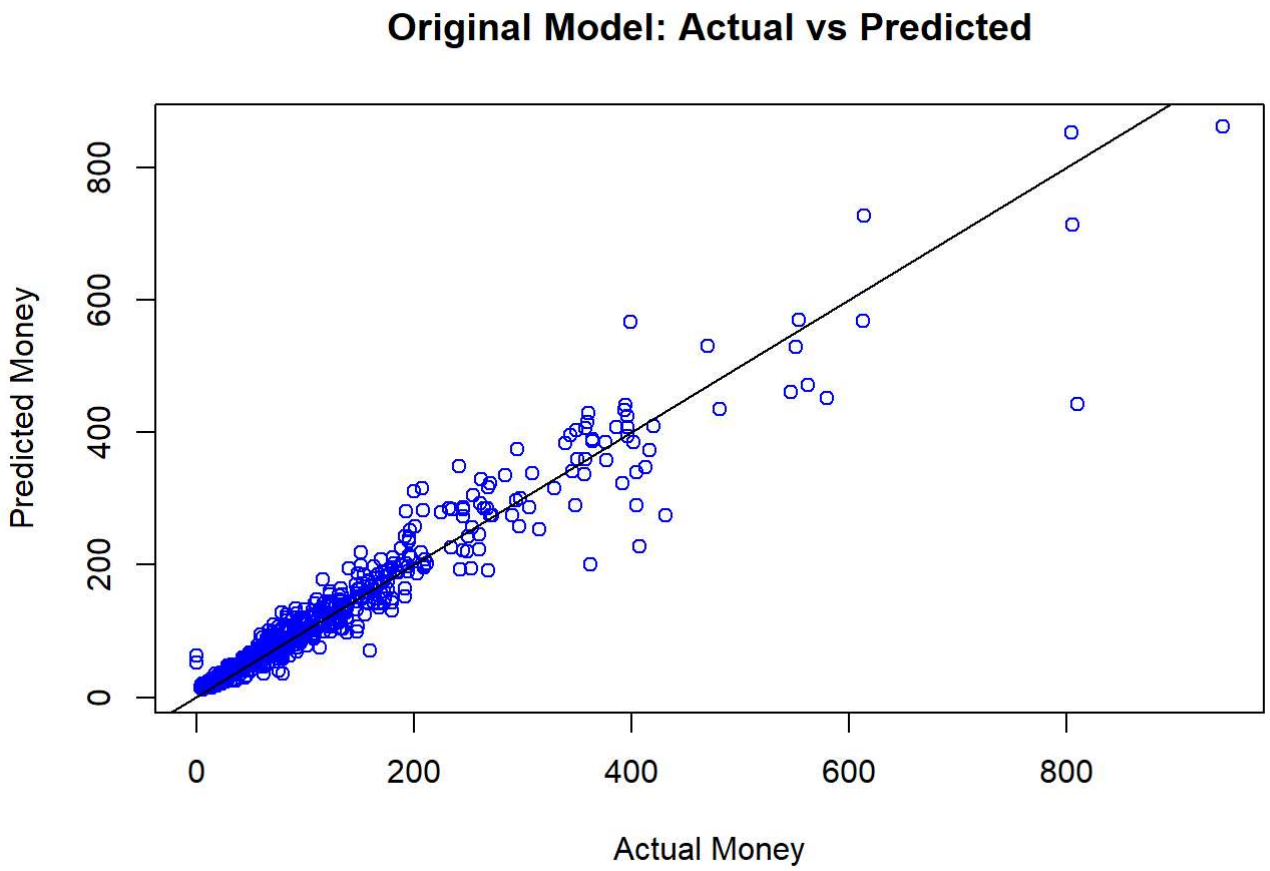
```
rf_model_new_predictions = predict(rf_pga_model_new, pga_tour_test[,3:17])
rf_model_new_predictions[1:20]
```

```
##         1         2         3         4         5         6         7         8
##  77.71674  76.73111  51.64390 279.71299 138.52980 322.90790  58.84001 346.19756
##         9        10        11        12        13        14        15        16
## 187.82760 169.48709  56.65670 251.10549  87.46649  45.26700  36.58783  39.45144
##        17        18        19        20
##  87.23881  72.70829  60.12583  68.38689
```

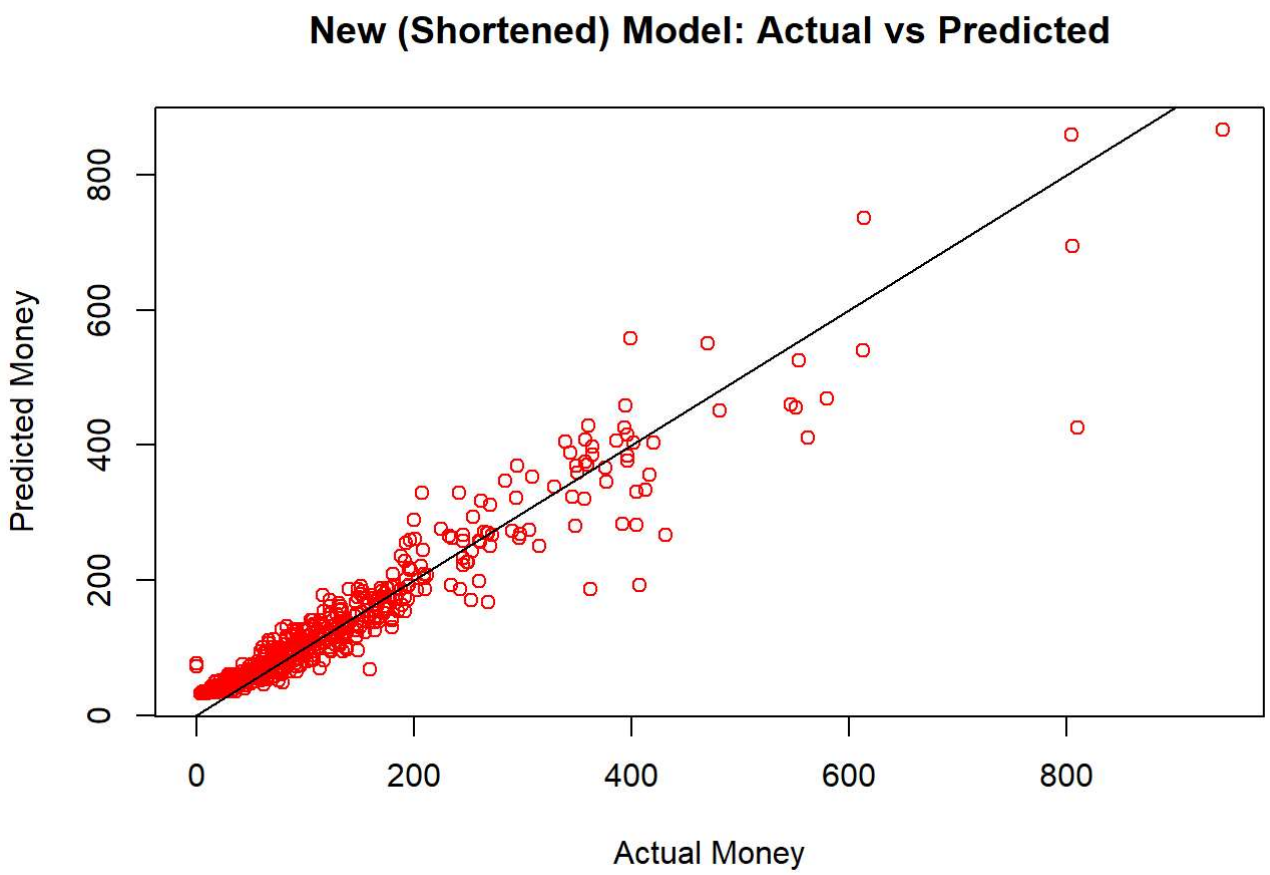
```
pga_tour_test$Money[1:20]
```

```
##  [1]  72.1024  75.5356  21.7495 348.6407 167.6695 345.5012  47.4923 376.4797
##  [9] 169.5144 122.7196  61.8171 269.9150  63.6373  61.4658  16.3134  18.6847
## [17]  71.8507  61.7253  37.3980  56.3121
```

```
#Below are plots of the actual money earned against the predicted money earned. The model with t
he most points closer to the line is the better one.

plot(pga_tour_test$Money, rf_model_predictions, main="Original Model: Actual vs Predicted", xlab
="Actual Money", ylab="Predicted Money", col="blue")
abline(0, 1)
```



```
plot(pga_tour_test$Money, rf_model_new_predictions, main="New (Shortened) Model: Actual vs Predi
cted", xlab="Actual Money", ylab="Predicted Money", col="red")
abline(0, 1)
```




```
#From the plots, it is difficult to tell which is the better model. The following metrics will h
elp us decide.



mse_model = mean((pga_tour_test$Money - rf_model_predictions)^2)
mse_new_model = mean((pga_tour_test$Money - rf_model_new_predictions)^2)
mse_model; mse_new_model
```

```
## [1] 1106.939
```

```
## [1] 1335.548
```

```
ss_total = sum((pga_tour_test$Money - mean(pga_tour_test$Money))^2)
ss_res_model = sum((pga_tour_test$Money - rf_model_predictions)^2)
ss_res_new_model = sum((pga_tour_test$Money - rf_model_new_predictions)^2)

r2_model = 1 - (ss_res_model / ss_total)
r2_new_model = 1 - (ss_res_new_model / ss_total)
r2_model; r2_new_model
```

```
## [1] 0.9291634
```

```
## [1] 0.9145341
```

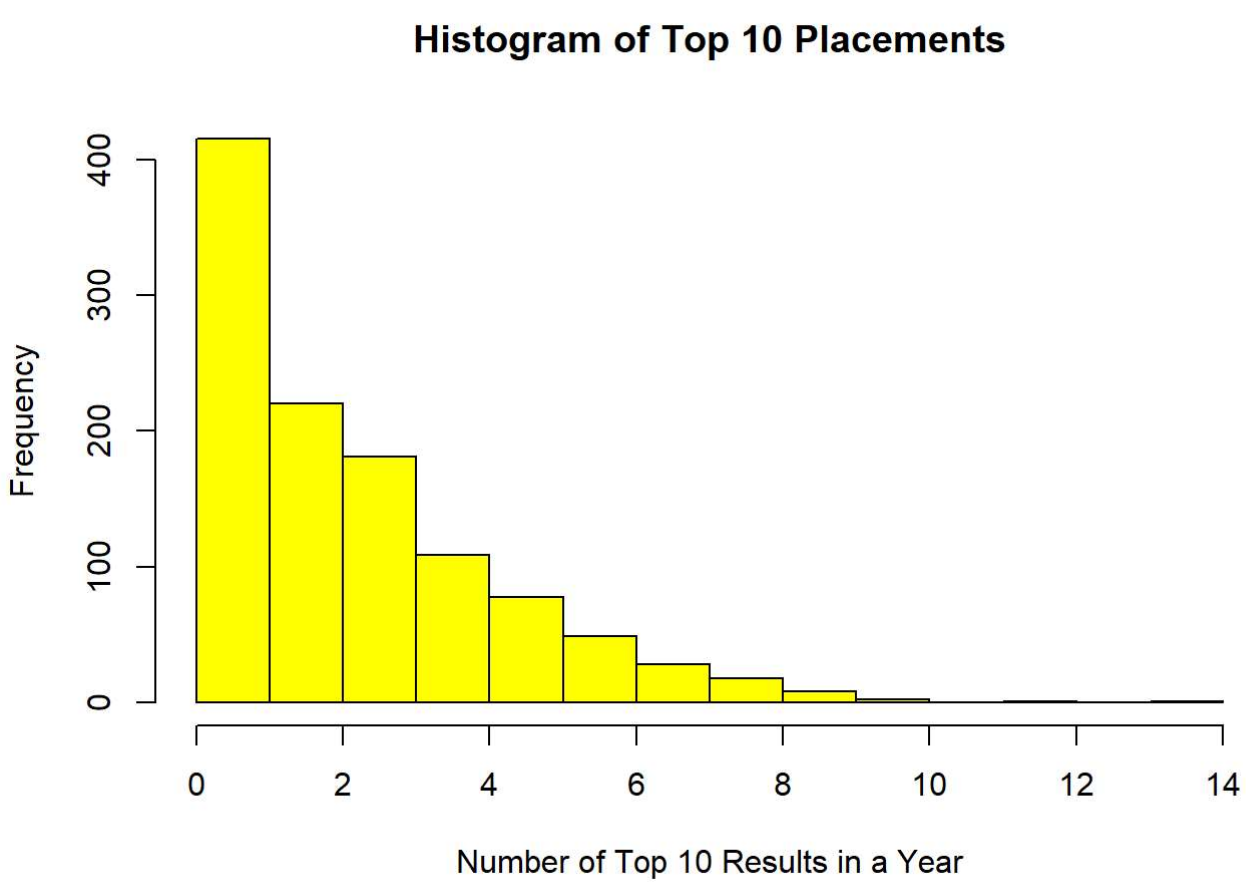
```
#We see that the metrics indicate that the original model is the best model. But because our mod
el has 10 less predictor variables, making it much more simple and efficient, and an r squared v
alue being 0.023 less than the original model, we will keep the new model.
```

Creating interaction terms to improve our model

```
#No interaction terms will be created with Wins because many rows for it are 0.
#No interaction terms will be created for Top.10 either because its data is quite skewed, as evi
denced by the histogram.

#Frequency table for Wins
table(pga_tour_train$Wins)
```

```
##
##   0   1   2   3   4   5
## 909 167  24   7   2   1
```

```
#Histogram
hist(pga_tour_train$Top.10, main="Histogram of Top 10 Placements", xlab="Number of Top 10 Result
s in a Year", col="yellow")
```


Histogram of Top 10 Placements
Frequency
0
100
200
300
400
0
2
4
6
8
10
12
14
Number of Top 10 Results in a Year

```
#Interaction terms will be created between the variables: Average.Score, Points, and Average.SG.
Total


#Creating the interaction terms for the training and testing datasets

pga_tour_train["Score_Points_Interaction"] = pga_tour_train$Average.Score * pga_tour_train$Point
s
pga_tour_train["Score_SG_Interaction"] = pga_tour_train$Average.Score * pga_tour_train$Average.S
G.Total

pga_tour_train["Points_SG_Interaction"] = pga_tour_train$Points * pga_tour_train$Average.SG.Tota
l
pga_tour_train["Score_Points_SG_Interaction"] = pga_tour_train$Average.Score * pga_tour_train$Po
ints * pga_tour_train$Average.SG.Total



pga_tour_test["Score_Points_Interaction"] = pga_tour_test$Average.Score * pga_tour_test$Points
pga_tour_test["Score_SG_Interaction"] = pga_tour_test$Average.Score * pga_tour_test$Average.SG.T
otal

pga_tour_test["Points_SG_Interaction"] = pga_tour_test$Points * pga_tour_test$Average.SG.Total
pga_tour_test["Score_Points_SG_Interaction"] = pga_tour_test$Average.Score * pga_tour_test$Point
s * pga_tour_test$Average.SG.Total







interaction_term_models = function(train_dataset, ntrees, predictor_columns, k_value, response_c
olumn, interaction_columns){

  #Holds the list of best models found in each combination of interaction terms.
  list_of_best_models = list()


  #Case where no interaction terms are included
  model_without_interactions = rf_model_tuning_cv(train_dataset, ntrees, predictor_columns, k_va
lue, response_column)
  list_of_best_models = append(list_of_best_models, list(list(model = model_without_interaction
s, combination = predictor_columns)))


  # Get all possible combinations of interaction terms (excluding the case with no interaction t
erms)
  all_combinations = list()


  for (i in 1:length(interaction_columns))
    all_combinations = c(all_combinations, combn(interaction_columns, i, simplify = FALSE))
```

```
  for (column_combination in all_combinations) {

    #Combine the column combination with the predictor columns
    new_predictor_columns = c(predictor_columns, unlist(column_combination))


    #Train and tune the model with cross-validation
    model = rf_model_tuning_cv(train_dataset, ntrees, new_predictor_columns, k_value, response_c
olumn)


    list_of_best_models = append(list_of_best_models, list(list(model = model, combination = new
_predictor_columns)))
  }

  return(list_of_best_models)
}



#best_model_in_theory chooses the model with the highest r^2 value.
best_model_on_training_data = function(itms_list){

  #Saves which index the best model is in
  best_model_index = 0


  highest_rsq_value = 0

  #Iterating over each model
  for(i in seq_along(itms_list)){
    given_model = itms_list[[i]]


    final_rsq_value = tail(given_model$model$rsq, 1)


    if(final_rsq_value > highest_rsq_value){
      highest_rsq_value = final_rsq_value
      best_model_index = i
    }

  }
  #Retrieving the best model found and its predictor variables that were used
  best_model = itms_list[[best_model_index]]

  return(best_model)
}
```

```
best_model_on_testing_data = function(itm_list, testing_dataset){

  best_r2 = 0
  best_model_index = 0

  for (i in seq_along(itm_list)){
    model_predictions = predict(itm_list[[i]]$model, pga_tour_test)


    r2_model = r2_calculation(model_predictions, pga_tour_test)


    if (best_r2 < r2_model){
      best_r2 = r2_model
      best_model_index = i
    }
  }
  return(itm_list[[best_model_index]])
}


r2_calculation = function(predictions, testing_data){
  ss_res = sum((testing_data$Money - predictions)^2)


  ss_total = sum((testing_data$Money - mean(testing_data$Money))^2)

  r2_model = 1 - (ss_res / ss_total)

  return(r2_model)
}
```

Running the interaction terms

```
pga_itms = interaction_term_models(pga_tour_train, c(125:150), c(9:12, 14), 5, "Money", c(19:2
2))


best_pga_interaction_model_theoretically = best_model_on_training_data(pga_itms)



print(paste("Predictor variables used in the theoretically best model that was found:", paste(co
lnames(pga_tour_train[, best_pga_interaction_model_theoretically$combination]), collapse = ",
")))
```

```
## [1] "Predictor variables used in the theoretically best model that was found: Average.Score,
Points, Wins, Top.10, Average.SG.Total, Score_Points_Interaction, Points_SG_Interaction"
```

```
best_pga_interaction_model_practically = best_model_on_testing_data(pga_itms, pga_tour_test)

theoretical_interaction_model_predictions = predict(best_pga_interaction_model_theoretically$mod
el, pga_tour_test)

practical_interaction_model_predictions = predict(best_pga_interaction_model_practically$model,
pga_tour_test)


best_pga_interaction_model_theoretically; best_pga_interaction_model_practically
```

```
## $model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 136
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1688.024
##                     % Var explained: 92.46
##
## $combination
## [1]  9 10 11 12 14 19 21
```

```
## $model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 135
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1878.856
##                     % Var explained: 91.66
##
## $combination
## [1]  9 10 11 12 14 19
```

```
r2_calculation(theoretical_interaction_model_predictions, pga_tour_test); r2_calculation(practic
al_interaction_model_predictions, pga_tour_test)
```

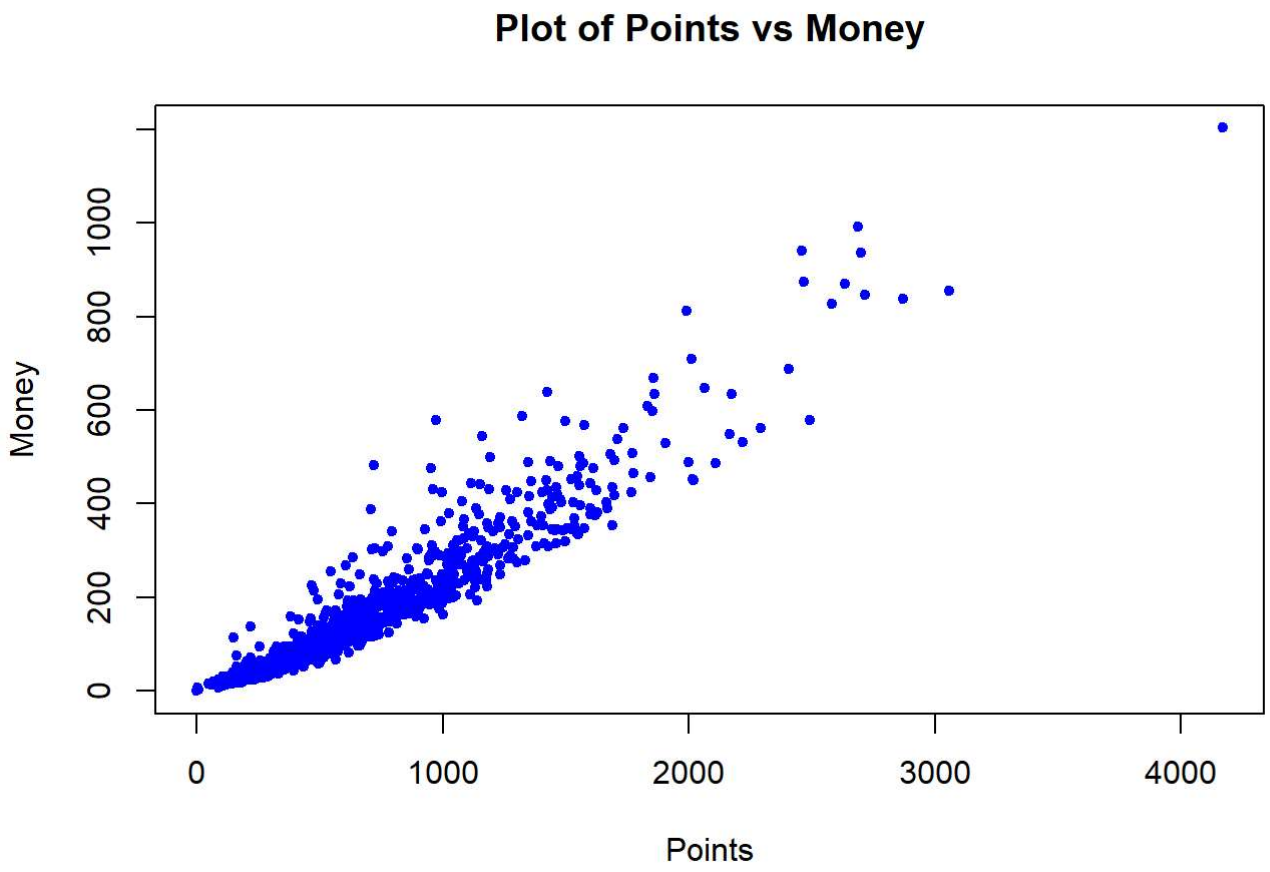
```
## [1] 0.917892
```

```
## [1] 0.9252906
```

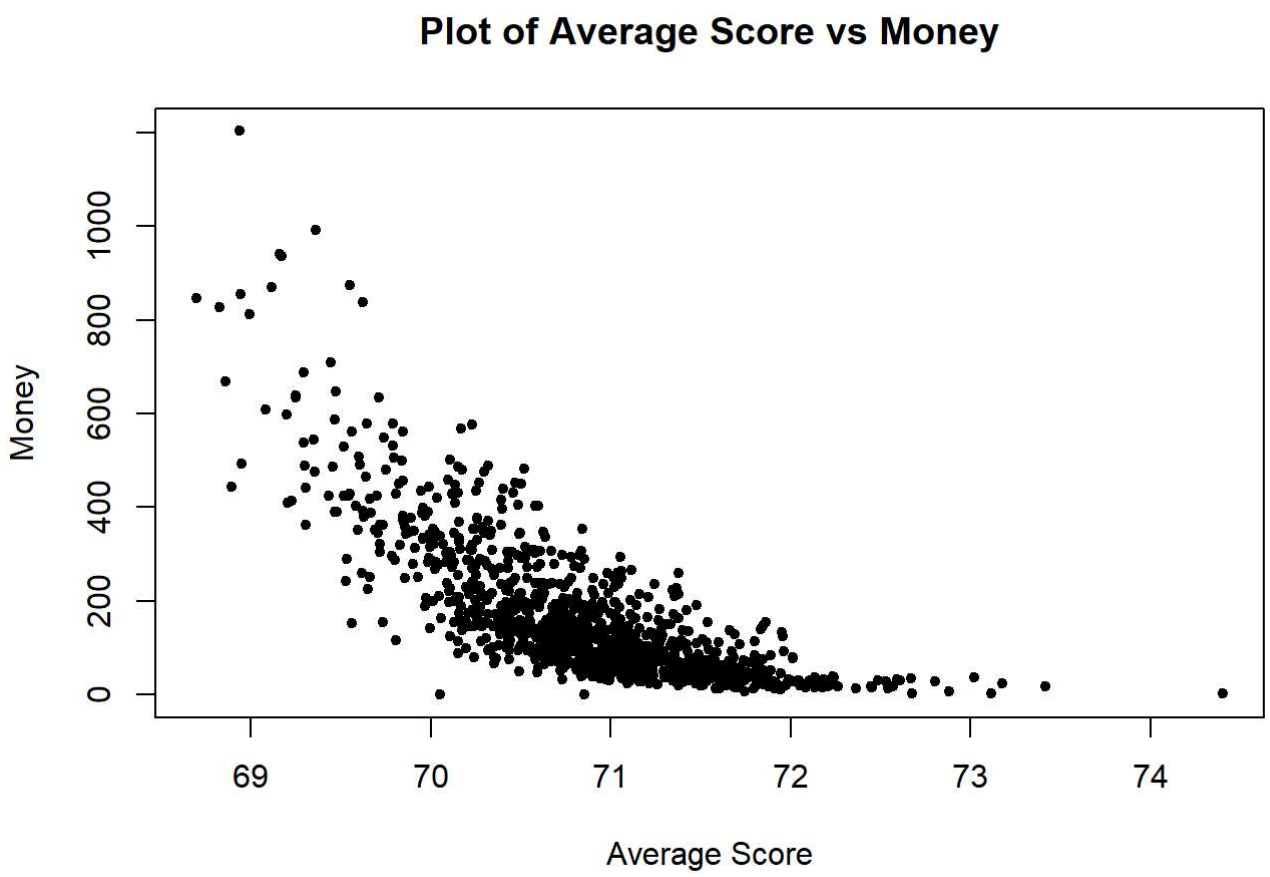
```
#The best model theoretically and practically perform very similarly. But because the best model on our training dataset has a slightly higher r^2 value and a lower MSE value, we will use this as our chosen interaction model.


best_interaction_model = best_pga_interaction_model_theoretically
best_interaction_model_r2 = r2_calculation(theoretical_interaction_model_predictions, pga_tour_test)
```

Effect of quadratic variables on Model predictions

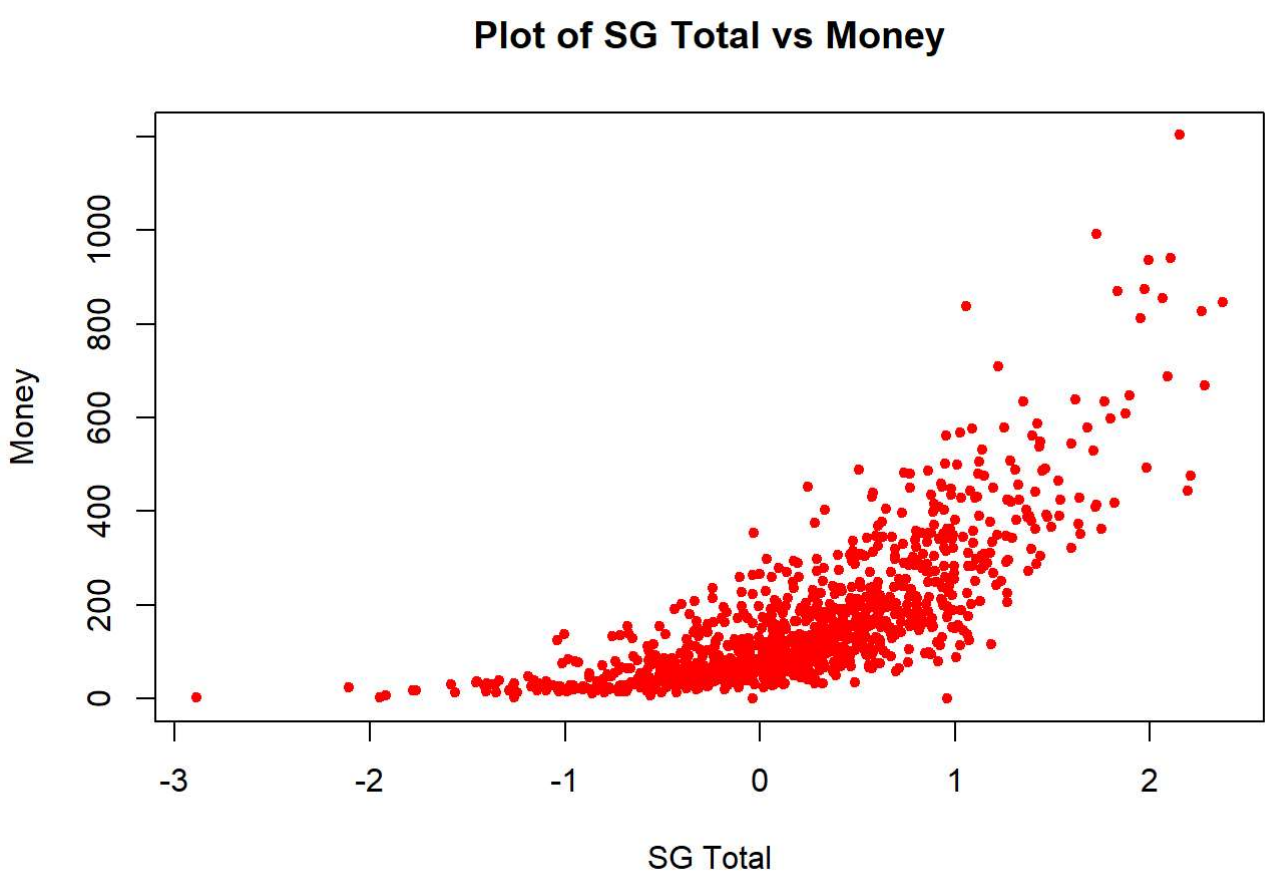
```
#From these plots, we observe that there appears to be a curved relationship between each of the variables and Money. Because of this, we will examine if including quadratic terms, will improve our model's performance.

plot(pga_tour_train$Points, pga_tour_train$Money, col = "blue", main = "Plot of Points vs Money", xlab="Points", ylab = "Money", pch=20)
```




```
plot(pga_tour_train$Average.Score, pga_tour_train$Money, main = "Plot of Average Score vs Money", xlab="Average Score", ylab = "Money", pch=20)
```




```
plot(pga_tour_train$Average.SG.Total, pga_tour_train$Money, main = "Plot of SG Total vs Money",
xlab="SG Total", ylab = "Money", col="red", pch=20)
```




```
#Given the nature of golf and the PGA tour, it is quite intuitive to assume that some predictor variables may have an exponential impact on how much money a player has earned. The following co de will test this assumption, and determine if changing one or more variables .
#Once again Wins won't be converted to a polynomial feature, given that most of its observations are 0s and 1s and that is acting as a categorical variable. Top.10 will also be excluded from be ing converted to polynomial given that its data is more categorical than continuous.


#Frequency table showing Top.10 behaving more categorical than continuous
table(pga_tour_train$Top.10)
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  12  14
## 185 230 220 181 109  78  49  28  18   8   2   1   1
```

```
pga_tour_train["Score_Squared"] = pga_tour_train$Average.Score ^ 2
pga_tour_train["Points_Squared"] = pga_tour_train$Points ^ 2
pga_tour_train["SG_Total_Squared"] = pga_tour_train$Average.SG.Total ^ 2


pga_tour_test["Score_Squared"] = pga_tour_test$Average.Score ^ 2
pga_tour_test["Points_Squared"] = pga_tour_test$Points ^ 2
pga_tour_test["SG_Total_Squared"] = pga_tour_test$Average.SG.Total ^ 2



itm_quadratic = interaction_term_models(pga_tour_train, c(125:150), best_interaction_model$combi
nation, 5, "Money", c(23:25))
best_quadratic_model_theoretically = best_model_on_training_data(itm_quadratic)
best_quadratic_model_theoretically
```

```
## $model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 137
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1750.982
##                     % Var explained: 92.3
##
## $combination
## [1]  9 10 11 12 14 19 21
```

```
best_quadratic_model_practically = best_model_on_testing_data(itm_quadratic, pga_tour_test)
best_quadratic_model_practically
```

```
## $model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 141
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1793.501
##                     % Var explained: 91.59
##
## $combination
## [1]  9 10 11 12 14 19 21 25
```

```
itm_quadratic
```

```
## [[1]]
## [[1]]$model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 137
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1750.982
##                     % Var explained: 92.3
##
## [[1]]$combination
## [1]  9 10 11 12 14 19 21
##
##
## [[2]]
## [[2]]$model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 136
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1818.184
##                     % Var explained: 92.05
##
## [[2]]$combination
## [1]  9 10 11 12 14 19 21 23
##
##
## [[3]]
## [[3]]$model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 142
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1924.675
##                     % Var explained: 91.76
##
## [[3]]$combination
## [1]  9 10 11 12 14 19 21 24
##
##
## [[4]]
```

```
## [[4]]$model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 141
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 1793.501
##                     % Var explained: 91.59
##
## [[4]]$combination
## [1]  9 10 11 12 14 19 21 25
##
##
## [[5]]
## [[5]]$model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 148
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 1883.286
##                     % Var explained: 91.64
##
## [[5]]$combination
## [1]  9 10 11 12 14 19 21 23 24
##
##
## [[6]]
## [[6]]$model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 147
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 1822.326
##                     % Var explained: 91.17
##
## [[6]]$combination
## [1]  9 10 11 12 14 19 21 23 25
##
##
## [[7]]
## [[7]]$model
```

```
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 127
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 1905.267
##                     % Var explained: 91.47
##
## [[7]]$combination
## [1]  9 10 11 12 14 19 21 24 25
##
##
## [[8]]
## [[8]]$model
##
## Call:
##  randomForest(x = train_fold[, predictor_columns], y = train_fold[[response_column]],      nt
ree = ntrees[i])
##                Type of random forest: regression
##                      Number of trees: 137
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 1836.909
##                     % Var explained: 91.78
##
## [[8]]$combination
##  [1]  9 10 11 12 14 19 21 23 24 25
```

```
#We can see that the best model when applied to the training data differs from the best performi
ng model which was applied to the testing data. But because the r^2 value of the best quadratic
models is almost greater by 0.4 than the second best one, we can conclude that the best quadrati
c model is the best theoretical one.


best_quadratic_model = best_quadratic_model_practically
best_quadratic_model_predictions = predict(best_quadratic_model$model, pga_tour_test)
best_quadratic_model_r2 = r2_calculation(best_quadratic_model_predictions, pga_tour_test)


#We also observe that the r^2 values of the quadratic models on both the testing and training da
ta is actually slightly lower as compared to the best interaction model. Given this, we choose t
o go with our interaction model as the best model, because it will be simpler and less prone to
overfitting.


#r^2 values on testing data
best_interaction_model_r2; best_quadratic_model_r2
```

```
## [1] 0.917892
```

```
## [1] 0.9238477
```

```
#r^2 values on training data
tail(best_interaction_model$model$rsq, 1); tail(best_quadratic_model$model$rsq, 1)
```

```
## [1] 0.9245863
```

```
## [1] 0.9158902
```