DSC 423: Data Analysis & Regression

Assignment 8: Residual Analysis

Name: Kushal Navghare
Student ID: 2116916
Honor Statement: I, Kushal Navghare, assure that I have completed this work independently. The solutions given are entirely my own work.

1. Short Essay - Read the short PDF on George Box. Explain in your own words the significance of "all models are wrong, but some are useful" as if you were interviewing for job in data science.

Ans: Author Guillem Barroso talks about how the results generated by all the models are all wrong, however, some models can be helpful in certain situations. He states that all models are flawed and can never fully mimic the reality. However, despite their imperfections, models can still be valuable if they are close enough to the real world. While these models will never be perfect, they can provide useful insights and will be helpful decision-making. It also helps if we understand the requirement of the real world application of the model results. Not all situations require a high accuracy model. The focus should be on developing models that provides an estimated outcome according to the specific situation. The role of human interpretation in the results produced by models is also important. A data scientist should know how to not only build models, but also how to convey the results produced by these models to the end users. A data scientists should be able to analyze and make sense of the model outputs in relation to the audience, context, objectives, and constraints of the problem at hand.

2. Previously, you used the PGA tour dataset to predict Prize Money. Use a log transformation to transform Prize Money into a new response variable. Apply your knowledge of regression analysis to fit a regression model using the remaining predictors in your dataset. If necessary, remove the non-significant variables. Remember to remove one variable at a time (variable with largest p- value is removed first) and refit the model, until all variables are significant.

```
# read file
raw_df <- read.csv(paste0(dir.path, 'data/pgatour2006.csv'))

# summary
dim(raw_df)
```

```
## [1] 196  11
```

```
summary(raw_df)
```

```
##      Name             PrizeMoney     AveDrivingDistance DrivingAccuracy
##  Length:196         Min.   :  2240   Min.   :265.9      Min.   :49.75
##  Class :character   1st Qu.: 17369   1st Qu.:283.6      1st Qu.:59.76
##  Mode  :character   Median : 36644   Median :288.2      Median :63.24
##                     Mean   : 50891   Mean   :289.5      Mean   :63.38
##                     3rd Qu.: 57915   3rd Qu.:295.5      3rd Qu.:66.97
##                     Max.   :662771   Max.   :319.6      Max.   :78.43
##       GIR        PuttingAverage  BirdieConversion   SandSaves
##  Min.   :56.87   Min.   :1.712   Min.   :23.17    Min.   :33.91
```

```
##  1st Qu.:63.52   1st Qu.:1.763   1st Qu.:27.51    1st Qu.:45.13
##  Median :65.36   Median :1.778   Median :29.01    Median :48.66
##  Mean   :65.19   Mean   :1.780   Mean   :28.98    Mean   :48.97
##  3rd Qu.:66.77   3rd Qu.:1.796   3rd Qu.:30.55    3rd Qu.:52.87
##  Max.   :74.15   Max.   :1.851   Max.   :35.66    Max.   :63.64
##    Scrambling      BounceBack    PuttsPerRound
##  Min.   :49.02   Min.   :12.29   Min.   :27.96
##  1st Qu.:55.26   1st Qu.:17.56   1st Qu.:28.91
##  Median :57.65   Median :19.62   Median :29.19
##  Mean   :57.49   Mean   :19.60   Mean   :29.20
##  3rd Qu.:59.46   3rd Qu.:21.31   3rd Qu.:29.48
##  Max.   :66.45   Max.   :25.93   Max.   :30.19
```

```
str(raw_df)
```

```
## 'data.frame':    196 obs. of  11 variables:
##  $ Name              : chr  "Aaron Baddeley" "Adam Scott" "Alex Aragon" "Alex Cejka" ...
##  $ PrizeMoney        : int  60661 262045 3635 17516 16683 107294 50620 57273 86782 23396 ...
##  $ AveDrivingDistance: num  288 301 303 289 288 ...
##  $ DrivingAccuracy   : num  60.7 62 51.1 66.4 63.2 ...
##  $ GIR               : num  58.3 69.1 59.1 67.7 64 ...
##  $ PuttingAverage    : num  1.75 1.77 1.79 1.78 1.76 ...
##  $ BirdieConversion  : num  31.4 30.4 29.9 29.3 29.3 ...
##  $ SandSaves         : num  54.8 53.6 37.9 45.1 52.4 ...
##  $ Scrambling        : num  59.4 57.9 50.8 54.8 57.1 ...
##  $ BounceBack        : num  19.3 19.4 16.8 17.1 18.2 ...
##  $ PuttsPerRound     : num  28 29.3 29.2 29.5 28.9 ...
```

a. (10 points) Check for multicollinear. Explain your process. For understanding the multicolinearity, we will first build a full model using all features to predict PrizeMoney. Then we will remove the features from the model based on the significance values (p-values) of the features starting from highest.

To understand multicolinearity, we will use VIF values (Variance Inflation Factor) to assess the multiconilearity. These values quantifies how much the variance of the estimated regression coefficients is increased due to multicollinearity. A high VIF value indicates a high degree of multicollinearity, suggesting that the corresponding predictor variables are strongly correlated with each other.

```
# select numeric columns only
num_df <- raw_df %>%
  select_if(is.numeric) %>%
  dplyr::select(-c(PuttingAverage, BounceBack, DrivingAccuracy,
                   AveDrivingDistance, PuttsPerRound))

# full model
full_model <- lm(log(PrizeMoney)~., data = raw_df %>% dplyr::select(-c(Name)))

# build final model
final_model <- lm(log(PrizeMoney)~., data = num_df)

# model output
summary(final_model)
```

```
##
```

```
## Call:
## lm(formula = log(PrizeMoney) ~ ., data = num_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7349 -0.4672 -0.0755  0.4409  2.1230
##
## Coefficients:
##                     Estimate Std. Error t value             Pr(>|t|)
## (Intercept)      -11.198268   1.451430  -7.715 0.000000000000651868 ***
## GIR                0.163349   0.018218   8.966 0.000000000000000282 ***
## BirdieConversion   0.199590   0.021880   9.122 < 0.0000000000000002 ***
## SandSaves          0.016673   0.009746   1.711               0.0888 .
## Scrambling         0.075263   0.018100   4.158 0.000048435815838413 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6628 on 191 degrees of freedom
## Multiple R-squared:  0.5522, Adjusted R-squared:  0.5428
## F-statistic: 58.87 on 4 and 191 DF,  p-value: < 0.00000000000000022
```

```
# multicolinearity in full model
vif(full_model)
```

```
## AveDrivingDistance    DrivingAccuracy                GIR     PuttingAverage
##             3.4508             2.7557             8.0924            15.8540
##   BirdieConversion          SandSaves         Scrambling         BounceBack
##             5.4303             1.5115             4.7371             1.5266
##      PuttsPerRound
##            20.9510
```

As seen in full model, we can see the VIF (Variance Inflation Factor) for few features is quite high(PuttsPerRound, PuttingAverage, GIR, etc.) These variables are likely contributing to multicollinearity issues. A general rule is to prioritize addressing variables with VIF values above the chosen threshold (e.g., 5 or 10).

b. Compare this model to the one you made in the previous assignment. How did performing a log transformation impact the quality of the model? Why?

Previously, we have used a model with below formula:

$\text{PrizeMoney} \sim \text{DrivingAccuracy}^{2+\text{GIR}}2 + \text{BirdieConversion}^{2+\text{Scrambling}}2$

```
# model (previous)
prev_model <- lm(PrizeMoney~DrivingAccuracy^2+GIR^2+BirdieConversion^2+Scrambling^2,
                 data = raw_df %>% dplyr::select(-c(Name)))

summary(prev_model)
```

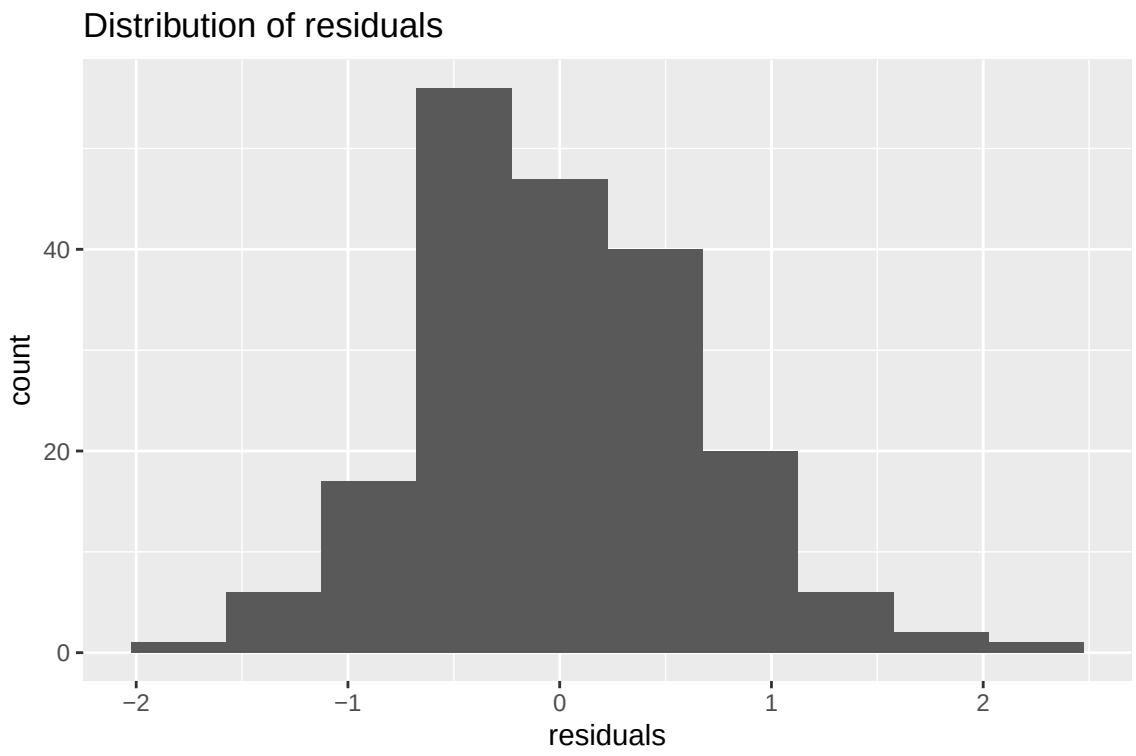
```
##
## Call:
## lm(formula = PrizeMoney ~ DrivingAccuracy^2 + GIR^2 + BirdieConversion^2 +
```

```
##     Scrambling^2, data = raw_df %>% dplyr::select(-c(Name)))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -85429 -27959  -7833  15674 422173
##
## Coefficients:
##                    Estimate Std. Error t value             Pr(>|t|)
## (Intercept)      -1094996.9   109585.4  -9.992 < 0.0000000000000002 ***
## DrivingAccuracy     -1964.1      815.7  -2.408                0.017 *
## GIR                  9742.9     1465.9   6.646        0.000000000306 ***
## BirdieConversion    10670.5     1703.7   6.263        0.000000002439 ***
## Scrambling           5670.4     1239.4   4.575        0.000008556442 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50080 on 191 degrees of freedom
## Multiple R-squared:  0.3984, Adjusted R-squared:  0.3858
## F-statistic: 31.62 on 4 and 191 DF,  p-value: < 0.00000000000000022
```

With this model, we were able to achieve 0.38 of adj R-squared error. Now, after applying log(PrizeMoney) on the outcome, we achieved 0.55 adj R-squared. This shows improvement in performance after applying log transformation to the outcome. This can happen when one or more assumptions about the linear regression are violated. It may be because the relationship between the predictors and the outcome may not be strictly linear. Or the assumption of constant variance of the errors across all levels of the predictors. In some cases, the variance of the response variable (y) may increase or decrease systematically with its mean value.

c. Analyze and discuss the residual plots.

```r
# residual plots
residuals <- as.data.frame(final_model$residuals)

plt_1 <- ggplot(data = residuals, aes(x = final_model$residuals))
plt_1 + geom_histogram( binwidth = .45) +
  xlab("residuals") +
  ylab("count") +
  ggtitle("Distribution of residuals")
```

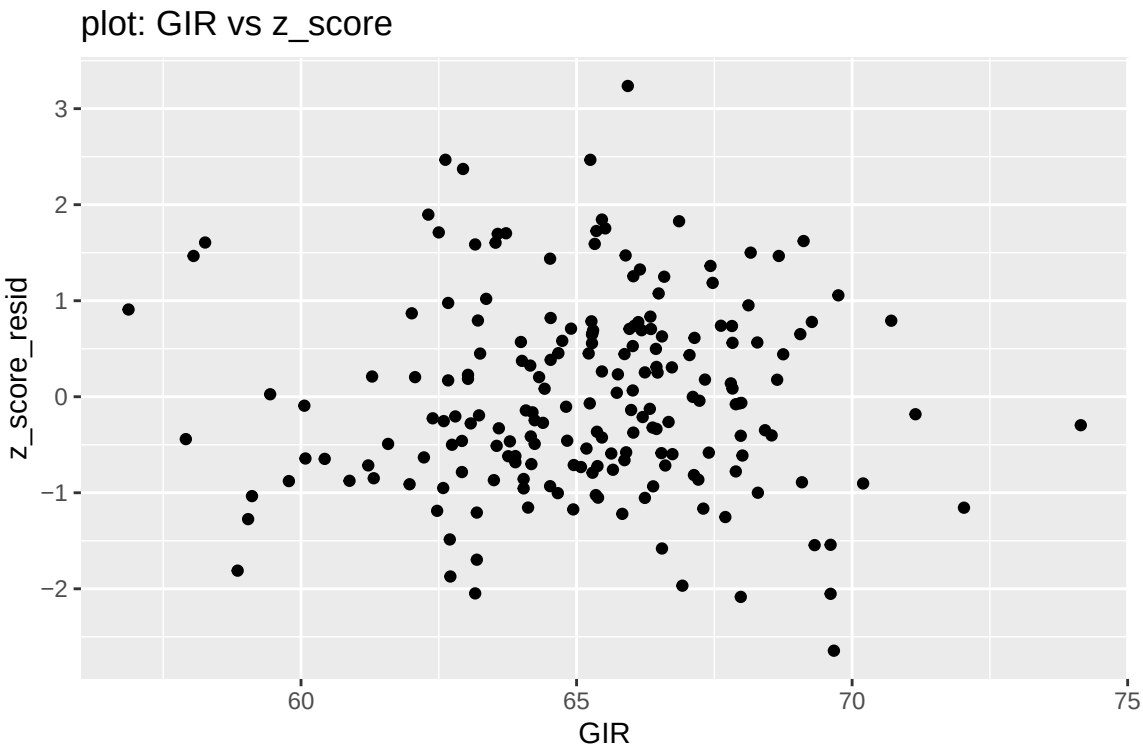


Now, let's look at residual plots with z-score.

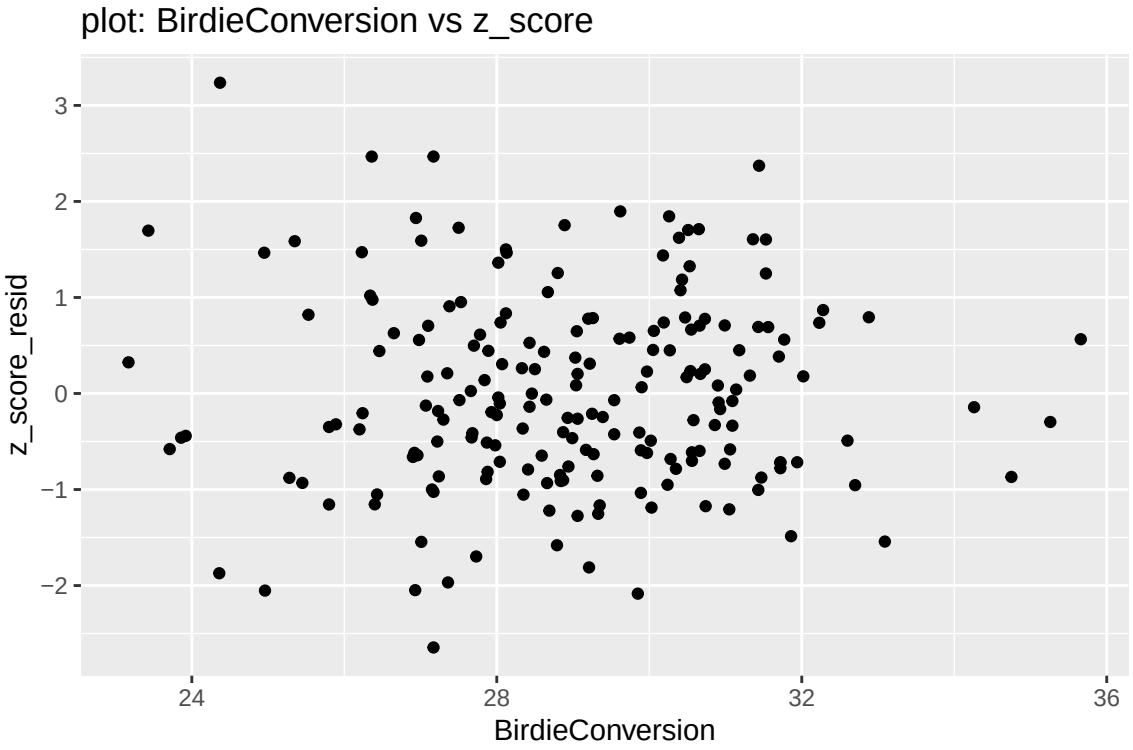
```
# calculate z-score
avg_resid <- mean(final_model$residuals)
std_resid <- sd(final_model$residuals)

z_score_resid <- (final_model$residuals - avg_resid)/std_resid

num_df <- num_df %>%
  mutate(z_score_resid = z_score_resid)
```

Now, we will look at the plots.

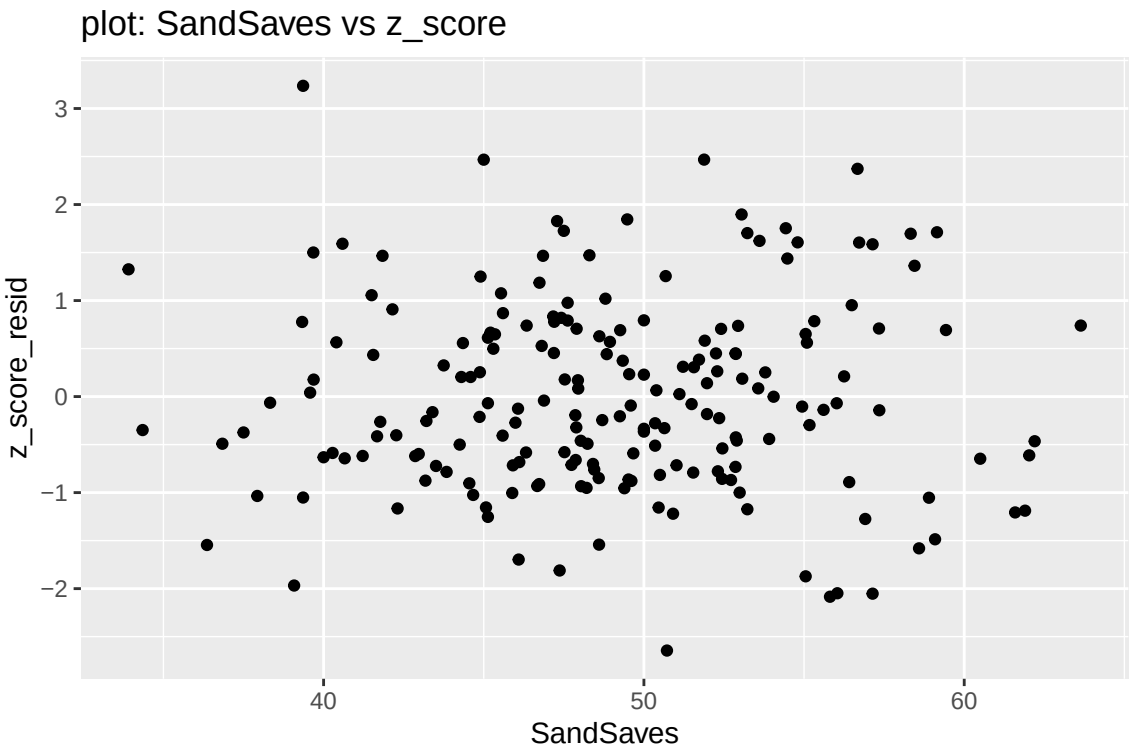
```
plt_2 <- ggplot(data = num_df, aes(x = GIR, y = z_score_resid))
plt_2 + geom_point() +
  ggtitle("plot: GIR vs z_score")
```



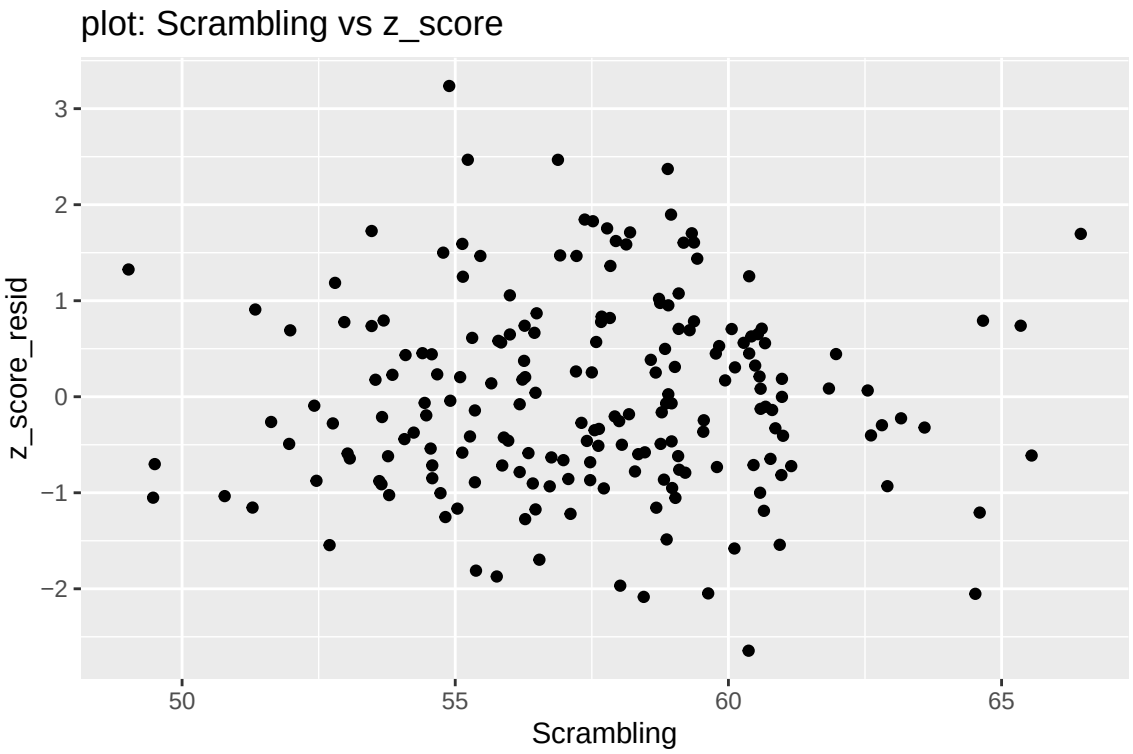

```
plt_3 <- ggplot(data = num_df, aes(x = BirdieConversion, y = z_score_resid))
plt_3 + geom_point() +
  ggtitle("plot: BirdieConversion vs z_score")
```



```
plt_4 <- ggplot(data = num_df, aes(x = SandSaves, y = z_score_resid))
plt_4 + geom_point() +
  ggtitle("plot: SandSaves vs z_score")
```




```
plt_4 <- ggplot(data = num_df, aes(x = Scrambling, y = z_score_resid))
plt_4 + geom_point() +
  ggtitle("plot: Scrambling vs z_score")
```

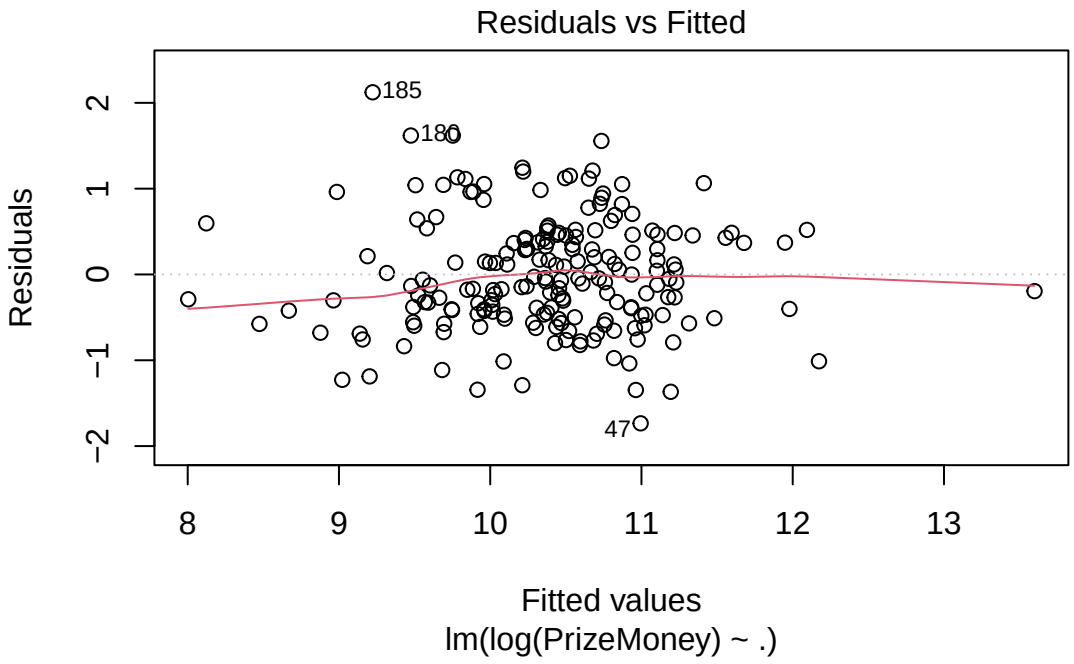


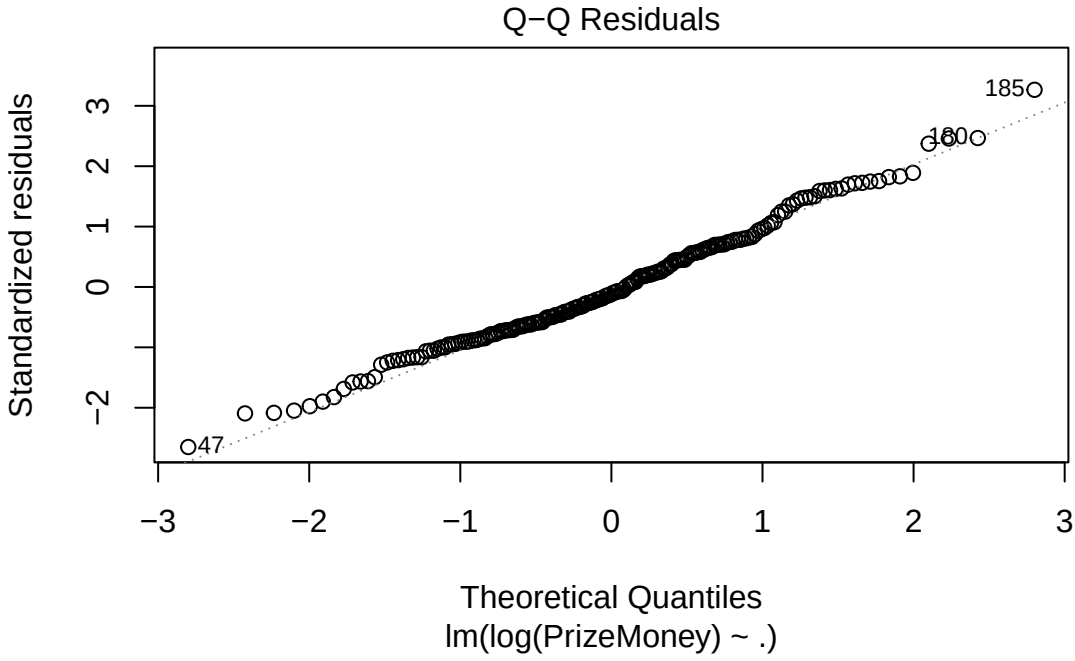
Looking at above residual plots, the residuals does not seem to have a constant variance. Which probably explains the low adj R-squared. Also, the line of fit is influenced by few outliers in the dataset.

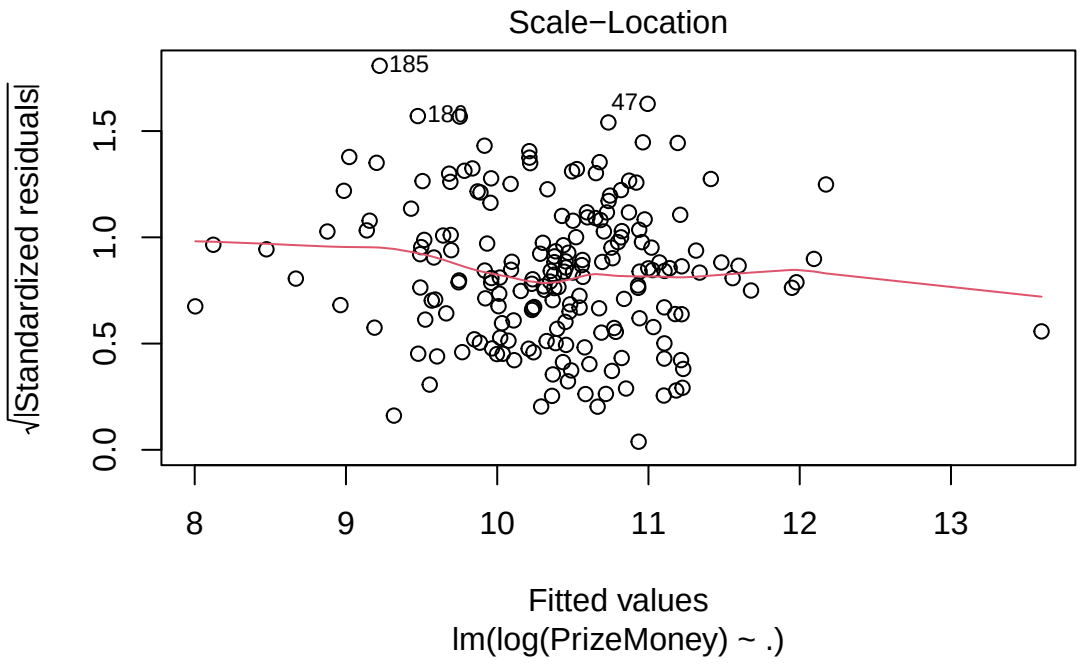
d. Analyze if there are any outliers and/or influential points. If there are points in the dataset that need to be investigated, give one or more reason to support each point chosen. Discuss your answer.

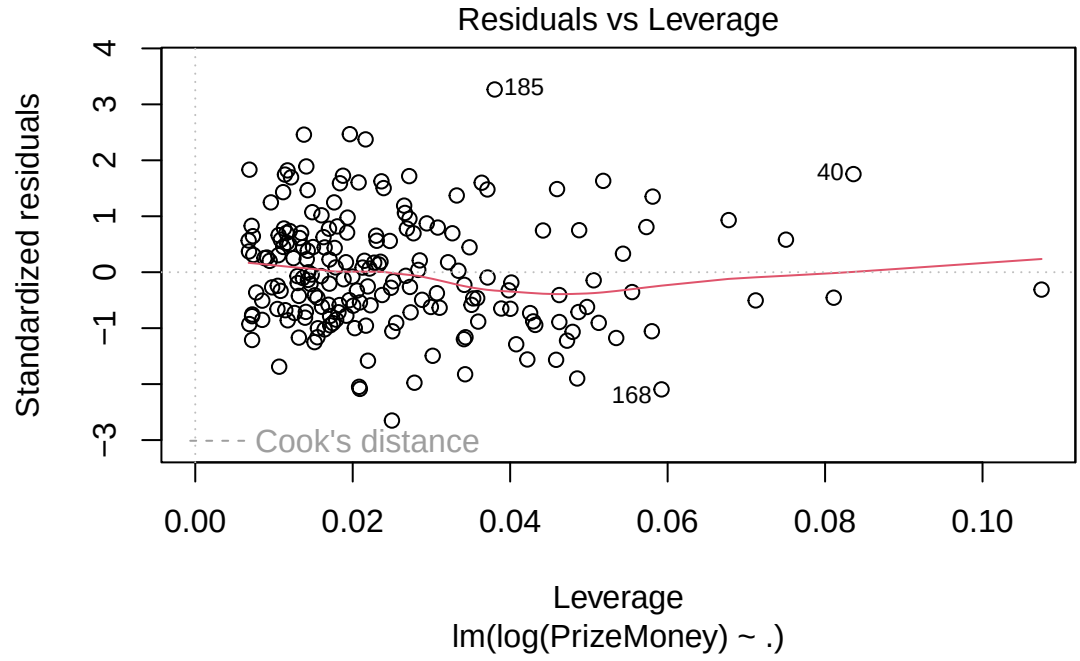
Ans: Outliers in the dataset can affect the line of fit, it will change the slope of the line by trying to fit the outlier. The outliers can be found in the model plots.

```
plot(final_model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(log(PrizeMoney) ~ .)
185
180
47
8
9
10
11
12
13
−2
−1
0
1
2


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(log(PrizeMoney) ~ .)
185
160
47
−3
−2
−1
0
1
2
3





From above plots, it is clear that observation #180, #185 & #47 are outliers in the dataset. 1. If looked at the final plot, the points are annotated. 2. Normal Q-Q plot also shows that the residual are normal, except of these 3 outliers which influence the best line of fit.

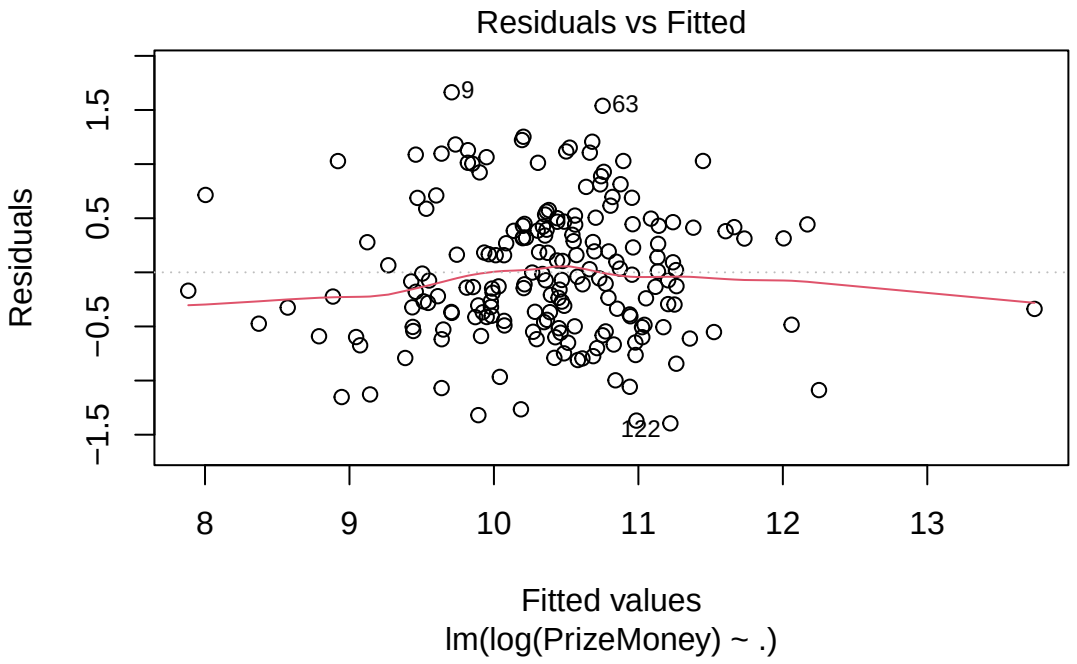
Let's see these outliers and their distribution in the data are affecting the model performance. We will test by removing such cases from the data.

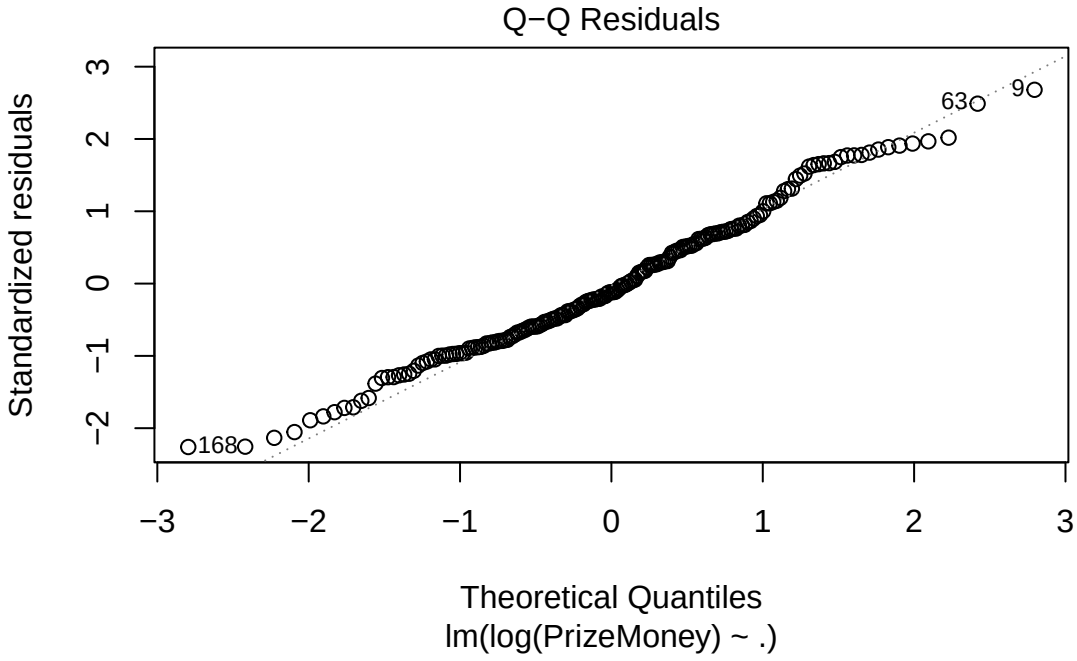
```
# remove outlier rows
df_no_outliers <- num_df[-c(47, 180, 185), ] %>%
  dplyr::select(-c(z_score_resid))

# model without outliers
model_no_outlier <- lm(log(PrizeMoney)~., data = df_no_outliers)

summary(model_no_outlier)
```

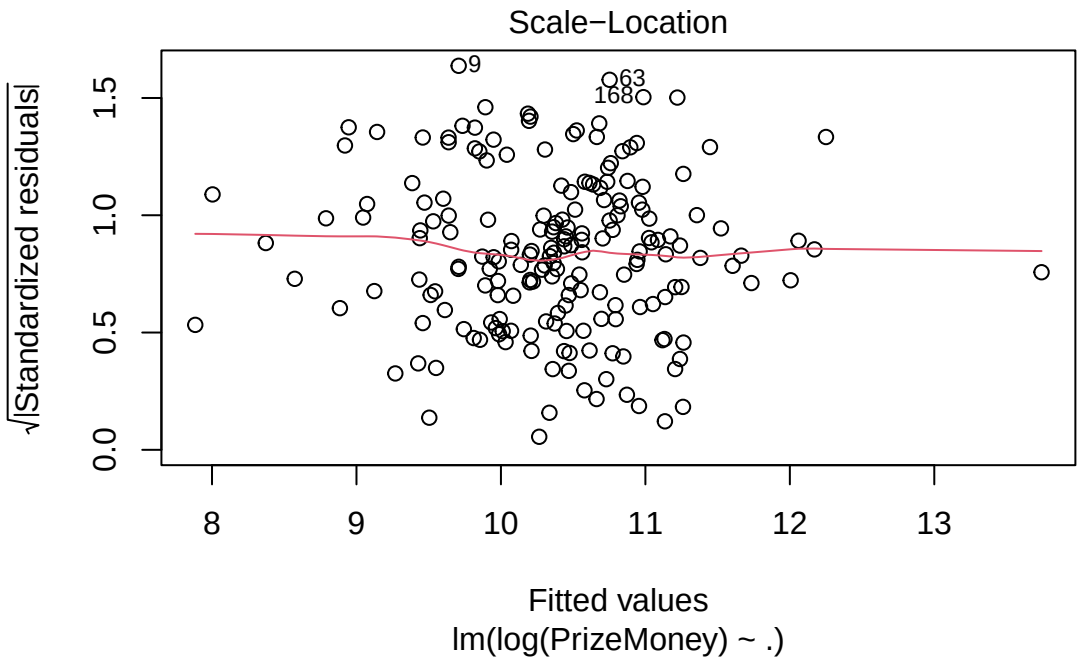
```
##
## Call:
## lm(formula = log(PrizeMoney) ~ ., data = df_no_outliers)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.39512 -0.46009 -0.07338  0.42241  1.66361
##
## Coefficients:
##                    Estimate Std. Error t value             Pr(>|t|)
## (Intercept)      -12.273212   1.385261  -8.860 0.000000000000000603 ***
## GIR                0.169317   0.017311   9.781 < 0.0000000000000002 ***
## BirdieConversion   0.209787   0.020938  10.019 < 0.0000000000000002 ***
## SandSaves          0.017798   0.009243   1.926               0.0557 .
## Scrambling         0.080906   0.017128   4.724 0.000004523876279156 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6252 on 188 degrees of freedom
## Multiple R-squared:  0.602,  Adjusted R-squared:  0.5936
## F-statistic:  71.1 on 4 and 188 DF,  p-value: < 0.00000000000000022
```

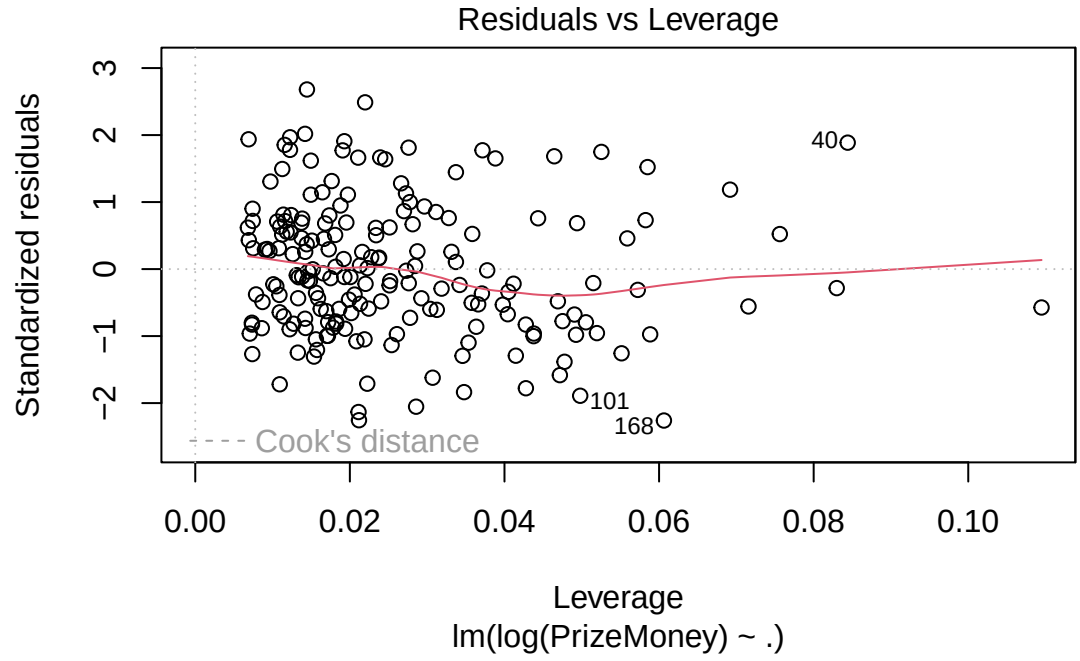
From the results, it looks like the performance improved slightly. However, there are new outliers which are shown in model plots.

```
plot(model_no_outlier)
```


Residuals vs Fitted
Residuals
1.5
0.5
−0.5
−1.5
9
63
122
8
9
10
11
12
13
Fitted values
lm(log(PrizeMoney) ~ .)


Q−Q Residuals
Standardized residuals
3
2
1
0
−1
−2
63
9
168
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(log(PrizeMoney) ~ .)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
9
63
168
8
9
10
11
12
13
Fitted values
lm(log(PrizeMoney) ~ .)


Residuals vs Leverage
Standardized residuals
−2
−1
0
1
2
3
40
101
168
Cook's distance
0.00
0.02
0.04
0.06
0.08
0.10
Leverage
lm(log(PrizeMoney) ~ .)