Data Science: Captone (HarvardX: PH125.9x) - CYP Report

Ivory Wu

17 June 2020

Executive Summary

This is a report of Choose Your Own project for Data Science: Captone (HarvardX: PH125.9x). The author chose a dataset of Video Games Sales 2016 from Kaggle for the project.

The objective of the project is to find the most suitable model to predict the user review score of individual games from the variables available in the datasets. It could be used to improve the developers' and publishers' understanding of user's preference for future games.

Dataset

The project uses the Video Games Sales dataset from Kaggle [link] (https://www.kaggle.com/gregorut/videogamesales). This dataset contains video games with sales greater than 100,000 copies as of 2016. It was generated by a scrape of vgchartz.com.

Structure of the dataset:

```
str(VideoGames)
```

```
## 'data.frame':    6199 obs. of  10 variables:
##  $ Name           : Factor w/ 11563 levels ""," Beyblade Burst",..: 11059 5572 11061 6693 11057 6696
##  $ Platform       : Factor w/ 31 levels "2600","3DO","3DS",..: 26 26 26 5 26 26 5 29 26 17 ...
##  $ Year_of_Release: num  2006 2008 2009 2006 2006 ...
##  $ Genre          : Factor w/ 13 levels "","Action","Adventure",..: 12 8 12 6 5 6 8 5 12 2 ...
##  $ Global_Sales   : num  82.5 35.5 32.8 29.8 28.9 ...
##  $ Critic_Score   : int  76 82 80 89 58 87 91 61 80 95 ...
##  $ Critic_Count   : int  51 73 73 65 41 80 64 45 33 80 ...
##  $ User_Score     : num  8 8.3 8 8.5 6.6 8.4 8.6 6.3 7.4 9 ...
##  $ User_Count     : int  322 709 192 431 129 594 464 106 52 1588 ...
##  $ Rating         : Factor w/ 9 levels "","AO","E","E10+",..: 3 3 3 3 3 3 3 3 3 7 ...
```

The dataset has 6199 observations with 10 variables, including 4 catagorical variables (Name, Platfrom, Genre & Rating) and 6 numeric variables (Year_of_Release, Global_Sales, Critic_Score, Critic_Count, User_Score & User_Count).

Variables

```
summary(VideoGames)
```

```
##                                        Name          Platform    Year_of_Release
##  Need for Speed: Most Wanted             :   8   PS2    :1025   Min.   :1996
##  LEGO Star Wars II: The Original Trilogy:   7   X360   : 768   1st Qu.:2004
##  Madden NFL 07                          :   7   PS3    : 695   Median :2007
##  Madden NFL 08                          :   7   PC     : 617   Mean   :2007
##  Spider-Man 3                           :   7   XB     : 517   3rd Qu.:2011
##  Cars                                   :   6   Wii    : 429   Max.   :2016
##  (Other)                                :6157   (Other):2148
##           Genre       Global_Sales      Critic_Score    Critic_Count
##  Action      :1457   Min.   : 0.0100   Min.   :13.00   Min.   :  3.00
##  Sports      : 867   1st Qu.: 0.1100   1st Qu.:62.00   1st Qu.: 14.00
##  Shooter     : 789   Median : 0.2900   Median :73.00   Median : 24.00
##  Role-Playing: 633   Mean   : 0.7842   Mean   :70.34   Mean   : 28.89
##  Racing      : 530   3rd Qu.: 0.7600   3rd Qu.:81.00   3rd Qu.: 39.00
##  Platform    : 377   Max.   :82.5300   Max.   :98.00   Max.   :113.00
##  (Other)     :1546
##    User_Score      User_Count        Rating
##  Min.   :0.500   Min.   :    4   T      :2130
##  1st Qu.:6.500   1st Qu.:   11   E      :1894
##  Median :7.500   Median :   27   M      :1293
##  Mean   :7.185   Mean   :  174   E10+   : 818
##  3rd Qu.:8.200   3rd Qu.:   89          :  61
##  Max.   :9.600   Max.   :10179   AO     :   1
##                                  (Other):   2
```

Catagorical Variables:

- Name: The name of individual games.
- Platform: The platform that a game software runs on, and can be catagorised into 4 major groups by manufactures:
  1. Play Station (PS, PS2, PS3, PS4, PSP, PSV)
  2. XBOX (XB, X360, XOne)
  3. Nitendo (GB, GBA, Wii,WiiU, DS, 3DS, DC)
  4. Microsoft (PC) PS2 was the post popular platform in terms of numbers of games released on the platfrom.
- Rating: Video game content rating system. Entertainment Software Rating Board (ESRB) is used in this dataset.
  1. E rating games is suitable for all age.
  2. E10+ for age 10 and above.
  3. T for age 13 and above.
  4. M for age 17 and above.
  5. AO for age 18 and above.
     T was the most used rating followed by E.
- Genres: A specific category of games related by similar gameplay characteristics. In this dataset there are 13 genres accounted for: Action, Advebture, Fighting, Misc, Platfrom, Puzzle, Racing, Role-Playing, Shooter, Simulation, Sports and Startegy.
  Action is the most popular Genre by a margin (1457) to Sports in the second place (867).

Numeric Variables:

- Year_of_Release: The year that a game was released by the publisher,ranging from 1996 to 2016.
- Global_Sales: The number of games copies sold worldwide by millions. Only the games sold more than 100,000 copies were recorded in the dataset.
- Critic_Score: Average review scores from recognised critics in the gaming industry. The score is between 0 and 100 points and the mean in this dataset is 70.34.

- Critic_Count: Number of reviews given by critics. In this dataset it ranges from 3 to 113 with a mean of 28.89.
- User_Score: Average review score submitted by users. It is a 10-point system with a mean of 7.185.
- User_Count: Number of reviews given by users. In this dataset it ranges from 4 to 10179 with a median of 27.

Objective

The objective of the project is to find the most suitable model to predict **User Score** from other variables in the dataset. The Root Mean Square Error (RMSE) bewteen the precited value and the actual result will be used to measure the effectiveness of the model. A random validation set that consist of 10% of the original dataset and is not used in the modeling will be used to verify the selected model.

```
RMSE <- function(true_rating, pred_rating){
  sqrt(mean((true_rating - pred_rating)^2))}
```

Key Steps

1. **Data Screening and Cleaning**: Going through the data for the first time to remove unused and nominal data, observations that contains N/A values and nomalize certain variables for further caculation needed in the later stage. Validation set is created at the end of this stage to avod it interfering with the building of the model.

2. **Data Exploration and Analysis**: Look at the structure and the summary of the dataset, including the specifity/class of each variables and the observations it contents. Fitting of linear model on all available variables is used to determine the coeficient and to prioritize further data visualization.

3. **Data visualization**:

   - catagorical Variables: Create boxplots of user scores by levels to visualise if there are significant variance between different levels.
   - Numeric Variables: Create Pointplots or lineplot to see if there is a trend between user scores and the variable.

4. **Create Train Set and Test Set**: Create a random test set that consist 10% of the dataset (after the validation set is created) to be used to determine the effectiveness of each model trained by the train set.

5. **Build Modeling**: 4 type of models were explored to find the most suitable with lowest RMSE:

   1. Generic Effect: Generic effect of Platform, Rating and Genre
   2. Linear model
   3. Random Forest
   4. Regression: Ridge & Lasoo

6. Compute RMSE with Validation Set: Verify the selected model with validation set

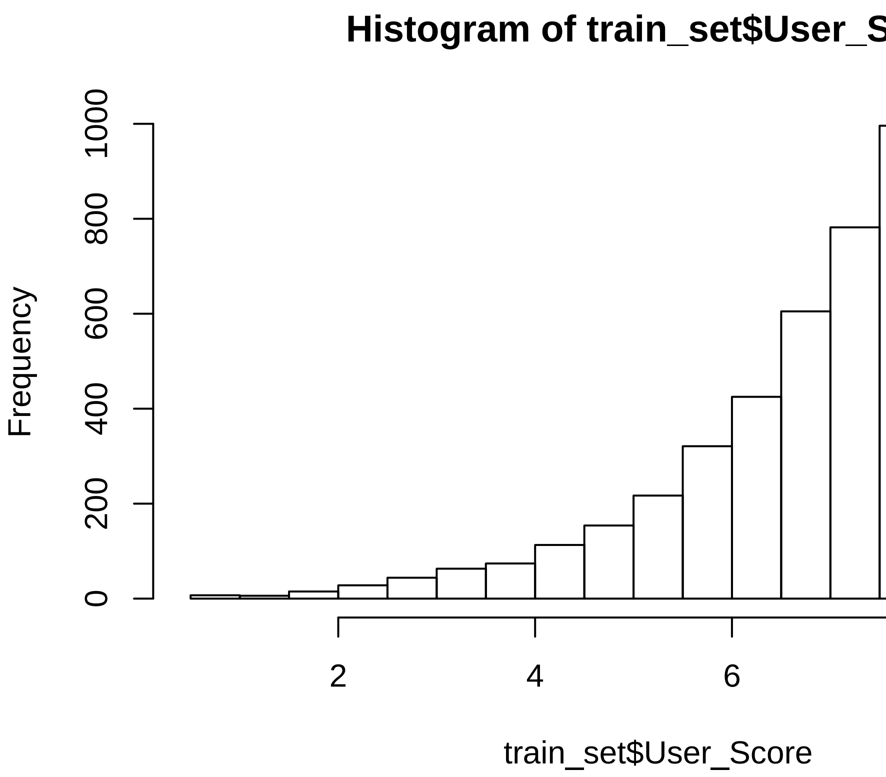
Method

Data Cleaning

```
# Drop NA, unused data and convert year to numeric
mydata_clean <- mydata %>% drop_na() %>%
    mutate(Year_of_Release = 1979 + as.numeric(Year_of_Release),
        User_Score = as.numeric(as.character(User_Score))) %>%
    select(-NA_Sales, -EU_Sales, -JP_Sales, -Other_Sales, -Publisher, -Developer) %>%
    filter(Year_of_Release %in% c(1996:2016))
```

- Remove observations that includes any NA value.
- Year_of_Release variable was originally catagorical and was transformed into continuous numeric variable for data visualization purposes.
- Remove the variables that are will not be used, nominal or consume too much memories when modeling (author is using a Macbook Air and has limitations).
- Filter Year_of_Release to comtemporary video games published after 1996.

Data Exploration & Visualisation



First we look at the **distribution** of User_Score:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   6.500   7.500   7.183   8.200   9.600
```

We can see from the summary and the histogram that the distribution is bell-shaped and has a flatter left shoulder to the sharp right shoulder. The mean (7.183) and median (7.5) are both within the 0.5 point range from the scale with most User Scores (7.5-8.0).

Then we fit a linear model to understand the relationship between User_Score and other variables in the dataset:

```
#Look at coeficient & p-value of variables
train_fit <- train_set %>% select(-Name)
fit <- lm (User_Score~. , train_fit)
summary(fit)
```

```
##
## Call:
```

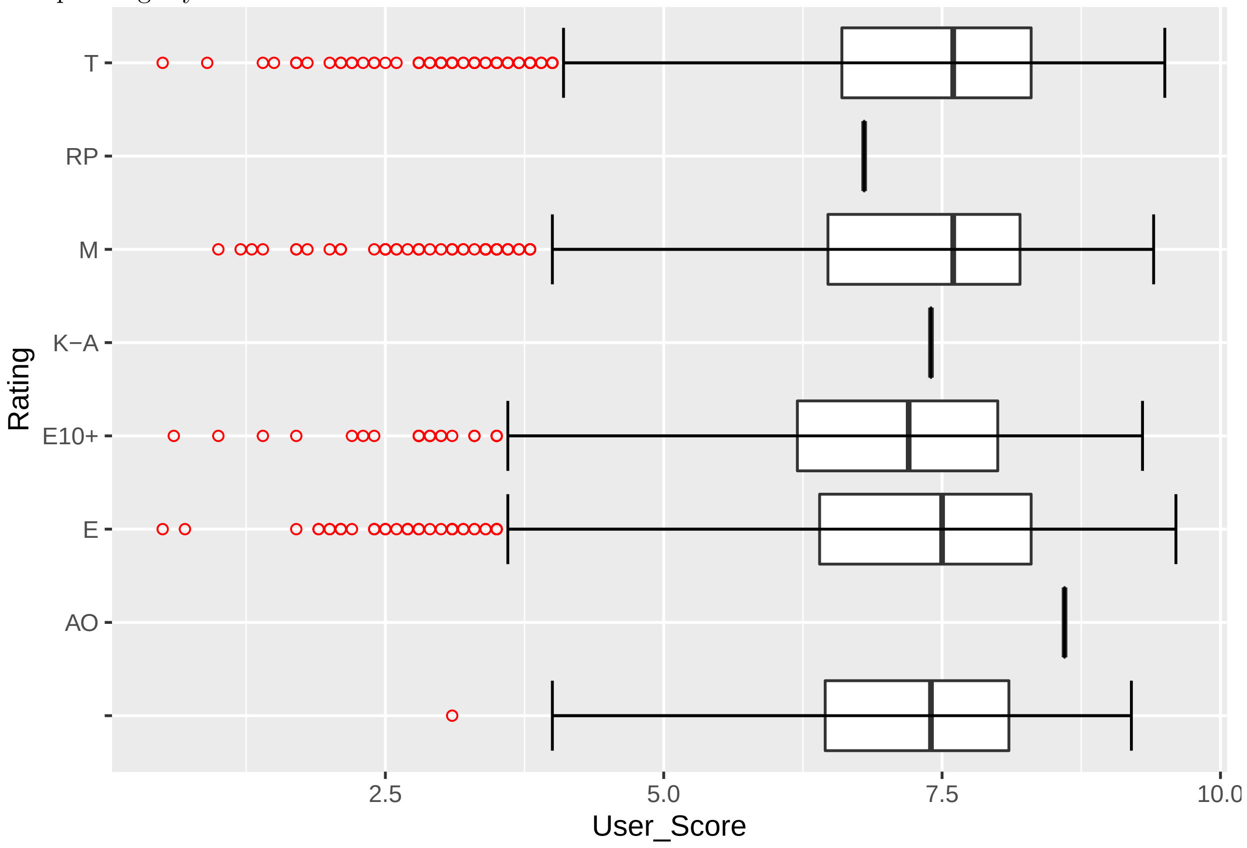
```
## lm(formula = User_Score ~ ., data = train_fit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7932 -0.5110  0.1244  0.6756  4.1088
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.323e+02  1.415e+01   9.352  < 2e-16 ***
## PlatformDC        -6.356e-01  3.658e-01  -1.737 0.082362 .
## PlatformDS        -7.718e-02  1.190e-01  -0.649 0.516594
## PlatformGBA        8.786e-02  1.432e-01   0.614 0.539501
## PlatformGC        -5.293e-02  1.358e-01  -0.390 0.696757
## PlatformPC        -5.329e-01  1.163e-01  -4.582 4.72e-06 ***
## PlatformPS        -1.080e-01  1.699e-01  -0.636 0.524943
## PlatformPS2        1.139e-01  1.231e-01   0.926 0.354572
## PlatformPS3       -4.215e-01  1.102e-01  -3.824 0.000133 ***
## PlatformPS4       -2.689e-01  1.266e-01  -2.124 0.033710 *
## PlatformPSP       -1.497e-02  1.223e-01  -0.122 0.902573
## PlatformPSV        3.912e-01  1.469e-01   2.663 0.007766 **
## PlatformWii        2.588e-02  1.174e-01   0.220 0.825529
## PlatformWiiU       8.500e-02  1.612e-01   0.527 0.597952
## PlatformX360      -4.021e-01  1.107e-01  -3.633 0.000283 ***
## PlatformXB        -1.208e-01  1.289e-01  -0.937 0.348820
## PlatformXOne      -5.230e-01  1.363e-01  -3.838 0.000126 ***
## Year_of_Release   -6.421e-02  7.020e-03  -9.146  < 2e-16 ***
## GenreAdventure     7.794e-02  8.051e-02   0.968 0.333045
## GenreFighting     -5.146e-02  7.016e-02  -0.733 0.463324
## GenreMisc         -2.758e-01  7.004e-02  -3.938 8.33e-05 ***
## GenrePlatform     -4.894e-02  7.001e-02  -0.699 0.484617
## GenrePuzzle       -2.890e-01  1.203e-01  -2.403 0.016305 *
## GenreRacing       -2.324e-01  6.245e-02  -3.721 0.000200 ***
## GenreRole-Playing  1.425e-01  5.527e-02   2.579 0.009925 **
## GenreShooter      -2.047e-01  5.181e-02  -3.951 7.87e-05 ***
## GenreSimulation   -1.286e-01  7.829e-02  -1.643 0.100496
## GenreSports       -4.758e-01  5.902e-02  -8.061 9.17e-16 ***
## GenreStrategy     -1.400e-01  8.180e-02  -1.711 0.087071 .
## Global_Sales      -4.912e-02  9.492e-03  -5.175 2.36e-07 ***
## Critic_Score       6.479e-02  1.235e-03  52.468  < 2e-16 ***
## Critic_Count       3.694e-03  1.015e-03   3.638 0.000277 ***
## User_Count        -2.211e-04  2.999e-05  -7.373 1.92e-13 ***
## RatingAO          -1.000e+00  1.083e+00  -0.923 0.355855
## RatingE           -6.093e-01  1.491e-01  -4.088 4.42e-05 ***
## RatingE10+        -6.074e-01  1.502e-01  -4.045 5.31e-05 ***
## RatingK-A         -2.309e+00  1.088e+00  -2.121 0.033952 *
## RatingM           -5.219e-01  1.486e-01  -3.512 0.000449 ***
## RatingRP           1.360e-01  1.083e+00   0.126 0.900065
## RatingT           -5.244e-01  1.469e-01  -3.569 0.000361 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 5538 degrees of freedom
## Multiple R-squared:  0.4536, Adjusted R-squared:  0.4497
## F-statistic: 117.9 on 39 and 5538 DF,  p-value: < 2.2e-16
```

While from the R-squared value (45%) it does show certain level of relevance between User_Score and other variables, it is too low to indicate a well-fitted model. However, if we divide the variables into two sets base on their class, we can see that the numeric variables all have p-values lower than 0.005, and the majority of the catagorical variables have p-values higher than 0.005. It shows that the numeric variables pass the significance test.

In the next section, we take a deeper look at the catagorical variables:

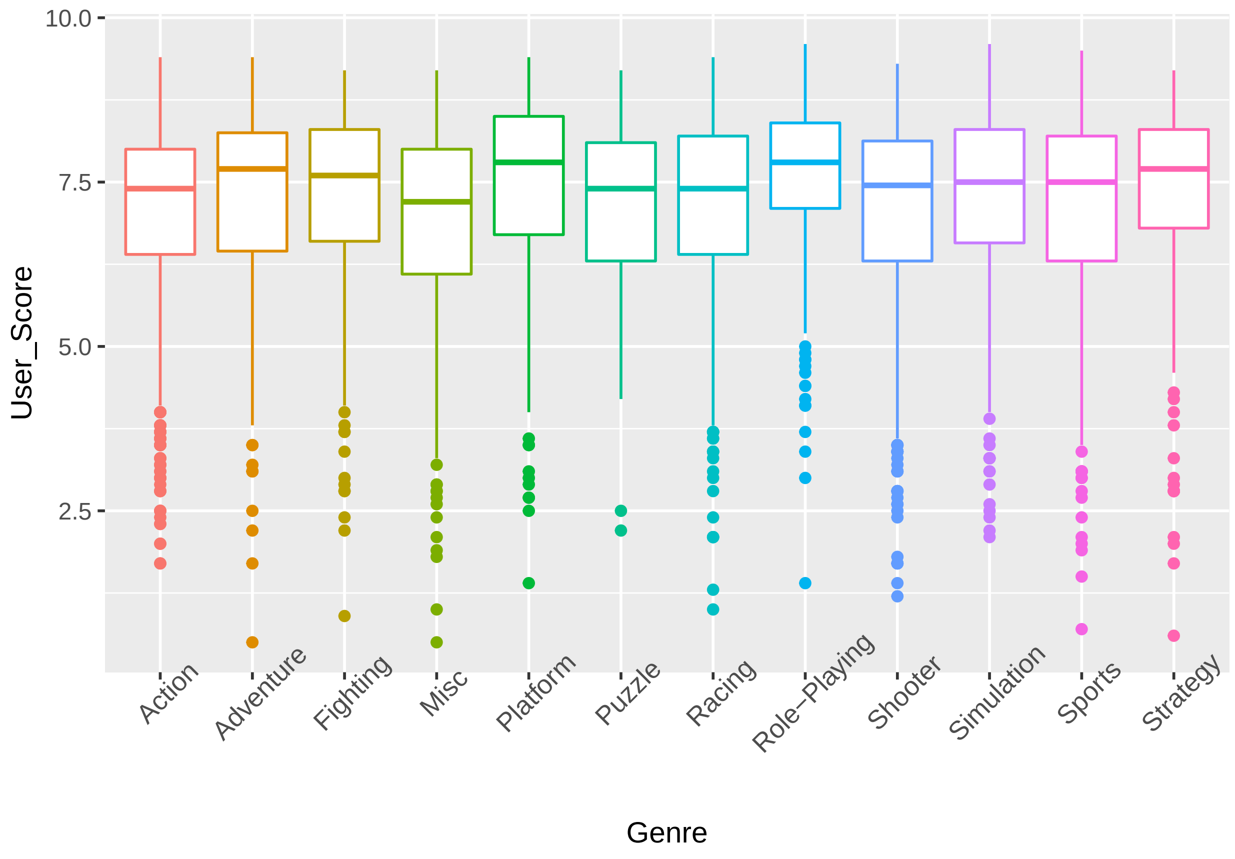
Rating vs. User Score

Apart from the three Rating that only contains one observation, all other Raings has similar median and box size plus slightly more variance in whiskers.



Genre vs. User Score

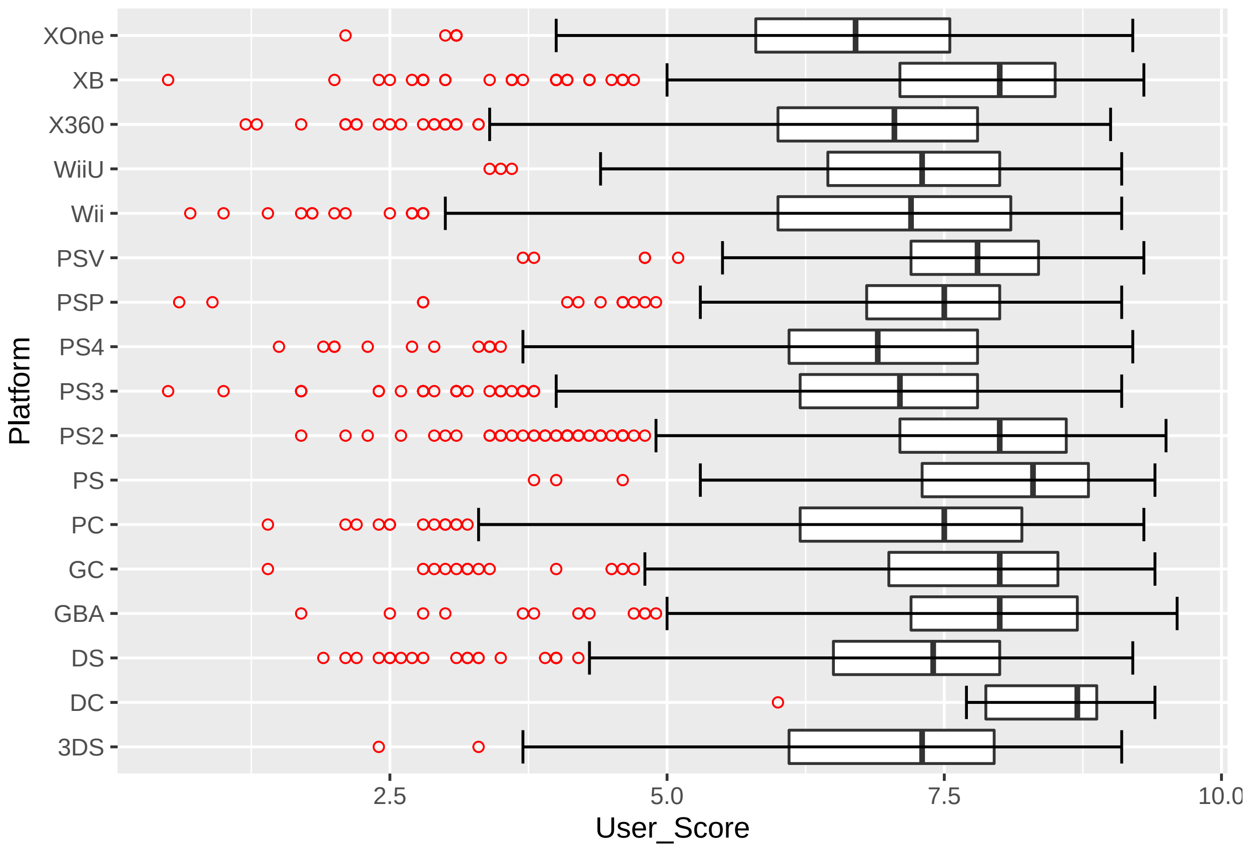
We see a similar boxplot as above. All genres have its User Score median very close to the oeverall median of 7.5 and similar box size. It implies that the variance of User Score between different levels within Rating/Genre variables are not significant, though the difference still exist and might be useful for applying individual effect in for-



mulas.

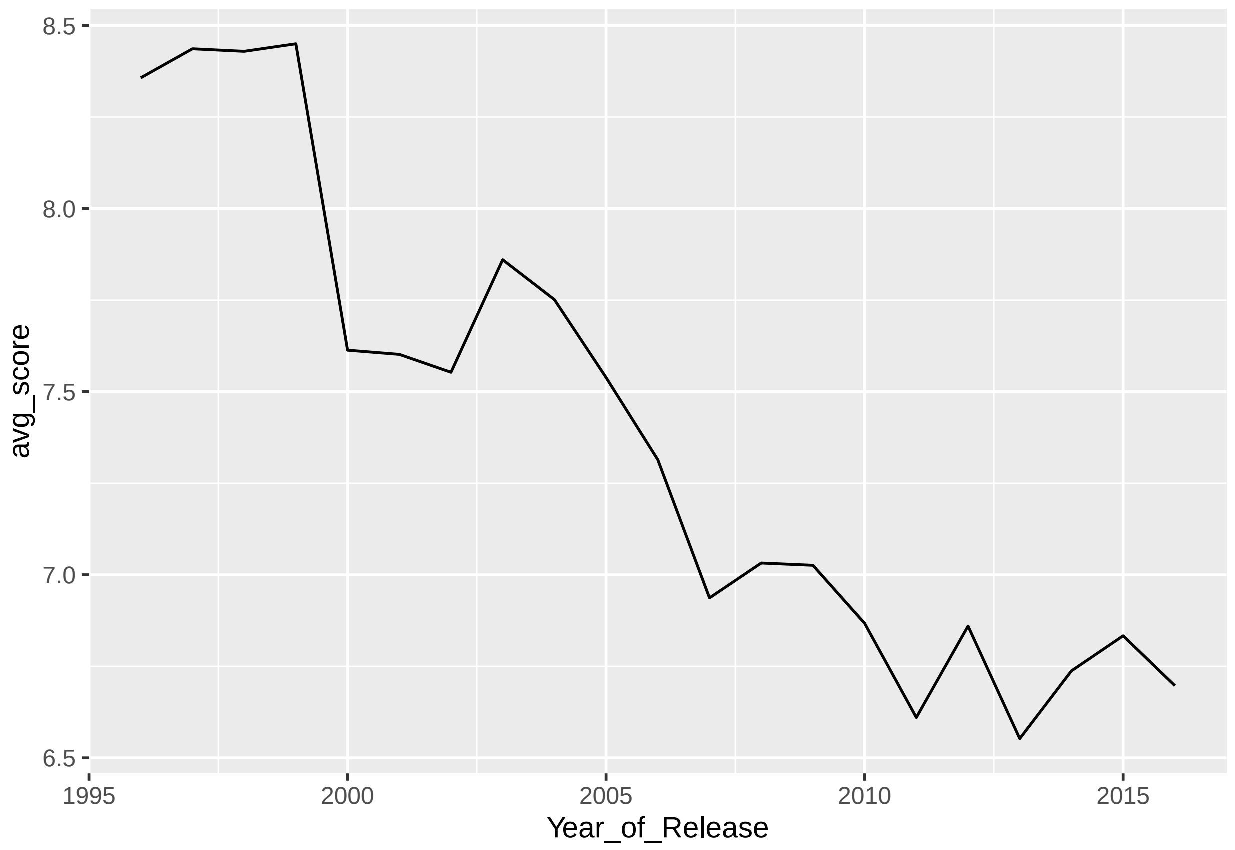
Platform vs. User Score

On the other hand, we see more variance of User Score between difference platfroms in the boxplot below, both in terms of the median and the distance between 1st and 3rd quartiles. One possible explanation is the fact that while a user would make purchases of video games across multiple genres and ratings, the platform they can play the game is restricted to the game consoles that a user owns. Thus the platform effect refects a fixed user base that creates a user effect on the User Score.



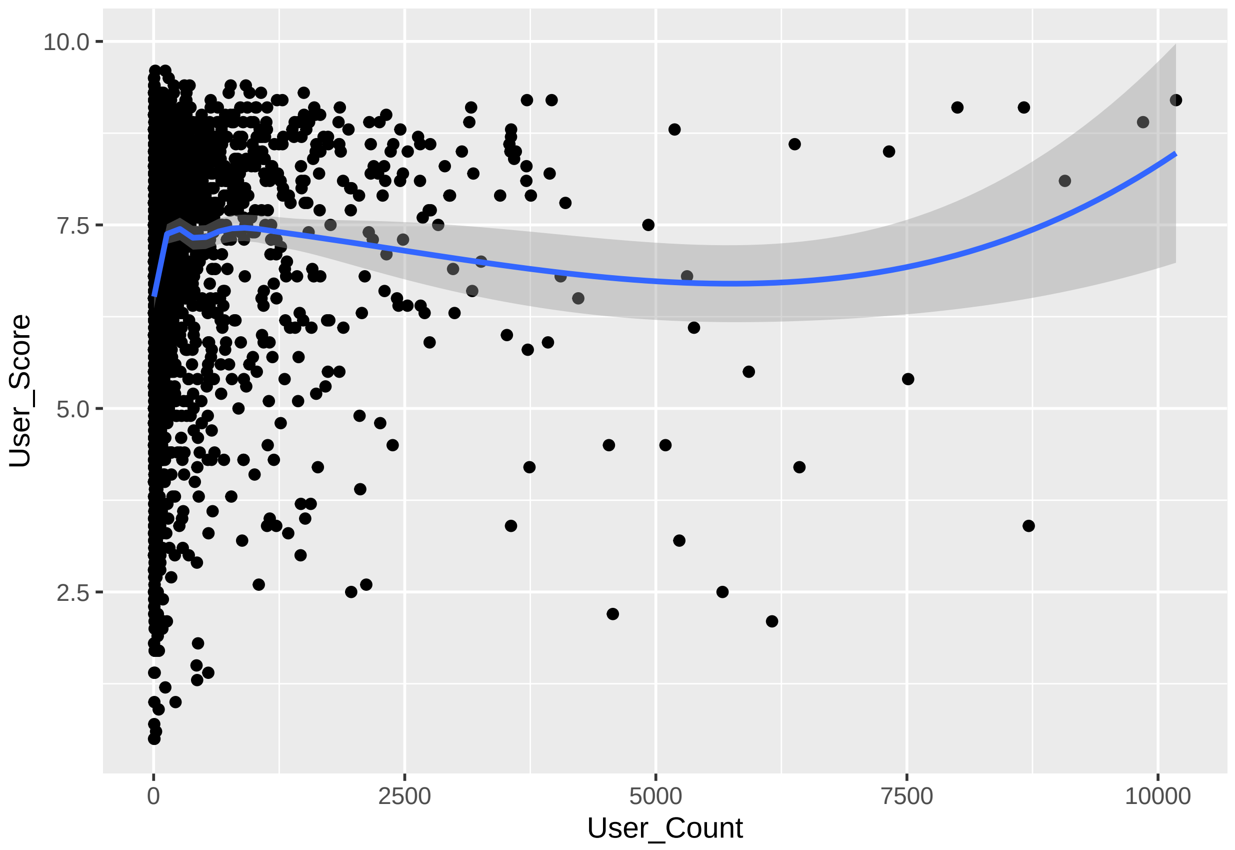
Year of Release vs Average User Score

We can see a trend of decreasing User Score from 1996 to 2010, then it bounced between the same range between the same range since then. It is consistant with the negative coeficient (-6.421e-02) observed in the linear model fitting and that Year of Realease and User Score has negative correlation.



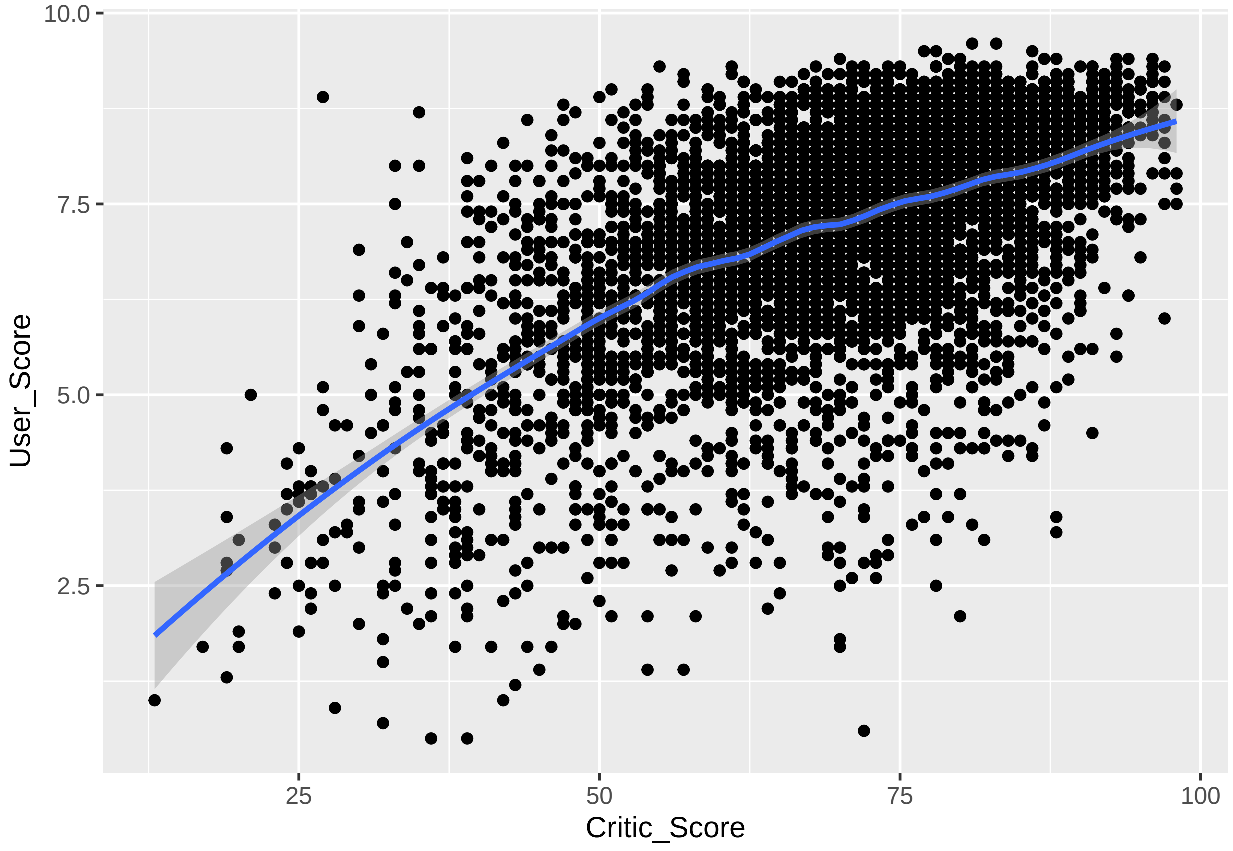
User Count vs User Score

Meanwhile, User Count only has positive correlations with User Score when the User Count is very small (< 250) or very big (> 7500). Between the two marks there are almost no coorelations between the two variables, which is consistant with its small coefficient (3.694e-03).



### Critic Score vs User Score

Lastly, Critic Score has positive correlations with User Scores across all range. It does make sense that Critics and Users have similar reviews towards a specific video game and it is consistant with its postitive coefficcient



(6.479e-02).

Modeling Approach

We first approach the modeling with basic General Effect Model with 3 catagorical variables:

$$\hat{Y} = mu + b_p + b_g + b_r$$

$\hat{Y}$: Predicted value
mu: Average User Score
b_p: Platform Effect
b_g: Genre Effect
b_r: Rating Effect

Model 1: Average User Score

$$\hat{Y} = mu$$

We calculate the average User Score across train set as predicted vales.

```r
# compute average User_Score
mu <- mean(train_set$User_Score)
# compute predicted value
pred_1 <- rep(mu, nrow(test_set))
# compute RMSE
rmse_1 <- RMSE(test_set$User_Score, pred_1)
# add value to results table
```

```r
rmse_results <- tibble(method = "Model 1: Average Score", RMSE = rmse_1)
rmse_results
```

```
## # A tibble: 1 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 Model 1: Average Score  1.39
```

Model 2: Platform Effect

$$\hat{Y} = mu + b_p$$

We calculate Platfrom effect (b_p) for each Platfrom from the train set then left join the value to test set by Platform to compute the predicted value for test set.

```r
#compute b_p: platfrom effect
Platform_effect <- train_set %>% group_by(Platform) %>%
  summarise(b_p = mean(User_Score) - mu)
train_effect <- train_set %>% left_join(Platform_effect, by = "Platform")

# compute predicted value
pred_2 <- test_set %>% mutate(mu = mu) %>%
  left_join(Platform_effect, by = "Platform") %>%
  mutate(pred = mu + b_p) %>% pull(pred)

#comput RMSE and update result table
rmse_2 <- RMSE(test_set$User_Score, pred_2)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 2: Platform Effect", RMSE = rmse_2))
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620

Model 3: Platform + Genre Effect

$$\hat{Y} = mu + b_p + b_g$$

Again we calculate the Genre effect (b_g) the same way as Model 2 but grouping the train set by Genre instead. Left join the value to test set by Genre to compute the predicted vale.

```r
#compute b_g: genre effect
Genre_effect <- train_effect %>% group_by(Genre) %>% summarise(b_g = mean(User_Score) - mu - mean(b_p))
train_effect <- train_effect %>% left_join(Genre_effect, by = "Genre")

# compute predicted value
pred_3 <- test_set %>% mutate(mu = mu) %>%
  left_join(Platform_effect, by = "Platform") %>%
  left_join(Genre_effect, by = "Genre") %>%
  mutate(pred = mu + b_p +b_g ) %>% pull(pred)

#comput RMSE and update result table
```

```r
rmse_3 <- RMSE(test_set$User_Score, pred_3)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 3: Platfrom + Genre Effect", RMSE = rmse_3
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459

Model 4: Platform + Genre + Rating Effect

$$\hat{Y} = mu + b_p + b_g + b_r$$

Repeat the same process for Rating Effect as the two models above.

```r
#compute b_r: rating effect
Rating_effect <- train_effect %>% group_by(Rating) %>% summarise(b_r = mean(User_Score) - mu - mean(b_p
train_effect <- train_effect %>% left_join(Rating_effect, by = "Rating")

# compute predicted value
pred_3 <- test_set %>% mutate(mu = mu) %>%
  left_join(Platform_effect, by = "Platform") %>%
  left_join(Genre_effect, by = "Genre") %>%
  left_join(Rating_effect, by = "Rating") %>%
  mutate(pred = mu + b_p +b_g + b_r ) %>% pull(pred)

#comput RMSE and update result table
rmse_4 <- RMSE(test_set$User_Score, pred_3)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 4: Platform + Genre + Rating Effect", RMSE
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459
Model 4: Platform + Genre + Rating Effect	1.332990

We can see that by adding each variables to the formula, RMSE decreases slightly each time but remains an unsatisfying result above 1.3. In order to include more variables to improve the RMSE and based on the earlier analysis that all numeric variable have p-values below 0.05, we will fit a linear model with all numeric variable first.

Model 5: Linear Model (Numeric Variables)

```r
# fit linear model for numeric values
fit <- lm(User_Score ~ Year_of_Release + Global_Sales + User_Count + Critic_Count + Critic_Score, train

# compute predicted value
pred_5 <- predict(fit, test_set)
```

```r
#comput RMSE and update result table
rmse_5 <- RMSE(test_set$User_Score, pred_5)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 5: Linear Model (Numeric Variables)", RMSE
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459
Model 4: Platform + Genre + Rating Effect	1.332990
Model 5: Linear Model (Numeric Variables)	1.073623

The RMSE from Model 5 is significantly lower that Model 1 to 4. In the earlier anaysis it also showed that some levels within each catagorical variables also have p-vaule lower than 0.005. Therefor we will try to fit the Linear Model again with all variables:

Model 6: Linear Model (All Variables)

```r
# fit linear model for all values
fit <- lm(User_Score ~ Platform + Genre + Rating + Global_Sales + User_Count + Critic_Count + Critic_Sc

# compute predicted value
pred_6 <- predict(fit, test_set)

#comput RMSE and update result table
rmse_6 <- RMSE(test_set$User_Score, pred_6)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 6: Linear Model (All Variables)", RMSE = r
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459
Model 4: Platform + Genre + Rating Effect	1.332990
Model 5: Linear Model (Numeric Variables)	1.073623
Model 6: Linear Model (All Variables)	1.052780

The RMSE result is slightly lower than Model 5. It implies the catagorical variables are worth including to improve the result though the impact is not significant.

Next, we move on to Random Forest model in pursue of higher accuracy. It should be a more suitable choice for the dataset as it consist multi-dimentional catagorical and continuous variables.

Model 7: Random Forest

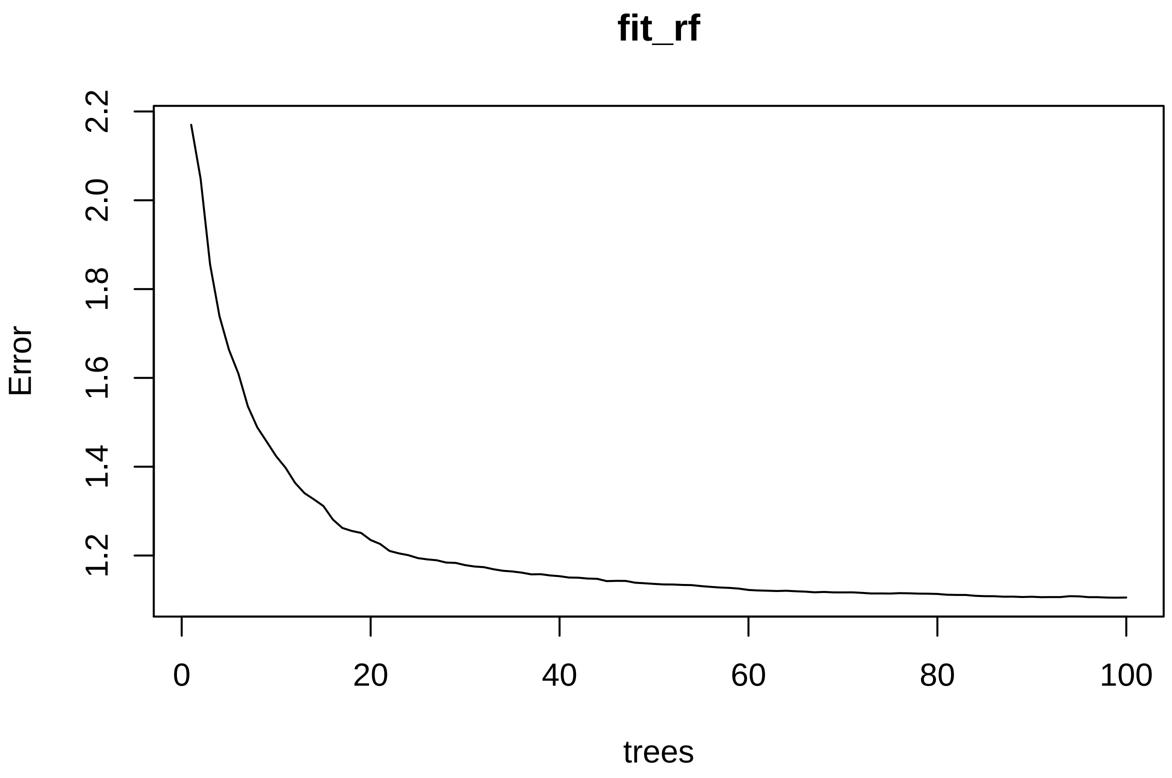
```r
train_rf <- train_set %>% select(-Name)

# fit random forest model with minimum OOB
```

```r
fit_rf <- randomForest(User_Score~., train_rf, ntree = 100)

# compute predicted value
pred_7 <- predict(fit_rf, test_set)

#comput RMSE and update result table
rmse_7 <- RMSE(test_set$User_Score, pred_7)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 7: Random Forest", RMSE = rmse_7))
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459
Model 4: Platform + Genre + Rating Effect	1.332990
Model 5: Linear Model (Numeric Variables)	1.073623
Model 6: Linear Model (All Variables)	1.052780
Model 7: Random Forest	1.000700

The plot below shows that we reach the minumum OOB at 100 for ntree.

```r
#Plot minimum OOB
plot(fit_rf)
```

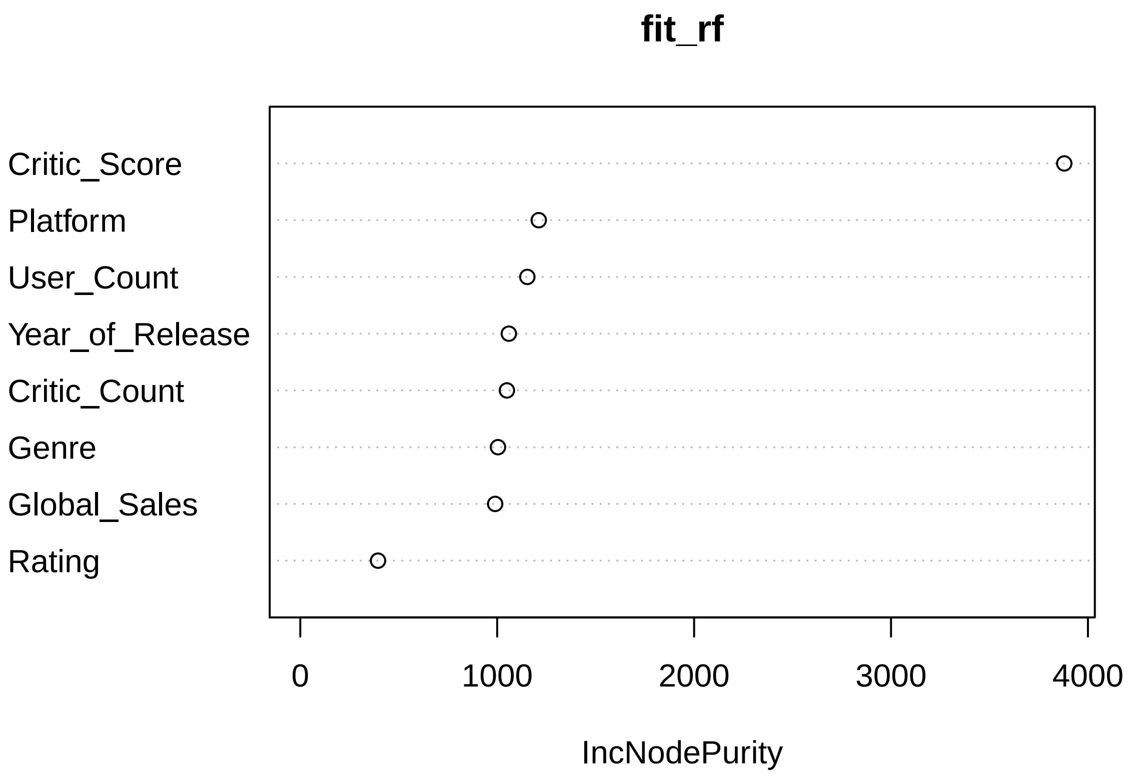


And the importance of each variables:

```r
# variable importance
importance(fit_rf)
```

```
##                    IncNodePurity
## Platform              1211.1257
## Year_of_Release       1059.1772
## Genre                 1004.0305
## Global_Sales           989.7013
## Critic_Score          3879.8726
## Critic_Count          1049.3799
## User_Count            1153.1353
## Rating                 394.8007
```

```
#Plot variable importance
varImpPlot(fit_rf)
```



Lastly we look at the Regularization models. We use Ridge and Lasso Regression as the dataset has certain level of multicollinearity (ex. User Count and Crtic Count).

Model 8: Ridge Regression

```
#Prep matrix for the model
x_var <- data.matrix(train_set[, -c(1,8)])
y_var <- train_set$User_Score
lambda_seq <- 10^seq(-2, 5, length = 100)

# fit the model for minimum lambda
fit_cv <- cv.glmnet(x_var, y_var, alpha = 0, lambda = lambda_seq)
lambda_min <- fit_cv$lambda.min

# fit ridge regression model with min lambda
fit_ridge <- glmnet(x_var, y_var, alpha = 0, lambda  = lambda_min)
```

```
# prep test set - remove name and User_score
test_x <- data.matrix(test_set[, -c(1,8)])
# compute predicted values
pred_8 <- predict(fit_ridge, s = lambda_min, newx = test_x)
# compute RMSE and update results table
rmse_8 <- RMSE(test_set$User_Score, pred_8)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 8: Ridge Regression ", RMSE = rmse_8))
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459
Model 4: Platform + Genre + Rating Effect	1.332990
Model 5: Linear Model (Numeric Variables)	1.073623
Model 6: Linear Model (All Variables)	1.052780
Model 7: Random Forest	1.000700
Model 8: Ridge Regression	1.062945

Model 9: Lasso Regression

```
# fit the model for minimum lambda
fit_la <- cv.glmnet(x_var, y_var, alpha = 1)
lambda_min <- fit_la$lambda.min

# fit lasso regression model with min lambda
fit_lasso <- glmnet(x_var, y_var, alpha = 1, lambda  = lambda_min)
# prep test set - remove name and User_score
test_x <- data.matrix(test_set[, -c(1,8)])
# compute predicted values
pred_9 <- predict(fit_lasso, s = lambda_min, newx = test_x)
# compute RMSE and update results table
rmse_9 <- RMSE(test_set$User_Score, pred_9)
rmse_results <- bind_rows(rmse_results, tibble(method="Model 9: Lasso Regression ", RMSE = rmse_9))
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459
Model 4: Platform + Genre + Rating Effect	1.332990
Model 5: Linear Model (Numeric Variables)	1.073623
Model 6: Linear Model (All Variables)	1.052780
Model 7: Random Forest	1.000700
Model 8: Ridge Regression	1.062945
Model 9: Lasso Regression	1.062693

Results

```
rmse_results %>% kable()
```

method	RMSE
Model 1: Average Score	1.389298
Model 2: Platform Effect	1.344620
Model 3: Platfrom + Genre Effect	1.336459
Model 4: Platform + Genre + Rating Effect	1.332990
Model 5: Linear Model (Numeric Variables)	1.073623
Model 6: Linear Model (All Variables)	1.052780
Model 7: Random Forest	1.000700
Model 8: Ridge Regression	1.062945
Model 9: Lasso Regression	1.062693

We can see from the table that **Random Forest** model produces the lowest RMSE, followed by **Linear Model** then **Regularization models** in a close 3rd place. Random Forest is the model that fits the best likely due to:
1. Its flexibility to accomodate multidimentional variables. 2. The dataset has both catagorical and continuous numeric values. 3. In the categorical variables, only certain levels has significant impact on the prediction and Random Forest can capture this trait.

We now verify the validation set with Random Forest:

```
# compute predicted value for validation set
pred_vali <- predict(fit_rf, validation)

#comput RMSE and update result table
rmse_vali <- RMSE(validation$User_Score, pred_vali)
rmse_vali
```

```
## [1] 1.032095
```

The RMSE for the validation set is 1.0320946.

Conclusion

Summary

The report focused on predicting the User Score from the variables avaiable from the Video Games 2016 dataset, including Catagorical variables (Platform, Rating and Genre) and numeric variables (Year of Release, Global Sales, User Count, Critic Count and Critic Score).

In data exploration and analysis we foused on the visualisation of User Score among the levels for catagorical variables, as well as the significance of each variables through linear model diagnosis.

The dataset was then divided into train set and test set to build modeling to predict values that produces the lowest RMSE. Among the Generic Effect (Platform, Rating and Genre), Linear Model, Random Forest and Regularizations (Ridge and Lasso Regression) models, we found that Random Forest produces the lowest RMSE thus the most suitable model to predict User Score.

Potential Impact

The algorithm will be able to help the Gaming industry predict what review scores a video games is likely to receive from the users, and use the machine learning concept to make better decisions based on the predicted user score from this algorithm.

Limitations

The author runs the project on an Macbook Air 2014 with OSX El Capitan. Due to the limitation on memory, some variables (Regional Sales and Publishers/Developers) had to be removed at the Data Cleaning stage to accomodate the fitting of Linear Model , Random Forest and Regularization.

It is very likely that the RMSE would be lower for all the models used in the project if the author can accomodation more variables, as Regional Sales will provide insights that reflects the preference of the users from different regions, as well as the correlation with Platforms and Genre. The Publisher/Developer variables will show us if brand loyalty affect User Scores.

Future Work

For the Gaming industry, they could further explore the optimized combination for the games that record higher user scores - especially for the parameters that could be pre-determined at the planning stage of game developement. For example, Sports games for PS4 receive higher user scores due to the comfort of the controller, or Thriller games tend to receive higher user scores when it is rated for older age groups.