|  |
| --- |
| --- |
|  | title: "Video Game Sales Analysis" |
|  | author: "Linh Hua" |
|  | date: "December 12, 2020" |
|  | output: pdf\_document |
|  | fontsize: 11pt |
|  | --- |
|  |  |
|  | ```{r setup, include=FALSE} |
|  | knitr::opts\_chunk$set(echo = TRUE) |
|  | ``` |
|  |  |
|  | **## Contents** |
|  |  |
|  | 1 Introduction ......................................................................... 1\ |
|  | 2 Methods and Analysis .............................................................. 2-9\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp; 2.1 Data Generation and Cleaning ............................................... 2-3\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp; 2.2 Data Visualization and Exploration .......................................... 3-9 \ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp; 2.3 Data Modeling .................................................,............ 9-16\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; 2.3.1 kNN Model ........................................................... 10-11\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; 2.3.2 Naive Bayes Model ................................................... 12-13\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; 2.3.3 Random Forests Model ................................................ 13-16\ |
|  | 3 Results and Conclusions ........................................................... 17-19\ |
|  | 4 Reference ........................................................................... 19\ |
|  | 5 Appendix - Environment ............................................................ 19\ |
|  |  |
|  |  |
|  | **## 1 Introduction** |
|  |  |
|  | As part of the HarvardX's Data Science Professional Certificate Program, students get the opportunity to perform formal data science analyses in our Capstone course using the knowledge gained from our coursework.\ |
|  |  |
|  | We have selected video games sales for our analysis. Since their debut, video games have exploded in popularity through the years. This year video games in the U.S alone, has produced on average over 1 billion dollars in sales per month as reported on www.statista.com. Being such a significant part of the market share, it is exciting to dive in and learn more about the video game business.\ |
|  |  |
|  | The main objective of our project is to apply machine learning techniques that go beyond standard linear regression using a publicly available data set to solve a problem of our choice. We will generate our data set, interpret the data, perform various algorithms/modelings and present our insights. We will review which video games were popular, explore and look for trends and develop some prediction models for video game ranking and sales.\ |
|  |  |
|  | **## 2 Methods and Analysis** |
|  |  |
|  | 2.1 Data Generation and Cleaning\ |
|  |  |
|  | The video games sales data was downloaded from Kaggle (https://www.kaggle.com/gregorut/videogamesales) as a vgsales.csv file. We imported the data programmatically with the R script below. We will use the below script to generate and prepare this vgsales data. We have cleaned the data by removing the rows that were not applicable from the 'Year' and 'Publisher' columns. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | ################################################################################ |
|  | # Generate the vgsales (video game sales) data |
|  | ################################################################################ |
|  |  |
|  | # Install libraries if they have not been installed yet |
|  | 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(dslabs)) install.packages("dslabs", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(dslabs)) install.packages("dslabs", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(class)) install.packages("class", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(knitr)) install.packages("knitr", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(kableExtra)) install.packages("kableExtra", |
|  | repos = "http://cran.us.r-project.org") |
|  |  |
|  | # Load libraries |
|  | library(tidyverse) |
|  | library(caret) |
|  | library(data.table) |
|  | library(dslabs) |
|  | library(class) |
|  | library(randomForest) |
|  | library(knitr) |
|  | library(kableExtra) |
|  |  |
|  | # Video games sales data set: |
|  | # https://www.kaggle.com/gregorut/videogamesales |
|  | # Manually downloaded and included the vgsales.csv file |
|  | # Use the below code to import programmically or use R Studio to import the data |
|  | filename <- "vgsales.csv" |
|  | dir <- system.file("extdata", package = "dslabs") |
|  | fullpath <- file.path(dir, filename) |
|  | file.copy(fullpath, "vgsales.csv") |
|  |  |
|  | vgsales <- read\_csv(filename) |
|  |  |
|  | ################################################################################ |
|  | # Clean the data |
|  | ################################################################################ |
|  |  |
|  | # Remove rows with N/A values in the Year column |
|  | vgsales <- vgsales %>% filter(Year != "N/A") |
|  |  |
|  | # Remove rows with N/A values in the Publisher column |
|  | vgsales <- vgsales %>% filter(Publisher != "N/A") |
|  |  |
|  | ``` |
|  |  |
|  | 2.2 Data Visualization and Exploration\ |
|  |  |
|  | To familiarize ourselves with the data, we ran the following scripts to analyze vgsales (see below). In doing so, we knew vgsales had 16,291 rows and 11 variables. The variables were:\ |
|  |  |
|  | Rank - Ranking of overall sales\ |
|  | Name - The games name\ |
|  | Platform - Platform of the games release (i.e. PC,PS4, etc.)\ |
|  | Year - Year of the game's release\ |
|  | Genre - Genre of the game\ |
|  | Publisher - Publisher of the game\ |
|  | NA\_Sales - Sales in North America (in millions)\ |
|  | EU\_Sales - Sales in Europe (in millions)\ |
|  | JP\_Sales - Sales in Japan (in millions)\ |
|  | Other\_Sales - Sales in the rest of the world (in millions)\ |
|  | Global\_Sales - Total worldwide sales\ |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # View the structure of the data set |
|  | str(vgsales) |
|  |  |
|  | ``` |
|  |  |
|  | ![Vgsales Structure](imgs\vgs\_structure.png)\ |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # View data summary |
|  | summary(vgsales) |
|  |  |
|  | ``` |
|  |  |
|  | ![Vgsales Summary](imgs\vgs\_summary.png)\ |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # View number of rows |
|  | nrow(vgsales) |
|  |  |
|  | # View number of columns |
|  | ncol(vgsales) |
|  |  |
|  | ``` |
|  |  |
|  | The video game sales data were from these 5 areas: North America, Europe, Japan, Other and Global. There are 576 game publishers and 31 platforms. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # View number of different countries |
|  | platforms <- unique(vgsales$Platform) |
|  | platforms |
|  |  |
|  | # Calculate the mean of game sales in North America |
|  | avg\_NA\_sales <- mean(vgsales$NA\_Sales) |
|  | avg\_NA\_sales |
|  |  |
|  | # Calculate the standard deviation of game sales in North America |
|  | sd\_NA\_sales <- sd(vgsales$NA\_Sales) |
|  | sd\_NA\_sales |
|  |  |
|  | # Calculate the mean of global game sales |
|  | avg\_G\_sales <- mean(vgsales$Global\_Sales) |
|  | avg\_G\_sales |
|  |  |
|  | # Calculate the standard deviation of global game sales |
|  | sd\_G\_sales <- sd(vgsales$Global\_Sales) |
|  | sd\_G\_sales |
|  |  |
|  | ``` |
|  |  |
|  | ![Video Game Platforms](imgs\vgs\_platforms.png)\ |
|  |  |
|  | The average video games sales was 0.265 million dollars and the standard deviation was 0.822 in the North America. In comparison, the average video games sales was 0.540 million dollars and the standard deviation was 1.567 globally. Larger standard deviations indicate the data were more spread out in both cases.\ |
|  |  |
|  | From the plots below, we gleaned that the top 3 game platforms were Nintendo DS, PlayStation 2 and PlayStation 3. The top 3 years to release the most games were 2008, 2009 and 2010. The top 3 game genres were Action, Sports and Miscellaneous. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # Bar plot: Platform counts |
|  | vgsales %>% |
|  | ggplot(aes(Platform)) + |
|  | labs(x = "Platforms") + |
|  | geom\_bar() + |
|  | coord\_flip() |
|  |  |
|  | # Bar plot: Games release by year counts |
|  | vgsales %>% |
|  | ggplot(aes(Year)) + |
|  | labs(x = "Game Release Year") + |
|  | geom\_bar(fill = "blue", col = "black") + |
|  | coord\_flip() |
|  |  |
|  | # Bar plot: Genre counts |
|  | vgsales %>% |
|  | ggplot(aes(Genre)) + |
|  | labs(x = "Genres") + |
|  | geom\_bar(fill = "green", col = "black") + |
|  | coord\_flip() |
|  |  |
|  | ``` |
|  |  |
|  | ![Platform Counts](imgs\vgs\_platforms\_bar.png)\ |
|  |  |
|  | ![Games Release by Year Counts](imgs\vgs\_game\_release\_count.png)\ |
|  |  |
|  | ![Genre Counts](imgs\vgs\_genres.png)\ |
|  |  |
|  | We compared the video games sales between North America and Global respectively below: |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # Scatter plot: North America sales by Year |
|  | vgsales %>% |
|  | ggplot() + |
|  | geom\_point(aes(x = Year, y = NA\_Sales)) + |
|  | labs(x = "Years", y = "North America Sales") + |
|  | theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | # Scatter plot: Global sales by Year |
|  | vgsales %>% |
|  | ggplot() + |
|  | geom\_point(aes(x = Year, y = Global\_Sales)) + |
|  | labs(x = "Years", y = "Global Sales") + |
|  | theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | ![North America Sales](imgs\north-america-sales.png)\ |
|  |  |
|  | ![Global Sales](imgs\global-sales.png)\ |
|  |  |
|  | The worst sellers were Monster Hunter Freedom 3 ($0) in North America and Turok ($0.01 million) globally. We know that Wii Sports was a top seller in North America and globally. Wii Sports had $41.49 million in North America and $82.74 million globally. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # View bottom game sales in North America |
|  | worst\_game\_sales <- which.min(vgsales$NA\_Sales) |
|  | vgsales$Name[worst\_game\_sales] |
|  | min(vgsales$NA\_Sales) |
|  |  |
|  | # View top game sales in North America |
|  | best\_game\_sales <- which.max(vgsales$NA\_Sales) |
|  | vgsales$Name[best\_game\_sales] |
|  | max(vgsales$NA\_Sales) |
|  |  |
|  | # View bottom game sales Globally |
|  | worst\_game\_sales\_g <- which.min(vgsales$Global\_Sales) |
|  | vgsales$Name[worst\_game\_sales\_g] |
|  | min(vgsales$Global\_Sales) |
|  |  |
|  | # View top game sales Globally |
|  | best\_game\_sales\_g <- which.max(vgsales$Global\_Sales) |
|  | vgsales$Name[best\_game\_sales\_g] |
|  | max(vgsales$Global\_Sales) |
|  |  |
|  | ``` |
|  |  |
|  | 2.3 Data Modeling\ |
|  |  |
|  | We focused primarily on North America game sales from here on. In our algorithm and modeling development, we applied the kNN, Naive Bayes and Random Forests approaches to determine which model yields highest accuracy in games reaching their sales goals.\ |
|  |  |
|  | We chose to split our training and test set with (70/30) in order to limit the amount of variance in estimates. We proceeded to partition the vgsales data to perform our modeling as follows:\ |
|  |  |
|  | 2.3.1 kNN Model\ |
|  |  |
|  | In the kNN Model, we began with selecting the predictors and left out the Genre data (and others) as its our target study. We built the model with the data normalized and used the square root of the number of observations to obtain the K value of 106 and 107. Then we calculated the accuracy of these created models. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | ##### kNN Model: ##### |
|  |  |
|  | ## Normalization |
|  | normalize <- function(x) { |
|  | return ((x - min(x)) / (max(x) - min(x))) } |
|  |  |
|  | ## Selecting our predictors and translating state to status |
|  | vgsales\_subset <- vgsales %>% |
|  | select(NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales, Global\_Sales) |
|  |  |
|  | ## Creating the normalized subset |
|  | vgsales\_subset\_n <- as.data.frame(lapply(vgsales\_subset[,1:5], normalize)) |
|  |  |
|  | ## Split data into a training set and a test set by randomly selection |
|  | set.seed(100, sample.kind = "Rounding") |
|  | d <- sample(1:nrow(vgsales\_subset\_n),size=nrow(vgsales\_subset\_n)\*0.7, |
|  | replace = FALSE) |
|  |  |
|  | train\_set <- vgsales\_subset[d,] # 70% training data |
|  | test\_set <- vgsales\_subset[-d,] # remaining 30% test data |
|  |  |
|  | ## Creating separate data frame for 'status' feature which is our target. |
|  | train\_labels <- vgsales\_subset[d,1] |
|  | test\_labels <- vgsales\_subset[-d,1] |
|  |  |
|  | ## Find the number of observations |
|  | NROW(train\_labels) |
|  |  |
|  | ## Square root of 11,403 is 106.78 and so we will create 2 models. |
|  | ## One with ‘K’ value as 106 and the other model with a ‘K’ value as 107. |
|  | knn\_106 <- knn(train=train\_set, test=test\_set, cl=train\_labels$NA\_Sales, k=106) |
|  | knn\_107 <- knn(train=train\_set, test=test\_set, cl=train\_labels$NA\_Sales, k=107) |
|  |  |
|  | ## Calculate the proportion of correct classification for k = 106, 107 |
|  | ACC\_106 <- 100 \* sum(test\_labels$NA\_Sales == knn\_106)/NROW(test\_labels$NA\_Sales) |
|  | ACC\_107 <- 100 \* sum(test\_labels$NA\_Sales == knn\_107)/NROW(test\_labels$NA\_Sales) |
|  | ACC\_106 |
|  | # [1] 54.50082 |
|  | ACC\_107 |
|  | # [1] 54.41899 |
|  |  |
|  | ``` |
|  |  |
|  | We also created a loop that calculates the accuracy of the KNN model for ‘K’ values ranging from 1 to 108. This way we can check which ‘K’ value will result in more accurate model. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | ## Loop that calculates the accuracy of the kNN model |
|  | i=1 |
|  | k\_optm=1 |
|  | for (i in 1:108){ |
|  | knn\_mod <- knn(train=train\_set, test=test\_set, cl=train\_labels$NA\_Sales, k=i) |
|  | k\_optm[i] <- 100 \* sum(test\_labels$NA\_Sales==knn\_mod)/NROW(test\_labels$NA\_Sales) |
|  | k=i |
|  | cat(k,'=',k\_optm[i],'') |
|  | } |
|  |  |
|  | ## Accuracy plot |
|  | plot(k\_optm, type="b", xlab="K- Value",ylab="Accuracy level") |
|  |  |
|  | ``` |
|  |  |
|  | The below graph shows that for ‘K’ value of 1 got the maximum accuracy at 76.636.\ |
|  |  |
|  | ![kNN Model Plot](imgs\kNN.png)\ |
|  |  |
|  | 2.3.2 Naive Bayes Model\ |
|  |  |
|  | In the Naive Bayes Model, we used the Genre variable to demonstrate our algorithm. |
|  | We translated the Genre variable into a new variable called Genre\_Spec. It was mutated to contain only the values "Action" or "Not Action". Then we perform the below steps to illustrate how we calculate the prevalence, average and standard deviation to achieve the naive bayes values with bias and without bias. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | ##### Naive Bayes Model: ##### |
|  |  |
|  | ## Selecting our predictors |
|  | vgsales\_subset <- vgsales %>% |
|  | select(Genre, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales, Global\_Sales) |
|  |  |
|  | # Translate the Genre column to Action or Not Action and save it to Genre\_Spec |
|  | vgsales\_subset <- vgsales\_subset %>% |
|  | mutate(Genre\_Spec = ifelse(Genre == "Action", "Action", "Not Action")) |
|  |  |
|  | ## Split data into train and test sets |
|  | set.seed(1995) |
|  | test\_index <- createDataPartition(vgsales\_subset, times = 1, p = 0.5, list = FALSE) |
|  | train\_set <- vgsales\_subset %>% slice(-test\_index) # 50% data |
|  | test\_set <- vgsales\_subset %>% slice(test\_index) # 50% remaining data |
|  |  |
|  | params <- train\_set %>% |
|  | group\_by(Genre\_Spec) %>% |
|  | summarize(avg = mean(NA\_Sales), sd = sd(NA\_Sales)) |
|  |  |
|  | pi <- train\_set %>% summarize(pi=mean(Genre\_Spec=="Action")) %>% pull(pi) |
|  | pi |
|  |  |
|  | x <- test\_set$NA\_Sales |
|  |  |
|  | f0 <- dnorm(x, params$avg[2], params$sd[2]) |
|  | f1 <- dnorm(x, params$avg[1], params$sd[1]) |
|  |  |
|  | p\_hat\_bayes <- f1\*pi / (f1\*pi + f0\*(1 - pi)) |
|  |  |
|  | y\_hat\_bayes <- ifelse(p\_hat\_bayes > 0.5, "Action", "Not Action") |
|  | sensitivity(data = factor(y\_hat\_bayes), reference = factor(test\_set$Genre\_Spec)) |
|  | # [1] 0 |
|  | specificity(data = factor(y\_hat\_bayes), reference = factor(test\_set$Genre\_Spec)) |
|  | # [1] 1 |
|  |  |
|  | ## Now we do the same as above but unbiased |
|  | p\_hat\_bayes\_unbiased <- f1 \* 0.5 / (f1 \* 0.5 + f0 \* (1 - 0.5)) |
|  | y\_hat\_bayes\_unbiased <- ifelse(p\_hat\_bayes\_unbiased > 0.5, "Action", "Not Action") |
|  |  |
|  | sensitivity(factor(y\_hat\_bayes\_unbiased), factor(test\_set$Genre\_Spec)) |
|  | # [1] 0.9342508 |
|  | specificity(factor(y\_hat\_bayes\_unbiased), factor(test\_set$Genre\_Spec)) |
|  | # [1] 0.0555767 |
|  |  |
|  | # Plotting the new rule at 1 |
|  | qplot(x, p\_hat\_bayes\_unbiased, geom = "line") + |
|  | geom\_hline(yintercept = 0.5, lty = 2) + |
|  | geom\_vline(xintercept = 1, lty = 2) |
|  |  |
|  | ``` |
|  |  |
|  | ![Naive Bayes Model Plot](imgs\naive-bayes.png)\ |
|  |  |
|  | 2.3.3 Random Forests Model\ |
|  |  |
|  | In the last Random Forests Model, we used the Genre\_No variable as the focus for this algorithm. The randomForest() function requires numeric input values so we translated the Genre character values to numeric values and saved them as the Genre\_No variable. |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | ##### Random Forests Model: ##### |
|  |  |
|  | ## Translate the values of Genre into Genre\_No as follows: |
|  | train\_set <- train\_set %>% |
|  | mutate(Genre\_No = case\_when( |
|  | Genre == "Strategy" ~ 1, |
|  | Genre == "Sports" ~ 2, |
|  | Genre == "Simulation" ~ 3, |
|  | Genre == "Shooter" ~ 4, |
|  | Genre == "Role-Playing" ~ 5, |
|  | Genre == "Racing" ~ 6, |
|  | Genre == "Puzzle" ~ 7, |
|  | Genre == "Platform" ~ 8, |
|  | Genre == "Misc" ~ 9, |
|  | Genre == "Fighting" ~ 10, |
|  | Genre == "Adventure" ~ 11, |
|  | Genre == "Action" ~ 12, |
|  | )) |
|  |  |
|  | test\_set <- test\_set %>% |
|  | mutate(Genre\_No = case\_when( |
|  | Genre == "Strategy" ~ 1, |
|  | Genre == "Sports" ~ 2, |
|  | Genre == "Simulation" ~ 3, |
|  | Genre == "Shooter" ~ 4, |
|  | Genre == "Role-Playing" ~ 5, |
|  | Genre == "Racing" ~ 6, |
|  | Genre == "Puzzle" ~ 7, |
|  | Genre == "Platform" ~ 8, |
|  | Genre == "Misc" ~ 9, |
|  | Genre == "Fighting" ~ 10, |
|  | Genre == "Adventure" ~ 11, |
|  | Genre == "Action" ~ 12, |
|  | )) |
|  |  |
|  | ## Summary of data in train and test sets |
|  | summary(train\_set) |
|  | summary(test\_set) |
|  |  |
|  | ## Create a Random Forest model with default parameters |
|  | model1 <- randomForest(Genre\_No ~ ., data = train\_set, importance = TRUE) |
|  | model1 |
|  |  |
|  | model2 <- randomForest(Genre\_No ~ ., data = train\_set, ntree = 500, mtry = 6, |
|  | importance = TRUE) |
|  | model2 |
|  |  |
|  | ``` |
|  |  |
|  | ![Random Forests Model 1](imgs\rf-model1.png)\ |
|  |  |
|  | ![Random Forests Model 2](imgs\rf-model2.png)\ |
|  |  |
|  | ```{r echo=TRUE, eval=FALSE} |
|  |  |
|  | # Predicting on train set |
|  | predTrain <- predict(model2, train\_set, type = "class") |
|  |  |
|  | # Checking classification accuracy |
|  | table(predTrain, train\_set$Genre\_No) |
|  |  |
|  | # Predicting on test set |
|  | predValid <- predict(model2, test\_set, type = "class") |
|  |  |
|  | # Checking classification accuracy |
|  | mean(predValid == test\_set$Genre\_No) |
|  | table(predValid, test\_set$Genre\_No) |
|  |  |
|  | # To check important variables |
|  | importance(model2) |
|  | varImpPlot(model2) |
|  |  |
|  | ``` |
|  |  |
|  | ![Random Forests Model 2 Importance](imgs\rf-imp.png)\ |
|  |  |
|  | ![Random Forests Model 2 Plot](imgs\random-forests-varImpPlot.png)\ |
|  |  |
|  | \newpage |
|  |  |
|  | **## 3 Results and Conclusions** |
|  |  |
|  | **## Data Discovery Results:** |
|  |  |
|  | ![Discovery Results](imgs\results-discovery.png)\ |
|  |  |
|  | **## kNN Results:** |
|  |  |
|  | In the kNN Model, we observed that training set produced starting points for the experiment with K = 106 and K = 107. They yielded 54.50082 and 54.41899 respectively. These values were average and did not perform as well as we had hope. We believe that data points being further apart contributed to the outcome. K = 1 produced the most accurate result as we saw in the plot prior, which also aligned correctly with the fact that the game ranking 1st was also that one that had the highest sales. This method can be further tuned.\ |
|  |  |
|  | ![kNN Results](imgs\results-kNN.png)\ |
|  |  |
|  | **## Naive Bayes Results:** |
|  |  |
|  | With the Naive Bayes Model, we ascertained that this method displayed strong independence assumptions between the features. Thus ultimately, the unbiased prediction returned a high value for sensitivity and the specificity value was low at 0.0555767 as we expected. This method performed decently. |
|  |  |
|  | ![Naive Bayes Results](imgs\results-NB.png)\ |
|  |  |
|  | **## Random Forests Results:** |
|  |  |
|  | Lastly, using Random Forests Model, we extrapolated that as the mean of squared residuals decrease then the % variance is increased. This is a common characteristic in the Random Forests Model. We chose Model 2 to further test and computed the variable importance values to see the effects. As shown, the Genre variable importance was 191.67, the NA\_Sales variable importance was 20.85 and the Genre\_Spec variable importance was 10.19. The decreasing variable importance trend seemed appropriate as we gave more weight to the Genre variable in developing this model. The model performed as we suspected but again, can be further refined. |
|  |  |
|  | ![Random Forests Results](imgs\results-RF.png)\ |
|  |  |
|  | Overall, we thought the Random Forests Model was the most complex of the 3 models we utilized. Transparently, we intended to experiment and learn from the experience. We will take our insights and improve our future analyses. We began the project with a broad look at the video game sales data and as we visualized and explored more, we begin to understand the trends and implications. With data-driven knowledge, we developed our algorithm and models to experiment and tuned our prediction practices. We applied machine learning techniques that went beyond standard linear regression for our video game sales data set. We generated our data set, interpreted the data, performed various algorithms/modelings and presented our insights. We reviewed which video games were popular, explored and developed some prediction models for video game ranking and sales using 3 methods: the kNN, Naive Bayes and Random Forests Models. |
|  |  |
|  | We have gained invaluable knowledge from our coursework and are excited to continue to learn and hone by practicing our data science skills. As technology improves exponentially over time, perhaps one day we will accomplish a nearly perfect prediction model among other powerful systems that will improve our lives. |
|  |  |
|  | **## 4 Reference** |
|  |  |
|  | Irizarry, Rafael A. (2020). Introduction to Data Science: Data Analysis and Prediction Algorithms with R |
|  |  |
|  | https://www.statista.com/chart/21492/video-game-industry-sales-in-the-united-states/ |
|  |  |
|  | **## 5 Appendix - Environment** |
|  |  |
|  | Video Game Sales Environment Information:\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp; OS &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Windows 10\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp; IDE &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; RStudio version 1.3.1093\ |
|  | &nbsp;&nbsp;&nbsp;&nbsp;&nbsp; Language &nbsp; R version 4.0.3 (2020-10-10)\ |

2. https://github.com/Golfik/Video-Games-Sales/blob/384e0ace4e22bda4993c9c1f0f9af3ff66782980/Video%20Games%20Sales%20project.Rmd

|  |  |
| --- | --- |
|  | --- |
|  | title: '\*\*Video Game Sales with Ratings project\*\*' |
|  | output: |
|  | pdf\_document: |
|  | toc: yes |
|  | number\_sections: yes |
|  | fig\_caption: yes |
|  | header-includes: \usepackage{booktabs} |
|  | --- |
|  | \pagebreak |
|  | **# Introduction** |
|  |  |
|  | **## Project scope** |
|  |  |
|  | In this project I explore and analyze *\*Video Game Sales with Ratings\** data set made by Rush Kirubi. For the analysis sake the data set is stored on my [GitHub repository](https://github.com/Golfik/Movielens-project) (as well as this report, full code in .R and .rmd file), but original can be found on <u>[Kaggle](https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings)</u>. |
|  |  |
|  | Besides data analysis I will try to find best possible prediction model for Global and North American Sales. |
|  |  |
|  | Libraries used for the sake of this project are: *\*tidyverse\**, *\*funModeling\**, *\*scales\**, *\*caret\**, *\*corrgram\**, *\*ggpubr\**, *\*wordcloud\**, *\*tm\** and *\*gridExtra\**. For Cubist and NNLS (Non-Negative Least Squares) models additional *\*cubist\** and *\*nnls\** packages needs to be installed. |
|  |  |
|  | In this project I've tried to create all plots in pallets that can be perceived by people affected by color vision deficiency, following *\*RColorBrewer\** palette guidelines. |
|  |  |
|  | This pdf report contains most code for any data exploration and machine learning I will be doing, but some other transformations and code for plots creations have been omitted for report clarity. You can find full code in the *\*Appendix/Full code\** session, or .rmd and .R files on my GitHub repository mentioned above. Small portions of the code presented in .rmd table might be different from .R file, for the sake of presenting plots and tables in final pdf report. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, warning=FALSE} |
|  | library(knitr) |
|  | library(kableExtra) |
|  |  |
|  | if(!require(tidyverse)) install.packages("tidyverse", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(funModeling)) install.packages("funModeling", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(caret)) install.packages("caret", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(corrgram)) install.packages("corrgram", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(scales)) install.packages("scales", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(ggpubr)) install.packages("ggpubr", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(tm)) install.packages("tm", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(wordcloud)) install.packages("wordcloud", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(gridExtra)) install.packages("gridExtra", |
|  | repos = "http://cran.us.r-project.org") |
|  | # For model in train function |
|  | if(!require(Cubist)) install.packages("Cubist", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(nnls)) install.packages("nnls", |
|  | repos = "http://cran.us.r-project.org") |
|  |  |
|  | library(tidyverse) |
|  | library(funModeling) |
|  | library(scales) |
|  | library(caret) |
|  | library(corrgram) |
|  | library(ggpubr) |
|  | library(wordcloud) |
|  | library(tm) |
|  | library(gridExtra) |
|  |  |
|  | url\_path <- |
|  | "https://raw.githubusercontent.com/Golfik/Video-Games-Sales/master/Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv" |
|  | raw.videosales <- read.csv(url(url\_path),na.strings = c("","NA")) |
|  | rm(url\_path) |
|  |  |
|  | video\_sales <- raw.videosales |
|  | ``` |
|  | **## Data overview** |
|  |  |
|  | *\*Video Games Sales with Ratings\** data set was imported to *\*video\_sales\** object, upon which I will be conducting the analysis (raw data set is saved in *\*raw.videosales\** object in case reference would be needed). |
|  |  |
|  | Taking first look into the data set by using *\*glimpse\** I identify 16 variables with 16,719 records. |
|  |  |
|  | ```{r echo=FALSE} |
|  | glimpse(video\_sales) |
|  | ``` |
|  | According to Kirubi (creator of the data set) and the glimpse above, the 16 variables corresponds to: |
|  |  |
|  | \* *\*Name\** - name of the game |
|  | \* *\*Platform\** - platform on which the game was released |
|  | \* *\*Year\_of\_Release\** - year of release of the game |
|  | \* *\*Genre\** - game's genre (category) |
|  | \* *\*Publisher\** - publisher of the game |
|  | \* *\*NA\_Sales\** - millions of units of selected game sold in North America, as per Dec 2016 data |
|  | \* *\*EU\_Sales\** - millions of units of selected game sold in European Union, as per Dec 2016 data |
|  | \* *\*JP\_Sales\** - millions of units of selected game sold in Japan, as per Dec 2016 data |
|  | \* *\*Other\_Sales\** - millions of units of selected game sold in the rest of the world, as per Dec 2016 data |
|  | \* *\*Global\_Sales\** - millions of units of selected game sold in the whole world, as per Dec 2016 data |
|  | \* *\*Critics\_Score\** - aggregated score by critics, compiled by *\*Metacritic\** site |
|  | \* *\*Critic\_Count\** - number of critics participating in scoring |
|  | \* *\*User\_Score\** - aggregated score of users compiled by *\*Metacritic\** site |
|  | \* *\*User\_Count\** - number of users participating in scoring |
|  | \* *\*Developer\** - official developer responsible for game creation |
|  | \* *\*Rating\** - official North America rating assigned by *\*ESRB\** (Entertainment Software Rating Board) |
|  |  |
|  | First, I check if any of the game titles is recorded more than once: |
|  | ```{r} |
|  | duplicated(video\_sales) %>% sum() |
|  | ``` |
|  | There are no duplicates in this data set, so I can assume that all 16,719 rows are unique games. Let's see if we can identify any significant issues and outliers, by using a *\*summary\** function. |
|  |  |
|  | ```{r echo=FALSE} |
|  | summary(video\_sales) |
|  | ``` |
|  | None of the character variables, like Name, Platform or Publisher, can be analyzed with standard numerical functions, so it might be worth to consider changing them into other format. In all sales besides total (*\*Global\_Sales\**) there is a minimum 0, which might be an actual *\*NA\** - also worth considering. Maximum value in *\*Global\_Sales\** is significantly higher than maximum values in sales of sub-sectors, so it needs to be investigated if it's not actual error. |
|  |  |
|  | More than that, it is already visible from glimpse that *\*Critic\_Score\**, *\*Critic\_Count\** and *\*User\_Count\** have many *\*NA\** values. It is highly possible, that had *\*User\_Score\** been also an integer, it would be also filled with *\*NA\*s - another point to investigate further in the data cleaning and exploration. Let's take a closer look into ratio of \*NA\*s and unique values number, by using a \*status\** function. |
|  |  |
|  | ```{r echo=FALSE} |
|  | status(video\_sales) %>% |
|  | kable(booktabs=T, caption="Video Sales - zeros, NAs and unique values of variables") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "hold\_position") |
|  | ``` |
|  | Seeing the table above, it is clear that it's not only Critic and User variables that have an issue with high *\*NA\*s ratio, but also \*Developer\** and *\*Rating\** variables. While a lot of prediction models can handle *\*NA\**s in the variables, and exclude them from their algorithms, it can create significant bias. That's why any variable with high (definitely above 50%) ratio of *\*NA\**s need to be treated carefully. Easy way would be to exclude them entirely, but since those ratios are high, we would be effectively removing half of our data set. That's why deeper analysis must be conducted and alternative ways (like median imputing) considered. |
|  |  |
|  | *\*Status\** function is also showing us how many unique values each variable has. While it is tempting to change most of the character variables to factors, if factor has high number of levels it may cause overfitting in later stages of modeling (especially in decision trees with low cardinality). More than that, changing variables with too low unique numbers to factors may cause malfunctioning and errors in modeling due to variables having near to zero variance in training sets. That's why I will refrain from transforming them to factors all at once and analyze each case one by one. |
|  |  |
|  | **# Data exploration, visualization and cleaning** |
|  |  |
|  | **## Global and regional sales variables** |
|  |  |
|  | I will begin the analysis from the main output, which is total and regional sales, counted as millions of units. Below there is a density plot of *\*Global\_Sales\** variable. It is immediately obvious that the data are right-skewed, with very long tail. |
|  |  |
|  | ```{r echo=FALSE, fig.cap="Density of Global Sales"} |
|  | ggplot(video\_sales,aes(Global\_Sales)) + |
|  | geom\_histogram(aes(y=..density..),binwidth = 0.5) + |
|  | geom\_density(alpha=0.2,fill="#EF3B2C") + |
|  | labs(x="Global Sales [M units]", y ="Density", |
|  | title="Density plot of Global Sales variable") + |
|  | theme\_bw() |
|  | ``` |
|  |  |
|  | Analyzing top 10 global sales and their corresponding games we see that mainly one game (*\*Wii Sports\**) is responsible for the long tail. |
|  | ```{r eval=FALSE} |
|  | video\_sales %>% |
|  | select(Name,Year\_of\_Release,Platform, Genre, User\_Count, Global\_Sales) %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | head(10) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | video\_sales %>% |
|  | select(Name,Year\_of\_Release,Platform, Genre, User\_Count, Global\_Sales) %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | head(10) %>% |
|  | kable(booktabs=T, caption="Top 10 best selling games globally") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "HOLD\_position") |
|  | ``` |
|  |  |
|  | *\*Wii Sports\** (the top seller) was released on *\*Wii\** platform and it's a sports game. Seeing as there are other games in top 10 that are sport type and were released on *\*Wii\**, it doesn't seem like the sales output for this title is an error. I decided to leave this title in the data set. Below is the density plot only up to 3rd quartile, so without the right tail created by best selling games. The right-skewedness is now clearly visible, with almost all game titles falling below the overall mean sale being 0.5M units, and highest density around 0.3M global sales. This means that most games sell over x275 less than the best selling game recorded in data set. |
|  |  |
|  | ```{r echo=FALSE, fig.cap="Density of Global Sales, exluding top quantile", fig.pos="H"} |
|  | tot.mean <- video\_sales %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | select(Name, Global\_Sales) %>% |
|  | summarise(tot.mean=mean(Global\_Sales)) %>% |
|  | unlist() |
|  | video\_sales %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | filter(between(Global\_Sales,0,0.47)) %>% |
|  | ggplot(aes(Global\_Sales)) + |
|  | geom\_histogram(aes(y=..density..),binwidth = 0.01) + |
|  | geom\_density(alpha=0.2,fill="#EF3B2C") + |
|  | geom\_vline(aes(xintercept=tot.mean), linetype="dashed", size=1) + |
|  | labs(x="Global Sales [M units]", y ="Density", |
|  | title="Density of Global Sales, exluding top quantile", |
|  | subtitle="Dashed line marking mean value of 0.5335 (all values)") + |
|  | theme\_bw() |
|  | rm(tot.mean) |
|  | ``` |
|  |  |
|  | Next let's analyze how games were faring through years. Below I explore Global Sales aggregated for all titles per each year on one plot, sales per one title each year on the second one, and the general number of titles released every year. In order to do that, I will change the format of the *\*Year\_of\_Release\** to numeric in the whole data set (as generally years should be consider numeric or date format). |
|  |  |
|  | ```{r message=FALSE, error=FALSE, warning=FALSE} |
|  | video\_sales <- video\_sales %>% |
|  | mutate(Year\_of\_Release=as.numeric(Year\_of\_Release)) |
|  | ``` |
|  |  |
|  | ```{r echo=FALSE, warning=FALSE, message=FALSE, fig.cap="Global Sales, units sold and number of games released through years", fig.pos="H"} |
|  | plot1 <- video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(sum\_of\_sales=sum(Global\_Sales)) %>% |
|  | ggplot() + |
|  | geom\_point(aes(x=Year\_of\_Release, y=sum\_of\_sales), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | geom\_line(aes(x=Year\_of\_Release, y=sum\_of\_sales), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | labs(x = "Year", y = "Global Sales", title="Global sales [M units] through years") + |
|  | theme\_bw() |
|  | plot2 <- video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(sales\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | ggplot() + |
|  | geom\_point(aes(x=Year\_of\_Release, y=sales\_pertitle), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | geom\_line(aes(x=Year\_of\_Release, y=sales\_pertitle), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | labs(x = "Year", y = "Sales per title", |
|  | title="Units sold [M units] per game title through years") + |
|  | theme\_bw() + |
|  | scale\_color\_brewer(palette = "RdYlBu") |
|  | plot3 <- video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(n=n()) %>% |
|  | ggplot() + |
|  | geom\_point(aes(x=Year\_of\_Release, y=n), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | geom\_line(aes(x=Year\_of\_Release, y=n), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | labs(x = "Year", y = "Number of games", |
|  | title="Number of games released through years") + |
|  | theme\_bw() |
|  | ggarrange(plot1,plot2,plot3,ncol=1, align="hv") |
|  | rm(plot1,plot2,plot3) |
|  | ``` |
|  |  |
|  | There is a visible trend starting sometime in the mid-90s. While the total sales has started to increase, at the same time the number of sold copies per title has been dropping. This leads to conclusion that after mid-90s there has been high increase in number of games on the market (as seen clearly on third plot on Figure 3), where most of them were having much smaller profit than in the past. There is a big peak in Global Sales around 2006 - again, contributed to best selling game *\*Wii Sport\** and similar games on *\*Wii\** platform published during the same period. |
|  |  |
|  | Another thing that needs to be re-checked are the values after the year 2016. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(Year\_of\_Release)) %>% |
|  | head(10) %>% |
|  | kbl(booktabs=T, caption="Number of games released in last 10 recorded years", align="c") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "HOLD\_position") |
|  | ``` |
|  |  |
|  | It seems that not only there is a gap between 2017 and 2020, but the numbers of 2017 and 2020 seems very off compared to previous years. More than that, original data set created by Kirubi has been created as of December 2016, so it is safe to say that all records after that date can be treated as errors and therefore erased. |
|  |  |
|  | ```{r} |
|  | video\_sales <- video\_sales %>% filter(Year\_of\_Release < 2017) |
|  | ``` |
|  |  |
|  | Lastly let's analyze how the sales are distributed between different regions. On Figure 4 below we can see this distribution through years. At the beginning of 1980 the most dominant market was North American one, but it soon had too compete with Japan. Japanese market tried to gain higher share up until middle of 1990, when they started to significantly lose again with North America region. But all across those years EU and Other markets have been growing step-by-step without much disruptions, and at the present times EU market has almost equal share to the NA market. |
|  |  |
|  | ```{r echo=FALSE, fig.cap="Share of sales between regions through years", fig.pos="H", fig.height=3.5, fig.width=5.5} |
|  | video\_sales %>% |
|  | select(Year\_of\_Release, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | gather(Region, Sales, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | group\_by(Year\_of\_Release,Region) %>% |
|  | ggplot() + |
|  | geom\_bar(aes(x=Year\_of\_Release, y=Sales, fill=Region), |
|  | position="fill", stat="identity") + |
|  | labs(x = "Year", y = "Sales", |
|  | title="Share of sales between regions through years") + |
|  | theme\_bw() + |
|  | scale\_y\_continuous(labels = percent\_format()) + |
|  | scale\_fill\_brewer(palette = "RdYlBu", |
|  | labels=c("EU", "Japan", "North America", "Other")) |
|  | ``` |
|  |  |
|  | The mid-1990 growth of NA and EU markets might be attributed to rise of popularity of PC games at that time (as seen on third plot on Figure 3), that were slowly taking away previously console-only market. Most of the PC games from that period had very small sales in Japan. This theory can be linked also to Figure 3, as rise of PC games popularity resulted in games being more popular and starting to be produced on mass scale. However, this is only just a theory, as besides pure sales data there are many other factors, like socio-political changes, happening at that period that could contribute to this change. |
|  |  |
|  | **## Publisher and Developer analysis** |
|  |  |
|  | Let's take into consideration *\*Publisher\** and *\*Developer\** variables. They are similar in their structure (both categorical) and meaning, so the approach to their analysis and exploration will be also similar. Ideally, we would like to replace their character type not by factor (as there are too many unique values to both variables, as shown in Table 1), but by numerical factor corresponding to each publisher and developer importance and influence on sales. |
|  |  |
|  | First of all, let's take a look at the top 10 best selling publishers and developers (on Figure 5 below), when comparing their total sales through all years of analysis. From Table 1 we know that *\*Developer\** variable has almost 40% of *\*NA\*s, so I've removed them from plotting (still remaining in \*video\_sales\** data set) for the sake of plot clarity. |
|  |  |
|  | ```{r echo=FALSE, warning=FALSE, message=FALSE, fig.cap="Top 10 best selling publishers and developers, as total sales made", fig.pos="H"} |
|  | # Global sales by publisher, as total sum of sales made |
|  | plot1 <- video\_sales %>% |
|  | gather(Region, Sales, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | group\_by(Publisher,Region) %>% |
|  | summarise(sum\_Sales=sum(Sales),total=sum(Global\_Sales)) %>% |
|  | arrange(desc(total)) %>% |
|  | head(40) %>% |
|  | ggplot(aes(x=reorder(Publisher,desc(total)), y=sum\_Sales,fill=Region)) + |
|  | geom\_bar(stat = "identity") + |
|  | theme\_bw() + |
|  | theme(axis.text.x = element\_text(angle = 15, hjust=0.95)) + |
|  | labs(x="Publisher", y ="Global Sales [M units]", |
|  | title="Top 10 best selling publishers globally, as total sales made") + |
|  | scale\_fill\_brewer(palette = "PRGn", labels=c("EU", "Japan", "North America", "Other")) |
|  |  |
|  | # Global Sales by developer, as total sum of sales |
|  | plot2 <- video\_sales %>% |
|  | filter(!is.na(Developer)) %>% |
|  | gather(Region, Sales, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | group\_by(Developer,Region) %>% |
|  | summarise(sum\_Sales=sum(Sales),total=sum(Global\_Sales)) %>% |
|  | arrange(desc(total)) %>% |
|  | head(40) %>% |
|  | ggplot(aes(x=reorder(Developer,desc(total)), y=sum\_Sales,fill=Region)) + |
|  | geom\_bar(stat = "identity") + |
|  | theme\_bw() + |
|  | theme(axis.text.x = element\_text(angle = 15, hjust=0.95)) + |
|  | labs(x="Developer", y ="Global Sales [M units]", |
|  | title="Top 10 best selling developers globally, as total sales made") + |
|  | scale\_fill\_brewer(palette = "PRGn", labels=c("EU", "Japan", "North America", "Other")) |
|  |  |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2,common.legend=TRUE, legend="bottom") |
|  | rm(plot1,plot2) |
|  | ``` |
|  |  |
|  | For both variables it's confirming that North America region has the biggest ratio of sales. It is also visible that while EU and Other regions have stable share in all 10 best selling publishers and developers, Japan has clearly playing favorites, favoring ones (like *\*Nintendo\**) and neglecting other names. |
|  |  |
|  | It it also visible that most of top 10 places are taken by well known modern corporations. *\*Nintendo\** is both the best selling Publisher, as well as the best selling Developer. Similarly *\*Ubisoft\** and *\*Electronic Arts\**/*\*EA\** are not far behind. However, seeing as those are big companies, it could be that those companies are releasing many mediocre titles, with only few really successful ones, and their total sum of sales is not guaranteeing that every title they release will be a success. Let's look then at top 10 best selling Publishers and Developers as calculated by sales per title, on Figure 6 below. |
|  |  |
|  | ```{r echo=FALSE, warning=FALSE, message=FALSE, fig.cap="Top 10 publishers and developers as sales per title made", fig.pos="H"} |
|  | # Global sales made per title by top publishers |
|  | plot1 <- video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | summarise(sale\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | arrange(desc(sale\_pertitle)) %>% |
|  | head(10) %>% |
|  | ggplot(aes(y=sale\_pertitle,x=reorder(Publisher,sale\_pertitle))) + |
|  | geom\_bar(stat = "identity", fill="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Publisher", y ="Sales per title [M units]", |
|  | title="Top 10 publishers by sales per title") + |
|  | coord\_flip() + |
|  | scale\_x\_discrete(labels = function(x) str\_wrap(x, width = 50)) |
|  |  |
|  | # Global sales per title by top developers |
|  | plot2 <- video\_sales %>% |
|  | filter(!is.na(Developer)) %>% |
|  | group\_by(Developer) %>% |
|  | summarise(sale\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | arrange(desc(sale\_pertitle)) %>% |
|  | head(10) %>% |
|  | ggplot(aes(y=sale\_pertitle,x=reorder(Developer,sale\_pertitle))) + |
|  | geom\_bar(stat = "identity", fill="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Developer", y ="Sales per title [M units]", |
|  | title="Top 10 developers by sales per title") + |
|  | coord\_flip() + |
|  | scale\_x\_discrete(labels = function(x) str\_wrap(x, width = 50)) |
|  |  |
|  | # Arranging two previous plots, and clearing environment |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2, align="v") |
|  | rm(plot1,plot2) |
|  | ``` |
|  |  |
|  | Looking at plots were we consider average sales per title, *\*Nintendo\** and other big players stop being the best selling publishers and developers in our data set. They are falling behind from top places, replaced by not well-known studios (like *\*Retro Studios, Entertainment Analysis & Development Division\** developer studio on second place). This suggest that we should model our sales not by publisher and developer name, but rather on a factors that would be corresponding to sales per title made by each publisher and developer. |
|  |  |
|  | Following this approach would however prove unfair advantage on the studios who just released one big title and almost nothing more, and resulting in false fitting in our future models. Let's see how does it look in the top 10 developers and publishers presented on Figure 6. |
|  |  |
|  | ```{r eval=FALSE} |
|  | # How many games has top Publisher released? |
|  | video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n()) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Publisher,sale\_per\_title, no\_of\_games) %>% unique() %>% head(10) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n()) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Publisher,sale\_per\_title, no\_of\_games) %>% |
|  | unique() %>% |
|  | head(10) %>% |
|  | kbl(booktabs=T, caption="Top 10 best selling publishers") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "HOLD\_position") |
|  | ``` |
|  | ```{r eval=FALSE} |
|  | # How many games has top Developer released? |
|  | video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n()) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Developer,sale\_per\_title, no\_of\_games) %>% unique() %>% head(10) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n()) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Developer,sale\_per\_title, no\_of\_games) %>% |
|  | unique() %>% |
|  | head(10) %>% |
|  | kbl(booktabs=T, caption="Top 10 best selling developers") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "hold\_position") |
|  | ``` |
|  |  |
|  | It is clear from both tables that creating Publisher and Developer factors based just on sales per title would result in favoring a lot of "one-shot" studios. In order to penalize the studios with smaller number of titles released, all sales per title would be multiplied by the function with limit to 1. Good example of such function, which I would use for *\*Publisher\** and *\*Developer\** variables is: |
|  | $$f(n)=n\*sin(1/n)$$ |
|  | where *\*n\** would be the number of titles released. The top selling Publisher and Developer tables with new penalized factors look now like that: |
|  | ```{r eval=FALSE} |
|  | # How will publisher factor look like if we penalize studios with small number |
|  | # of titles with function with limit to 1? |
|  | video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n(), |
|  | adj.factor=no\_of\_games\*sin(1/no\_of\_games), |
|  | Publisher.fct=sale\_per\_title\*no\_of\_games\*sin(1/no\_of\_games)) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Publisher,sale\_per\_title, no\_of\_games, adj.factor, Publisher.fct) %>% |
|  | unique() %>% |
|  | head(10) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n(), |
|  | adj.factor=no\_of\_games\*sin(1/no\_of\_games), |
|  | Publisher.fct=sale\_per\_title\*no\_of\_games\*sin(1/no\_of\_games)) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Publisher,sale\_per\_title, no\_of\_games, adj.factor, Publisher.fct) %>% |
|  | unique() %>% |
|  | head(10) %>% |
|  | kbl(booktabs=T, caption="Top 10 best selling publishers with penalized factor") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "hold\_position") |
|  | ``` |
|  | ```{r eval=FALSE} |
|  | # How will developer factor look like if we penalize studios with small number |
|  | # of titles with function with limit to 1? |
|  | video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n(), |
|  | adj.factor=no\_of\_games\*sin(1/no\_of\_games), |
|  | Developer.fct=sale\_per\_title\*no\_of\_games\*sin(1/no\_of\_games)) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Developer,sale\_per\_title, no\_of\_games, adj.factor, Developer.fct) %>% |
|  | unique() %>% |
|  | head(10) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n(), |
|  | adj.factor=no\_of\_games\*sin(1/no\_of\_games), |
|  | Developer.fct=sale\_per\_title\*no\_of\_games\*sin(1/no\_of\_games)) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Developer,sale\_per\_title, no\_of\_games, adj.factor, Developer.fct) %>% |
|  | unique() %>% |
|  | head(10) %>% |
|  | kable(booktabs=T, caption="Top 10 best selling developers with penalized factor") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_option=c("HOLD\_position","scale\_down")) |
|  | ``` |
|  | Now that I've created new numerical factors for *\*Publisher\** and *\*Developer\** variables, we can remove the old character type ones and include the new ones in *\*video\_sales\** data set, to be used in future modeling. |
|  |  |
|  | ```{r} |
|  | video\_sales <- video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(Publisher.fct=(sum(Global\_Sales)/n())\*n()\*sin(1/n())) %>% |
|  | ungroup() |
|  | video\_sales <- video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(Developer.fct=(sum(Global\_Sales)/n())\*n()\*sin(1/n())) %>% |
|  | ungroup() |
|  | video\_sales <- video\_sales %>% |
|  | select(-Publisher,-Developer) |
|  | ``` |
|  |  |
|  | **## Platform and Genre analysis** |
|  |  |
|  | Next variables that I will look into are *\*Platform\** and *\*Genre\**, currently also a character type (see Table 1). Let's first take a look into game titles spread through different types of *\*Platforms\**, on below Figure 7. For the sake of plot clarity, only 15 top Platforms in terms of number of titles released were shown. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Top 15 platforms with highest number of games released", fig.pos="H"} |
|  | video\_sales %>% |
|  | group\_by(Platform) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(15) %>% |
|  | ggplot(aes(x=reorder(Platform,n),y=n)) + |
|  | geom\_segment(aes(x=reorder(Platform,n), |
|  | xend=reorder(Platform,n), y=0, yend=n), color="grey") + |
|  | geom\_point(color="#FD8D3C", size=4) + |
|  | coord\_flip() + |
|  | theme\_bw() + |
|  | labs(x="Platform", y ="Number of titles released", |
|  | title="Top 15 platforms with highest number of games released") |
|  | ``` |
|  |  |
|  | The indisputable kings, having almost twice as much titles released on them than 3rd place, are *\*PS2 (PlayStation 2)\** console belonging to *\*Sony\** and *\*DS\** console belonging to *\*Nintendo\**. In fact, those two companies own almost all other consoles (12 out of 15) visible on Figure 6, such as *\*PS3 (PlayStation 3)\**, *\*PSP (PlayStation Portable)\**, *\*PS (PlayStation)\**, *\*PSV (PlayStation Vita)\**, *\*PS4 (PlayStation 4)\** belonging to *\*Sony\**, and *\*Wii\**, *\*GBA (Game Boy Advance)\**, *\*GC (GameCube)\**, *\*3DS\**, *\*N64\** belonging to *\*Nintendo\**. The only other *\*Platforms\** from top 15 are *\*X360 (Xbox 360)\** and *\*XB (Xbox)\** belonging to *\*Microsoft\**, and universal *\*PC\** platform. |
|  |  |
|  | *\*Platform\** variable has 31 unique levels, so changing it into a factor may cause overfitting due to too large number of levels (as mentioned in the *\*Introduction\**). Factorizing this variable can improve slightly the speed of future modeling (tested on the current code with 4%=5min improvement), but it also caused drop of fit (higher RMSE and lower R-squared) due to noise caused by zero variance variables (due to platforms with very small number of titles). I've decided not to eliminate *\*Platform\** levels with such near-zero variance, in order not to reduce and disturb the data set, and I'll leave the *\*Platform\** variable as a character-type. |
|  |  |
|  | Now let's take a look if the top selling games were released on any of the *\*Platforms\** from Figure 7, and if we can distinguish any pattern in terms of games genre, on Figure 8 below. |
|  |  |
|  | ```{r echo=FALSE, fig.cap="Top 15 selling games globally", fig.pos="H"} |
|  | video\_sales %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | head(15) %>% |
|  | ggplot(aes(y=Global\_Sales,x=reorder(Name,Global\_Sales),fill=Genre)) + |
|  | geom\_bar(stat="identity") + geom\_text(aes(label=Platform), hjust=1.5) + |
|  | coord\_flip() + |
|  | theme\_bw() + |
|  | labs(x="Global Sales [M of units]", y ="Game title and its platform", |
|  | title="Top 15 selling games globally") + |
|  | scale\_fill\_brewer(palette = "RdYlBu") |
|  | ``` |
|  |  |
|  | It is quite a surprise to see that none of 15 top selling games up until 2016 were released on *\*PS2\**, which was a *\*Platform\** leader, and only 3 were released on *\*DS\**, which was a runner-up. With that we can assume that there is no high correlation between type of Platform the game was released and how well it will sell. |
|  |  |
|  | Looking again at Figure 8, it seems that there is also no strong connection between *\*Genre\** of the game and its selling numbers. The *\*Sport\** and *\*Platform\** genres have very small advantage on other titles from the plot, but we're looking here only at the top 15 selling games. The number of titles released might be another influencer. |
|  |  |
|  | Let's take a look at the summary of total sales and number of games released of each game genre, in Table 8 below. |
|  |  |
|  | ```{r eval=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Genre) %>% |
|  | summarise(n\_titles=n(), total\_sales=sum(Global\_Sales)) %>% |
|  | arrange(desc(total\_sales)) |
|  | ``` |
|  | ```{r echo=FALSE, message=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Genre) %>% |
|  | summarise(n\_titles=n(), total\_sales=sum(Global\_Sales)) %>% |
|  | arrange(desc(total\_sales)) %>% |
|  | kable(booktabs=T, caption="Summary of total sales and number of games released per genre") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_option="HOLD\_position") |
|  | ``` |
|  |  |
|  | First thing that is visible from Table 8 is that there are two titles with *\*NA\** genre. It doesn't make any sense to try to impute them, as *\*Genre\** is clearly categorical variable, and since it's just two titles I will erase them from *\*video\_sales\** data set. |
|  |  |
|  | ```{r} |
|  | video\_sales <- video\_sales %>% |
|  | filter(!is.na(Genre)) |
|  | ``` |
|  |  |
|  | Since there is only 12 different values of *\*Genre\** variable, and from Table 8 we can see that none of them are represented in very small amount, we can safely assume that transforming *\*Genre\** into factor is the best choice and will not result in overfitting or modeling disruption due to near-zero variances of particular levels. |
|  |  |
|  | ```{r} |
|  | video\_sales$Genre <- as.factor(video\_sales$Genre) |
|  | ``` |
|  |  |
|  | Looking back to Table 8 and comparing numbers of titles per genre and total sales, it seems that it would be good to follow approach of *\*Publisher\** and *\*Developer\** variable, and check the sales per one title per game genre. This is represented on Figure 9 below. \pagebreak |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Sales per one title per game genre", fig.pos="H"} |
|  | video\_sales %>% |
|  | group\_by(Genre) %>% |
|  | summarise(sale\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | arrange(desc(sale\_pertitle)) %>% |
|  | ggplot(aes(y=sale\_pertitle,x=reorder(Genre,desc(sale\_pertitle)))) + |
|  | geom\_bar(stat = "identity", fill="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Genre", y ="Sales per title [M units]", |
|  | title="Sales per one title per game genre") + |
|  | theme(axis.text.x = element\_text(angle = 15, hjust=0.95)) |
|  | ``` |
|  |  |
|  | It seems that for some genres, considering number of titles released makes a small difference, like for *\*Platform\** genre. However, for many of them they are placed almost on the same level as in Table 8, like *\*Shooter\** or *\*Role-Playing\**. Taking that into consideration, and also the fact that *\*Genre\** doesn't have many levels in total, I take a final decision that it would be best to leave it as non-adjusted factor than change it to numerical representative like for *\*Publisher\** and *\*Developer\**. |
|  |  |
|  | **## User and Critic Score and Count analysis** |
|  |  |
|  | Another variables that will be taken into consideration are *\*User\_Score\**, *\*User\_Count\**, *\*Critic\_Score\** and *\*Critic\_Count\**. It needs to be noted that if we were to try to predict sales upon game release, or just before it, it would be impossible to know *\*User\**-type variables, as user statistics are created only after the game was released. That's why in this project it will be assumed that we're trying to predict final sales after players (users) had enough time to check the game and give their opinion, as the whole data in the data set was collected also way after release of each game. |
|  |  |
|  | First let's take a look at *\*User\_Score\** and *\*Critic\_Score\** frequencies, on Figure 10 below. Dark background theme was selected for better visibility of very bright colors differentiating different game *\*Genre\** on each plot. As it was pointed in Table 1, both *\*User\_Score\** and *\*Critic\_Score\** have very high number of *\*NA\**s. For now I will omit them in the visualization. \pagebreak |
|  |  |
|  | ```{r echo=FALSE, warning=FALSE, message=FALSE, fig.cap="Distribution of User and Critic Scores across game genres", fig.pos="H"} |
|  | plot1 <- ggplot(video\_sales, aes(as.numeric(User\_Score), color=Genre)) + |
|  | geom\_freqpoly(size=1) + |
|  | theme\_dark() + |
|  | scale\_color\_brewer(palette = "Paired") + |
|  | labs(x="User Score", y ="Count", |
|  | title="Distribution of User Scores across game genres") |
|  | plot2 <- ggplot(video\_sales, aes(as.numeric(Critic\_Score), color=Genre)) + |
|  | geom\_freqpoly(size=1) + |
|  | theme\_dark() + |
|  | scale\_color\_brewer(palette = "Paired") + |
|  | labs(x="Critic Score", y ="Count", |
|  | title="Distribution of Critic Scores across game genres") |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2,common.legend=TRUE, legend="bottom") |
|  | rm(plot1,plot2) |
|  | ``` |
|  |  |
|  | It seems that distribution of User and Critic Scores is quite similar, both having its peak around 80% of max scoring value. Both of the variables have also similar distribution of scoring according to game *\*Genre\**, with highest scores achieved (mostly) by *\*Action\** genre. The only major difference here is *\*Sports\** genre, which Critics favor more than Users do. |
|  |  |
|  | As it was mentioned before (and also in Table 1), *\*User\_Score\** is a character-type. After removing *\*NA\**s from the first plot on Figure 10, it seems that it's quite obvious that there are no actual character-type values, but they could be skipped due to the nature of the plot. Let's explore this variable further below. |
|  |  |
|  | ```{r eval=FALSE} |
|  | video\_sales %>% |
|  | group\_by(User\_Score) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(10) |
|  | ``` |
|  | ```{r echo=FALSE, message=FALSE} |
|  | video\_sales %>% |
|  | group\_by(User\_Score) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(10) %>% |
|  | kbl(booktabs=T, caption="Count of User Scores per type (top 10)", align="c") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_option="HOLD\_position") |
|  | ``` |
|  |  |
|  | It is immediately clear while looking at Table 9, that *\*tbd\** value (to-be-decided) should actually be *\*NA\**. That would leave only numerical values, so it makes sense to change this variable type to numerical. |
|  |  |
|  | ```{r} |
|  | video\_sales$User\_Score[video\_sales$User\_Score=="tbd"] <- NA |
|  | video\_sales$User\_Score <- as.numeric(video\_sales$User\_Score) |
|  | ``` |
|  |  |
|  | Let's see if the same can be said about *\*Critic\_Score\** variable, which is also a character-type. |
|  |  |
|  | ```{r eval=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Critic\_Score) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(10) |
|  | ``` |
|  | ```{r echo=FALSE, message=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Critic\_Score) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(10) %>% |
|  | kbl(booktabs=T, caption="Count of Critic Scores per type (top 10)",align="c") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_option="HOLD\_position") |
|  | ``` |
|  |  |
|  | There are no clearly character-type values, so we can also transform this variable into numeric type. |
|  |  |
|  | ```{r} |
|  | video\_sales$Critic\_Score <- as.numeric(video\_sales$Critic\_Score) |
|  | ``` |
|  |  |
|  | \pagebreak |
|  | Scoring coming from Users and Critics seems to have peaks in the same place, as seen on Figure 10. Let's see how they are looking against each other on Figure 11 below. For this plot all *\*NA\** values were skipped. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Correlation between User and Critic Score", fig.pos="H", fig.height=4} |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | ggplot(aes(x=User\_Score,y=Critic\_Score)) + |
|  | geom\_point() + |
|  | geom\_smooth(method="loess") + |
|  | theme\_bw() + |
|  | labs(x="User Score", y ="Critic Score", |
|  | title="Correlation between User and Critic Score") |
|  | ``` |
|  |  |
|  | There seems to be a correlation between *\*User\_Score\** and *\*Critic\_Score\**, but not very strong one. It is especially visible on the geom\_smooth line on Figure 11 (using *\*loess\** method), as the line seems to be slightly wavy and not all straight, indicating that it's not fully linear. \pagebreak |
|  |  |
|  | But what about those two variables simultaneous influence on total sales? Figure 12 shows influence of scoring difference between *\*User\_Score\** and *\*Critic\_Score\** and its impact on *\*Global\_Sales\**. Here I also skipped *\*NA\*s from the analysis, and for the sake of comparing like-to-like both scorings were normalized to 100, meaning that all values of \*User\_Score\** had to be multiplied by 10. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Influence of scoring difference between Users and Critics on Global Sales", fig.pos="H"} |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | mutate(diff=(User\_Score\*10-Critic\_Score)) %>% |
|  | group\_by(diff) %>% |
|  | summarise(sums=sum(Global\_Sales)) %>% |
|  | ggplot(aes(x=diff,y=sums)) + |
|  | geom\_point(color="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Difference between User and Critic Score", |
|  | y ="Global Sales [M of units]", |
|  | title="Influence of scoring difference between Users and Critics on Global Sales") |
|  |  |
|  | ``` |
|  |  |
|  | Figure 12 shows that scorings towards each other follow normal distribution, with the highest output (*\*Global\_Sales\**) achieved when both scorings are exactly the same, and lowest when they are different from each other by at least 30%. \pagebreak |
|  |  |
|  | Knowing that both variables should be in agreement for the highest output is not answering the question of at what values of score the sales would be the highest. This question is explored o Figure 13 below, once again omitting *\*NA\**s and normalizing scoring to 100% for visualization purpose. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Global Sales vs. Critic and User Score", fig.pos="H"} |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | mutate(User\_Score=User\_Score\*10) %>% |
|  | gather(Type, Score, User\_Score, Critic\_Score) %>% |
|  | group\_by(Type,Score) %>% |
|  | summarise(totalsales=sum(Global\_Sales)) %>% |
|  | ggplot(aes(x=Score,y=totalsales, fill=Type)) + |
|  | geom\_area(position="identity") + |
|  | facet\_grid(Type~.) + |
|  | theme\_bw() + |
|  | scale\_fill\_brewer(palette="Dark2", label=c("Critic Score", "User Score\*10")) + |
|  | labs(x="Score", y ="Global Sales [M of units]", |
|  | title="Global Sales vs. Critic and User Score") |
|  | ``` |
|  |  |
|  | What is interesting from Figure 13 is that the games that were selling highest number of units were not the games that achieved highest possible scoring from both Users and Critics. Both input sources have similar impact on sales, and the games that were the best sellers were actually the ones that achieved 75-80% of possible score. It seems that even if you're making a masterpiece you still can't please everyone. \pagebreak |
|  |  |
|  | Now let's look at the count of scores that were cast. Where their number also influencing *\*Global\_Sales\**? Where there any tendencies - the more the better or less is better? This is explored on Figure 14. *\*NA\**s were also skipped on this plot, but since there is naturally more users (players) than critics, the x-scale could not be the same. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Global Sales vs. Critic and User score Count", fig.pos="H"} |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | gather(Type, Count, User\_Count, Critic\_Count) %>% |
|  | group\_by(Type,Count) %>% |
|  | summarise(totalsales=sum(Global\_Sales)) %>% |
|  | ggplot(aes(x=Count,y=totalsales,fill=Type)) + |
|  | geom\_area(position="identity") + |
|  | facet\_grid(~Type, scales="free") + |
|  | theme\_bw() + |
|  | labs(x="Count", y ="Global Sales [M of units]", |
|  | title="Global Sales vs. Critic and User score Count") + |
|  | scale\_fill\_brewer(palette="Dark2", label=c("Critic Count", "User Count")) |
|  | ``` |
|  |  |
|  | There seems to be no relation between two Counts and no clear connection between those two variables and *\*Global\_Sales\**. *\*User\_Count\** has no clear big peak, and it's influence is spread across many values. Meanwhile *\*Critic\_Count\** is clearly right-skewed, with wide-spread peak near lower count values (around 23), and few very singular peaks (similar to whole *\*User\_Count\** graph) around higher values. \pagebreak |
|  |  |
|  | Since we were looking at correlation of all four User and Critic variables towards *\*Global\_Sales\** we now have a pretty good picture of those. But how does the distribution of those values look like across years documented in the data set? This can be seen on Figure 15 below. Like previously, *\*NA\**s were skipped on plots, and scores for Users and Critics are shown as average of all scores per each year, normalized to 100. |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Distribution of Critic and User Scores and Counts across years", fig.pos="H"} |
|  | plot1 <- video\_sales %>% |
|  | na.omit() %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(totalusers=mean(User\_Score\*10/User\_Count), |
|  | totalcritics=mean(Critic\_Score/Critic\_Count)) %>% |
|  | gather(Type, Count, totalusers, totalcritics) %>% |
|  | ggplot() + |
|  | geom\_line(aes(x=Year\_of\_Release, y=Count, color=Type), size=1) + |
|  | theme\_bw() + |
|  | scale\_color\_brewer(palette = "Dark2", label=c("Critics", "Users")) + |
|  | labs(x="Year of Release", y ="Sum of avg Scores (normalized) ", |
|  | title="Distribution of Critic and User Scores (avg per title) across years") |
|  | plot2 <- video\_sales %>% |
|  | na.omit() %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(totalusers=sum(User\_Count), |
|  | totalcritics=sum(Critic\_Count)) %>% |
|  | gather(Type, Count, totalusers, totalcritics) %>% |
|  | ggplot() + |
|  | geom\_line(aes(x=Year\_of\_Release, y=Count, color=Type), size=1) + |
|  | theme\_bw() + |
|  | scale\_color\_brewer(palette = "Dark2", label=c("Critics", "Users")) + |
|  | labs(x="Year of Release", y ="Sum of Counts", |
|  | title="Distribution of Critic and User score Counts across years") |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2,common.legend=TRUE, legend="bottom", align="v") |
|  | rm(plot1,plot2) |
|  | ``` |
|  |  |
|  | Looking at the average scores cast each year (first plot on Figure 15), there is no clear trend across the years. There has been a small decrease in scores around 1990, with (again) big peak near mid-1990s, but since then both scoring in average seem to be on similar level. This is interesting, seeing as the count of scores cast is increasing since that period. It needs to be noted that average score per year is never exceeding 18, and seeing as this is normalized scale up to 100, it is very low. It signals that there must be a lot of pretty badly scored games, and only a few ones are being above-average titles. This trend seems to continue even with increase of number of titles released in recent years (as seen on Figure 3). |
|  |  |
|  | As for the counts on Figure 15 (second plot), we can again see a trend starting right after mid-1990s. It could be caused by popularizing gaming industry in total, but in case of User and Critic Scores it is clearly contributed to *\*Metacritic\** portal (which is one of the most popular platform gathering this kind of scores), which started in 1999. Since then we see a drastically big increase of number of scores. Games released previous to 1999 are also having some scoring, as portal was gathering data retroactively and Users can score back, but it's clear there is much less of them. |
|  |  |
|  | Since we've noticed there is a significantly less counted scores before mid-1990s, let's check if there are any games that are not scored either from Users or Critics, and are disturbing our data set. |
|  |  |
|  | ```{r} |
|  | video\_sales %>% |
|  | filter(User\_Count==0 | Critic\_Count==0) %>% |
|  | select(Name,Year\_of\_Release,Global\_Sales,User\_Score,User\_Count, |
|  | Critic\_Score,Critic\_Count) |
|  | ``` |
|  |  |
|  | There are no titles like that, so we can assume that for this context our data set is pure (although it has a title that has been scored 0 - pretty bad game it seems). |
|  |  |
|  | Now that we've analyzed all four variables of User and Critic Scores and Count, let's deal with multiple *\*NA\*s that are disturbing them. We can't leave them in, as they would cause errors in several types of machine learning methods, like \*Random Forest\**, so they need to be dealt with. As mentioned before, removing the titles with them would mean significantly (~50%) decreasing our data set, so the alternative is to impute them with another value. Looking at distribution analysis (like on Figure 10) it seems that both Critic and User values have similar distributions, and inside each other they have clear peaks. Hence it makes sense to impute *\*User\_Score\**, *\*User\_Count\**, *\*Critic\_Score\** and *\*Critic\_Count\** *\*NA\**s with medians of those variables. |
|  |  |
|  | ```{r} |
|  | video\_sales$User\_Score[is.na(video\_sales$User\_Score)] <- |
|  | median(video\_sales$User\_Score, na.rm = TRUE) |
|  | video\_sales$User\_Count[is.na(video\_sales$User\_Count)] <- |
|  | median(video\_sales$User\_Count, na.rm = TRUE) |
|  | video\_sales$Critic\_Score[is.na(video\_sales$Critic\_Score)] <- |
|  | median(video\_sales$Critic\_Score, na.rm = TRUE) |
|  | video\_sales$Critic\_Count[is.na(video\_sales$Critic\_Count)] <- |
|  | median(video\_sales$Critic\_Count, na.rm = TRUE) |
|  | ``` |
|  |  |
|  | **## Rating variable** |
|  |  |
|  | Next one is *\*Rating\** variable. As seen before in Table 1 *\*Rating\** has significant number of *\*NA\** values. It's shown currently as character and has 8 unique values in total. Let's see what are those in the Table 11 below. |
|  |  |
|  | ```{r eval=FALSE} |
|  | video\_sales %>% group\_by(Rating) %>% summarise(n=n()) %>% arrange(desc(n)) |
|  | ``` |
|  | ```{r echo=FALSE, message=FALSE} |
|  | video\_sales %>% |
|  | group\_by(Rating) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | kbl(booktabs=T, caption="Unique values of Rating variable",align="c") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_option="HOLD\_position") |
|  | ``` |
|  |  |
|  | Rating system in North America is governed and assigned by ESRB. According to 2016 standards all of the values presented in Table 11 exist (not counting *\*NA\** of course), besides *\*K-A\** and *\*RP\**. *\*K-A\** rating is an old one, replaced by *\*E\** rating, and *\*RP\** means *\*Rating Pending\**, so equal to *\*NA\** in the data set. Let's change the values of *\*Rating\** accordingly. |
|  |  |
|  | ```{r} |
|  | video\_sales$Rating[video\_sales$Rating=="RP"] <- NA |
|  | video\_sales$Rating[video\_sales$Rating=="K-A"] <- "E" |
|  | ``` |
|  |  |
|  | Now, *\*Rating\** variable should have 6 unique values (*\*NA\** not counted), so let's change it to factor for ease of modeling. |
|  |  |
|  | ```{r} |
|  | video\_sales$Rating <- as.factor(video\_sales$Rating) |
|  | ``` |
|  |  |
|  | Similar to what I did before to other variables, let's explore what would be the influence of game *\*Rating\** on total sales per one title released, grouped by each *\*Rating\**. This can be seen on Figure 16 below (\*NA\*s have been omitted). |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Global Sales per one title per game rating", fig.pos="H"} |
|  | video\_sales %>% |
|  | filter(!is.na(Rating)) %>% |
|  | group\_by(Rating) %>% |
|  | summarise(sales=sum(Global\_Sales)/n(),n=n()) %>% |
|  | ggplot(aes(x=reorder(Rating,sales), y=sales)) + |
|  | geom\_segment( aes(x=reorder(Rating,sales), xend=reorder(Rating,sales), |
|  | y=0, yend=sales), color="black") + |
|  | geom\_point( color="blue", size=4) + |
|  | geom\_text(aes(label=paste("#games:",n)),hjust=0.9, vjust=1.75) + |
|  | coord\_flip() + |
|  | theme\_bw() + |
|  | labs(x="Rating", y ="Global Sales per title [M units]", |
|  | title="Global Sales per one title per game rating") |
|  | ``` |
|  |  |
|  | If we discard the outlier values from Figure 16, which would be Rating *\*AO\** and Rating *\*EC\**, as they have the smallest number of titles released, there is a small advantage of *\*M\** (Mature) *\*Rating\** over the rest in terms of influence on sales. However, all of those ratings are not exceeding 1M units sold per game title, which can be considered low if we compare it to top selling games from Table 2. |
|  |  |
|  | The *\*Rating\** variable has significant (41% according to Table 1) number of *\*NA\** values, which would be disturbing our future modeling, as said before. Seeing as the *\*Rating\** variable is clearly a categorical (and now a factor) variable, we can't impute it with median value. Looking also at the distribution of number of games per rating on Figure 16, we don't see any clear "obvious-choice" to fill *\*NA\*s with. That's why, even if it's reducing the data set substantially, I've decided to exclude rows from the data set where \*Rating\** is *\*NA\**. |
|  |  |
|  | ```{r} |
|  | video\_sales <- video\_sales %>% filter(!is.na(Rating)) |
|  | ``` |
|  |  |
|  | **## Game title analysis** |
|  |  |
|  | The last variable that is left to be analyzed is *\*Name\** variable. As game titles are (usually) belonging to one game only and never repeating, it seems like we should automatically discard this variable from machine learning models. But is there any pattern in titles? If yes, what would be this pattern's influence on sales output? |
|  |  |
|  | To help with those questions, I want to create a wordcloud of most used words in the game titles. Using *\*wordcloud\** package, I first create a corpus out of *\*Name\** variable called *\*text\**, that can be used in the actual function. |
|  |  |
|  | ```{r} |
|  | text <- Corpus(VectorSource(video\_sales$Name)) |
|  | ``` |
|  |  |
|  | After that, I clean *\*text\** from numbers, punctuation, unnecessary spaces and stopwords common for English language. |
|  |  |
|  | ```{r warning=FALSE} |
|  | text <- text %>% |
|  | tm\_map(removeNumbers) %>% |
|  | tm\_map(removePunctuation) %>% |
|  | tm\_map(stripWhitespace) %>% |
|  | tm\_map(removeWords, stopwords("english")) |
|  | ``` |
|  |  |
|  | Next, I change cleaned *\*text\** into matrix format representing term document matrix (appearance of words in cleaned *\*text\** as numbers). Then I sum the instances when each word is appearing and rearrange words in decreasing manner. Finally I create a *\*text.df\** data frame with each word and its frequency. |
|  |  |
|  | ```{r} |
|  | text <- as.matrix(TermDocumentMatrix(text)) |
|  | words <- sort(rowSums(text),decreasing=TRUE) |
|  | text.df <- data.frame(word = names(words),freq=words) |
|  | ``` |
|  |  |
|  | Now that I have final data frame for wordcloud graph, let's inspect top words that were calculated. |
|  |  |
|  | ```{r eval=FALSE} |
|  | text.df %>% select(freq) %>% head(10) |
|  | ``` |
|  |  |
|  | ```{r echo=FALSE, message=FALSE} |
|  | text.df %>% |
|  | select(freq) %>% |
|  | head(10) %>% |
|  | kbl(booktabs=T, caption="Top 10 most frequent words in game titles",align="c") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_option="HOLD\_position") |
|  | ``` |
|  |  |
|  | Looking at Table 12 most of the top words seem to be valid, except top one. Since "the" is used in game titles mostly as an article, I would like for it to not be counted in the analysis. As it was not removed automatically during previous cleaning of *\*text\**, I will exclude it manually from *\*text.df\** data frame. |
|  |  |
|  | ```{r} |
|  | text.df <- text.df[!text.df$word=="the",] |
|  | ``` |
|  |  |
|  | Other than that, it is also visible from Table 12 that only *\*world\** word is popping out as high frequency word, and the rest of top 20 has pretty similar count. |
|  |  |
|  | Now that the data frame created from chopped text string of *\*Name\** is prepared, I can create actual wordcloud below. |
|  |  |
|  | ```{r warning=FALSE, fig.cap="Most used words in game titles", fig.pos="H", fig.height=4} |
|  | wordcloud(text.df$word, text.df$freq, min.freq = 50, max.words=200, |
|  | random.order=FALSE, max.freq=400, rot.per=0.35, |
|  | colors=brewer.pal(12, "Paired")) |
|  | ``` |
|  |  |
|  | \pagebreak |
|  | Knowing frequency of each word in game titles, let's try to see how (and if) they are influencing *\*Global\_Sales\**. |
|  |  |
|  | To find if there is any correlation between those, first I create an additional column called *\*sales\**, that shows sum of *\*Global\_Sales\** of all games that had the particular word in the title. To speed up the code I've created a *\*c\** vector beforehand. |
|  |  |
|  | ```{r} |
|  | c <- count(text.df)[[1]] |
|  | for (i in 1:c) { |
|  | text.df$sales[i] <- |
|  | video\_sales$Global\_Sales[str\_detect(video\_sales$Name, |
|  | fixed(text.df$word[i], |
|  | ignore\_case=TRUE))] %>% |
|  | sum() |
|  | } |
|  | ``` |
|  |  |
|  | Now that I've changed *\*text.df\** data frame, I can create a correlation plot between sum of *\*Global\_Sales\** and frequency of words. It is depicted, together with line smoothed with *\*loess\** method, on Figure 18 below. \pagebreak |
|  |  |
|  | ```{r echo=FALSE, fig.cap="Sales per frequency of words used in the game title", message=FALSE, fig.pos="H"} |
|  | text.df %>% |
|  | ggplot(aes(x=freq, y=sales)) + |
|  | geom\_point() + |
|  | geom\_smooth(method=loess, se=TRUE) + |
|  | theme\_bw() + |
|  | labs(x="Frequency of the word used in game title", y ="Global Sales [M units]") |
|  | ``` |
|  |  |
|  | Looking at Figure 18 it doesn't seem like there would be a very strong correlation between words used and *\*Global\_Sales\** output (the smoothed line is definitely curving). Due to that I decide that *\*Name\** variable can be removed from *\*video\_sales\** data set without replacing it with adjusted word factor, and without hurting significantly the performance of prediction models. |
|  |  |
|  | ```{r} |
|  | video\_sales <- video\_sales %>% select(-Name) |
|  | ``` |
|  |  |
|  | ```{r echo=FALSE} |
|  | # Cleaning the environment |
|  | rm(text.df, text, c, i, words) |
|  | ``` |
|  |  |
|  | **## Final check of the data set** |
|  |  |
|  | Now, after cleaning and transforming *\*video\_sales\** data set, let's explore one more time how the summary of it looks like. |
|  |  |
|  | ```{r} |
|  | summary(video\_sales) |
|  | ``` |
|  |  |
|  | It doesn't seem like there are any outliers that we didn't explore, and all variables are in norm. It doesn't seem also like there are any *\*NA\** values left, but let's double-check. |
|  |  |
|  | ```{r} |
|  | is.na(video\_sales) %>% sum() |
|  | ``` |
|  |  |
|  | As thought, there are no *\*NA\*s left that could disturb modeling. A final look into the status of \*video\_sales\** below tells us, that besides *\*Platform\** there are no other character variables. Only regional sales have values equal to 0 (assuming that those titles were not sold in those regions), and one instance of 0 in *\*User\_Score\** was already explored and explained as valid. The *\*video\_sales\** data set should be ready for modeling in next steps. |
|  |  |
|  | ```{r} |
|  | status(video\_sales) |
|  | ``` |
|  |  |
|  | **# Regression models** |
|  |  |
|  | **## Correlation between variables and PCA** |
|  |  |
|  | Now that the basic data set for modeling is ready, let's explore different options for training set and used formula. |
|  |  |
|  | First, let's think about formula that I will be using. The idea is to first start with predicting total sales, so putting *\*Global\_Sales\** as an output. Choosing that, obviously I can't use all of regional sales variables (like *\*NA\_Sales\** or *\*JP\_Sales\**) as they all add to the same value of total *\*Global\_Sales\**. |
|  |  |
|  | To check if any of the variables left need to be marked as interacting with each other, I will create a correlation plot of all variables in *\*video\_sales\**. |
|  |  |
|  | ```{r, fig.cap="Correlogram of variables selected for modeling", fig.pos="H", fig.height=4} |
|  | corrgram(video\_sales[,sapply(video\_sales, is.numeric)], |
|  | order=TRUE, lower.panel=panel.shade, |
|  | upper.panel=panel.cor, text.panel=panel.txt, |
|  | main="Correlogram of variables selected for modeling") |
|  | ``` |
|  |  |
|  | As it's visible on plot above, variables that could be considered cross-depended are most of sales variables, especially *\*Global\_Sales\**, *\*NA\_Sales\** and *\*EU\_Sales\** (with R>0.8). I assume that rest of the variables are independent from each other. |
|  |  |
|  | Second thing I wanted to look at before starting modeling was how to define training and test sets for models. From the first glance on the data set (Table 1) and correlation plot above, it doesn't make much sense to follow PCA (Principal Component Analysis), as there are not so many variables and they are not tightly interacting with one another (a lot of variables have R<0.3 between each other). However, I was quite curious to see how many components would be needed to explain most of variance, and check if this still might be better approach than just slicing *\*video\_sales\** into training and testing sets. |
|  |  |
|  | For now, for the purpose of this PCA I've focused only on numerical variables, creating a *\*pca\_data\** out of *\*video\_sales\**, keeping in mind that if we would be following this approach all variables would need to be transformed to numerical equivalents. |
|  |  |
|  | ```{r} |
|  | pca\_data <- video\_sales %>% select(-Platform,-Genre,-Rating) |
|  | ``` |
|  |  |
|  | Next, I've calculated actual components split out of *\*pca\_data\**. |
|  |  |
|  | ```{r} |
|  | pr.vsales <- prcomp(pca\_data, scale=TRUE) |
|  | summary(pr.vsales) |
|  | ``` |
|  |  |
|  | Having done that, I've calculated and created plots of variance and cumulative variance of PCA calculated one step before. |
|  |  |
|  | ```{r} |
|  | pr.var <- pr.vsales$sdev^2 |
|  | pve <- pr.var/sum(pr.var) |
|  | ``` |
|  |  |
|  | ```{r eval=FALSE} |
|  | par(mfrow=c(2,1)) |
|  | plot(pve, type="b", lwd=2, col="#FD8D3C", |
|  | main="Variance of PCA", |
|  | xlab="Number of Principal Components", |
|  | ylab="% of Variance") |
|  | plot(cumsum(pve), type="b", lwd=2, col="#41B6C4", |
|  | main="Cumulative variance of PCA", |
|  | xlab="Number of Principal Components", |
|  | ylab="% of Cumulative Variance") |
|  | abline(h=0.9, lty=2) |
|  | ``` |
|  | \pagebreak |
|  |  |
|  | ```{r echo=FALSE, fig.cap="Variance and cumulative variance of PCA"} |
|  | par(mfrow=c(2,1)) |
|  | plot(pve, type="b", lwd=2, col="#FD8D3C", |
|  | main="Variance of PCA", |
|  | xlab="Number of Principal Components", |
|  | ylab="% of Variance") |
|  | plot(cumsum(pve), type="b", lwd=2, col="#41B6C4", |
|  | main="Cumulative variance of PCA", |
|  | xlab="Number of Principal Components", |
|  | ylab="% of Cumulative Variance") |
|  | abline(h=0.9, lty=2) |
|  | ``` |
|  |  |
|  | The dashed line on cumulative variance plot above marks moment when components explain 90% of variance. It's evident that we would need 7 components for that, out of 12 (keeping in mind that three variables are missing from this analysis as they were not transformed into numerical ones). This would mean that using PCA method for modeling I would still need at least 60% of whole data set. Given that we don't have as many variables or very big data set in the first place, I've decided to follow classic approach of simple splitting, instead of PCA. |
|  |  |
|  | ```{r echo=FALSE} |
|  | # Cleaning the environment |
|  | rm(pca\_data,pr.vsales,pr.var,pve) |
|  | ``` |
|  |  |
|  | **## Predicting global sales** |
|  |  |
|  | As a first step before training prediction models for *\*Global\_Sales\** as an output, I create a new data set out of cleaned *\*video\_sales\** one. This one, called *\*globalsales\** will consist only of variables that are used as predictors and the output. |
|  |  |
|  | Despite being the most correlated towards *\*Global\_Sales\** (see Figure 19 above), regional sales would not be helpful in actual prediction model. The sum of them is adding up to value of *\*Global\_Sales\**, so they can't be used all together as they would be self-explaining the output. Using some of them could be an approach that some wants to follow, but I've decided to first build a model based on predictors other than sales. In this case, using even one of the regional sales variables could drown the other predictors. |
|  |  |
|  | ```{r} |
|  | globalsales <- video\_sales %>% |
|  | select(-NA\_Sales,-EU\_Sales,-Other\_Sales, -JP\_Sales) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | set.seed(1) |
|  | ``` |
|  |  |
|  | Now, let's create a train and test sub-sets by splitting *\*globalsales\** following 80/20 Pareto Principle, with 80% of *\*globalsales\** data set selected for training in *\*global.train\** and the rest for testing in *\*global.test\**. |
|  |  |
|  | ```{r} |
|  | global.test\_index <- createDataPartition(y = globalsales$Global\_Sales, |
|  | times = 1, p = 0.2, list = FALSE) |
|  | global.test <- globalsales %>% slice(global.test\_index) |
|  | global.train <- globalsales %>% slice(-global.test\_index) |
|  | ``` |
|  |  |
|  | As some of the variables are not numerical, let's make sure that their unique representation is present equally in both *\*global.train\** and *\*global.test\** sets. |
|  |  |
|  | ```{r} |
|  | global.train <- global.train %>% |
|  | semi\_join(global.test, by = "Platform") %>% |
|  | semi\_join(global.test, by="Genre") %>% |
|  | semi\_join(global.test, by="Rating") |
|  | ``` |
|  |  |
|  | For modeling I've selected five regression models: |
|  |  |
|  | \* Linear Regression (LM) |
|  | \* Cubist |
|  | \* Regression Trees (Rpart) |
|  | \* Random Forest (RF) |
|  | \* Non-Negative Least Squares (NNLS) |
|  |  |
|  | All of them I'll run with *\*train\** function from *\*caret\** package (for Cubist and NNLS additional packages are required), and create a vector of predicted outputs using *\*predict\** function. All models would be 10-fold cross-validated, to try to generalize as much as possible with different data sets and avoid overfitting. Additionally to that, in all models input variables will be scaled and centered as pre-processing. Models without pre-processing were also tested, but mostly they were performing worse, with higher RMSE and lower R-square results. |
|  |  |
|  | Starting from Linear Regression, the code for training and predicting is presented below. Linear Regression function doesn't have any additional parameters, so none could be tuned. |
|  |  |
|  | ```{r message=FALSE, warning=FALSE} |
|  | global.lm.fit <- train(Global\_Sales~.,data=global.train, method="lm", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.lm.pred <- predict(global.lm.fit,global.test) |
|  | ``` |
|  |  |
|  | Second model trained was following Cubist method. It was also pre-processed and 10-fold cross-validated. *\*Global.cubist.fit\** fitted model and *\*global.cubist.pred\** prediction vector were created. |
|  |  |
|  | ```{r message=FALSE} |
|  | global.cubist.fit <- train(Global\_Sales~.,data=global.train, method="cubist", |
|  | trControl=trainControl(method="cv",number=10), |
|  | tuneGrid=data.frame(committees=seq(1,9,1), |
|  | neighbors=seq(1,9,1)), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.cubist.pred <- predict(global.cubist.fit,global.test) |
|  | ``` |
|  |  |
|  | As Cubist can have *\*committees\** and *\*neighbors\** parameters tuned, I've been trying to find best fit using *\*tuneGrid\** parameter in *\*train\** function. Both tuned parameters can be integers with values less than 10, so range from 1 to 9 seems to be the simplest one. Plot showing the best fit of those parameters that minimizes RMSE is shown below. |
|  |  |
|  | ```{r, fig.cap="RMSE values for different tuned parameters for Cubist model of Global Sales"} |
|  | plot(global.cubist.fit,highlight = TRUE) |
|  | ``` |
|  |  |
|  | The best tune value is therefore: |
|  |  |
|  | ```{r} |
|  | global.cubist.fit$bestTune |
|  | ``` |
|  |  |
|  | Third model that I was training was Regression Trees. It was also pre-processed and 10-fold cross-validated. Regression Trees method (represented by *\*rpart\**) has tunable complexity parameter *\*cp\**. The code for model and prediction vector creation is visible below. |
|  |  |
|  | ```{r message=FALSE} |
|  | global.rpart.fit <- train(Global\_Sales~.,data=global.train, method="rpart", |
|  | tuneGrid=data.frame(cp=seq(0,0.05,0.005)), |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.rpart.pred <- predict(global.rpart.fit,global.test) |
|  | ``` |
|  |  |
|  | As *\*cp\** is also tunable parameter, let's see on the graph at which value we can achieve minimal RMSE. |
|  |  |
|  | ```{r fig.cap="RMSE values for different tuned parameters for Rpart model of Global Sales"} |
|  | ggplot(global.rpart.fit, highlight=TRUE) |
|  | global.rpart.fit$bestTune |
|  | ``` |
|  |  |
|  | The best fit according to code and plot above is for *\*cp\** equal to 0. In Regression Tree method, *\*cp\** equal 0 means that it is better to not prune the tree at all and create as many splits as possible. Due to that I've decided not to plot created tree, as it would be unreadable. It is also first indicator that the fit here might not be very good, or our data set is too small for this prediction. |
|  |  |
|  | Fourth model that I was training was Random Forest type. Here, an additional fixed parameter besides 10-fold cross-validation and pre-processing was introduced - number of trees *\*ntree\**. I set is up as a standard 100, as any higher number was heavily influencing calculation time with not significant improvement in results. Tuned parameter here is *\*mtry\**, for which I've tried to find best fit from integers between 1 and 25. |
|  |  |
|  | ```{r message=FALSE} |
|  | global.rf.fit <- train(Global\_Sales~., data = global.train, method="rf", |
|  | trControl=trainControl(method="cv", number=10), |
|  | tuneGrid=data.frame(mtry=seq(1,25,1)), |
|  | ntree=100, preProcess = c("center", "scale")) |
|  |  |
|  | global.rf.pred <- predict(global.rf.fit,global.test) |
|  | ``` |
|  |  |
|  | To see what value of *\*mtry\** is defined as best and minimizing RMSE, I plot the tuned outputs and check within model *\*bestTune\** variable. |
|  |  |
|  | ```{r fig.cap="RMSE values for different tuned parameters for RF model of Global Sales", fig.pos="H"} |
|  | global.rf.fit$bestTune |
|  | ggplot(global.rf.fit, highlight=TRUE) |
|  | ``` |
|  |  |
|  | It seams that best value of *\*mtry\** would be around 24, however by looking closely it seems that since 10 the RMSE value is up and down just slightly and overall could be perceived as constant. |
|  |  |
|  | Last model that I was training here was Non-Negative Least Squares (also requiring additional *\*nnls\** package for calculations). This model does not have any tunable parameters, so it was only pre-processed and cross-validated, same as previous models. |
|  |  |
|  | ```{r message=FALSE} |
|  | global.nnls.fit <- train(Global\_Sales~.,data=global.train, method="nnls", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.nnls.pred <- predict(global.nnls.fit,global.test) |
|  | ``` |
|  |  |
|  | Now that all models have been created, let's compare the RMSE and R-squared results of all of them. I've created a list of fitted models, that would be then used as input for resampling. |
|  |  |
|  | ```{r} |
|  | models <- list(lm=global.lm.fit,cubist=global.cubist.fit,rpart=global.rpart.fit, |
|  | rf=global.rf.fit,nnls=global.nnls.fit) |
|  | ``` |
|  |  |
|  | This list is then used as input for RMSE, MAE and R-squared boxplots, which we get if we *\*resample\** the list beforehand. |
|  |  |
|  | ```{r fig.cap="Global Sales learning models comparison"} |
|  | bwplot(resamples(models),metric=c("RMSE","Rsquared","MAE"), |
|  | main="Global Sales learning models comparison") |
|  | ``` |
|  |  |
|  | The boxplot above gives us already an indication which models might be performing better than others, but since their results are quite close to one another, it's hard to tell exactly median values for each. That's why I've created a *\*global.results\** data frame below, that lists median values of RMSE, MAE and R-squared of all models from all their runs that have been calculated above. |
|  |  |
|  | ```{r} |
|  | mae <- list(median(global.lm.fit$resample$MAE), |
|  | median(global.cubist.fit$resample$MAE), |
|  | median(global.rpart.fit$resample$MAE), |
|  | median(global.rf.fit$resample$MAE), |
|  | median(global.nnls.fit$resample$MAE)) |
|  | rmse <- list(median(global.lm.fit$resample$RMSE), |
|  | median(global.cubist.fit$resample$RMSE), |
|  | median(global.rpart.fit$resample$RMSE), |
|  | median(global.rf.fit$resample$RMSE), |
|  | median(global.nnls.fit$resample$RMSE)) |
|  | rsqr <- list(median(global.lm.fit$resample$Rsquared), |
|  | median(global.cubist.fit$resample$Rsquared), |
|  | median(global.rpart.fit$resample$Rsquared), |
|  | median(global.rf.fit$resample$Rsquared), |
|  | median(global.nnls.fit$resample$Rsquared)) |
|  |  |
|  | global.results <- data.frame(Model=c("LM","Cubist","Rpart","RandomForest","NNLS"), |
|  | Median\_MAE=unlist(mae), Median\_RMSE=unlist(rmse), |
|  | Median\_Rsquared=unlist(rsqr)) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | global.results %>% |
|  | kable(booktabs=T, caption="Median MAE, RMSE and R-squared results for Global Sales prediction models") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "hold\_position") |
|  | ``` |
|  |  |
|  | From the table above we can see that none of the selected models are statistically significant, as they all have R<0.65 and quite high RMSE. The closest ones are Cubist and Random Forest (RF). |
|  |  |
|  | Let's take a final look on all models outputs (actual and predicted ones) on the figure below.\pagebreak |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Actual vs Predicted output values for all calculated Global Sales prediction models", fig.pos="H"} |
|  | plot.lm <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.lm.pred) %>% mutate(Type="LM") |
|  | plot.cubist <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.cubist.pred) %>% mutate(Type="Cubist") |
|  | plot.rpart <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.rpart.pred) %>% mutate(Type="Rpart") |
|  | plot.rf <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.rf.pred) %>% mutate(Type="RandomForest") |
|  | plot.nnls <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.nnls.pred) %>% mutate(Type="NNLS") |
|  |  |
|  | final.plot <- rbind(plot.lm,plot.cubist,plot.rpart,plot.rf,plot.nnls) |
|  |  |
|  | ggplot(final.plot, aes(x=Predicted, y=Actuals)) + |
|  | geom\_point() + |
|  | geom\_smooth(method="loess", se=TRUE) + |
|  | facet\_wrap(.~Type) + |
|  | theme\_bw() |
|  | ``` |
|  |  |
|  | On Figure 25 above all plots are close to follow linear regression for the values closest to 0. However, the further we go for higher values, then more plots start to deviate from it (fastest deviating being LM an NNLS models). This proves that actually none of those models are good at predicting *\*Global\_Sales\** output. |
|  |  |
|  | ```{r echo=FALSE} |
|  | # Cleaning the environment |
|  | rm(final.plot,mae,models,rmse,rsqr, plot.cubist,plot.lm,plot.nnls,plot.rf,plot.rpart) |
|  | ``` |
|  |  |
|  |  |
|  | **## Predicting North America sales** |
|  |  |
|  | So if I can't obtain good model for predicting *\*Global\_Sales\** from *\*video\_sales\** data set, is there any other information that we can pull from it? Let's maybe focus on the most correlated with *\*Global\_Sales\** variable (see Figure 19), which was *\*NA\_Sales\**. I've excluded it from previous prediction models, as I didn't want to use any of regional sales for total sum prediction, but maybe following this path could lead to better results. |
|  |  |
|  | I can guess that trying to predict *\*NA\_Sales\** with the same set of parameters that I chose for *\*Global\_Sales\** would lead to similar results, so let's try different approach. What if we wanted to predict *\*NA\_Sales\** (so the biggest regional consumer - see Figure 4) for a game that was already released, gathered reviews from both Critics and Users AND we know how it performed in other regions? This way we would add more variables correlated strongly to the output we want to predict. |
|  |  |
|  | First, let's again create a new data frame for modeling, that consists of all our variables, without *\*Global\_Sales\**. This new data frame will be called *\*NAsales\**. |
|  |  |
|  | ```{r} |
|  | NAsales <- video\_sales %>% select(-Global\_Sales) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | set.seed(1) |
|  | ``` |
|  |  |
|  | Then, similarly to previous modeling for *\*Global\_Sales\**, I've created test and train sets, following Pareto Principle of 80/20 split. |
|  |  |
|  | ```{r} |
|  | na.test\_index <- createDataPartition(y = NAsales$NA\_Sales, |
|  | times = 1, p = 0.2, list = FALSE) |
|  | na.test <- NAsales%>% slice(na.test\_index) |
|  | na.train <- NAsales %>% slice(-na.test\_index) |
|  | ``` |
|  |  |
|  | As a final preparation step before modeling, I've joined *\*na.train\** and *\*na.test\** sets on the non-numerical variables, to make sure that the same unique variables can be found in both sets, which will prevent models crashing due to near-zero variables importance in those variables. Using semi-join, we will be able to eliminate most of those instances (however *\*Rating\** might still not have enough representatives in *\*EC\** group). |
|  |  |
|  | ```{r} |
|  | na.train <- na.train %>% |
|  | semi\_join(na.test, by="Rating") %>% |
|  | semi\_join(na.test, by="Platform") %>% |
|  | semi\_join(na.test, by="Genre") |
|  | ``` |
|  |  |
|  | For comparison sake, I've used same regression models as during modeling *\*Global\_Sales\** output: Linear Regression, Cubist, Regression Trees, Random Forest and Non-Negative Least Squares. All models for *\*NA\_Sales\** prediction were also 10-fold cross-validated and pre-processed (scaled and centered variables). |
|  |  |
|  | First, let's model Linear Regression model and create prediction vector. |
|  |  |
|  | ```{r message=FALSE, warning=FALSE} |
|  | na.lm.fit <- train(NA\_Sales~.,data=na.train, method="lm", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.lm.pred <- predict(na.lm.fit,na.test) |
|  | ``` |
|  |  |
|  | Following that is the Cubist model. It has a possibility to tune *\*committees\** and *\*neighbors\** parameters, for which I set a range same as for previous Cubist modeling (integer from 1 to 9). |
|  |  |
|  | ```{r message=FALSE, warning=FALSE} |
|  | na.cubist.fit <- train(NA\_Sales~.,data=na.train, method="cubist", |
|  | trControl=trainControl(method="cv",number=10), |
|  | tuneGrid=data.frame(committees=seq(1,9,1), |
|  | neighbors=seq(1,9,1)), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.cubist.pred <- predict(na.cubist.fit,na.test) |
|  | ``` |
|  |  |
|  | Tuned parameters can be seen on the plot below, with highlighted best value for them that is used in final model. The best value of tuned parameter is then 5. |
|  |  |
|  | ```{r eval=FALSE} |
|  | na.cubist.fit$bestTune |
|  | plot(na.cubist.fit,highlight = TRUE) |
|  | ``` |
|  |  |
|  | ```{r echo=FALSE} |
|  | na.cubist.fit$bestTune |
|  | ``` |
|  |  |
|  | \pagebreak |
|  |  |
|  | ```{r echo=FALSE, fig.cap="RMSE values for different tuned parameters for Cubist model of NA Sales"} |
|  | plot(na.cubist.fit,highlight = TRUE) |
|  | ``` |
|  |  |
|  | Next one is Regression Trees (rpart) model, where tunable parameter is *\*complexity parameter (cp)\**, for which we also set same range of tuning as in predicting *\*Global\_Sales\** output. |
|  |  |
|  | ```{r messsage=FALSE, warning=FALSE} |
|  | na.rpart.fit <- train(NA\_Sales~.,data=na.train, method="rpart", |
|  | tuneGrid=data.frame(cp=seq(0,0.05,0.005)), |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.rpart.pred <- predict(na.rpart.fit,na.test) |
|  | ``` |
|  |  |
|  | Similarly like in the *\*Global\_Sales\** predicting, cp parameter is equal to zero, which implies that the tree should not be pruned and be as complex as possible to achieve lowest RMSE. |
|  |  |
|  | This can be seen on tuning plot below, where *\*cp\** parameter with lowest RMSE is highlighted. |
|  |  |
|  | ```{r } |
|  | na.rpart.fit$bestTune |
|  | ``` |
|  |  |
|  | ```{r fig.cap="RMSE values for different tuned parameters for Rpart model of NA Sales", fig.pos="H", fig.height=5} |
|  | ggplot(na.rpart.fit,highlight = TRUE) |
|  | ``` |
|  |  |
|  | Fourth model that I will compare is Random Forest (rf) model. Besides tunable parameter *\*mtry\** it can have modified number of trees made *\*ntree\**. Similarly to before, for both of these I've set range and value same as for previous Random Forest modeling, equal to 100 trees and integer range from 1 to 25. |
|  |  |
|  | ```{r message=FALSE, warning=FALSE} |
|  | na.rf.fit <- train(NA\_Sales~.,data=na.train, method="rf", |
|  | tuneGrid=data.frame(mtry=seq(1,25,1)), ntree=100, |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.rf.pred <- predict(na.rf.fit,na.test) |
|  | ``` |
|  |  |
|  | The best value of *\*mtry\** that gives lowest RMSE is 19, which can be seen on plot below. \pagebreak |
|  |  |
|  | ```{r fig.cap="RMSE values for different tuned parameters for RF model of NA Sales", fig.pos="H", fig.height=5} |
|  | ggplot(na.rf.fit,highlight = TRUE) |
|  | ``` |
|  |  |
|  | ```{r } |
|  | na.rf.fit$bestTune |
|  | ``` |
|  |  |
|  | The last model that I will be calculating is NNLS. For this one there were no tunable parameters. |
|  |  |
|  | ```{r message=FALSE, warning=FALSE} |
|  | na.nnls.fit <- train(NA\_Sales~.,data=na.train, method="nnls", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.nnls.pred <- predict(na.nnls.fit,na.test) |
|  | ``` |
|  |  |
|  | Now that all models for *\*NA\_Sales\** predictions have been computed, I've created a box plot containing RMSE, MAE and R-squared results for all of them. |
|  |  |
|  | ```{r fig.cap="NA Sales learning models comparison"} |
|  | models <- list(lm=na.lm.fit, cubist=na.cubist.fit, rpart=na.rpart.fit, |
|  | randomforest=na.rf.fit, nnls=na.nnls.fit) |
|  |  |
|  | bwplot(resamples(models),metric=c("RMSE","Rsquared","MAE"), |
|  | main="NA Sales learning models comparison") |
|  | ``` |
|  |  |
|  | Above box plot clearly shows that all models have much better results than those for *\*Global\_Sales\** prediction. It also indicates that all of them would be statistically significant models (with R>0.65), but for the actual confirmation of that let's check more precise numbers. |
|  |  |
|  | Below I've computed median numbers of all three results indicators for all models computations (similarly like it was done for *\*Global\_Sales\**). |
|  |  |
|  | ```{r} |
|  | mae <- list(median(na.lm.fit$resample$MAE), |
|  | median(na.cubist.fit$resample$MAE), |
|  | median(na.rpart.fit$resample$MAE), |
|  | median(na.rf.fit$resample$MAE), |
|  | median(na.nnls.fit$resample$MAE)) |
|  | rmse <- list(median(na.lm.fit$resample$RMSE), |
|  | median(na.cubist.fit$resample$RMSE), |
|  | median(na.rpart.fit$resample$RMSE), |
|  | median(na.rf.fit$resample$RMSE), |
|  | median(na.nnls.fit$resample$RMSE)) |
|  | rsqr <- list(median(na.lm.fit$resample$Rsquared), |
|  | median(na.cubist.fit$resample$Rsquared), |
|  | median(na.rpart.fit$resample$Rsquared), |
|  | median(na.rf.fit$resample$Rsquared), |
|  | median(na.nnls.fit$resample$Rsquared)) |
|  |  |
|  | na.results <- data.frame(Model=c("LM","Cubist","Rpart","RandomForest","NNLS"), |
|  | Median\_MAE=unlist(mae), Median\_RMSE=unlist(rmse), |
|  | Median\_Rsquared=unlist(rsqr)) |
|  | ``` |
|  | ```{r echo=FALSE} |
|  | na.results %>% |
|  | kable(booktabs=T, caption="Median MAE, RMSE and R-squared results for NA Sales prediction models") %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = "hold\_position") |
|  | ``` |
|  |  |
|  | From the table above it's now clear that all models were achieving much better results than during predicting *\*Global\_Sales\** output. The worst performance comes from NNLS model, with highest RMSE (0.52) and lowest R-Squared (0.68). The best one would be then Cubist model, with lowest RMSE (0.27), highest R-Squared (0.88) and lowest MAE (0.08). |
|  |  |
|  | Let's now take a look at Actual vs Predicted *\*NA\_Sales\** output plots below. |
|  |  |
|  | ```{r message=FALSE} |
|  | plot.lm <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.lm.pred) %>% mutate(Type="LM") |
|  | plot.cubist <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.cubist.pred) %>% mutate(Type="Cubist") |
|  | plot.rpart <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.rpart.pred) %>% mutate(Type="Rpart") |
|  | plot.rf <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.rf.pred) %>% mutate(Type="RandomForest") |
|  | plot.nnls <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.nnls.pred) %>% mutate(Type="NNLS") |
|  |  |
|  | final.plot <- rbind(plot.lm,plot.cubist,plot.rpart,plot.rf,plot.nnls) |
|  | ``` |
|  |  |
|  | \pagebreak |
|  |  |
|  | ```{r echo=FALSE, message=FALSE, fig.cap="Actual vs Predicted output values for all calculated Global Sales prediction models", fig.pos="H", fig.height=4} |
|  | ggplot(final.plot, aes(x=Predicted, y=Actuals)) + |
|  | geom\_point() + |
|  | geom\_smooth(method="loess", se=TRUE) + |
|  | facet\_wrap(.~Type) + theme\_bw() |
|  | ``` |
|  |  |
|  | Almost all plots above show much straighter smoothed lines, almost linear in lower ranges. The best ones are also Cubist and Random Forest, while the worst ones (having slight curve even in the lowest range) are NNLS and LM. Both of those cases prove what was shown already in Table 14, where Cubist and Random Forest are the best performing, while NNLS and LM are the worst. |
|  |  |
|  | Knowing now that all chosen models could be used (with better or worse results) for predicting *\*NA\_Sales\**, let's take final look into variable importance for each of them. On Table 15 below I've created a summary of top 10 most important variables for each model, using *\*varImp\** function. |
|  |  |
|  | ```{r} |
|  | na.lm.imp <- varImp(na.lm.fit) |
|  | na.cubist.imp <- varImp(na.cubist.fit) |
|  | na.rpart.imp <- varImp(na.rpart.fit) |
|  | na.rf.imp <- varImp(na.rf.fit) |
|  | na.nnls.imp <- varImp(na.nnls.fit) |
|  | ``` |
|  |  |
|  | \pagebreak |
|  |  |
|  | ```{r echo=FALSE} |
|  | imp.lm <- data.frame(Variable=rownames(arrange(na.lm.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.lm.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.cubist <- data.frame(Variable=rownames(arrange(na.cubist.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.cubist.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.rpart <- data.frame(Variable=rownames(arrange(na.rpart.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.rpart.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.rf <- data.frame(Variable=rownames(arrange(na.rf.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.rf.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.nnls <- data.frame(Variable=rownames(arrange(na.nnls.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.nnls.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  |  |
|  | imp.table <- cbind(imp.cubist,imp.rf, imp.lm, imp.rpart, imp.nnls) |
|  |  |
|  | kable(imp.table,booktabs=T, caption="Variables importance in five checked regression models for NA Sales") %>% |
|  | add\_header\_above(c("Cubist"=2, "RF"=2,"LM"=2,"Rpart"=2, "NNLS"=2)) %>% |
|  | add\_header\_above(c("Best performing models"=4,"Other models"=6)) %>% |
|  | kable\_classic() %>% |
|  | kable\_styling(latex\_options = c("hold\_position", "scale\_down")) |
|  | ``` |
|  |  |
|  | What is clearly visible from above Table is that for the best models (which are Cubist and RF) most of variance is almost equally explained by both *\*Other\_Sales\** and *\*EU\_Sales\** variables (both oscillating near 90-100%). For the rest of them only one variable is outstanding with it's importance - *\*EU\_Sales\** explaining 100% variance in all three models of LM, Rpart and NNLS. |
|  |  |
|  | None other variable and its levels are consistent in their importance in all five considered models, and some (like *\*Genre\** and *\*Rating\**) are actually not in top 10 at all. |
|  |  |
|  | ```{r echo=FALSE} |
|  | # Cleaning the environment |
|  | rm(final.plot,mae,models,rmse,rsqr, plot.cubist,plot.lm,plot.nnls,plot.rf,plot.rpart, imp.table) |
|  | ``` |
|  |  |
|  |  |
|  | **# Summary & conclusions** |
|  |  |
|  | When talking about *\*video\_sales\** data set and prediction models calculated from it, we need to be remember that if we were trying to predict games sales in reality, more research into unknown variables would have to be conducted. Those missing variables are currently creating "noise" in our models, and including them could lead to improvement in prediction accuracy. However, most of them would be either hard to get or requiring joining few other new sources to existing *\*video\_sales\** data set. Looking into real world data, the most obvious missing variables that could be big influencers are: |
|  |  |
|  | \* cost spend on marketing (was it global campaign, were any known faces involved) |
|  | \* socio-political changes of the region (is there an economical depression or pandemic happening) |
|  | \* social media influencers generating organic hype (was this game featured in un-sponsored "let's play" by known streamer or Youtuber) |
|  | \* demographics of players (age, gender, nationality, ethnicity) |
|  | \* price of the game and/or platform needed to play |
|  |  |
|  | Another thing that needs to be noted is that I've assumed in prediction models that we're trying to find out final sales value *\*after\** game has been released. Due to that assumptions User and Critic variables (*\*User\_Score\**, *\*User\_Count\**, *\*Critic\_Score\** and *\*Critic\_Count\**) have been included in the models. If models were to be used in prediction of sales upon game release day, those variables would have to be excluded, as they are not known at that point of time (especially User ones, as there might be few Critics with pre-release access). In this case all models for both *\*Global\_Sales\** and *\*NA\_Sales\** would have to be recalculated. |
|  |  |
|  | Regarding machine learning modeling done for two different outputs, it is clear that *\*NA\_Sales\** prediction models have been performing much better than those of *\*Global\_Sales\**. *\*NA\_Sales\** has main difference of including also other regional sales values, being strongly correlated with desired output. All 5 models for this configuration have performed well and could be used to predict actual *\*NA\_Sales\** values. However, *\*Global\_Sales\** models were not critically bad, and I strongly believe it's possible to use at least Cubist and Random Forest models for this prediction, with disclaimer for double-checking actual results. The performance of the models in both cases could be improved if we also take into consideration mentioned before different configuration of input variables, include models that put higher importance on the unexplained noise or enlarger the data set with more up-to-date values. |
|  |  |
|  | The *\*Complexity Parameter (cp)\** in Regression Tree models (Rpart) in both instances had a best fit of 0. The general rule is that the smaller *\*cp\** value the better. This can indicate that I've been really successful in preparing the *\*video\_sales\** data set before and it doesn't require pruning after that, but it may also indicate that we're close to having models being overfitted as the data set is too small. Seeing as in both prediction cases *\*rpart\** model was not the best performing one when confronted with test set, that could indicate that either this is really not the best model for this kind of data or we do really have a case of overfitting that is not so visible in other models. |
|  |  |
|  | On the other hand, we also have an example of *\*Rating\** variable having near-zero variance (for *\*EC\** level) for both prediction cases, despite not showing such tendencies before modeling. Not a single level of this variable is in top 10 most important variables (see Table 15). This could be an indication that this *\*Rating\** level has too little instances and therefore the data set on which we are training the models is too small. As mentioned before, this could be also avoided by removing *\*Rating\** from the variable used for predicting, with the cost of slightly worse results (which are already quite satisfying). |
|  |  |
|  | In summary, possible **\*\*next steps for video game sales prediction models improvement\*\*** could be: |
|  |  |
|  | \* considering adding data after 2016 to the data set to make it larger, |
|  | \* analyze actual gaming related data and events happening through time to understand undiscovered influencers, and possibly include them in prediction models, |
|  | \* removing *\*Rating\** variable as predictor, since it has very low (and almost zero in some instances) variance and importance, |
|  | \* to predict sales expected at the day of release of the game remove User (and possibly also Critic) Score and Count variables. |
|  |  |
|  | \pagebreak |
|  |  |
|  | **# Appendix** |
|  |  |
|  | **## Session info** |
|  |  |
|  | ```{r} |
|  | sessionInfo() |
|  | ``` |
|  |  |
|  | **## Full code** |
|  |  |
|  | ```{r eval=FALSE} |
|  |  |
|  | # This code contains data exploration and prediction models on video game sales. |
|  | # Data set used here is Video Game Sales with Rating data set. |
|  | # This data set has been created by Rush Kirubi on Kaggle |
|  | # Created in Dec 2020. |
|  |  |
|  |  |
|  | # 1. Download and prepare the data set ------------------------------------ |
|  |  |
|  | # Ensuring all packages are installed and loaded |
|  | if(!require(tidyverse)) install.packages("tidyverse", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(funModeling)) install.packages("funModeling", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(caret)) install.packages("caret", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(corrgram)) install.packages("corrgram", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(scales)) install.packages("scales", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(ggpubr)) install.packages("ggpubr", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(tm)) install.packages("tm", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(wordcloud)) install.packages("wordcloud", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(gridExtra)) install.packages("gridExtra", |
|  | repos = "http://cran.us.r-project.org") |
|  | # For model in train function |
|  | if(!require(Cubist)) install.packages("Cubist", |
|  | repos = "http://cran.us.r-project.org") |
|  | if(!require(nnls)) install.packages("nnls", |
|  | repos = "http://cran.us.r-project.org") |
|  |  |
|  | library(tidyverse) |
|  | library(funModeling) |
|  | library(scales) |
|  | library(caret) |
|  | library(corrgram) |
|  | library(ggpubr) |
|  | library(wordcloud) |
|  | library(tm) |
|  | library(gridExtra) |
|  |  |
|  | # Downloading Video Game Sales with Rating data set, |
|  | # created by Rush Kirubi from my GitHub repository |
|  | url\_path <- |
|  | "https://raw.githubusercontent.com/Golfik/Video-Games-Sales/master/ |
|  | Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv" |
|  |  |
|  | raw.videosales <- read.csv(url(url\_path),na.strings = c("","NA")) |
|  | rm(url\_path) |
|  |  |
|  | # Create a duplicate of data\_set for analysis (leaving raw.videosales for reference) |
|  | video\_sales <- raw.videosales |
|  |  |
|  |  |
|  |  |
|  | # 2. Data overview -------------------------------------------------------- |
|  |  |
|  | # First look in the data set |
|  | glimpse(video\_sales) |
|  |  |
|  | # Checking out duplicate values - non found |
|  | duplicated(video\_sales) %>% sum() |
|  |  |
|  | # Looking deeper to see if there are any obvious outliers |
|  | summary(video\_sales) |
|  |  |
|  | # Checking out if there are many NAs and how many unique values |
|  | status(video\_sales) |
|  |  |
|  |  |
|  |  |
|  | # 3. Exploring, visualizing and cleaning the data set --------------------- |
|  |  |
|  |  |
|  | # 3.1. Global and regional sales variables ---- |
|  |  |
|  | # Analyzing density of Global Sales |
|  | ggplot(video\_sales,aes(Global\_Sales)) + |
|  | geom\_histogram(aes(y=..density..),binwidth = 0.5) + |
|  | geom\_density(alpha=0.2,fill="#EF3B2C") + |
|  | labs(x="Global Sales [M units]", y ="Density", |
|  | title="Density plot of Global Sales variable") + |
|  | theme\_bw() |
|  |  |
|  | # Checking out top 10 best selling games globally, |
|  | # to see what outliers may influence long tail on previous plot |
|  | video\_sales %>% |
|  | select(Name,Platform, Genre, User\_Count, Global\_Sales) %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | head(10) |
|  |  |
|  | # Global Sales density plot of all up to 3rd quantile, |
|  | # with shown mean value (zooming to values without tail) |
|  | tot.mean <- video\_sales %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | select(Name, Global\_Sales) %>% |
|  | summarise(tot.mean=mean(Global\_Sales)) %>% |
|  | unlist() |
|  | video\_sales %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | filter(between(Global\_Sales,0,0.47)) %>% |
|  | ggplot(aes(Global\_Sales)) + |
|  | geom\_histogram(aes(y=..density..),binwidth = 0.01) + |
|  | geom\_density(alpha=0.2,fill="#EF3B2C") + |
|  | geom\_vline(aes(xintercept=tot.mean), linetype="dashed", size=1) + |
|  | labs(x="Global Sales [M units]", y ="Density", |
|  | title="Density of Global Sales, exluding top quantile", |
|  | subtitle="Dashed line marking mean value of 0.5335 (all values)") + |
|  | theme\_bw() |
|  | rm(tot.mean) |
|  |  |
|  | # Changing the Year of Release to numeric type |
|  | video\_sales <- video\_sales %>% |
|  | mutate(Year\_of\_Release=as.numeric(Year\_of\_Release)) |
|  |  |
|  | # Global Sales, profit per title and number of games through years |
|  | plot1 <- video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(sum\_of\_sales=sum(Global\_Sales)) %>% |
|  | ggplot() + |
|  | geom\_point(aes(x=Year\_of\_Release, y=sum\_of\_sales), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | geom\_line(aes(x=Year\_of\_Release, y=sum\_of\_sales), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | labs(x = "Year", y = "Global Sales", title="Global sales [M units] through years") + |
|  | theme\_bw() |
|  | plot2 <- video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(sales\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | ggplot() + |
|  | geom\_point(aes(x=Year\_of\_Release, y=sales\_pertitle), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | geom\_line(aes(x=Year\_of\_Release, y=sales\_pertitle), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | labs(x = "Year", y = "Sales per title", |
|  | title="Units sold [M units] per game title through years") + |
|  | theme\_bw() + |
|  | scale\_color\_brewer(palette = "RdYlBu") |
|  | plot3 <- video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(n=n()) %>% |
|  | ggplot() + |
|  | geom\_point(aes(x=Year\_of\_Release, y=n), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | geom\_line(aes(x=Year\_of\_Release, y=n), |
|  | stat = 'identity',show.legend = FALSE) + |
|  | labs(x = "Year", y = "Number of games", |
|  | title="Number of games released through years") + |
|  | theme\_bw() |
|  | ggarrange(plot1,plot2,plot3,ncol=1, align="hv") |
|  | rm(plot1,plot2,plot3) |
|  |  |
|  | # Remove 2020 as weird and 2017 as not enough data (insight from previous graph), |
|  | # as well as the name of original data set suggest is shouldn't be included |
|  | video\_sales %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(Year\_of\_Release)) |
|  | video\_sales <- video\_sales %>% filter(Year\_of\_Release < 2017) |
|  |  |
|  | # Share of sales between regions through years |
|  | video\_sales %>% |
|  | select(Year\_of\_Release, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | gather(Region, Sales, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | group\_by(Year\_of\_Release,Region) %>% |
|  | ggplot() + |
|  | geom\_bar(aes(x=Year\_of\_Release, y=Sales, fill=Region), |
|  | position="fill", stat="identity") + |
|  | labs(x = "Year", y = "Sales", |
|  | title="Share of sales between regions through years") + |
|  | theme\_bw() + |
|  | scale\_y\_continuous(labels = percent\_format()) + |
|  | scale\_fill\_brewer(palette = "RdYlBu", |
|  | labels=c("EU", "Japan", "North America", "Other")) |
|  |  |
|  |  |
|  | # 3.2. Publisher and Developer analysis ----------------------------------- |
|  |  |
|  | # Global sales by publisher, as total sum of sales made |
|  | plot1 <- video\_sales %>% |
|  | gather(Region, Sales, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | group\_by(Publisher,Region) %>% |
|  | summarise(sum\_Sales=sum(Sales),total=sum(Global\_Sales)) %>% |
|  | arrange(desc(total)) %>% |
|  | head(40) %>% |
|  | ggplot(aes(x=reorder(Publisher,desc(total)), y=sum\_Sales,fill=Region)) + |
|  | geom\_bar(stat = "identity") + |
|  | theme\_bw() + |
|  | theme(axis.text.x = element\_text(angle = 15, hjust=0.95)) + |
|  | labs(x="Publisher", y ="Global Sales [M units]", |
|  | title="Top 10 best selling publishers globally, as total sales made") + |
|  | scale\_fill\_brewer(palette = "PRGn", labels=c("EU", "Japan", "North America", "Other")) |
|  |  |
|  | # Global Sales by developer, as total sum of sales |
|  | plot2 <- video\_sales %>% |
|  | filter(!is.na(Developer)) %>% |
|  | gather(Region, Sales, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales) %>% |
|  | group\_by(Developer,Region) %>% |
|  | summarise(sum\_Sales=sum(Sales),total=sum(Global\_Sales)) %>% |
|  | arrange(desc(total)) %>% |
|  | head(40) %>% |
|  | ggplot(aes(x=reorder(Developer,desc(total)), y=sum\_Sales,fill=Region)) + |
|  | geom\_bar(stat = "identity") + |
|  | theme\_bw() + |
|  | theme(axis.text.x = element\_text(angle = 15, hjust=0.95)) + |
|  | labs(x="Developer", y ="Global Sales [M units]", |
|  | title="Top 10 best selling developers globally, as total sales made") + |
|  | scale\_fill\_brewer(palette = "PRGn", labels=c("EU", "Japan", "North America", "Other")) |
|  |  |
|  | # Arranging two previous plots, and clearing environment |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2,common.legend=TRUE, legend="bottom") |
|  | rm(plot,plot2) |
|  |  |
|  | # Global sales made per title by top publishers |
|  | plot1 <- video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | summarise(sale\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | arrange(desc(sale\_pertitle)) %>% |
|  | head(10) %>% |
|  | ggplot(aes(y=sale\_pertitle,x=reorder(Publisher,sale\_pertitle))) + |
|  | geom\_bar(stat = "identity", fill="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Publisher", y ="Sales per title [M units]", |
|  | title="Top 10 publishers by sales per title") + |
|  | coord\_flip() + |
|  | scale\_x\_discrete(labels = function(x) str\_wrap(x, width = 50)) |
|  |  |
|  | # Global sales per title by top developers |
|  | plot2 <- video\_sales %>% |
|  | filter(!is.na(Developer)) %>% |
|  | group\_by(Developer) %>% |
|  | summarise(sale\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | arrange(desc(sale\_pertitle)) %>% |
|  | head(10) %>% |
|  | ggplot(aes(y=sale\_pertitle,x=reorder(Developer,sale\_pertitle))) + |
|  | geom\_bar(stat = "identity", fill="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Developer", y ="Sales per title [M units]", |
|  | title="Top 10 developers by sales per title") + |
|  | coord\_flip() + |
|  | scale\_x\_discrete(labels = function(x) str\_wrap(x, width = 50)) |
|  |  |
|  | # Arranging two previous plots, and clearing environment |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2, align="v") |
|  | rm(plot1,plot2) |
|  |  |
|  | # How many games has top Publisher released? |
|  | video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n()) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Publisher,sale\_per\_title, no\_of\_games) %>% |
|  | unique() %>% |
|  | head(10) |
|  |  |
|  | # How many games has top Developer released? |
|  | video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n()) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Developer,sale\_per\_title, no\_of\_games) %>% |
|  | unique() %>% |
|  | head(10) |
|  |  |
|  | # How will publisher factor look like if we penalize studios with small number |
|  | # of titles with function with limit to 1? |
|  | video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n(), |
|  | adj.factor=no\_of\_games\*sin(1/no\_of\_games), |
|  | Publisher.fct=sale\_per\_title\*no\_of\_games\*sin(1/no\_of\_games)) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Publisher,sale\_per\_title, no\_of\_games, adj.factor, Publisher.fct) %>% |
|  | unique() %>% |
|  | head(10) |
|  |  |
|  | # How will developer factor look like if we penalize studios with small number |
|  | # of titles with function with limit to 1? |
|  | video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(sale\_per\_title=sum(Global\_Sales)/n(),no\_of\_games=n(), |
|  | adj.factor=no\_of\_games\*sin(1/no\_of\_games), |
|  | Developer.fct=sale\_per\_title\*no\_of\_games\*sin(1/no\_of\_games)) %>% |
|  | arrange(desc(sale\_per\_title)) %>% |
|  | select(Developer,sale\_per\_title, no\_of\_games, adj.factor, Developer.fct) %>% |
|  | unique() %>% |
|  | head(10) |
|  |  |
|  | # Add new Developer and Publisher factors to the video\_sales data set, |
|  | # and removing original Publisher and Developer |
|  | video\_sales <- video\_sales %>% |
|  | group\_by(Publisher) %>% |
|  | mutate(Publisher.fct=(sum(Global\_Sales)/n())\*n()\*sin(1/n())) %>% |
|  | ungroup() |
|  | video\_sales <- video\_sales %>% |
|  | group\_by(Developer) %>% |
|  | mutate(Developer.fct=(sum(Global\_Sales)/n())\*n()\*sin(1/n())) %>% |
|  | ungroup() |
|  | video\_sales <- video\_sales %>% |
|  | select(-Publisher,-Developer) |
|  |  |
|  |  |
|  | # 3.3. Platform and Genre analysis ---------------------------------------- |
|  |  |
|  | # Number of game titles released per platform (only top) |
|  | video\_sales %>% |
|  | group\_by(Platform) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(15) %>% |
|  | ggplot(aes(x=reorder(Platform,n),y=n)) + |
|  | geom\_segment(aes(x=reorder(Platform,n), |
|  | xend=reorder(Platform,n), y=0, yend=n), color="grey") + |
|  | geom\_point(color="#FD8D3C", size=4) + |
|  | coord\_flip() + |
|  | theme\_bw() + |
|  | labs(x="Platform", y ="Number of titles released", |
|  | title="Top 15 platforms with highest number of games released") |
|  |  |
|  | # Top selling games, and their genres and platforms |
|  | video\_sales %>% |
|  | arrange(desc(Global\_Sales)) %>% |
|  | head(15) %>% |
|  | ggplot(aes(y=Global\_Sales,x=reorder(Name,Global\_Sales),fill=Genre)) + |
|  | geom\_bar(stat="identity") + geom\_text(aes(label=Platform), hjust=1.5) + |
|  | coord\_flip() + |
|  | theme\_bw() + |
|  | labs(x="Global Sales [M of units]", y ="Game title and its platform", |
|  | title="Top 15 selling games globally") + |
|  | scale\_fill\_brewer(palette = "RdYlBu") |
|  |  |
|  | # Sales per genre |
|  | video\_sales %>% |
|  | group\_by(Genre) %>% |
|  | summarise(n\_titles=n(), total\_sales=sum(Global\_Sales)) %>% |
|  | arrange(desc(total\_sales)) |
|  |  |
|  | # Erasing NAs from Genre |
|  | video\_sales <- video\_sales %>% |
|  | filter(!is.na(Genre)) |
|  |  |
|  | # Changing Genre to a factor |
|  | video\_sales$Genre <- as.factor(video\_sales$Genre) |
|  |  |
|  | # Average sales per one title by each game genre |
|  | video\_sales %>% |
|  | group\_by(Genre) %>% |
|  | summarise(sale\_pertitle=sum(Global\_Sales)/n()) %>% |
|  | arrange(desc(sale\_pertitle)) %>% |
|  | ggplot(aes(y=sale\_pertitle,x=reorder(Genre,desc(sale\_pertitle)))) + |
|  | geom\_bar(stat = "identity", fill="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Genre", y ="Sales per title [M units]", |
|  | title="Sales per one title per game genre") + |
|  | theme(axis.text.x = element\_text(angle = 15, hjust=0.95)) |
|  |  |
|  |  |
|  | # 3.4. User and Critic Score and Count analysis --------------------------- |
|  |  |
|  | # User Score and Critic Score frequency plot, |
|  | # across different Genre (dark theme for visibility) |
|  | plot1 <- ggplot(video\_sales, aes(as.numeric(User\_Score), color=Genre)) + |
|  | geom\_freqpoly(size=1) + |
|  | theme\_dark() + |
|  | scale\_color\_brewer(palette = "Paired") + |
|  | labs(x="User Score", y ="Count", |
|  | title="Distribution of User Scores across game genres") |
|  | plot2 <- ggplot(video\_sales, aes(as.numeric(Critic\_Score), color=Genre)) + |
|  | geom\_freqpoly(size=1) + |
|  | theme\_dark() + |
|  | scale\_color\_brewer(palette = "Paired") + |
|  | labs(x="Critic Score", y ="Count", |
|  | title="Distribution of Critic Scores across game genres") |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2,common.legend=TRUE, legend="bottom") |
|  | rm(plot1,plot2) |
|  |  |
|  | # Checking out User\_Score - why it's character, |
|  | # then change tbd to NA, and variable type to numeric |
|  | video\_sales %>% |
|  | group\_by(User\_Score) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(10) |
|  |  |
|  | video\_sales$User\_Score[video\_sales$User\_Score=="tbd"] <- NA |
|  |  |
|  | video\_sales$User\_Score <- as.numeric(video\_sales$User\_Score) |
|  |  |
|  | #Check the same for Critic Score. |
|  | # There are no character values, and it can be changed to numeric |
|  | video\_sales %>% |
|  | group\_by(Critic\_Score) %>% |
|  | summarise(n=n()) %>% |
|  | arrange(desc(n)) %>% |
|  | head(10) |
|  |  |
|  | video\_sales$Critic\_Score <- as.numeric(video\_sales$Critic\_Score) |
|  |  |
|  | # Correlation plot between User Score and Critic Score |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | ggplot(aes(x=User\_Score,y=Critic\_Score)) + |
|  | geom\_point() + |
|  | geom\_smooth(method="loess") + |
|  | theme\_bw() + |
|  | labs(x="User Score", y ="Critic Score", |
|  | title="Correlation between User and Critic Score") |
|  |  |
|  | # Influence of scoring difference between User and Critic Score on Global Sales |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | mutate(diff=(User\_Score\*10-Critic\_Score)) %>% |
|  | group\_by(diff) %>% |
|  | summarise(sums=sum(Global\_Sales)) %>% |
|  | ggplot(aes(x=diff,y=sums)) + |
|  | geom\_point(color="#006CD1") + |
|  | theme\_bw() + |
|  | labs(x="Difference between User and Critic Score", |
|  | y ="Global Sales [M of units]", |
|  | title="Influence of scoring difference between Users and Critics on Global Sales") |
|  |  |
|  | # User and Critic Score impact on sales |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | mutate(User\_Score=User\_Score\*10) %>% |
|  | gather(Type, Score, User\_Score, Critic\_Score) %>% |
|  | group\_by(Type,Score) %>% |
|  | summarise(totalsales=sum(Global\_Sales)) %>% |
|  | ggplot(aes(x=Score,y=totalsales, fill=Type)) + |
|  | geom\_area(position="identity") + |
|  | facet\_grid(Type~.) + |
|  | theme\_bw() + |
|  | scale\_fill\_brewer(palette="Dark2", label=c("Critic Score", "User Score\*10")) + |
|  | labs(x="Score", y ="Global Sales [M of units]", |
|  | title="Global Sales vs. Critic and User Score") |
|  |  |
|  | # Count analysis of users and critics, and their influence on Global Sales |
|  | video\_sales %>% |
|  | na.omit() %>% |
|  | gather(Type, Count, User\_Count, Critic\_Count) %>% |
|  | group\_by(Type,Count) %>% |
|  | summarise(totalsales=sum(Global\_Sales)) %>% |
|  | ggplot(aes(x=Count,y=totalsales,fill=Type)) + |
|  | geom\_area(position="identity") + |
|  | facet\_grid(~Type, scales="free") + |
|  | theme\_bw() + |
|  | labs(x="Count", y ="Global Sales [M of units]", |
|  | title="Global Sales vs. Critic and User score Count") + |
|  | scale\_fill\_brewer(palette="Dark2", label=c("Critic Count", "User Count")) |
|  |  |
|  | # User and Critic Score (as average) and Count across years |
|  | plot1 <- video\_sales %>% |
|  | na.omit() %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(totalusers=mean(User\_Score\*10/User\_Count), |
|  | totalcritics=mean(Critic\_Score/Critic\_Count)) %>% |
|  | gather(Type, Count, totalusers, totalcritics) %>% |
|  | ggplot() + |
|  | geom\_line(aes(x=Year\_of\_Release, y=Count, color=Type), size=1) + |
|  | theme\_bw() + |
|  | scale\_color\_brewer(palette = "Dark2", label=c("Critics", "Users")) + |
|  | labs(x="Year of Release", y ="Sum of avg Scores (normalized) ", |
|  | title="Distribution of Critic and User Scores (avg per title) across years") |
|  | plot2 <- video\_sales %>% |
|  | na.omit() %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarise(totalusers=sum(User\_Count), |
|  | totalcritics=sum(Critic\_Count)) %>% |
|  | gather(Type, Count, totalusers, totalcritics) %>% |
|  | ggplot() + |
|  | geom\_line(aes(x=Year\_of\_Release, y=Count, color=Type), size=1) + |
|  | theme\_bw() + |
|  | scale\_color\_brewer(palette = "Dark2", label=c("Critics", "Users")) + |
|  | labs(x="Year of Release", y ="Sum of Counts", |
|  | title="Distribution of Critic and User score Counts across years") |
|  | ggarrange(plot1,plot2,ncol=1,nrow=2,common.legend=TRUE, legend="bottom", align="v") |
|  | rm(plot1,plot2) |
|  |  |
|  | # Are there any titles that has not been scored by Users or by Critics? |
|  | video\_sales %>% filter(User\_Count==0 | Critic\_Count==0) %>% |
|  | select(Name,Year\_of\_Release,Global\_Sales,User\_Score,User\_Count, |
|  | Critic\_Score,Critic\_Count) |
|  |  |
|  | # Imputing User and Critic Score and Count NAs with medians |
|  | video\_sales$User\_Score[is.na(video\_sales$User\_Score)] <- |
|  | median(video\_sales$User\_Score, na.rm = TRUE) |
|  | video\_sales$User\_Count[is.na(video\_sales$User\_Count)] <- |
|  | median(video\_sales$User\_Count, na.rm = TRUE) |
|  | video\_sales$Critic\_Score[is.na(video\_sales$Critic\_Score)] <- |
|  | median(video\_sales$Critic\_Score, na.rm = TRUE) |
|  | video\_sales$Critic\_Count[is.na(video\_sales$Critic\_Count)] <- |
|  | median(video\_sales$Critic\_Count, na.rm = TRUE) |
|  |  |
|  |  |
|  | # 3.5. Rating variable ---------------------------------------------------- |
|  |  |
|  | # Exploring unique values of Rating variable |
|  | video\_sales %>% group\_by(Rating) %>% summarise(n=n()) %>% arrange(desc(n)) |
|  |  |
|  | # Changing the K-A rating to E, and RP to NA value |
|  | video\_sales$Rating[video\_sales$Rating=="RP"] <- NA |
|  | video\_sales$Rating[video\_sales$Rating=="K-A"] <- "E" |
|  |  |
|  | # Chaining Rating variable to factor |
|  | video\_sales$Rating <- as.factor(video\_sales$Rating) |
|  |  |
|  | # Analysis of game rating influence on global sales per one title |
|  | video\_sales %>% |
|  | filter(!is.na(Rating)) %>% |
|  | group\_by(Rating) %>% |
|  | summarise(sales=sum(Global\_Sales)/n(),n=n()) %>% |
|  | ggplot(aes(x=reorder(Rating,sales), y=sales)) + |
|  | geom\_segment( aes(x=reorder(Rating,sales), xend=reorder(Rating,sales), |
|  | y=0, yend=sales), color="black") + |
|  | geom\_point( color="blue", size=4) + |
|  | geom\_text(aes(label=paste("#games:",n)),hjust=0.9, vjust=1.75) + |
|  | coord\_flip() + |
|  | theme\_bw() + |
|  | labs(x="Rating", y ="Global Sales per title [M units]", |
|  | title="Global Sales per one title per game rating") |
|  |  |
|  | # Excluding Rating NA values from data set (as they can't be imputed with median) |
|  | video\_sales <- video\_sales %>% filter(!is.na(Rating)) |
|  |  |
|  |  |
|  | # 3.6. Game title analysis ------------------------------------------------ |
|  |  |
|  |  |
|  | # 3.6.1. Creating a wordcloud --------------------------------------------- |
|  |  |
|  | # Creating text corpus from Name string |
|  | text <- Corpus(VectorSource(video\_sales$Name)) |
|  |  |
|  | # Cleaning Name string |
|  | text <- text %>% |
|  | tm\_map(removeNumbers) %>% |
|  | tm\_map(removePunctuation) %>% |
|  | tm\_map(stripWhitespace) %>% |
|  | tm\_map(removeWords, stopwords("english")) |
|  |  |
|  | # Changing the format of the data |
|  | text <- as.matrix(TermDocumentMatrix(text)) |
|  |  |
|  | # Sum count of words in text, rearrange and change into data frame |
|  | words <- sort(rowSums(text),decreasing=TRUE) |
|  | text.df <- data.frame(word = names(words),freq=words) |
|  |  |
|  | # Inspecting new data frame |
|  | text.df %>% head(10) |
|  |  |
|  | # Excluding "the" as word |
|  | text.df <- text.df[!text.df$word=="the",] |
|  |  |
|  | # Creating wordcloud of cleaned Name string |
|  | wordcloud(text.df$word, text.df$freq, min.freq = 50, max.words=200, |
|  | random.order=FALSE, max.freq=400, rot.per=0.35, |
|  | colors=brewer.pal(12, "Paired")) |
|  |  |
|  |  |
|  | # 3.6.2. Checking correlation between words and Global Sales -------------- |
|  |  |
|  | # Creating a sum sales column to each word in text.df |
|  | c <- count(text.df)[[1]] |
|  | for (i in 1:c) { |
|  | text.df$sales[i] <- |
|  | video\_sales$Global\_Sales[str\_detect(video\_sales$Name, |
|  | fixed(text.df$word[i], |
|  | ignore\_case=TRUE))] %>% |
|  | sum() |
|  | } |
|  |  |
|  | # Global Sales vs frequency of the word plot |
|  | text.df %>% |
|  | ggplot(aes(x=freq, y=sales)) + |
|  | geom\_point() + |
|  | geom\_smooth(method=loess, se=TRUE) + |
|  | theme\_bw() + |
|  | labs(x="Frequency of the word used in game title", |
|  | y ="Global Sales [M units]", |
|  | title="Sales per frequency of words used in the game title") |
|  |  |
|  | # Removing Name variable from video\_sales data set |
|  | video\_sales <- video\_sales %>% select(-Name) |
|  |  |
|  | # Cleaning the environment |
|  | rm(text.df, text, c, i, words) |
|  |  |
|  |  |
|  | # 3.7. Final check of the data set ---------------------------------------- |
|  |  |
|  | # Checking out medians and outliers |
|  | summary(video\_sales) |
|  |  |
|  | # Checking if there are any NAs left |
|  | is.na(video\_sales) %>% sum() |
|  |  |
|  | # Checking out the status |
|  | status(video\_sales) |
|  |  |
|  |  |
|  |  |
|  | # 4. Modeling ------------------------------------------------------------- |
|  |  |
|  |  |
|  | # 4.1. Correlation between variables and PCA ------------------------------ |
|  |  |
|  | # Correlation plot of variables in video\_sales data set |
|  | corrgram(video\_sales[,sapply(video\_sales, is.numeric)], |
|  | order=TRUE, lower.panel=panel.shade, |
|  | upper.panel=panel.cor, text.panel=panel.txt, |
|  | main="Correlogram of variables selected for modeling") |
|  |  |
|  |  |
|  | # Creating a dataframe with only numerical values |
|  | # from \*video\_sales\* to use in PCA analysis |
|  | pca\_data <- video\_sales %>% select(-Platform,-Genre,-Rating) |
|  |  |
|  | # PCA calculations |
|  | pr.vsales <- prcomp(pca\_data, scale=TRUE) |
|  | summary(pr.vsales) |
|  |  |
|  | # Variance and cumulative variance plots |
|  | pr.var <- pr.vsales$sdev^2 |
|  | pve <- pr.var/sum(pr.var) |
|  | par(mfrow=c(2,1)) |
|  | plot(pve, type="b", lwd=2, col="#FD8D3C", |
|  | main="Variance of PCA", |
|  | xlab="Number of Principal Components", |
|  | ylab="% of Variance") |
|  | plot(cumsum(pve), type="b", lwd=2, col="#41B6C4", |
|  | main="Cumulative variance of PCA", |
|  | xlab="Number of Principal Components", |
|  | ylab="% of Cumulative Variance") |
|  | abline(h=0.9, lty=2) |
|  |  |
|  | # Cleaning the environment |
|  | rm(pca\_data,pr.vsales,pr.var,pve) |
|  |  |
|  |  |
|  | # 4.2. Global Sales prediction models and their results ------------------- |
|  |  |
|  |  |
|  | # 4.2.1. Creating a training and testing sub-sets ------------------------- |
|  |  |
|  | # Removed the regional sales variables, as they make Global Sales 1, |
|  | # and we don't want to use them as predictors |
|  | globalsales <- video\_sales %>% |
|  | select(-NA\_Sales,-EU\_Sales,-Other\_Sales, -JP\_Sales) |
|  |  |
|  | # Setting seed for reproducibility |
|  | set.seed(1) |
|  |  |
|  | # Creating train and test sets following 80/20 Pareto Principle |
|  | global.test\_index <- createDataPartition(y = globalsales$Global\_Sales, |
|  | times = 1, p = 0.2, list = FALSE) |
|  | global.test <- globalsales %>% slice(global.test\_index) |
|  | global.train <- globalsales %>% slice(-global.test\_index) |
|  |  |
|  | # Making sure that both test and train have same set of data |
|  | global.train <- global.train %>% |
|  | semi\_join(global.test, by = "Platform") %>% |
|  | semi\_join(global.test, by="Genre") %>% |
|  | semi\_join(global.test, by="Rating") |
|  |  |
|  |  |
|  | # 4.2.2. Models for predicting Global Sales ------------------------------- |
|  |  |
|  | # Linear regression model (10-fold cv, pre-processed) |
|  | global.lm.fit <- train(Global\_Sales~.,data=global.train, method="lm", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.lm.pred <- predict(global.lm.fit,global.test) |
|  |  |
|  | # Cubist (10-fold cv, pre-processed), with tuned committees and neighbors |
|  | global.cubist.fit <- train(Global\_Sales~.,data=global.train, method="cubist", |
|  | trControl=trainControl(method="cv",number=10), |
|  | tuneGrid=data.frame(committees=seq(1,9,1), |
|  | neighbors=seq(1,9,1)), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.cubist.pred <- predict(global.cubist.fit,global.test) |
|  |  |
|  | plot(global.cubist.fit,highlight = TRUE) |
|  | global.cubist.fit$bestTune |
|  |  |
|  | # Regression trees (10-fold cv, pre-processed), with tuned cp |
|  | global.rpart.fit <- train(Global\_Sales~.,data=global.train, method="rpart", |
|  | tuneGrid=data.frame(cp=seq(0,0.05,0.005)), |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.rpart.pred <- predict(global.rpart.fit,global.test) |
|  |  |
|  | ggplot(global.rpart.fit, highlight=TRUE) |
|  | global.rpart.fit$bestTune |
|  |  |
|  | # Random forest (10-fold cv, pre-processed), with 100 trees and tuned mtry |
|  | global.rf.fit <- train(Global\_Sales~., data = global.train, method="rf", |
|  | trControl=trainControl(method="cv", number=10), |
|  | tuneGrid=data.frame(mtry=seq(1,25,1)), |
|  | ntree=100, preProcess = c("center", "scale")) |
|  |  |
|  | global.rf.pred <- predict(global.rf.fit,global.test) |
|  |  |
|  | global.rf.fit$bestTune |
|  | ggplot(global.rf.fit, highlight=TRUE) |
|  |  |
|  | # Non-negative least squares NNLS (10-fold cv, pre-processed) |
|  | global.nnls.fit <- train(Global\_Sales~.,data=global.train, method="nnls", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | global.nnls.pred <- predict(global.nnls.fit,global.test) |
|  |  |
|  |  |
|  | # 4.2.3. Predicting Global Sales results ---------------------------------- |
|  |  |
|  | # Creating a list of fitted models, and plotting RMSE, MAE and R-squared results |
|  | models <- list(lm=global.lm.fit,cubist=global.cubist.fit,rpart=global.rpart.fit, |
|  | rf=global.rf.fit,nnls=global.nnls.fit) |
|  |  |
|  | bwplot(resamples(models),metric=c("RMSE","Rsquared","MAE"), |
|  | main="Global Sales learning models comparison") |
|  |  |
|  | # Creating a data frame out of results for fitted models |
|  | mae <- list(median(global.lm.fit$resample$MAE), |
|  | median(global.cubist.fit$resample$MAE), |
|  | median(global.rpart.fit$resample$MAE), |
|  | median(global.rf.fit$resample$MAE), |
|  | median(global.nnls.fit$resample$MAE)) |
|  | rmse <- list(median(global.lm.fit$resample$RMSE), |
|  | median(global.cubist.fit$resample$RMSE), |
|  | median(global.rpart.fit$resample$RMSE), |
|  | median(global.rf.fit$resample$RMSE), |
|  | median(global.nnls.fit$resample$RMSE)) |
|  | rsqr <- list(median(global.lm.fit$resample$Rsquared), |
|  | median(global.cubist.fit$resample$Rsquared), |
|  | median(global.rpart.fit$resample$Rsquared), |
|  | median(global.rf.fit$resample$Rsquared), |
|  | median(global.nnls.fit$resample$Rsquared)) |
|  |  |
|  | global.results <- data.frame(Model=c("LM","Cubist","Rpart","RandomForest","NNLS"), |
|  | Median\_MAE=unlist(mae), Median\_RMSE=unlist(rmse), |
|  | Median\_Rsquared=unlist(rsqr)) |
|  | global.results |
|  |  |
|  | # Actual vs Predicted output plots for all models |
|  | plot.lm <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.lm.pred) %>% mutate(Type="LM") |
|  | plot.cubist <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.cubist.pred) %>% mutate(Type="Cubist") |
|  | plot.rpart <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.rpart.pred) %>% mutate(Type="Rpart") |
|  | plot.rf <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.rf.pred) %>% mutate(Type="RandomForest") |
|  | plot.nnls <- data.frame(Actuals=global.test$Global\_Sales, |
|  | Predicted=global.nnls.pred) %>% mutate(Type="NNLS") |
|  |  |
|  | final.plot <- rbind(plot.lm,plot.cubist,plot.rpart,plot.rf,plot.nnls) |
|  |  |
|  | ggplot(final.plot, aes(x=Predicted, y=Actuals)) + |
|  | geom\_point() + |
|  | geom\_smooth(method="loess", se=TRUE) + |
|  | facet\_wrap(.~Type) + |
|  | theme\_bw() |
|  |  |
|  | # Cleaning the environment |
|  | rm(final.plot,mae,models,rmse,rsqr, plot.cubist,plot.lm,plot.nnls,plot.rf,plot.rpart) |
|  |  |
|  |  |
|  | # 4.3. NA Sales prediction models and their results ----------------------- |
|  |  |
|  |  |
|  | # 4.3.1. Creating a training and testing sub-sets ------------------------- |
|  |  |
|  | # Creating a new data frame for modeling of NA\_Sales output. |
|  | # Global Sales removed as the variable that we should not know |
|  | # (all sales shares are equal to 1) |
|  | NAsales <- video\_sales %>% select(-Global\_Sales) |
|  |  |
|  | # Setting seed for reproducibility |
|  | set.seed(1) |
|  |  |
|  | # Creating train and test sets following 80/20 Pareto Principle |
|  | na.test\_index <- createDataPartition(y = NAsales$NA\_Sales, |
|  | times = 1, p = 0.2, list = FALSE) |
|  | na.test <- NAsales%>% slice(na.test\_index) |
|  | na.train <- NAsales %>% slice(-na.test\_index) |
|  |  |
|  | # Making sure that both test and train have same set of data |
|  | na.train <- na.train %>% |
|  | semi\_join(na.test, by="Rating") %>% |
|  | semi\_join(na.test, by="Platform") %>% |
|  | semi\_join(na.test, by="Genre") |
|  |  |
|  |  |
|  | # 4.3.2. Models for predicting NA Sales ----------------------------------- |
|  |  |
|  | # Linear regression model (10-fold cv, pre-processed) |
|  | na.lm.fit <- train(NA\_Sales~.,data=na.train, method="lm", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.lm.pred <- predict(na.lm.fit,na.test) |
|  |  |
|  | # Cubist (10-fold cv, pre-processed), with tuned committees and neighbors parameter |
|  | na.cubist.fit <- train(NA\_Sales~.,data=na.train, method="cubist", |
|  | trControl=trainControl(method="cv",number=10), |
|  | tuneGrid=data.frame(committees=seq(1,9,1), |
|  | neighbors=seq(1,9,1)), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.cubist.pred <- predict(na.cubist.fit,na.test) |
|  |  |
|  | plot(na.cubist.fit,highlight = TRUE) |
|  | na.cubist.fit$bestTune |
|  |  |
|  | # Rpart (10-fold cv, pre-processed), with tuned cp parameter |
|  | na.rpart.fit <- train(NA\_Sales~.,data=na.train, method="rpart", |
|  | tuneGrid=data.frame(cp=seq(0,0.05,0.005)), |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.rpart.pred <- predict(na.rpart.fit,na.test) |
|  |  |
|  | ggplot(na.rpart.fit,highlight = TRUE) |
|  | na.rpart.fit$bestTune |
|  |  |
|  | # Random Forest (10-fold cv, pre-processed), with 100 trees and tunable mtry parameter |
|  | na.rf.fit <- train(NA\_Sales~.,data=na.train, method="rf", |
|  | tuneGrid=data.frame(mtry=seq(1,25,1)), ntree=100, |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.rf.pred <- predict(na.rf.fit,na.test) |
|  |  |
|  | ggplot(na.rf.fit,highlight = TRUE) |
|  | na.rf.fit$bestTune |
|  |  |
|  | # Non-negative least squares NNLS (10-fold cv, pre-processed) |
|  | na.nnls.fit <- train(NA\_Sales~.,data=na.train, method="nnls", |
|  | trControl=trainControl(method="cv",number=10), |
|  | preProcess = c("center", "scale")) |
|  |  |
|  | na.nnls.pred <- predict(na.nnls.fit,na.test) |
|  |  |
|  |  |
|  | # 4.3.3. Predicting NA Sales results -------------------------------------- |
|  |  |
|  | # Creating a list of fitted models, and plotting RMSE, MAE and R-squared results |
|  | models <- list(lm=na.lm.fit, cubist=na.cubist.fit, rpart=na.rpart.fit, |
|  | randomforest=na.rf.fit, nnls=na.nnls.fit) |
|  |  |
|  | bwplot(resamples(models),metric=c("RMSE","Rsquared","MAE"), |
|  | main="NA Sales learning models comparison") |
|  |  |
|  | # Creating a data frame out of results for fitted models |
|  | mae <- list(median(na.lm.fit$resample$MAE), |
|  | median(na.cubist.fit$resample$MAE), |
|  | median(na.rpart.fit$resample$MAE), |
|  | median(na.rf.fit$resample$MAE), |
|  | median(na.nnls.fit$resample$MAE)) |
|  | rmse <- list(median(na.lm.fit$resample$RMSE), |
|  | median(na.cubist.fit$resample$RMSE), |
|  | median(na.rpart.fit$resample$RMSE), |
|  | median(na.rf.fit$resample$RMSE), |
|  | median(na.nnls.fit$resample$RMSE)) |
|  | rsqr <- list(median(na.lm.fit$resample$Rsquared), |
|  | median(na.cubist.fit$resample$Rsquared), |
|  | median(na.rpart.fit$resample$Rsquared), |
|  | median(na.rf.fit$resample$Rsquared), |
|  | median(na.nnls.fit$resample$Rsquared)) |
|  |  |
|  | na.results <- data.frame(Model=c("LM","Cubist","Rpart","RandomForest","NNLS"), |
|  | Median\_MAE=unlist(mae), Median\_RMSE=unlist(rmse), |
|  | Median\_Rsquared=unlist(rsqr)) |
|  | na.results |
|  |  |
|  | # Comparison of models results - Actual vs Predicted plots |
|  | plot.lm <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.lm.pred) %>% mutate(Type="LM") |
|  | plot.cubist <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.cubist.pred) %>% mutate(Type="Cubist") |
|  | plot.rpart <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.rpart.pred) %>% mutate(Type="Rpart") |
|  | plot.rf <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.rf.pred) %>% mutate(Type="RandomForest") |
|  | plot.nnls <- data.frame(Actuals=na.test$NA\_Sales, |
|  | Predicted=na.nnls.pred) %>% mutate(Type="NNLS") |
|  |  |
|  | final.plot <- rbind(plot.lm,plot.cubist,plot.rpart,plot.rf,plot.nnls) |
|  |  |
|  | ggplot(final.plot, aes(x=Predicted, y=Actuals)) + |
|  | geom\_point() + |
|  | geom\_smooth(method="loess", se=TRUE) + |
|  | facet\_wrap(.~Type) + theme\_bw() |
|  |  |
|  | # Variable importance table (top 10), arranged side-by-side |
|  | na.lm.imp <- varImp(na.lm.fit) |
|  | na.cubist.imp <- varImp(na.cubist.fit) |
|  | na.rpart.imp <- varImp(na.rpart.fit) |
|  | na.rf.imp <- varImp(na.rf.fit) |
|  | na.nnls.imp <- varImp(na.nnls.fit) |
|  |  |
|  |  |
|  | imp.lm <- data.frame(LM.Variable=rownames(arrange(na.lm.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.lm.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.cubist <- data.frame(Cubist.Variable=rownames(arrange(na.cubist.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.cubist.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.rpart <- data.frame(Rpart.Variable=rownames(arrange(na.rpart.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.rpart.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.rf <- data.frame(RF.Variable=rownames(arrange(na.rf.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.rf.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  | imp.nnls <- data.frame(NNLS.Variable=rownames(arrange(na.nnls.imp$importance, |
|  | desc(Overall)))[1:10], |
|  | Overall=arrange(na.nnls.imp$importance, |
|  | desc(Overall))[1:10,]) |
|  |  |
|  | grid.arrange(tableGrob(imp.cubist),tableGrob(imp.rf),nrow=1) |
|  | grid.arrange(tableGrob(imp.lm),tableGrob(imp.rpart),tableGrob(imp.nnls),nrow=1) |
|  |  |
|  | # Cleaning the environment |
|  | rm(final.plot,mae,models,rmse,rsqr, plot.cubist,plot.lm,plot.nnls,plot.rf,plot.rpart) |
|  |  |
|  |  |
|  | ``` |
|  |  |

3. https://github.com/mjvatt/Machine-Learning/blob/360850ed1ecf803bd47c450c66804a8fb4216d1c/Harvard%20Capstone/Vatt\_M\_JAN2020\_Video%20Games%20Sales%20RMarkdown.Rmd

|  |
| --- |
|  |
| --- |
|  | title: "VGChartz Video Games Sales Dataset \n HarvardX: Capstone Project - PH125.9x" |
|  | author: "Michael Vatt" |
|  | date: "09 JAN 2020" |
|  | output: |
|  | pdf\_document: |
|  | toc: yes |
|  | toc\_depth: 2 |
|  | --- |
|  |  |
|  | ```{r setup, include=FALSE} |
|  |  |
|  | knitr::opts\_chunk$set(echo = TRUE) |
|  |  |
|  | ``` |
|  |  |
|  | **# Overview** |
|  |  |
|  | This project stemmed from the HarvardX Data Science Capstone Course PH125.9x. This time, the students were tasked with choosing their own dataset to wrangle, explore, and develop for their Machine Learning Algorithm. Due to my love of video games, I chose the Video Games Sales dataset from Kaggle. This project is outlined as follows: (1) Necessary packages are installed, (2) Dataset is loaded and setup configured, (3) Data Wrangling and familiarization with the dataset, (4) Perform exploratory data analysis, (5) Methods and analysis, (6) Modeling development and training, (7) Results of models and using best model on the validation set, and (8) the Conclusion. Exploratory analysis cannot be underestimated as it is critical to develop a machine learning algorithm that is both informative and meaningful in predicting anything. The final model is then explained and hopefully successful. |
|  |  |
|  | - Let's Install all Necessary Packages: |
|  | \ |
|  |  |
|  | ```{r, echo = TRUE, message = FALSE, warning = FALSE, eval = TRUE} |
|  |  |
|  | ############################################################# |
|  | # Create Test Set, Validation Set, and Final Prediction |
|  | ############################################################# |
|  |  |
|  | # Note: This Process Could Take a Couple of Minutes for Loading Required Packages |
|  |  |
|  | 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(doParallel)) install.packages("doParallel", repos = "http://cran.us.r-project.org") |
|  | if(!require(dplyr)) install.packages("caret", repos = "http://cran.us.r-project.org") |
|  | if(!require(ggplot2)) install.packages("caret", repos = "http://cran.us.r-project.org") |
|  | if(!require(ggpubr)) install.packages("caret", repos = "http://cran.us.r-project.org") |
|  | if(!require(gridExtra)) install.packages("caret", repos = "http://cran.us.r-project.org") |
|  | if(!require(knitr)) install.packages("caret", repos = "http://cran.us.r-project.org") |
|  | if(!require(scales)) install.packages("tidyverse", repos = "http://cran.us.r-project.org") |
|  | if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org") |
|  |  |
|  | ``` |
|  |  |
|  |  |
|  | **## Introduction** |
|  |  |
|  | This machine learning algorithm is dependent upon Video Games Sales from the VGChartz Dataset and corresponding ratings from Metacritic. This dataset was motivated by Gregory Smith's web scrape of VGChartz Video Game Sales. This dataset adds variables with another web scrape of Metacritic. Some observations in the dataset are incomplete, thus some data wrangling is necessary. This dataset includes the following variables: |
|  | \ |
|  |  |
|  | ``` {r dataset variables, echo = FALSE} |
|  |  |
|  | variables <- c("Name", "Platform", "Year\_of\_Release", "Genre", "Publisher", "NA\_Sales", "EU\_Sales", "JP\_Sales", "Other\_Sales", "Global\_Sales", "Critic\_Score", "Critic\_Count", "User\_Score", "User\_Count", "Developer") |
|  | variables %>% kable(col.names = "Dataset Variables") |
|  |  |
|  | ``` |
|  |  |
|  | **## Objective** |
|  |  |
|  | Our desire is to develop an ability to predict Global Sales with accuracy. The term used to give evaluation value to the algorithm is the Root Mean Square Error - also known as RMSE. This measure is a popular method for statistics, estimation, mathematics, and applications. It is used to measure the differences between values predicted by a model and the values observed. |
|  |  |
|  | RMSE is a measure of accuracy - without it, the algorithm is impractical and ineffective. The lower an RMSE score, the better! The goal of this project is to develop a model that will produce an RMSE below 0.865 - a lofty goal. |
|  |  |
|  | The effect errors have on the RMSE is proportional to the size of the squared error - larger the error, larger the effect. A means to help navigate this problem is built into the formula by reducing the impact of the errors through the square root function. |
|  |  |
|  | This project's machine learning algorithm was established through an incremental and iterative development process. Different models were established in layers to compare RMSE results in order to evaluate their prediction qualities. This will enable the best model to be determined and used to predict movie ratings on the dataset. |
|  |  |
|  | **## Dataset ##** |
|  |  |
|  | - WARNING: Some Models in this Project Require a Lot of Computing Power. Splitting the Cores Will Aid in Parallel Processing of CPUs |
|  |  |
|  | - NOTE: The Support Vector Machine and Random Forest Models Will Take the Longest Time |
|  | \ |
|  |  |
|  | ``` {r split cores} |
|  |  |
|  | split <- detectCores() |
|  | split # To Make Sure it Worked |
|  | cl <- makePSOCKcluster(split) |
|  | registerDoParallel(cl) |
|  |  |
|  | ``` |
|  |  |
|  | Let’s Load the Dataset from a CSV File Located in my GitHub Account |
|  | \ |
|  |  |
|  | ``` {r load the dataset} |
|  |  |
|  | file <- tempfile() |
|  | download.file("https://raw.githubusercontent.com/mjvatt/Machine-Learning/main/Harvard%20Capstone/Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv", file) |
|  | vgs <- fread("Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv", header = TRUE) |
|  |  |
|  | ``` |
|  |  |
|  | Familiarization with the Dataset: |
|  | \ |
|  |  |
|  | ``` {r data familiarization} |
|  |  |
|  | dim(vgs) # For Familiarity |
|  | str(vgs) # Notice that Some Columns Are Not The Same Class |
|  | class(vgs) # Let's Ensure it is Correct Format |
|  |  |
|  | ``` |
|  |  |
|  | Now that we are familiar with the dataset, we need to do some data wrangling to clean and prepare the data for ease of use. |
|  | \ |
|  |  |
|  | ``` {r data wrangling} |
|  |  |
|  | vgs$User\_Count <- as.numeric(as.character(vgs$User\_Count)) |
|  | vgs$User\_Score <- as.numeric(as.character(vgs$User\_Score)) |
|  |  |
|  | vgs2 <- na.omit(vgs) |
|  | dim(vgs2) # To See How it Has Changed |
|  |  |
|  | ``` |
|  |  |
|  | I noticed a variable that did not exist in the Dataset that I think would be fun to have in there - so let's create it. |
|  | \ |
|  |  |
|  | ``` {r create companies} |
|  |  |
|  | # First, Let's Make Some Company Variables |
|  |  |
|  | microsoft <- c('X360', 'XB', 'XOne') |
|  | nintendo <- c('Wii', 'NES', 'GB', 'DS', 'SNES', 'WiiU', '3DS', 'GBA', 'GC', 'N64') |
|  | sony <- c('PS', 'PS2', 'PS3', 'PS4', 'PSP', 'PSV') |
|  |  |
|  | # Here We Define the Criteria to Identify What Belongs to Each Company |
|  |  |
|  | companies <- function(c) { |
|  | if(c %in% microsoft) {return('Microsoft')} |
|  | else if(c %in% nintendo) {return('Nintendo')} |
|  | else if(c %in% sony) {return('Sony')} |
|  | } |
|  |  |
|  | # Now We Can Create This New Column |
|  |  |
|  | vgs2$companies <- sapply(vgs2$Platform, companies) |
|  |  |
|  | # Let's Check it Out to Make Sure it Worked |
|  |  |
|  | vgs2$companies[9:15] %>% kable(col.names = "Company") |
|  |  |
|  | ``` |
|  |  |
|  | **# Methods and Analysis** |
|  |  |
|  | **## Exploratory Data Analysis** |
|  |  |
|  | **### Distribution of Global Sales Across Genres and Ratings** |
|  |  |
|  | First, let's look at the Distribution of Global Sales Across Genres in the Data. We will also look at Video Games Sales based on their Rating (NOTE: Rating means "Maturity Rating Level" and has no connection or implication on User or Critic Scores given to the video games.) |
|  | \ |
|  |  |
|  | ``` {r distribution of global sales and ratings} |
|  |  |
|  | vgs2 %>% ggplot(aes(log(Global\_Sales))) + stat\_density(aes(color = Genre), geom = "line") + |
|  | labs(x = "Log Transformation of Global Sales", y = "Density") + ggtitle("Global Sales Across Genre") + |
|  | theme\_classic() |
|  |  |
|  | vgs2 %>% select(Rating) %>% filter(!Rating == '') %>% count(Rating) %>% |
|  | ggplot(aes(Rating, n, fill = Rating, color = Rating)) + |
|  | geom\_bar(stat = "identity") + ylab("Count of Games Per Rating") + |
|  | ggtitle("Sales by Video Game Age Rating") + |
|  | theme\_classic() # Notice we Have Three Ratings with No Data (We'll Remove Below) |
|  |  |
|  | ``` |
|  |  |
|  | I'd say this chart showed about what would be expected as certain types of games have more mass appeal, across genders, races and languages. Whilst others - for example, sports - while popular in the US, may not have such an appeal for the global masses. The Ratings chart was also not a surprise - but notice that we have three ratings with no data, we'll remove those types as they are not informative. |
|  |  |
|  | **### Relationship Between Critic Score/Critic Count and User Score/User Count** |
|  |  |
|  | Now, let's take a look at a possible relationship between Critic's data and User's data: |
|  | \ |
|  |  |
|  | ``` {r relationship between critics/users} |
|  |  |
|  | plot3 <- vgs2 %>% ggplot(aes(Critic\_Count, Critic\_Score)) + stat\_binhex() + |
|  | scale\_fill\_gradientn(colors = c("black", "orange"), name = "Count") + |
|  | labs(x = "Critic Count", y = "Critic Score") + ggtitle("Critic Score per Critic Count") + |
|  | theme\_classic() |
|  |  |
|  | plot4 <- vgs2 %>% ggplot(aes(User\_Count, User\_Score)) + stat\_binhex() + |
|  | scale\_fill\_gradientn(colors = c("black", "orange"), name = "Count") + |
|  | labs(x = "User Count", y = "User Score") + ggtitle("User Score per User Count") + |
|  | theme\_classic() |
|  |  |
|  | # For Easy Visual Comparison |
|  |  |
|  | grid.arrange(plot3, plot4, nrow=1, ncol=2) |
|  |  |
|  | ``` |
|  |  |
|  | **### Correlation Between Critic and User Scores** |
|  |  |
|  | Is there a correlation between Critic and User Scores? |
|  | \ |
|  |  |
|  | ``` {r correlation of critic and user scores?} |
|  |  |
|  | vgs2 %>% ggplot(aes(Critic\_Score, User\_Score)) + stat\_binhex() + |
|  | scale\_fill\_gradientn(colors = c("black", "orange"), name = "Count") + |
|  | ggtitle("Correlation Between Critic and User Scores") + theme\_classic() |
|  |  |
|  | ``` |
|  |  |
|  | It appears that there is some sort of correlation. But the User Scores appear to be more "favorable" or "positive" than critics. Let's actually see what that percentage is for scores: |
|  | \ |
|  |  |
|  | ``` {r correlation between critics/users} |
|  |  |
|  | cor(vgs2$User\_Score, vgs2$Critic\_Score, use = "complete.obs") %>% kable() |
|  |  |
|  | ``` |
|  |  |
|  | **## Scores Comparison** |
|  |  |
|  | Now for more detailed scores comparisons: |
|  | \ |
|  |  |
|  | ``` {r scores per genre comparison} |
|  |  |
|  | plot9 <- vgs2 %>% select(Genre, User\_Score) %>% ggplot(aes(Genre, User\_Score, color = Genre)) + |
|  | geom\_jitter(alpha = .3) + labs(x = "Video Game Genre", y = "User Score") + ggtitle("User Score Per Genre") + |
|  | theme\_classic() + theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | plot10 <- vgs2 %>% select(Genre, Critic\_Score) %>% ggplot(aes(Genre, Critic\_Score, color = Genre)) + |
|  | geom\_jitter(alpha = .3) + labs(x = "Video Game Genre", y = "Critic Score") + |
|  | ggtitle("Critic Score Per Genre") + theme\_classic() + theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | # For Easy Visual Comparison |
|  |  |
|  | grid.arrange(plot9, plot10, nrow = 1, ncol = 2) |
|  |  |
|  | ``` |
|  |  |
|  | This definitely seems to support the idea that User Scores are generally more favorable or positive towards the games - irrespective of genre - than Critic Scores. |
|  |  |
|  | **## Critic vs. User Scores Scaled** |
|  |  |
|  | Did you notice that the User Scores and Critic Scores were measured differently? Let's correct that and see if it changes anything. |
|  | \ |
|  |  |
|  | ``` {r critic vs user scores scaled} |
|  |  |
|  | vgs3 <- vgs2 |
|  |  |
|  | # Let's Do Some More Comparison Between Critic and User Scores |
|  | # We Should Scale the User Score to Make it Easier to Compare to Critics (NOTE: Critics had a 0:100 scale; Users used a 0:10 scale) |
|  |  |
|  | vgs3$User\_Score\_Scaled <- as.numeric(as.character(vgs3$User\_Score)) \* 10 |
|  |  |
|  | # Now, Let's Compare the Critic and User Scores Once Again |
|  |  |
|  | vgs3 %>% ggplot(aes(Critic\_Score, User\_Score\_Scaled)) + geom\_point(aes(color = Platform)) + |
|  | geom\_smooth(method = "lm", size = 1) + labs(x = "Critic Score", y = "User Score") + |
|  | ggtitle("Comparison of Critic and User Scores on Equal Scales") + theme\_classic() |
|  |  |
|  | ``` |
|  |  |
|  | Mathematically, scaling the user scores would explain that, no, it will not change anything as it still represents the same figures despite the change in what the figures display as. |
|  |  |
|  | **## Let's Compare Critic and User Scores Through Ratings and Systems** |
|  |  |
|  | What about a comparison of Critic and User Scores due to maturity ratings and systems? |
|  | \ |
|  |  |
|  | ``` {r critic vs. user scores via ratings and systems} |
|  |  |
|  | vgs3 %>% filter(Rating %in% c("E", "E10+", "M", "T")) %>% |
|  | ggplot(aes(Critic\_Score, User\_Score\_Scaled)) + geom\_point(aes(color = Platform)) + |
|  | geom\_smooth(method = "lm", size = 1) + facet\_wrap(~Rating) + |
|  | labs(x = "Critic Score", y = "User Score") + |
|  | ggtitle("Comparison of Critic and User Scores on Equal Scales") + theme\_classic() |
|  |  |
|  | ``` |
|  |  |
|  | They all seem to follow a well-developed theme. |
|  |  |
|  | **### Global Sales Comparison with Regional Sales** |
|  |  |
|  | Let's take a peek at some regional sales: |
|  | \ |
|  |  |
|  | ``` {r sales comparison} |
|  |  |
|  | plot6 <- vgs2 %>% select(Global\_Sales, NA\_Sales, Genre) %>% |
|  | ggplot(aes(Global\_Sales, NA\_Sales, color = Genre)) + stat\_smooth(method = "lm") + |
|  | labs(x = "Global Sales", y = "NA Sales") + ggtitle("GS vs. NA Sales across Genres") + |
|  | theme\_classic() |
|  |  |
|  | plot7 <- vgs2 %>% select(Global\_Sales, JP\_Sales, Genre) %>% |
|  | ggplot(aes(Global\_Sales, JP\_Sales, color = Genre)) + stat\_smooth(method = "lm") + |
|  | labs(x = "Global Sales", y = "JP Sales") + ggtitle("GS vs. JP Sales across Genres") + |
|  | theme\_classic() |
|  |  |
|  | # For Easy Visual Comparison |
|  |  |
|  | grid.arrange(plot6, plot7, nrow=1, ncol=2) |
|  |  |
|  | ``` |
|  |  |
|  | That is interesting but it doesn't appear to be highly enlightening just yet. |
|  |  |
|  | **# Data Analysis #** |
|  |  |
|  | **### Now to Figure Out How Many Video Games Are Released Each Year** |
|  |  |
|  | We definitely need to see how the video games industry has behaved in terms of numbers of video games released by year: |
|  | \ |
|  |  |
|  | ``` {r video games released each year} |
|  |  |
|  | vgs\_per\_year <- vgs2 %>% select(Name, Genre, Year\_of\_Release, Rating) %>% |
|  | group\_by(Year\_of\_Release, Genre) %>% summarize(gamescount = n()) |
|  |  |
|  | vgs\_per\_year %>% ggplot(aes(x = Year\_of\_Release, y = gamescount, group = Genre, color = Genre)) + |
|  | stat\_sum(size = 1) + geom\_line() + labs(x = "Year Video Game Released", y = "Number of Games Released") + |
|  | ggtitle("Number of Games Released by Year") + theme\_classic() + |
|  | theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | Here we can see the temporal trend of the video game industry the past three decades. Started incredibly slow, a massive expansion, and then a downturn in the third decade. |
|  |  |
|  | **## What are the Annual Video Games Sales?** |
|  |  |
|  | Annual sales of video games is an important insight into the trend of the video gaming industry. Let's take a look: |
|  | \ |
|  |  |
|  | ``` {r annual sales} |
|  |  |
|  | vgs2 %>% select(Year\_of\_Release, Global\_Sales) %>% group\_by(Year\_of\_Release) %>% |
|  | summarize(GS = sum(Global\_Sales)) %>% |
|  | ggplot(aes(Year\_of\_Release, GS, fill = Year\_of\_Release, color = Year\_of\_Release)) + |
|  | geom\_area() + geom\_point() + |
|  | labs(x = "Year of Release", y = "Total Sales") + ggtitle("Total Sales by Year") + |
|  | theme\_classic() + theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | This seems to greatly support the temporal trend of video games released by year. |
|  |  |
|  | **## Top Video Game Companies** |
|  |  |
|  | Who are the top video game companies by Global Sales? |
|  | \ |
|  |  |
|  | ``` {r top video game companies} |
|  |  |
|  | vgs2 %>% select(Publisher, Global\_Sales) %>% group\_by(Publisher) %>% summarize(GS = sum(Global\_Sales)) %>% |
|  | arrange(desc(GS)) %>% head(10) %>% |
|  | ggplot(aes(Publisher, GS, fill = Publisher)) + geom\_bar(stat="identity") + |
|  | labs(x = "Year of Release", y = "Total Sales") + ggtitle("Global Sales by Company") + theme\_classic() + |
|  | theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | **## Top Video Game Systems** |
|  |  |
|  | What are the best-selling video game systems? |
|  | \ |
|  |  |
|  | ``` {r top video game systems} |
|  |  |
|  | systems <- vgs2 %>% select(Platform, Global\_Sales) %>% group\_by(Platform) %>% |
|  | summarize(GS = sum(Global\_Sales)) |
|  | systems$market\_share <- round(systems$GS / sum(systems$GS) \* 100) # Calculating Market Share of Each System |
|  | systems$market\_share <- sort(systems$market\_share) # Simply Sorting it for Ease and Clarity |
|  |  |
|  | systems %>% ggplot(aes(Platform, market\_share, fill = Platform)) + geom\_bar(stat = "identity") + |
|  | labs(x = "Video Game System", y = "Total Sales") + ggtitle("Total Sales by Video Game System") + |
|  | theme\_classic() + theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | **## Best-Selling Video Games** |
|  |  |
|  | What about the best-selling video games? |
|  | \ |
|  |  |
|  | ``` {r best selling video games} |
|  |  |
|  | vgs2 %>% select(Name, Global\_Sales) %>% group\_by(Name) %>% |
|  | summarize(GS = sum(Global\_Sales)) %>% arrange(desc(GS)) %>% |
|  | head(10) %>% kable(col.names = c("Video Game Name", "Global Sales")) |
|  |  |
|  | ``` |
|  |  |
|  | **## Top 10 Best-Rated Games by Users** |
|  |  |
|  | Now, let's view the differences (or similarities) between the top 10 best-rated games by Users and Critics. |
|  | \ |
|  |  |
|  | ``` {r best-rated games by users} |
|  |  |
|  | vgs2 %>% select(Name, User\_Score) %>% group\_by(Name) %>% summarize(meanus = mean(User\_Score)) %>% |
|  | arrange(desc(meanus)) %>% head(10) %>% kable(caption = "Top 10 Rated Games by Users", col.names = c("Video Game Name", "Avg User Score")) |
|  |  |
|  |  |
|  | ``` |
|  |  |
|  | **## Top 10 Best-Rated Games by Critics** |
|  |  |
|  | Seems to be very little in common for the Top 10 best-rated video games for the two groups. |
|  | \ |
|  |  |
|  | ``` {r best-rated games by critics} |
|  |  |
|  | vgs2 %>% select(Name, Critic\_Score) %>% group\_by(Name) %>% summarize(meancs = mean(Critic\_Score)) %>% |
|  | arrange(desc(meancs)) %>% head(10) %>% kable(caption = "Top 10 Rated Games by Critics", col.names = c("Video Game Name", "Avg Critic Score")) |
|  |  |
|  | ``` |
|  |  |
|  | **## Compounded Effect: Genre and System?** |
|  |  |
|  | What about the possibility of a compounded effect on sales of the variables Genre and System together? |
|  | \ |
|  |  |
|  | ``` {r compounded effect: genre and system} |
|  |  |
|  | vgs2 %>% select(Platform, Genre, Global\_Sales) %>% group\_by(Platform, Genre) %>% |
|  | summarize(GS = sum(Global\_Sales)) %>% ggplot(aes(Genre, Platform, fill = GS)) + |
|  | geom\_tile() + scale\_fill\_gradientn(colors = c("black", "orange")) + |
|  | labs(x = "Genre", y = "Video Game System") + ggtitle("Genre + System Compounded Effect?") + |
|  | theme\_classic() + theme(legend.position = "none", axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | Definitely seems like there's a correlation with hot spots in specific locations. |
|  |  |
|  | **## Popularity Due to Rating?** |
|  |  |
|  | Maybe it's simply the maturity rating that is driving popularity? |
|  | \ |
|  |  |
|  | ``` {r popularity due to rating?} |
|  |  |
|  | vgs2 %>% select(Year\_of\_Release, Global\_Sales, Rating) %>% |
|  | filter(Rating %in% c("E", "E10+", "M", "T")) %>% group\_by(Year\_of\_Release, Rating) %>% |
|  | summarize(GS = sum(Global\_Sales)) %>% arrange(desc(GS)) %>% |
|  | ggplot(aes(Year\_of\_Release, GS, group = Rating, fill = Rating)) + |
|  | geom\_bar(stat = "identity", position = "fill") + scale\_y\_continuous(labels = percent) + |
|  | labs(x = "Years of Release", y = "Percentage of Games Sold by Rating") + |
|  | ggtitle("Video Games Sold by Maturity Ratings for Certain Years") + theme\_classic() + |
|  | theme(legend.position = "none", axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | Nope, this chart would suggest that maturity rating has very little impact on popularity or sales. |
|  |  |
|  | **## Overall Sales by Regions** |
|  |  |
|  | What about sales driven by specific regions? We know North America is a hotbed for video games industry, but what about others? |
|  | \ |
|  |  |
|  | ``` {r overall sales by regions} |
|  |  |
|  | vgs2 %>% select(Year\_of\_Release, NA\_Sales, JP\_Sales, EU\_Sales, Other\_Sales) %>% |
|  | group\_by(Year\_of\_Release) %>% |
|  | summarize(NAS = sum(NA\_Sales), JP = sum(JP\_Sales), EU = sum(EU\_Sales), Other = sum(Other\_Sales)) %>% |
|  | ggplot(aes(Year\_of\_Release, NAS, group = 1, color = "NAS")) + |
|  | geom\_line() + geom\_line(aes(y = JP, color = "JP")) + geom\_line(aes(y = EU, color = "EU")) + |
|  | geom\_line(aes(y = Other, color = "Other")) + labs(x = "Year of Release", y = "Total Sales") + |
|  | ggtitle("Sales of Video Games by Region") + theme\_classic() + theme(axis.text.x = element\_text(angle = 90)) |
|  |  |
|  | ``` |
|  |  |
|  | This chart makes absolute sense. It is as predicted for North America; Japan is a major hub of technology and is where Nintendo was founded and continues to develop but is overall a smaller market in comparison; Europe is a larger market and actually has many game development and visual FX studios there; Lastly, the "Other Region" is comprised with the rest of the world and is not surprising. Despite areas like South Korea, these are incredibly small markets in comparison to the others. The strict rules of China, despite being a highly advanced technological society, does not indicate that it has a large video gaming following. |
|  |  |
|  | **## Predicting Sales as a Function of Critic Scores** |
|  |  |
|  | Since critic scores seem to be less favorable or negative towards video games than user scores, let's take a look at global sales as a function of critic scores. The scores go from negative to positive (left to right): |
|  | \ |
|  |  |
|  | ``` {r sales as a function of critic scores, message = FALSE, warning = FALSE} |
|  |  |
|  | # Let's Look at the Three Major Video Game Companies - Nintendo, Microsoft, and Sony |
|  |  |
|  | # Microsoft |
|  |  |
|  | microsoft\_plot <- vgs3 %>% filter(vgs3$companies == 'Microsoft') %>% |
|  | ggplot(aes(Critic\_Score, NA\_Sales)) + geom\_point(aes(color = Genre)) + |
|  | ylim(0, 5) + geom\_smooth() + labs(x = "Critic Score", y = "Total Sales") + |
|  | ggtitle("Microsoft") + theme\_classic() + theme(axis.text.x = element\_blank()) |
|  |  |
|  | # Nintendo |
|  |  |
|  | nintendo\_plot <- vgs3 %>% filter(vgs3$companies == 'Nintendo') %>% |
|  | ggplot(aes(Critic\_Score, Global\_Sales)) + geom\_point(aes(color = Genre)) + |
|  | ylim(0, 5) + geom\_smooth() + labs(x = "Critic Score", y = "Total Sales") + |
|  | ggtitle("Nintendo") + theme\_classic() + theme(axis.text.x = element\_blank()) |
|  |  |
|  | # Sony |
|  |  |
|  | sony\_plot <- vgs3 %>% filter(vgs3$companies == 'Sony') %>% ggplot(aes(Critic\_Score, NA\_Sales)) + |
|  | geom\_point(aes(color = Genre)) + ylim(0, 5) + geom\_smooth() + |
|  | labs(x = "Critic Score", y = "Total Sales") + ggtitle("Sony") + theme\_classic() + |
|  | theme(axis.text.x = element\_blank()) |
|  |  |
|  | # Let's Compare those 3 Visually Together |
|  |  |
|  | ggarrange(nintendo\_plot, microsoft\_plot, sony\_plot, ncol = 3) |
|  |  |
|  | ``` |
|  |  |
|  | As expected, with greater critic scores, a greater level of global sales. However, this is not a linear function of critic scores - it's exponential. This would imply that critic scores are highly important. |
|  |  |
|  | **# Modeling** |
|  |  |
|  | **## Model 1: Simple Prediction** |
|  |  |
|  | As labeled, this will be the simplest possible model to predict Global Sales from the entire population. |
|  | \ |
|  |  |
|  | ``` {r model 1: simple prediction} |
|  |  |
|  | set.seed(1) |
|  |  |
|  | RMSE <- function(true\_ratings, predicted\_ratings){ |
|  | sqrt(mean((true\_ratings - predicted\_ratings) ^ 2, na.rm = T)) |
|  | } |
|  |  |
|  |  |
|  | val\_ind <- createDataPartition(vgs3$Global\_Sales, times = 1, p = 0.3, list = FALSE) |
|  | train\_set <- vgs3[-val\_ind,] |
|  | val\_set <- vgs3[val\_ind,] |
|  |  |
|  | # Simple Prediction Mean |
|  |  |
|  | mu <- mean(train\_set$Global\_Sales) |
|  |  |
|  | # What was the Mu? |
|  |  |
|  | mu |
|  |  |
|  | # Simple Prediction |
|  |  |
|  | rmse\_model1 <- RMSE(train\_set$Global\_Sales, mu) |
|  | rmse\_preds <- data.frame(Method = "Model 1: Simple Prediction Model", RMSE = rmse\_model1) |
|  |  |
|  | # Check Results |
|  |  |
|  | rmse\_preds %>% kable() |
|  |  |
|  | ``` |
|  |  |
|  | **## Model 2: Generalized Linear Model** |
|  |  |
|  | This is a linear model to see if we can do any better. |
|  | \ |
|  |  |
|  | ```{r model2, results="hide", warning=FALSE, message=FALSE} |
|  | train\_glm <- train(Global\_Sales ~ Platform + User\_Score + Critic\_Score + Critic\_Count + Genre + Year\_of\_Release + Rating, method = "glm", data = train\_set, na.action = na.exclude) |
|  |  |
|  | rmse\_model2 <- getTrainPerf(train\_glm)$TrainRMSE |
|  | rmse\_preds <- add\_row(rmse\_preds, Method = "Model 2: Generalized Linear Model", RMSE = rmse\_model2) |
|  |  |
|  | # Check Results |
|  |  |
|  |  |
|  |  |
|  | rmse\_preds %>% kable() |
|  |  |
|  | ``` |
|  |  |
|  | **## Model 3: Knn** |
|  |  |
|  | Now, we move onto the K-Nearest Neighbors algorithm: it is a non-parametric method used to compare across all parameters and considerations. |
|  | \ |
|  |  |
|  | ``` {r model 3: knn} |
|  |  |
|  | train\_knn <- train(Global\_Sales ~ Platform + User\_Score + Critic\_Score + Critic\_Count + Genre + Year\_of\_Release + Rating, method = "knn", data = train\_set, na.action = na.exclude) |
|  |  |
|  | rmse\_model3 <- getTrainPerf(train\_knn)$TrainRMSE |
|  | rmse\_preds <- add\_row(rmse\_preds, Method = "Model 3: K-Nearest Neighbors Model", RMSE = rmse\_model3) |
|  |  |
|  | # Check Results |
|  |  |
|  | rmse\_preds %>% kable() |
|  |  |
|  | ``` |
|  |  |
|  | **## Model 4: Support Vector Machines** |
|  |  |
|  | Next is the Support Vector Machines algorithm: it is a discriminative classifier that maximizes the margin between two classes. |
|  | \ |
|  |  |
|  | ``` {r model 4: svm} |
|  |  |
|  | train\_svm <- train(Global\_Sales ~ Platform + User\_Score + Critic\_Score + Critic\_Count + Genre + Year\_of\_Release + Rating, method = "svmLinear", data = train\_set, na.action = na.exclude, scale = FALSE) |
|  |  |
|  | rmse\_model4 <- getTrainPerf(train\_svm)$TrainRMSE |
|  | rmse\_preds <- add\_row(rmse\_preds, Method = "Model 4: Support Vector Machines Model", RMSE = rmse\_model4) |
|  |  |
|  | # Check Results |
|  |  |
|  | rmse\_preds %>% kable() |
|  |  |
|  | ``` |
|  |  |
|  | **## Model 5: Random Forest** |
|  |  |
|  | Lastly for our training models, is the Random Forest algorithm: it is a method that operates from decision trees and outputs classification of the individual trees that also helps correct against overfitting training sets. |
|  | \ |
|  |  |
|  | ``` {r model 5: rf} |
|  |  |
|  | train\_rf <- train(Global\_Sales ~ Platform + User\_Score + Critic\_Score + Critic\_Count + Genre + Year\_of\_Release + Rating, method = "rf", data = train\_set, na.action = na.exclude, ntree = 40, metric="RMSE", trControl = trainControl(method = "repeatedcv", number = 10, repeats = 3), importance = T) |
|  |  |
|  | rmse\_model5 <- getTrainPerf(train\_rf)$TrainRMSE |
|  | rmse\_preds <- add\_row(rmse\_preds, Method = "Model 5: Random Forest", RMSE = rmse\_model5) |
|  |  |
|  | # Check Results |
|  |  |
|  | rmse\_preds %>% kable() |
|  |  |
|  | ``` |
|  |  |
|  | **## Variable Importance Per Model** |
|  |  |
|  | Here, we should look at the variables with the highest importance to see if there are trends and valuable insights: |
|  | \ |
|  |  |
|  | ``` {r varImp} |
|  |  |
|  | varImp(train\_glm) |
|  | varImp(train\_knn) |
|  | varImp(train\_svm) |
|  | varImp(train\_rf) |
|  |  |
|  | ``` |
|  |  |
|  | **# Results** |
|  |  |
|  | Our best model was Model 5: Random Forest. Therefore, we will select this algorithm to run our final model against the validation set. Now, let's run the final model to see if we were successful in training our model: |
|  |  |
|  | **## Final Model on Validation Set** |
|  |  |
|  | ```{r final on validation set} |
|  |  |
|  | train\_final <- train(Global\_Sales ~ Platform + User\_Score + Critic\_Score + Critic\_Count + Genre, |
|  | method = "rf", data = train\_set, na.action = na.exclude, ntree = 40, metric="RMSE", trControl = trainControl(method = "repeatedcv", number = 10, repeats = 3), importance = T) |
|  |  |
|  | predicted <- predict(train\_final, newdata = val\_set) |
|  | rmse\_model6 <- RMSE(val\_set$Global\_Sales, predicted) |
|  | rmse\_preds <- add\_row(rmse\_preds, Method = "Model 6: Final on Validation", RMSE = rmse\_model6) |
|  |  |
|  | ``` |
|  |  |
|  | We were successful in further reducing our RMSE. |
|  |  |
|  | But let's take a look at the variable importance for the validation set: |
|  |  |
|  | ``` {r varImp final} |
|  |  |
|  | varImp(train\_final) |
|  |  |
|  | ``` |
|  | ``` {r final results of all models} |
|  |  |
|  | # Check Final Results |
|  |  |
|  | rmse\_preds %>% kable() |
|  |  |
|  | ``` |
|  |  |
|  | It looks like we did better - success! |
|  |  |
|  | **# Conclusion** |
|  |  |
|  | We were able to get the RMSE down to 1.330436! This shows that the model works pretty well as a machine learning algorithm to predict Global Sales based on VGChartz' Video Games Sales Dataset. The optimal model thus far is based on the Random Forest Model. However, I had to remove two variables in the final RF model due to returned errors - I could not debug in time but figured out that removing the variables did provide at least a good model. Another limitation was hardware - this was difficult to manage even with the parallel processing. My final report took roughly 25 mins to produce with parallel processing, but was successful nonetheless. |
|  |  |
|  | Future work would be to further tune the RF model and spend more time finding optimal numbers for ntree, mtry, and variables to use. Seems like Critic\_Score and Critic\_Count were the heavy favorites for variable importance - it would be interesting if we could develop this further to see if we can refine our prediction algorithm based on those factors. |
|  |  |
|  | The final RMSE is still much larger than desired but was successful in improving our algorithm by 34%! For reference, our worst model - the Simple Prediction Model - received an RMSE of 2.015945 It would be beneficial if we could garner more complete data as well through web scraping. One model I wished I had the time to adopt at the end was Matrix Factorization - I think this would be beneficial for future research. |
|  |  |
|  |  |
|  | **## Admin Note** |
|  |  |
|  | ``` {r stop parallel processing} |
|  |  |
|  | stopCluster(cl) # Stops Parallel Processing |
|  |  |
|  | ``` |
|  |  |