Video Game Sales Analysis

Linh Hua

December 12, 2020

Contents

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.

```
##########################################################################
# 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")
```

```r
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
# View the structure of the data set
str(vgsales)
```

```
> str(vgsales)
tibble [16,291 x 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Rank        : num [1:16291] 1 2 3 4 5 6 7 8 9 10 ...
 $ Name        : chr [1:16291] "Wii Sports" "Super Mario Bros." "Mario Kart Wii" "Wii Sports Resort" ...
 $ Platform    : chr [1:16291] "Wii" "NES" "Wii" "Wii" ...
 $ Year        : chr [1:16291] "2006" "1985" "2008" "2009" ...
 $ Genre       : chr [1:16291] "Sports" "Platform" "Racing" "Sports" ...
 $ Publisher   : chr [1:16291] "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
 $ NA_Sales    : num [1:16291] 41.5 29.1 15.8 15.8 11.3 ...
 $ EU_Sales    : num [1:16291] 29.02 3.58 12.88 11.01 8.89 ...
 $ JP_Sales    : num [1:16291] 3.77 6.81 3.79 3.28 10.22 ...
 $ Other_Sales : num [1:16291] 8.46 0.77 3.31 2.96 1 0.58 2.9 2.85 2.26 0.47 ...
 $ Global_Sales: num [1:16291] 82.7 40.2 35.8 33 31.4 ...
 - attr(*, "spec")=
  .. cols(
  ..   Rank = col_double(),
  ..   Name = col_character(),
  ..   Platform = col_character(),
  ..   Year = col_character(),
  ..   Genre = col_character(),
  ..   Publisher = col_character(),
  ..   NA_Sales = col_double(),
  ..   EU_Sales = col_double(),
  ..   JP_Sales = col_double(),
  ..   Other_Sales = col_double(),
  ..   Global_Sales = col_double()
  .. )
```

```r
# View data summary
summary(vgsales)
```

```
> summary(vgsales)
      Rank           Name             Platform            Year              Genre
 Min.   :    1   Length:16291       Length:16291       Length:16291       Length:16291
 1st Qu.: 4132   Class :character   Class :character   Class :character   Class :character
 Median : 8292   Mode  :character   Mode  :character   Mode  :character   Mode  :character
 Mean   : 8290
 3rd Qu.:12440
 Max.   :16600
  Publisher            NA_Sales          EU_Sales          JP_Sales         Other_Sales
 Length:16291       Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.00000   Min.   : 0.00000
 Class :character   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.00000   1st Qu.: 0.00000
 Mode  :character   Median : 0.0800   Median : 0.0200   Median : 0.00000   Median : 0.01000
                    Mean   : 0.2656   Mean   : 0.1477   Mean   : 0.07883   Mean   : 0.04843
                    3rd Qu.: 0.2400   3rd Qu.: 0.1100   3rd Qu.: 0.04000   3rd Qu.: 0.04000
                    Max.   :41.4900   Max.   :29.0200   Max.   :10.22000   Max.   :10.57000
  Global_Sales
 Min.   : 0.0100
 1st Qu.: 0.0600
 Median : 0.1700
 Mean   : 0.5409
 3rd Qu.: 0.4800
 Max.   :82.7400
```

```r
# 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
# 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
```

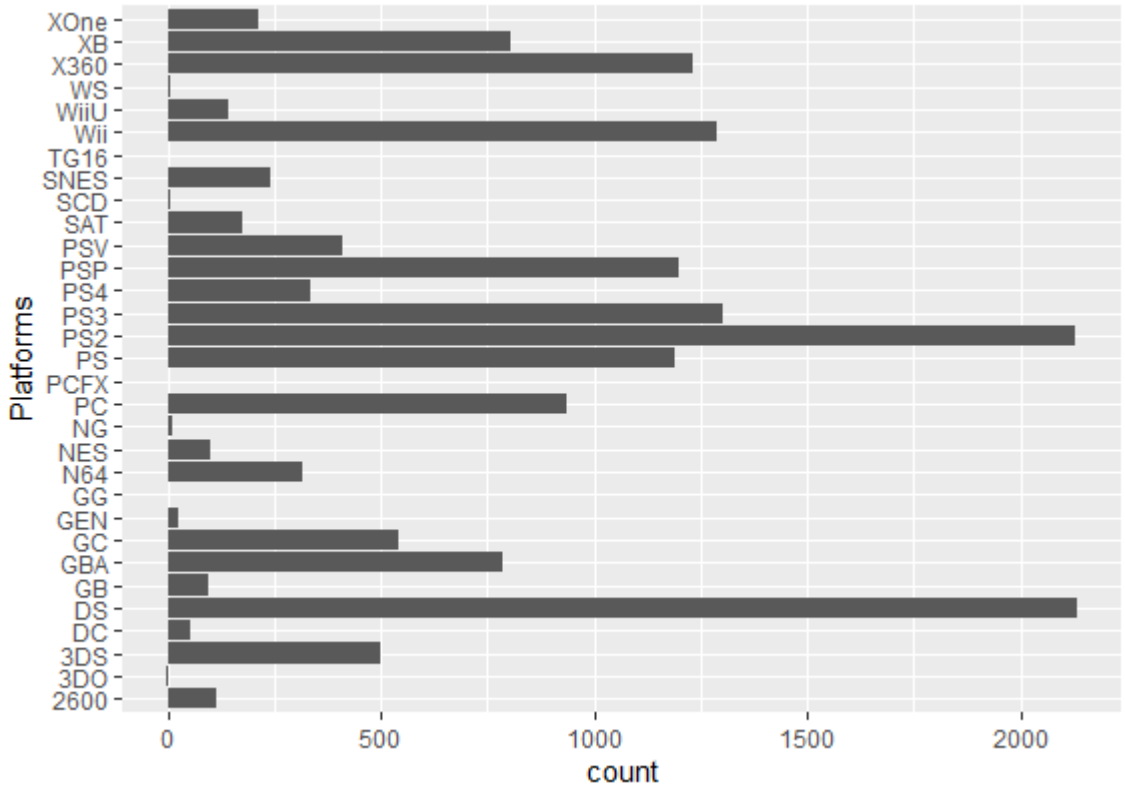
```
> platforms
 [1] "Wii"  "NES"  "GB"   "DS"   "X360" "PS3"  "PS2"  "SNES" "GBA"  "3DS"  "PS4"  "N64"  "PS"   "XB"
[15] "PC"   "2600" "PSP"  "XOne" "GC"   "WiiU" "GEN"  "DC"   "PSV"  "SAT"  "SCD"  "WS"   "NG"   "TG16"
[29] "3DO"  "GG"   "PCFX"
```

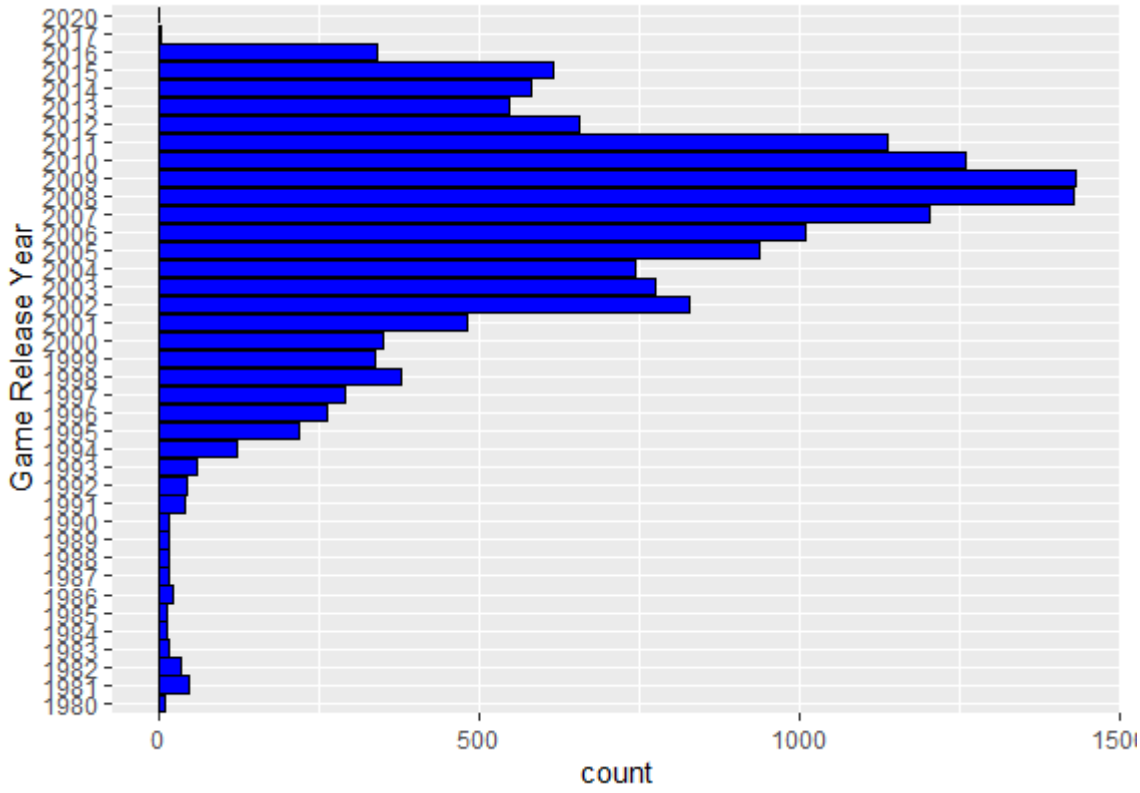
The average video games sales was 0.265 million dollars and the standard deviation was 0.822 in the North America. In comparison, the average video games sales was 0.540 million dollars and the standard deviation was 1.567 globally. Larger standard deviations indicate the data were more spread out in both cases.

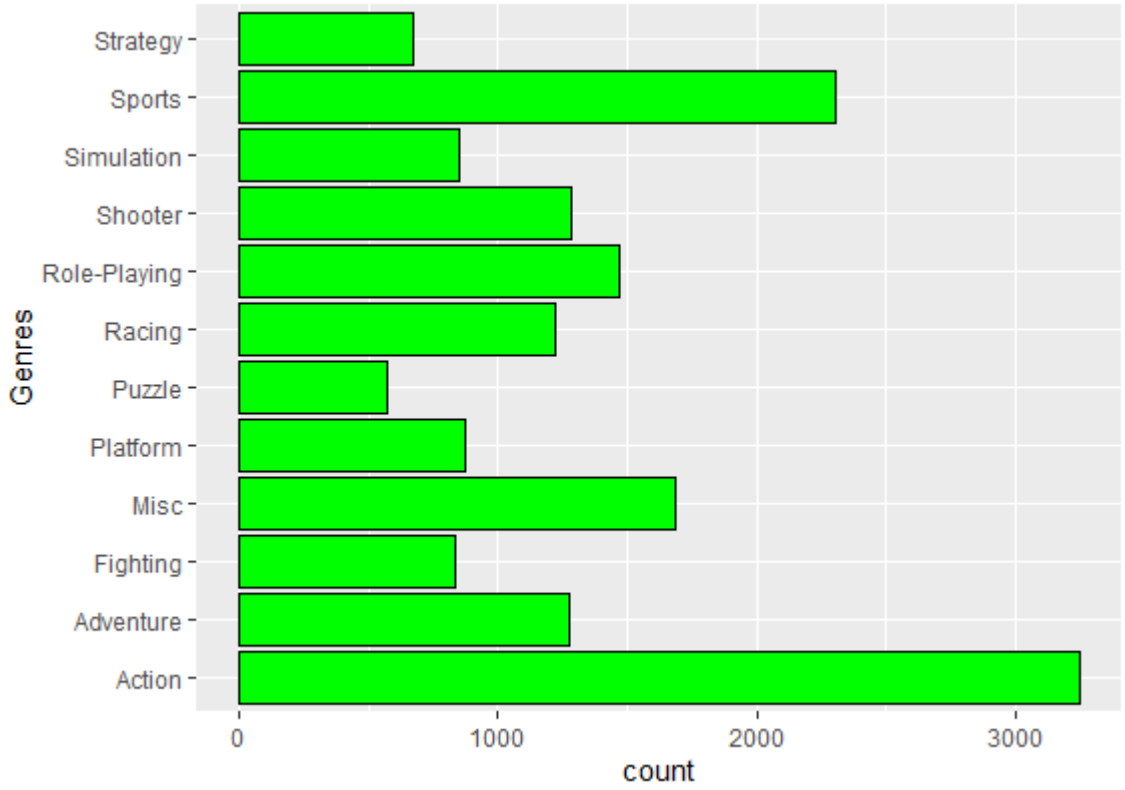
From the plots below, we gleaned that the top 3 game platforms were Nintendo DS, PlayStation 2 and PlayStation 3. The top 3 years to release the most games were 2008, 2009 and 2010. The top 3 game genres were Action, Sports and Miscellaneous.

```r
# 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()
```

```r
# Bar plot: Genre counts
vgsales %>%
  ggplot(aes(Genre)) +
  labs(x = "Genres") +
  geom_bar(fill = "green", col = "black") +
  coord_flip()
```

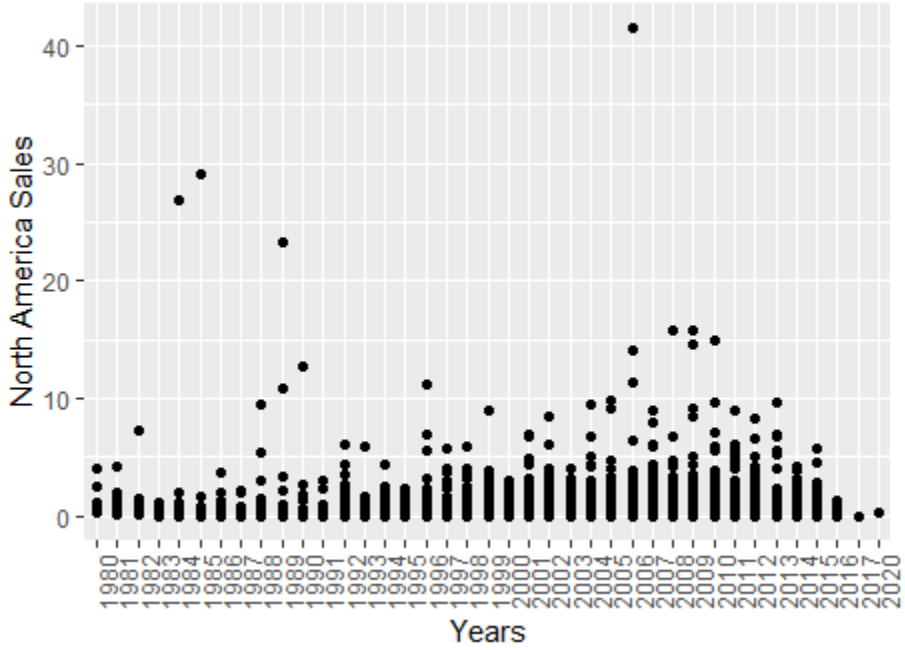


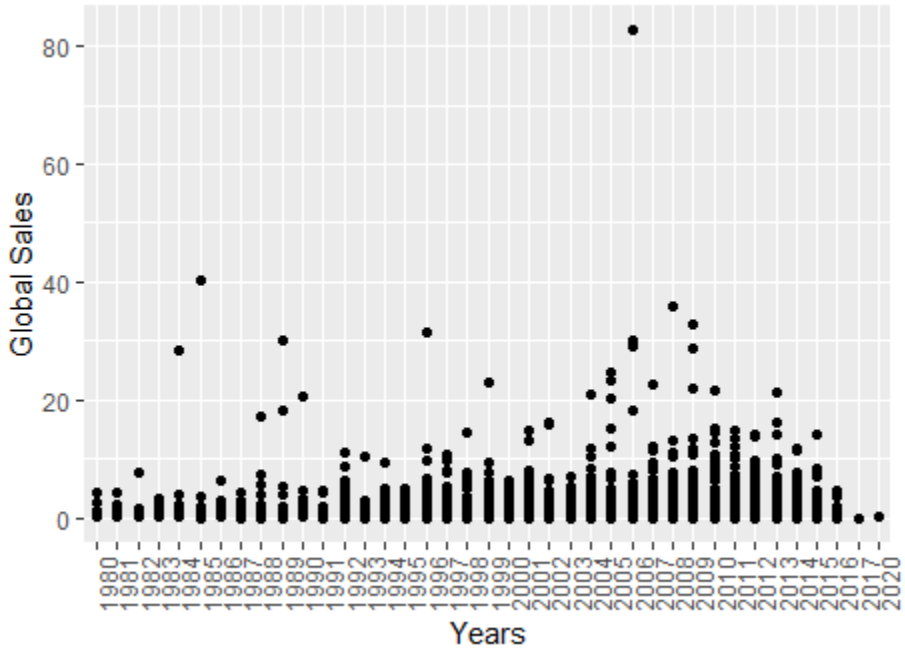

Game Release Year
2020
2017
2016
2015
2014
2013
2012
2011
2010
2009
2008
2007
2006
2005
2004
2003
2002
2001
2000
1999
1998
1997
1996
1995
1994
1993
1992
1991
1990
1989
1988
1987
1986
1985
1984
1983
1982
1981
1980
0
500
1000
count


Genres
Strategy
Sports
Simulation
Shooter
Role-Playing
Racing
Puzzle
Platform
Misc
Fighting
Adventure
Action
0
1000
2000
3000
count

We compared the video games sales between North America and Global respectively below:

```
# 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))
```





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.

```
# 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.

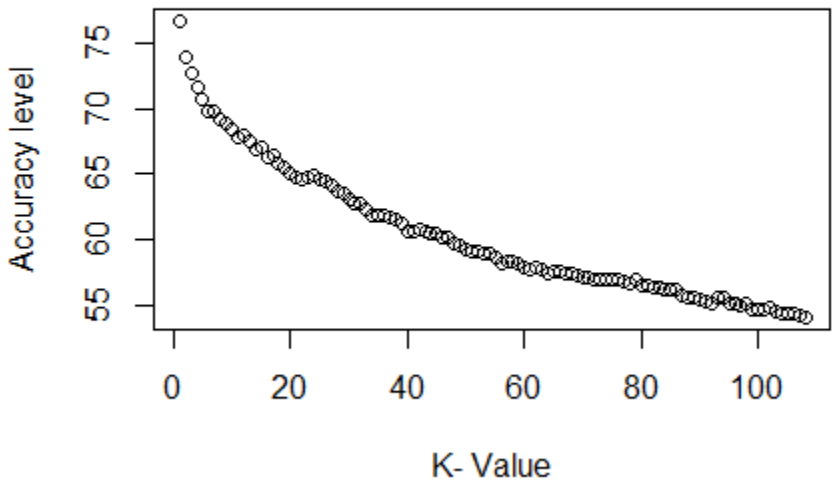
```
##### kNN Model: #####

## 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.

```
## 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.

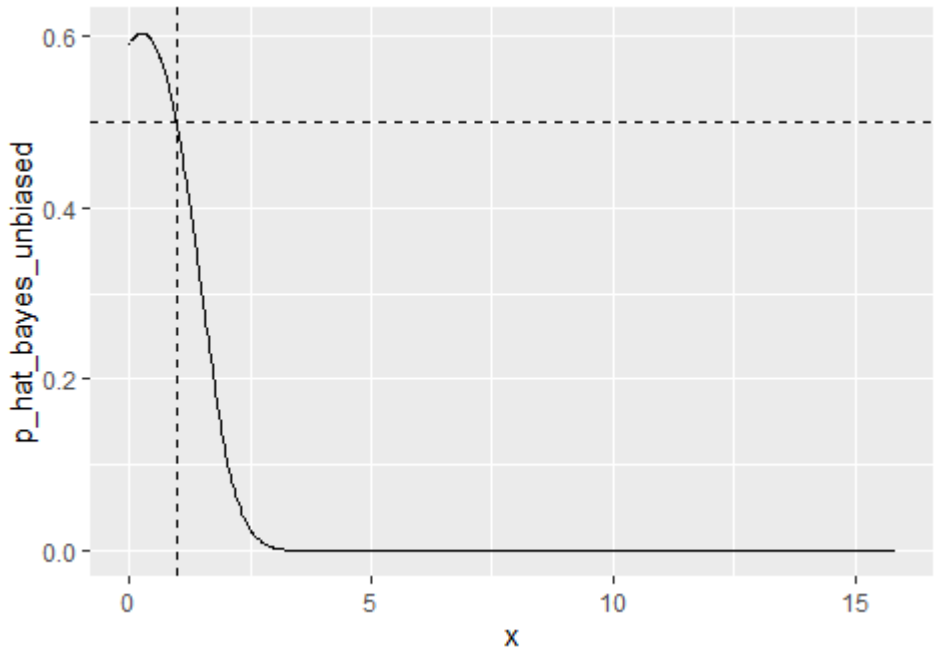


2.3.2 Naive Bayes Model

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.

```
#####  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")
```

```r
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)
```



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
#####  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,
```

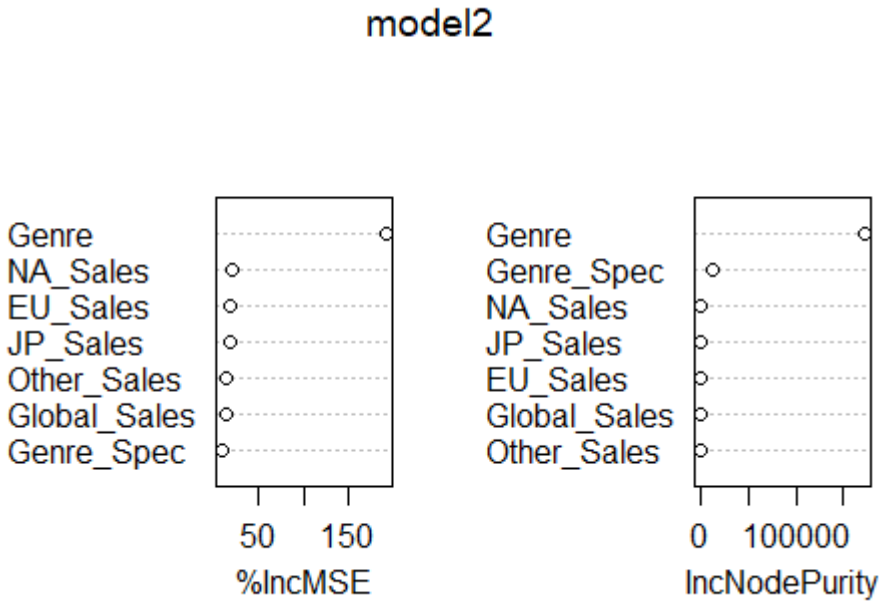
```
    Genre == "Racing" ~ 6,
    Genre == "Puzzle" ~ 7,
    Genre == "Platform" ~ 8,
    Genre == "Misc" ~ 9,
    Genre == "Fighting" ~ 10,
    Genre == "Adventure" ~ 11,
    Genre == "Action" ~ 12,
  ))

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
```

```
> model1

Call:
 randomForest(formula = Genre_No ~ ., data = train_set, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 0.0776066
                    % Var explained: 99.45
```

```
> model2

Call:
 randomForest(formula = Genre_No ~ ., data = train_set, ntree = 500,      mtry = 6, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 6

          Mean of squared residuals: 0.0002617268
                    % Var explained: 100
```

```r
# 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)
```

```
> importance(model2)
               %IncMSE IncNodePurity
Genre        191.67169  1.725509e+05
NA_Sales      20.84653  4.845962e+01
EU_Sales      19.21741  1.920287e+01
JP_Sales      18.22255  4.508181e+01
Other_Sales   14.78067  8.900055e+00
Global_Sales  13.55422  1.791590e+01
Genre_Spec    10.18792  1.209096e+04
```


model2
Genre
NA_Sales
EU_Sales
JP_Sales
Other_Sales
Global_Sales
Genre_Spec
50
150
%IncMSE
Genre
Genre_Spec
NA_Sales
JP_Sales
EU_Sales
Global_Sales
Other_Sales
0
100000
IncNodePurity

3 Results and Conclusions

Data Discovery Results:

Description	Value
Data rows	16,291
Variables	11
Number of places games were sold	5
Names of places games were sold	North America, Europe, Japan, Other and Global
Number of Publishers	576
Number of Platforms	31
Average video games sales (North America)	0.265 million
Standard deviation (North America)	0.822
Average video games sales (Globally)	0.540 million
Standard deviation (Globally)	1.567
Top 3 game platforms	Nintendo DS, PlayStation 2 and PlayStation 3
Top 3 years to release the most games	2008, 2009 and 2010
Top 3 game genres	Action, Sports and Miscellaneous
Worst seller (North America)	Monster Hunter Freedom 3 ($0)
Worst seller (Globally)	Turok (0.01 million)
Best seller (North America)	Wii Sports (41.49 million)
Best seller (Globally)	Wii Sports (82.74 million)

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.

Description	Result
ACC_106	54.50082
ACC_106	54.41899

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.

Description	Result
Sensitivity factor (y_hat_bayes_unbiased)	0.9342508
Specificity factor (y_hat_bayes_unbiased)	0.0555767

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.

Description	Result
Model 1: Mean of squared residuals	0.0776066
Model 1: % var explained	99.45
Model 2: Mean of squared residuals	0.0002617268
Model 2: % var explained	100
Importance: Genre	191.67
Importance: NA_Sales	20.85
Importance: Genre_Spec	10.19

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:

OS	Windows 10
IDE	RStudio version 1.3.1093
Language	R version 4.0.3 (2020-10-10)