Choose Your Own Project Submission

Michael Krog

June 2019

Introduction

For my Choose your own project submission I have chosen to try and predict the success of a video game globally in relation to other games, based on the sales data from a given region.

To categorise the success factors, I've graded global sales within the dataset as being in the top 100 selling games worldwide, the top 1000 selling games worldwide (excluding the top 100), and the rest.

The goal is then to be able to apply the algorithm herein to the sales of a given region and predict the likelihood that a game will be a top 100 game, a top 1000 game or something outside of this.

The video game data is from VgChartz, with review ratings from Metacritic. The data is available from Kaggle here:https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings/downloads/video-game-sales-with-ratings.zip/2

Method/Analysis

My objective here was to be able to build a KNN model that would be able to help us predict global sales based on regional sales with high accuracy.

To be able to achieve this I have had to run through a process of downloading and cleaning the data, running data analysis, preprocessing and finally training the data to be able to format a successful KNN model.

Downloading and cleaning the data

```
#Load the necessary packages:
library(tidyverse)
library(caret)

#Load the CSV data file:
video_games_full <- read_csv("Video_Games_Sales_as_at_22_Dec_2016.csv")


#Cleaning the data set:
#Remove NA's:
video_games_full <- video_games_full %>% filter(!is.na(video_games_full$Critic_Score))
video_games_full <- video_games_full %>% filter(!is.na(video_games_full$User_Score))
video_games_full <- video_games_full %>% filter(!is.na(video_games_full$Critic_Count))
video_games_full <- video_games_full %>% filter(!is.na(video_games_full$User_Count))



#Changing user score columns to numeric from character:
video_games_full$User_Score <- as.numeric(video_games_full$User_Score)
```

```r
#Confirming class of column
class(video_games_full$User_Score)
```

```
## [1] "numeric"
```

Data Exploration:

```r
#Count of the number of rows & columns
nrow(video_games_full)
```

```
## [1] 7017
```

```r
ncol(video_games_full)
```

```
## [1] 16
```

```r
#Number of user scores below or equal to six, and above or equal to seven
length(which(video_games_full$User_Score <= 6))
```

```
## [1] 1322
```

```r
length(which(video_games_full$User_Score >= 7))
```

```
## [1] 4601
```

```r
#Number of different games:
diff_games <- unique(video_games_full$Name)
length(diff_games)
```

```
## [1] 4471
```

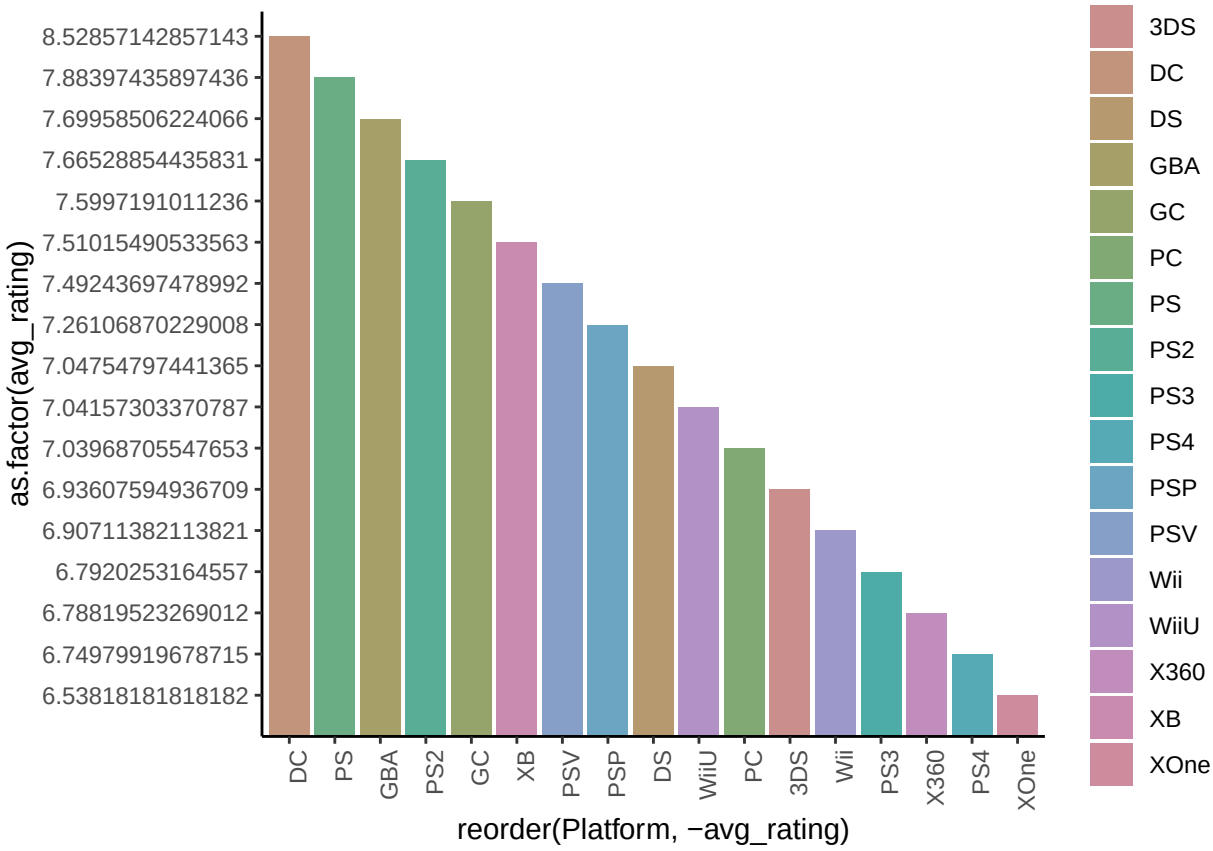
```r
#Number of different platforms:
n_distinct(video_games_full$Platform)
```

```
## [1] 17
```

```r
#Number of different publishers:
n_distinct(video_games_full$Publisher)
```

```
## [1] 272
```

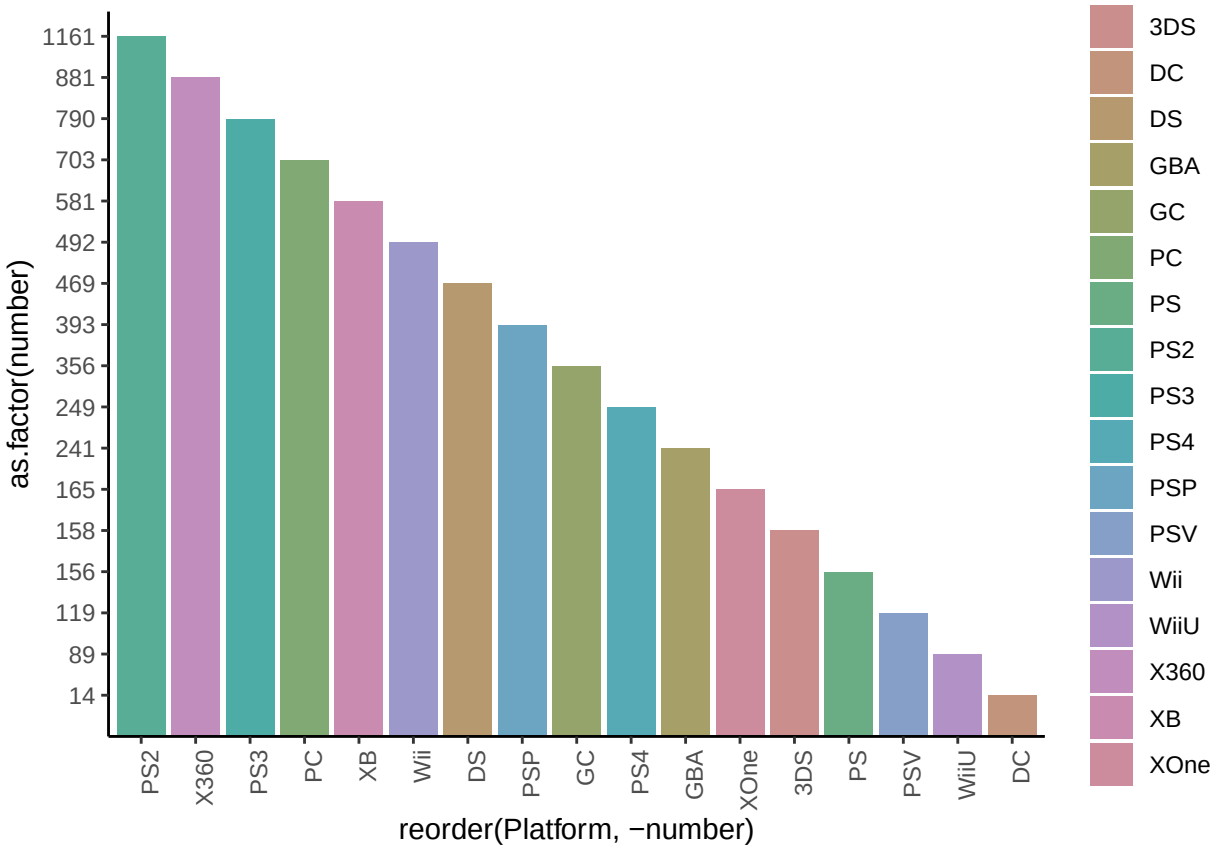
```r
#Avg score per platform:
#Create vector 'plat_ratings'
plat_ratings <- video_games_full %>%
  #Pipe from video_games_full and group by platform.
  group_by(Platform) %>%
  #Summarise avg_rating as the mean user score.
  summarise(avg_rating = mean(User_Score)) %>%
  #Arrange in descending order based on avg_rating.
  arrange(desc(avg_rating))

#Plot platform_ratings in descending order as a bar graph, with Platforms as x, and avg rating as y.
ggplot(plat_ratings, aes(x= reorder(Platform, -avg_rating), y= as.factor(avg_rating), fill = Platform))
  #bar graph.
  geom_bar(stat = "identity") +
  #Classic theme
  theme_classic() +
  #A hue fill
  scale_fill_hue(c = 40) +
  #Adjust axis text to 90 degrees.
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



It's interesting to see that the average rating on lesser known or older platforms is higher than on newer or more relevant platforms. This is likely due to the fact that older or lesser known platforms have less games, and hence less scores overall, and may be skewed as a result.

Let's explore this now:

```r
#Looking at number of games per platform:
#Create vector title_count_per_platform piped from full data set.
title_count_per_platform <- video_games_full %>%
  #Group by platform.
  group_by(Platform) %>%
  #summarise the count.
  summarise(number = n()) %>%
  #Arrange in descending order.
  arrange(desc(number))

#Plot title_count_per_platform in descending order.
ggplot(title_count_per_platform, aes(x= reorder(Platform, -number), y= as.factor(number), fill = Platfo
  #As a bar plot.
  geom_bar(stat = "identity") +
  #Classic theme
  theme_classic() +
  #Hue fill
  scale_fill_hue(c = 40) +
  #Adjust text angle to 90 degrees.
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

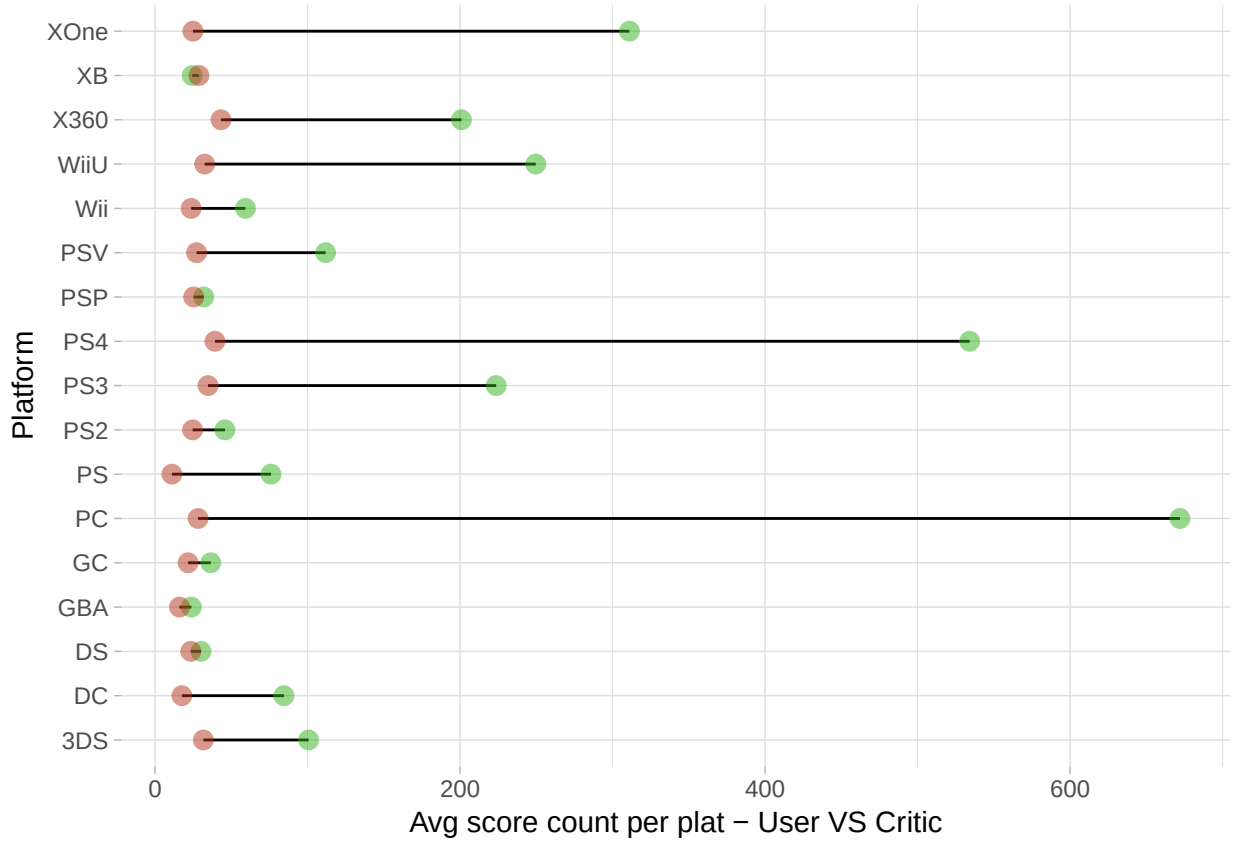


This makes a lot more sense where we can see that highly popular platforms like PS2 and Xbox 360 have far more games on each platfrom which makes sense. This also confirms the thought that platforms with high avg scores might be skewed because of having very few titles, which we can see for DC was the case.

Knowing the number of games per platform, let's take a guess at the platform popularity based on the

number of user and critic scores, and see if this aligns with the number of titles per platform.

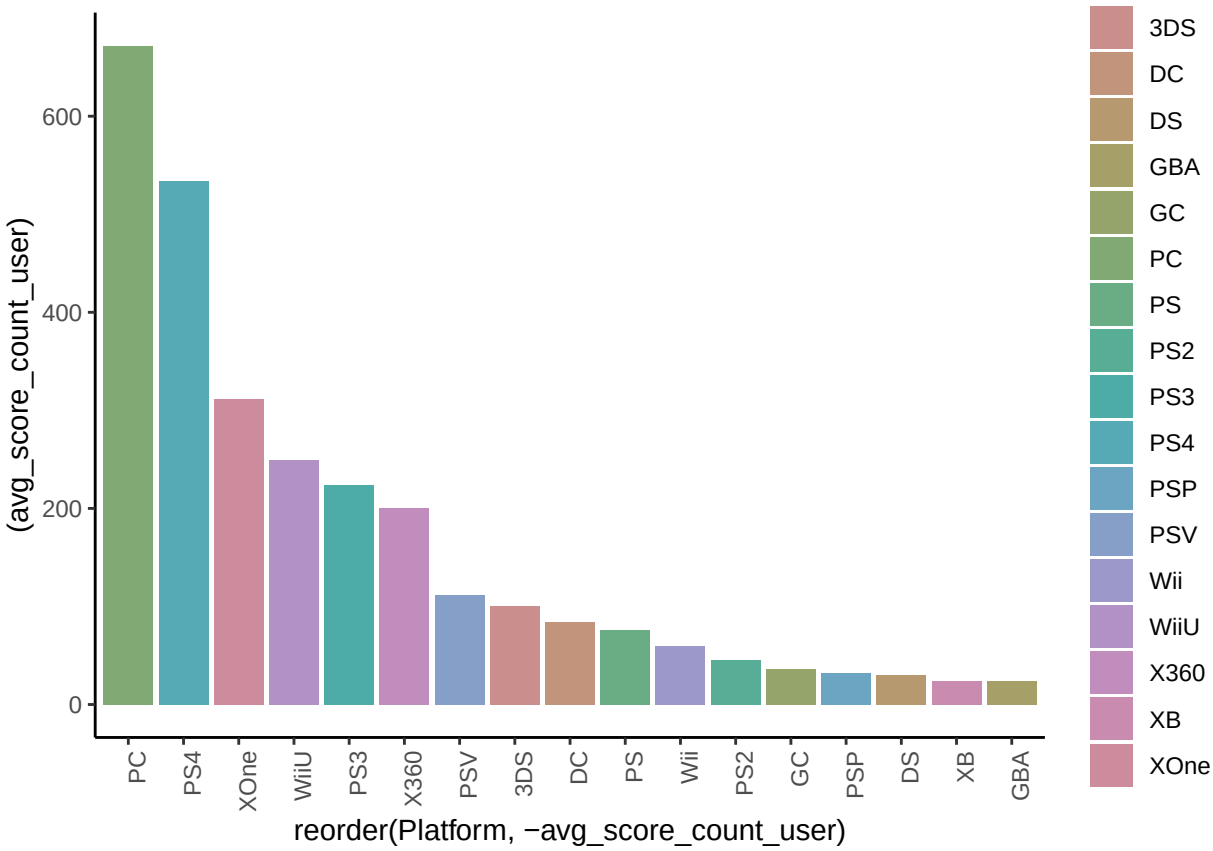
```
#Create vector avg_score_count_per_platfrom piped from full data set.
avg_score_count_per_plat <- video_games_full %>%
  #Group by platform.
  group_by(Platform) %>%
  #Summarise by average score count of users, and of critics.
  summarise(avg_score_count_user = mean(User_Count), avg_score_count_critic = mean(Critic_Count)) %>%
  #Arrange in descending order based on avg_score_count_user.
  arrange(desc(avg_score_count_user))


#Plot average score count per platform.
ggplot(avg_score_count_per_plat) +
  #Segment by platform (user and critic avg scores will be plotted with a black line)
  geom_segment( aes(x=Platform, xend=Platform, y=avg_score_count_user, yend=avg_score_count_critic), col
  #avg_user_score_counts plotted in green.
  geom_point( aes(x=Platform, y=avg_score_count_user), color=rgb(0.2,0.7,0.1,0.5), size=3, show.legend =
  #avg critic score counts plotted in red.
  geom_point( aes(x=Platform, y=avg_score_count_critic), color=rgb(0.7,0.2,0.1,0.5), size=3, show.legend
  #Flip x & y.
  coord_flip()+
  #Light theme.
  theme_light() +
  #Legend position at bottom.
  theme(
    legend.position = "bottom",
    panel.border = element_blank(),
  ) +
  # X label.
  xlab("Platform") +
  # Y label.
  ylab("Avg score count per plat - User VS Critic") +
  #Scale colour.
  scale_color_manual(name = "Year", labels = c("critic", "user"))
```



It's interesting to see that on some platforms there are far more user scores (green), than critic scores (red). This makes sense in as many users can score games on their preferred platform, but critics are probably paid to score games across platfroms. We could probably induce some measure of platform popularity from this as well based on the number of user scores. E.g. PC has the highest count of user scores followed by PS4 and Xbox One which makes sense in terms of popularity.

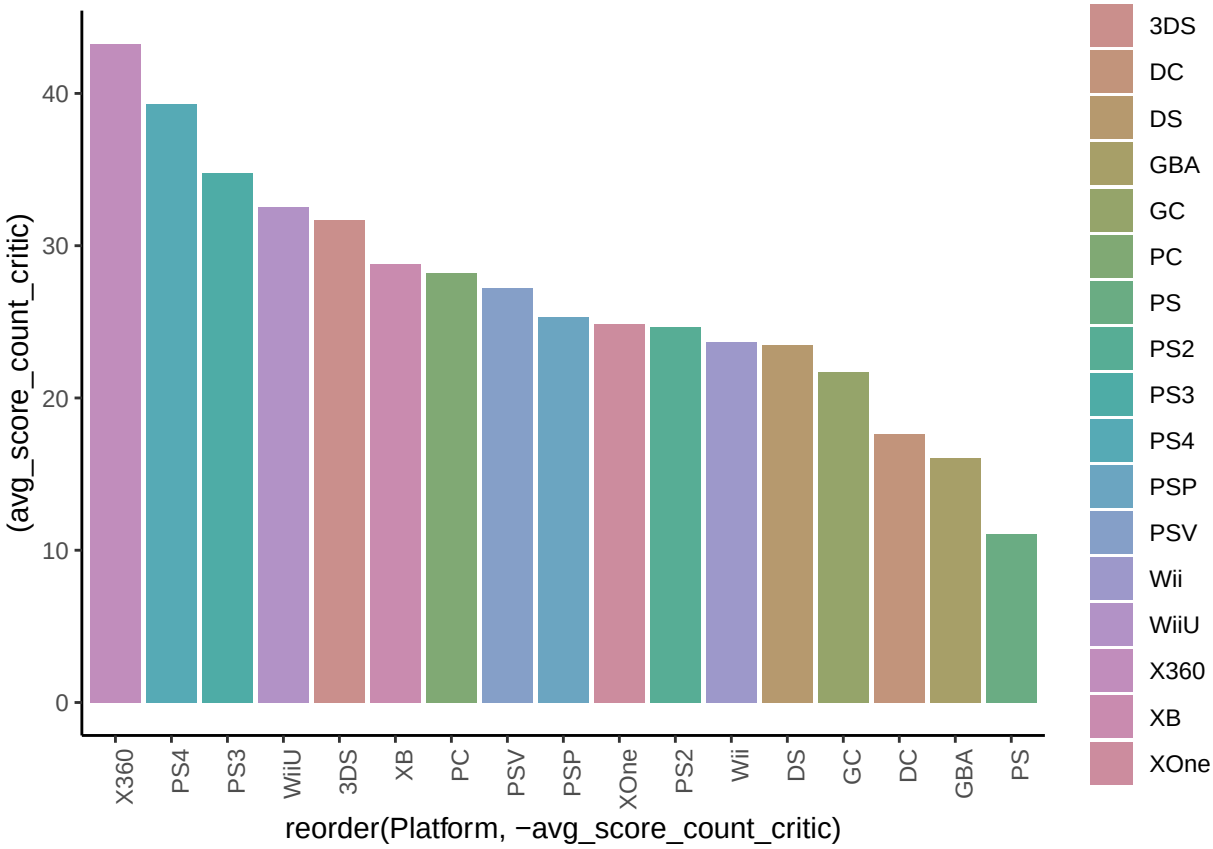
Let's plot this estimated popularity based on user score count:

```
#Avg count of scores users:
#Plot average score count per platform in descending order.
ggplot(avg_score_count_per_plat, aes(x= reorder(Platform, -avg_score_count_user), y= (avg_score_count_us
  #Create a bar plot
  geom_bar(stat = "identity") +
  #Use 'classic' theme.
  theme_classic() +
  #Use a hue fill.
  scale_fill_hue(c = 40) +
  #Rotate axis text 90 degrees.
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Out of interest let's do the same for critic score count. Although much more closely aligned than user count, this may give us some insight into which platforms company's are paying critics to review on.

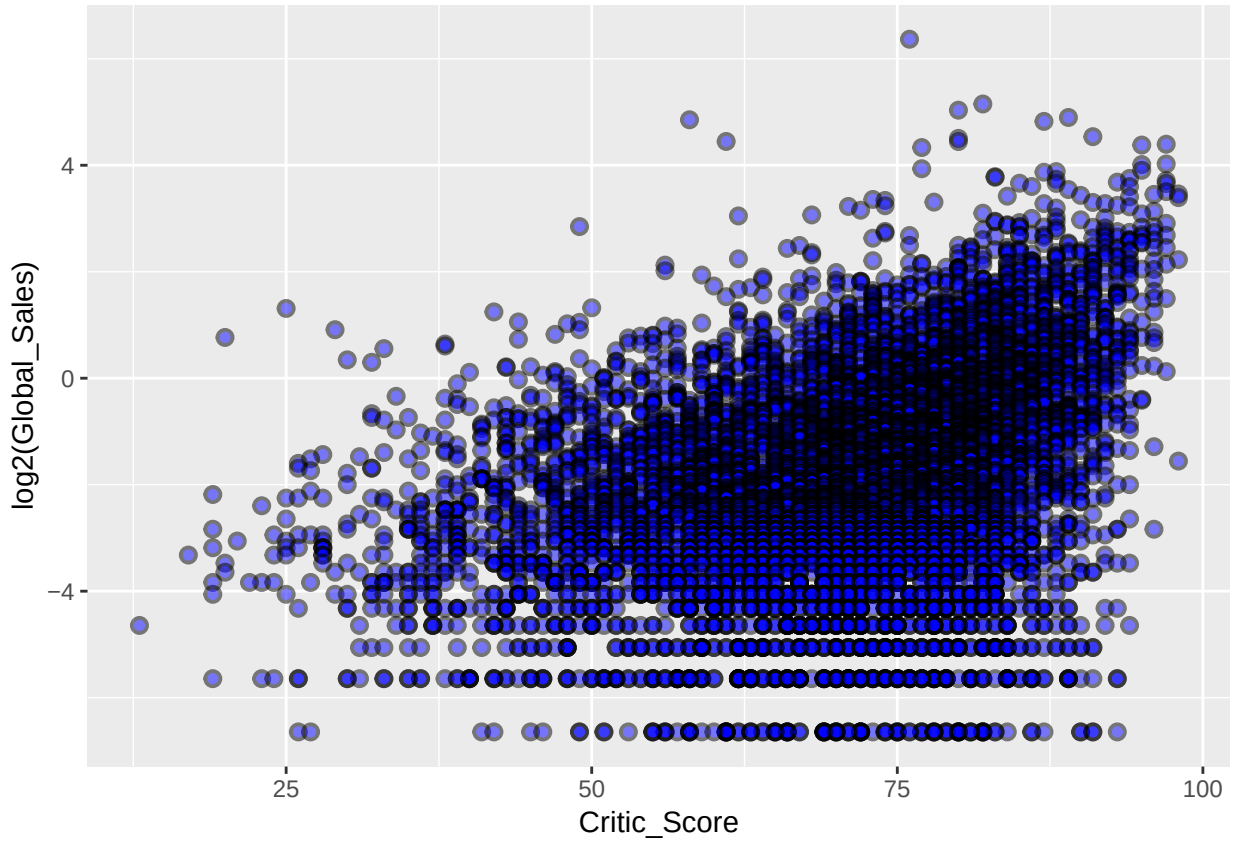
```
#Avg score count critic:
#Plot average score count per platform in descending order.
ggplot(avg_score_count_per_plat, aes(x= reorder(Platform, -avg_score_count_critic), y= (avg_score_count
  #Create a bar plot.
  geom_bar(stat = "identity") +
  #Classic theme.
  theme_classic() +
  #Hue scale colour fill.
  scale_fill_hue(c = 40) +
  #Rotate label text 90 degrees.
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



It's really interesting to see that the platforms getting the most critic reviews are different to the platforms getting the most customer reviews. There might be a misalignment here that could be remedied by the platforms critics focus their reviews on.

Ok so now let's also look at the global sales data and see if there's any correlation with critic scores.

```
#Plat critic score against the log2 of global sales.
ggplot(video_games_full, aes(x=Critic_Score, y= log2(Global_Sales))) +
  #Make points blue with black border, circle size 2.
  geom_point(
        color="black",
        fill="blue",
        shape=21,
        alpha=0.5,
        size=2,
        stroke = 1
        )
```



So there is a definite relationship between the critic score and global sales, which makes sense. Games with better scores are also better selling games.

Let's also have a look at the sales by region:

```
#Total sales value North America:
sum(video_games_full$NA_Sales)
```

```
## [1] 2731.65
```

```
#Total sales value Europe:
sum(video_games_full$EU_Sales)
```

```
## [1] 1635.63
```

```
#Total sales value Japan:
sum(video_games_full$JP_Sales)
```

```
## [1] 441.73
```

```
#Total sales value other regions:
sum(video_games_full$Other_Sales)
```

```
## [1] 572.06
```

It's interesting to see that North America had (at least in this data set) by far the highest number of sales, followed by Europe, Japan and other regions.

Determining Outcomes

To determine possible outcomes for our eventual predictions, I've grouped global sales into 3 levels:

1) Sales that equal the top 100 games.
2) Sales that our outside the top 100 but inside the top 1000
3) Sales outside the top 1000

This will lead to our prediction model of predicting the category a game will fall into in terms of global sales based on the sales in a given region.

To create these groups I have used the method below with trial and error on the numbers that give the closest possible results:

```
#Count of video games with global sales bigger than or equal to 6.08
plyr::count(video_games_full$Global_Sales >= 6.08)
```

```
##       x freq
## 1 FALSE 6917
## 2  TRUE  100
```

As can be seen above this equates perfectly to the top 100 games.

```
#Count of video games with global sales smaller than or equal to 6.08 & bigger than or equal to 1.279
plyr::count(video_games_full$Global_Sales <= 6.08 & video_games_full$Global_Sales >= 1.279)
```

```
##       x freq
## 1 FALSE 6119
## 2  TRUE  898
```

As can be seen this is 898 which is as close as I could get to the figure of 900 to make a perfect 1000.

I've then created these groups into a column in the full data set as below, and removed the unnecessary columns for our model:

```
#Create column 'performance', if global sales bigger than or equal to 6.08 = "top_hundred".
video_games_full$performance <- ifelse(video_games_full$Global_Sales >= 6.08,"top_hundred",
    #If smaller than 6.08 or bigger than or equal to 1.279 = "top_thousand".
        ifelse(video_games_full$Global_Sales < 6.08 & video_games_full$Global_Sales >= 1.279,"top_thousa
        #If smaller than 1.279 = "thousand_plus" otherwise "other".
        ifelse(video_games_full$Global_Sales < 1.279, "thousand_plus", "other"
        )))

#Remove unnecessary columns:
video_games_full = subset(video_games_full, select = -c(Name, Platform, Year_of_Release, Genre, Publishe
```

Creating the test & training sets

I've created the training set as 80% of the data and the test set as 20%

```r
#Creating test and training sets:
set.seed(1)
index_video_games <- createDataPartition(video_games_full$Other_Sales, times = 1, p = 0.2, list = FALSE)

train_set_vg <- video_games_full[index_video_games, ]
test_set_vg <- video_games_full[-index_video_games, ]
```

Creating the KNN prediction model.

```r
#Looking at a table of the performance factors:
table(train_set_vg$performance)
```

```
##
## thousand_plus   top_hundred  top_thousand
##          1192            22           191
```

```r
#Making sure the performance column is a factor:
train_set_vg[["performance"]] = factor(train_set_vg[["performance"]])

#Creating the model using trainControl from Caret package.
#Using repeated cross validation method for re-sampling.
#Creating 10 re-sampling iterations (folds)
#Using 3 repeats.
train_control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
#Setting the seed:
set.seed(3)
#Using KNN method setting the target variable 'performance' using all classifiers.
knn_fit <- train(performance ~., data = train_set_vg, method = "knn",
#trControl passed with results of the trainControl method.
 trControl=train_control,
#Pre-processing the data - centering & scaling.
#Create mean value approximately '0' and standard deviation '1'
 preProcess = c("center", "scale"),
#Tune the algorithm
 tuneLength = 10)

#See results in knn_fit:
knn_fit
```

```
## k-Nearest Neighbors
##
## 1405 samples
##    4 predictor
##    3 classes: 'thousand_plus', 'top_hundred', 'top_thousand'
##
## Pre-processing: centered (4), scaled (4)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1264, 1265, 1265, 1265, 1265, 1264, ...
## Resampling results across tuning parameters:
##
```

```
##   k   Accuracy   Kappa
##    5  0.9845660  0.9402660
##    7  0.9843279  0.9387941
##    9  0.9836186  0.9355597
##   11  0.9812444  0.9258958
##   13  0.9793465  0.9182846
##   15  0.9781627  0.9136943
##   17  0.9765028  0.9069775
##   19  0.9765045  0.9065182
##   21  0.9769790  0.9083793
##   23  0.9746048  0.8981415
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

As can be seen above k=5 produced the best result of 98.4% accuracy. Our model has now been tuned to use this k parameter.

Running our KNN Algorithm on the test set

We are now ready to run the algorith on the test set.

```
#Creating test_pred vector using 'predict'
test_pred <- predict(knn_fit, newdata = test_set_vg)

#Make sure that 'perfomance' is a factor before running the confusion matrix.
test_set_vg[["performance"]] = factor(test_set_vg[["performance"]])

#Run the confusion matrix of predictions vs actuals:
confusionMatrix(test_pred, test_set_vg$performance)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      thousand_plus top_hundred top_thousand
##   thousand_plus          4812           0           42
##   top_hundred               0          55            5
##   top_thousand             16          23          659
##
## Overall Statistics
##
##                Accuracy : 0.9847
##                  95% CI : (0.9811, 0.9877)
##     No Information Rate : 0.8603
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9362
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: thousand_plus Class: top_hundred
## Sensitivity                        0.9967            0.70513
## Specificity                        0.9464            0.99910
## Pos Pred Value                     0.9913            0.91667
## Neg Pred Value                     0.9789            0.99586
## Prevalence                         0.8603            0.01390
## Detection Rate                     0.8574            0.00980
## Detection Prevalence               0.8649            0.01069
## Balanced Accuracy                  0.9716            0.85211
##                      Class: top_thousand
## Sensitivity                       0.9334
## Specificity                       0.9921
## Pos Pred Value                    0.9441
## Neg Pred Value                    0.9904
## Prevalence                        0.1258
## Detection Rate                    0.1174
## Detection Prevalence              0.1244
## Balanced Accuracy                 0.9627
```

As can be seen our model as worked well with an accuracy of 98.47%

Results

The result is that the model is able to predict whether a game falls into the categories, top 100, top 1000, or outside the top 1000 for global sales based on the performance of sales from any given region with a 98.4% accuracy.

Conclusion

Using the KNN method for making this prediction has proven effective. The hope is of course that this model could now be take and applied to any regional sales data to be able to make a prediction of global performance.

Hopefully I have been able to achieve that.