European Video Games Bestsellers

Olga Loginova

19/06/2020

Introduction

Overview

The project is part of the HervardX: PH125.9x Data Science: Capstone course. The entire course consists of two parts: MovieLens Project and CYO Project, respectively. The current document is dedicated to CYO Project.

For this project, students must apply machine learning techniques other than the standard linear regression. Students are allowed to use any publicly available dataset to solve the problem of their choice. There should be at least two machine learning models the data is trained and tested on.

Libraries

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------------------------------------------------

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------------------------------- tidyve
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
## Loading required package: caretEnsemble
```

```
##
## Attaching package: 'caretEnsemble'
```

```
## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## Loading required package: DT
```

```
## Loading required package: wordcloud
```

```
## Loading required package: RColorBrewer
```

Dataset

The dataset used in the project is a combination of VGChartz Video Games Sales and corresponding ratings from Metacritic Video.

This dataset extends the number of variables from VGChartz Video Games Sales (Name, Platform, Year_of_Release, Genre, Publisher, NA_Sales, EU_Sales, JP_Sales, Other_Sales, Global_Sales) with a web scrape from Metacritic adding Critic_score, Critic_Count (the number of critics who gave a Critic_Score), User_Score, User_Count, Developer (game developer), and Rating (ESRB rating).

There are 16719 records in the data set:

```
nrow(data)
```

```
## [1] 16719
```

First six rows (to check the column names and information in them):

```
##                        Name Platform Year_of_Release        Genre Publisher
## 1                Wii Sports      Wii            2006       Sports  Nintendo
## 2         Super Mario Bros.      NES            1985     Platform  Nintendo
## 3            Mario Kart Wii      Wii            2008       Racing  Nintendo
## 4         Wii Sports Resort      Wii            2009       Sports  Nintendo
## 5 Pokemon Red/Pokemon Blue       GB            1996 Role-Playing  Nintendo
## 6                    Tetris       GB            1989       Puzzle  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
## 1    41.36    28.96     3.77        8.45        82.53           76           51
## 2    29.08     3.58     6.81        0.77        40.24           NA            NA
## 3    15.68    12.76     3.79        3.29        35.52           82            73
## 4    15.61    10.93     3.28        2.95        32.77           80            73
## 5    11.27     8.89    10.22        1.00        31.37           NA            NA
## 6    23.20     2.26     4.22        0.58        30.26           NA            NA
##   User_Score User_Count Developer Rating
## 1          8        322  Nintendo      E
## 2                    NA
## 3        8.3        709  Nintendo      E
## 4          8        192  Nintendo      E
## 5                    NA
## 6                    NA
```

The data seems to be in tidy format, but there are missing information and NAs in at least Critic_Score, Critic_Count, User_Score, User_Count, Developer. Critic_Score is essential for predicting models, so we need to tackle the NAs and empty strings there.

There are no duplicates in the data:

```
## [1] 0
```

Let's see the summary statistics of each column and data structure:

```
## 'data.frame':    16719 obs. of  16 variables:
##  $ Name           : chr  "Wii Sports" "Super Mario Bros." "Mario Kart Wii" "Wii Sports Resort" ...
##  $ Platform       : chr  "Wii" "NES" "Wii" "Wii" ...
##  $ Year_of_Release: chr  "2006" "1985" "2008" "2009" ...
##  $ Genre          : chr  "Sports" "Platform" "Racing" "Sports" ...
##  $ Publisher      : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ NA_Sales       : num  41.4 29.1 15.7 15.6 11.3 ...
##  $ EU_Sales       : num  28.96 3.58 12.76 10.93 8.89 ...
##  $ JP_Sales       : num  3.77 6.81 3.79 3.28 10.22 ...
##  $ Other_Sales    : num  8.45 0.77 3.29 2.95 1 0.58 2.88 2.84 2.24 0.47 ...
##  $ Global_Sales   : num  82.5 40.2 35.5 32.8 31.4 ...
##  $ Critic_Score   : int  76 NA 82 80 NA NA 89 58 87 NA ...
##  $ Critic_Count   : int  51 NA 73 73 NA NA 65 41 80 NA ...
##  $ User_Score     : chr  "8" "" "8.3" "8" ...
##  $ User_Count     : int  322 NA 709 192 NA NA 431 129 594 NA ...
##  $ Developer      : chr  "Nintendo" "" "Nintendo" "Nintendo" ...
##  $ Rating         : chr  "E" "" "E" "E" ...
```

Year_of_Release is a character vector. Let's convert it into the numeric one.

```r
data$Year_of_Release <- as.numeric(data$Year_of_Release)
```

```
## Warning: NAs introduced by coercion
```

As long as the project is focused on EU Sales, we will remove all zero values in the EU_Sales column (apparently, the games were not distributed on the European market).

There are also some missing year data Year_of_Release.

```r
y_data <- data %>% filter(is.na(Year_of_Release))
head(y_data)
```

```
##                          Name Platform Year_of_Release    Genre
## 1             Madden NFL 2004      PS2              NA   Sports
## 2            FIFA Soccer 2004      PS2              NA   Sports
## 3 LEGO Batman: The Videogame      Wii              NA   Action
## 4 wwe Smackdown vs. Raw 2006      PS2              NA Fighting
## 5             Space Invaders     2600              NA  Shooter
## 6                  Rock Band     X360              NA     Misc
##                                   Publisher NA_Sales EU_Sales JP_Sales Other_Sales
## 1                           Electronic Arts     4.26     0.26     0.01        0.71
## 2                           Electronic Arts     0.59     2.36     0.04        0.51
## 3 Warner Bros. Interactive Entertainment     1.80     0.97     0.00        0.29
## 4                                       N/A     1.57     1.02     0.00        0.41
## 5                                     Atari     2.36     0.14     0.00        0.03
## 6                           Electronic Arts     1.93     0.33     0.00        0.21
##   Global_Sales Critic_Score Critic_Count User_Score User_Count
## 1         5.23           94           29        8.5        140
## 2         3.49           84           20        6.4         76
## 3         3.06           74           17        7.9         22
## 4         3.00           NA           NA                    NA
## 5         2.53           NA           NA                    NA
## 6         2.47           92           72        8.2        178
##              Developer Rating
## 1           EA Tiburon      E
## 2            EA Canada      E
## 3      Traveller's Tales   E10+
## 4
## 5
## 6 Harmonix Music Systems      T
```

It seems that there's a pattern for some Names where the year of release is written in the end of the Name. So we can extract the year from such Names:

```
pattern <- "\\s(\\d{4})$"
for (i in which(y_data$Name %in% str_subset(y_data$Name, pattern)))
  y_data[i, 3] <- as.numeric(str_sub(y_data[i, 1],-4,-1))
```

The rest of the missing years were populated in accordance with the Internet data.

```
## Joining, by = c("Name", "Platform", "Year_of_Release", "Genre", "Publisher", "NA_Sales", "EU_Sales",

##                       Name Platform Year_of_Release        Genre Publisher
## 1               Wii Sports      Wii            2006       Sports  Nintendo
## 2          Super Mario Bros.      NES            1985     Platform  Nintendo
## 3            Mario Kart Wii      Wii            2008       Racing  Nintendo
## 4          Wii Sports Resort      Wii            2009       Sports  Nintendo
## 5 Pokemon Red/Pokemon Blue       GB            1996 Role-Playing  Nintendo
## 6                     Tetris       GB            1989       Puzzle  Nintendo
##   NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales Critic_Score Critic_Count
## 1    41.36    28.96     3.77        8.45        82.53           76           51
## 2    29.08     3.58     6.81        0.77        40.24           NA           NA
## 3    15.68    12.76     3.79        3.29        35.52           82           73
## 4    15.61    10.93     3.28        2.95        32.77           80           73
## 5    11.27     8.89    10.22        1.00        31.37           NA           NA
## 6    23.20     2.26     4.22        0.58        30.26           NA           NA
##   User_Score User_Count Developer Rating
## 1          8        322  Nintendo      E
## 2                    NA
## 3        8.3        709  Nintendo      E
## 4          8        192  Nintendo      E
## 5                    NA
## 6                    NA
```

Now let's check empty values in other columns. As we saw earlier, there are some NAs in Critic Scores:

```
data %>% filter(is.na(Critic_Score)) %>% nrow()
```

```
## [1] 4000
```

And an empty record in Genre:

```
data %>% filter(Genre == "") %>% nrow()
```

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

We will delete these records, even though the dataset will significantly shrink (for now we will consider it more appropriate than, say, mean imputation):

```
data <- data %>% filter(Genre != "" & !is.na(Critic_Score))
```

Further, we will consolidate the Platform information into its Producer, making destinction between Sony, Nintendo, Microsoft vs the rest (the diversity in Platform may lead to the situation when we don't have some values in the training set as opposed to the test set):

```
producer <- function(x) {
  if(x %in% c("PS", "PS2", "PS3", "PS4", "PSP", "PSV") == TRUE) {return("Sony")}
  else if(x %in% c("3DS", "DS", "GBA", "GC", "N64", "Wii", "WiiU") == TRUE) {return("Nintendo")}
  else if(x %in% c("PC", "X360", "XB", "XOne") == TRUE) {return("Microsoft")}
  else {return("Other")}
  }
data$Producer <- sapply(data$Platform, producer)
```

Aim

The aim of the project is to make a video games bestseller prediction on the European Market. For this purpose, we need to define a Bestseller and add the corresponding column.

Bestseller is a video game with more that half a million copies sold on the European market. The dataset Bestseller column will have two values - either it's a Bestseller or a Regular game.

```
bestseller <-function(s){ifelse(s >= 0.5, "Bestseller", "Regular")}
data$Bestseller <- sapply(data$EU_Sales, bestseller)
```

In the current project the predictors we will restricted to Critic_Score, Genre, Year_of_Release, and Producer. Each model will be trained independently, and the results will be combined.

Thus, we will try Generalised Linear Model (GLM), kNN Model, Classification Tree Model (CART), Random Forest Model, as well as test their Ensemble. The results will be discussed in the Result section.

Before splitting the dataset into the training and test sets, let's get rid of the unused columns and rename Year_of_Release and EU_Sales for the sake of simplicity:

```
data <- data %>% select(-NA_Sales, -JP_Sales, -Other_Sales, -Global_Sales, -Critic_Count, -User_Score, 
data <- data %>% rename(Year = Year_of_Release, Sales = EU_Sales)
```

The models will be fitted using only the training set. Once a model is done, it will be evaluated using the test set. The evaluaton criterion is defined as the proportion of cases that were correctly predicted as Bestsellers in the test set, i.e. overall accuracy. We will also explore the weighted average of Precision and Recall, F1 score, that takes both false positives and false negatives into account. Accuracy works best if false positives and false negatives have similar cost, but we have quite an uneven class distribution, even though we care about Sensitivity more (obviously, there are fewer bestsellers than regular games).

The dataset now has

```
## [1] 6845
```

rows. If we take 20% of the data as the test set, statistically, it should be Ok in terms of underfitting and variance overfitting.

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = data$Bestseller, times = 1, p = 0.2, list = FALSE)
train_set <- data[-test_index,]
test_set <- data[test_index,]
```

Now that the train and test sets are clean and tidy, we are ready and steady to explore the values.

Data Analysis and Models

Data Analysis

Overview

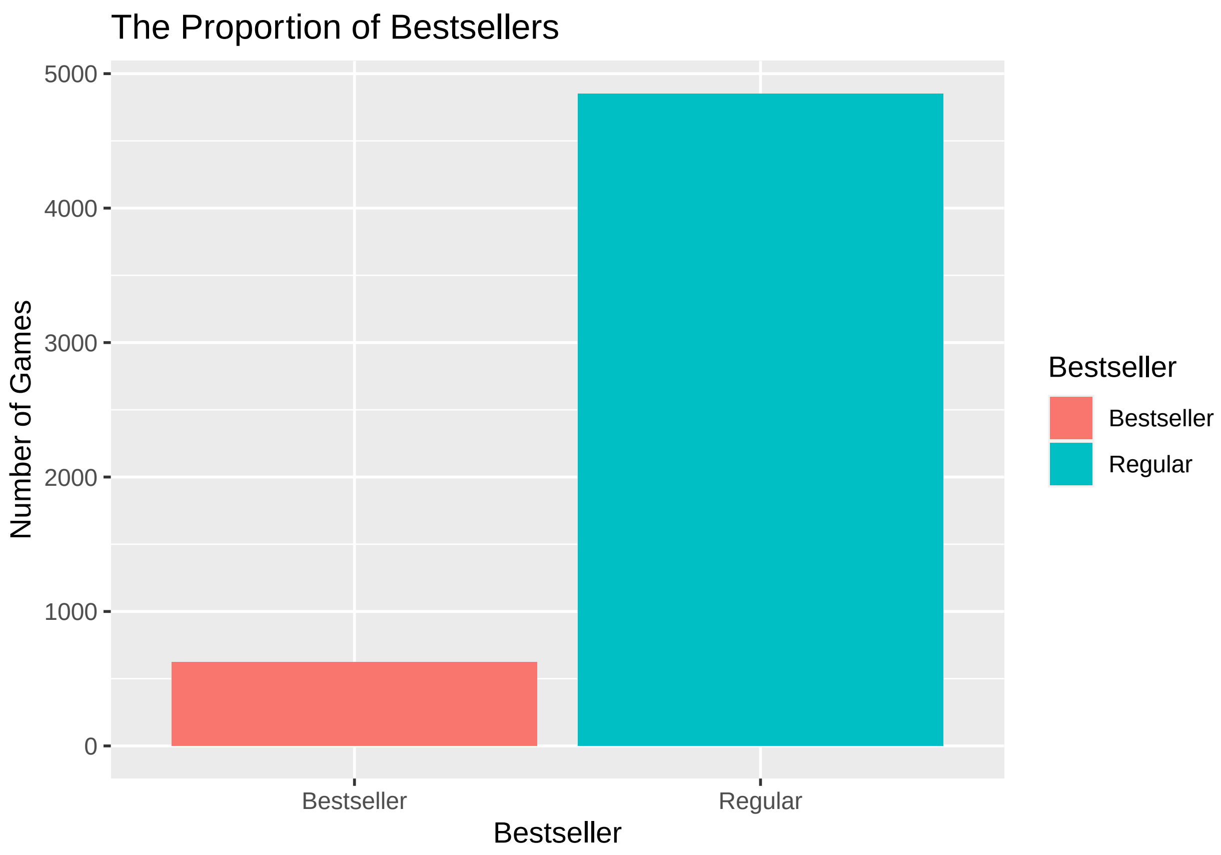
Here are the numbers of unique values in some columns:

```
##   n_games n_genre n_producer
## 1    3681      12          4
```

Judging by n_games, some games, apparently, differ by Name + Platform, rather than have a unique record in the Name column. So we may have duplicates. Let's check them.

```
##                      Name Year   Genre Sales Critic_Score  Producer Bestseller
## 1                  NHL 07 2006  Sports  0.02           76      Sony    Regular
## 2 Clive Barker's Jericho 2007 Shooter  0.01           63 Microsoft    Regular
## 3 Clive Barker's Jericho 2007 Shooter  0.01           60      Sony    Regular
## 4                  NHL 07 2006  Sports  0.02           76      Sony    Regular
## 5 Colin McRae Rally 2005 2004  Racing  0.01           83 Microsoft    Regular
## 6                  NHL 07 2006  Sports  0.01           75 Microsoft    Regular
## 7 Colin McRae Rally 2005 2004  Racing  0.01           83 Microsoft    Regular
## 8 Clive Barker's Jericho 2007 Shooter  0.01           63 Microsoft    Regular
```

All of them are Regular, so the duplicates with united Platforms will not mess up the prediction. Let's see what the proportion of Bestsellers in the training set:

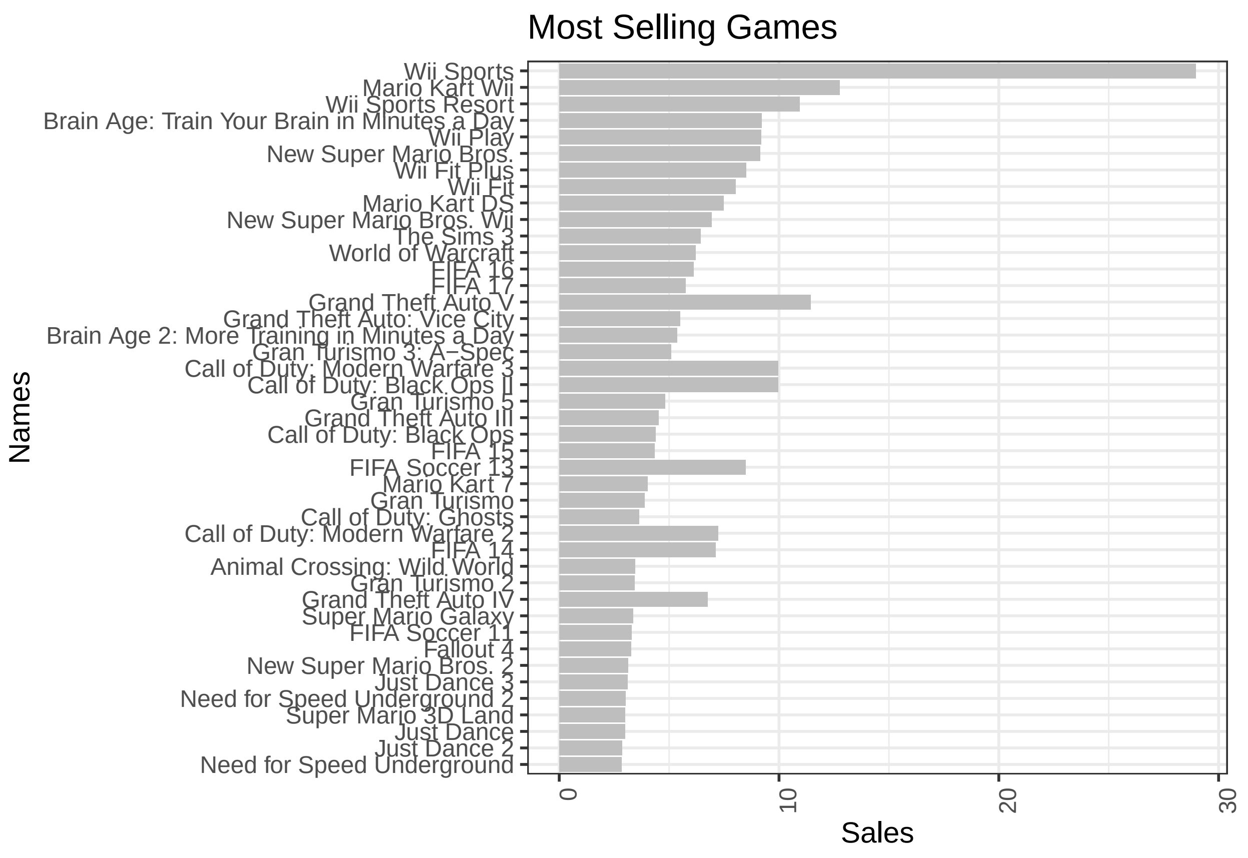


```
## [1] 0.1139518
```

There are about 11% of Bestsellers.

Sales Data

The general plot of 50 top-selling games reveals that there is one uncconditional outlier, Wii Sports.

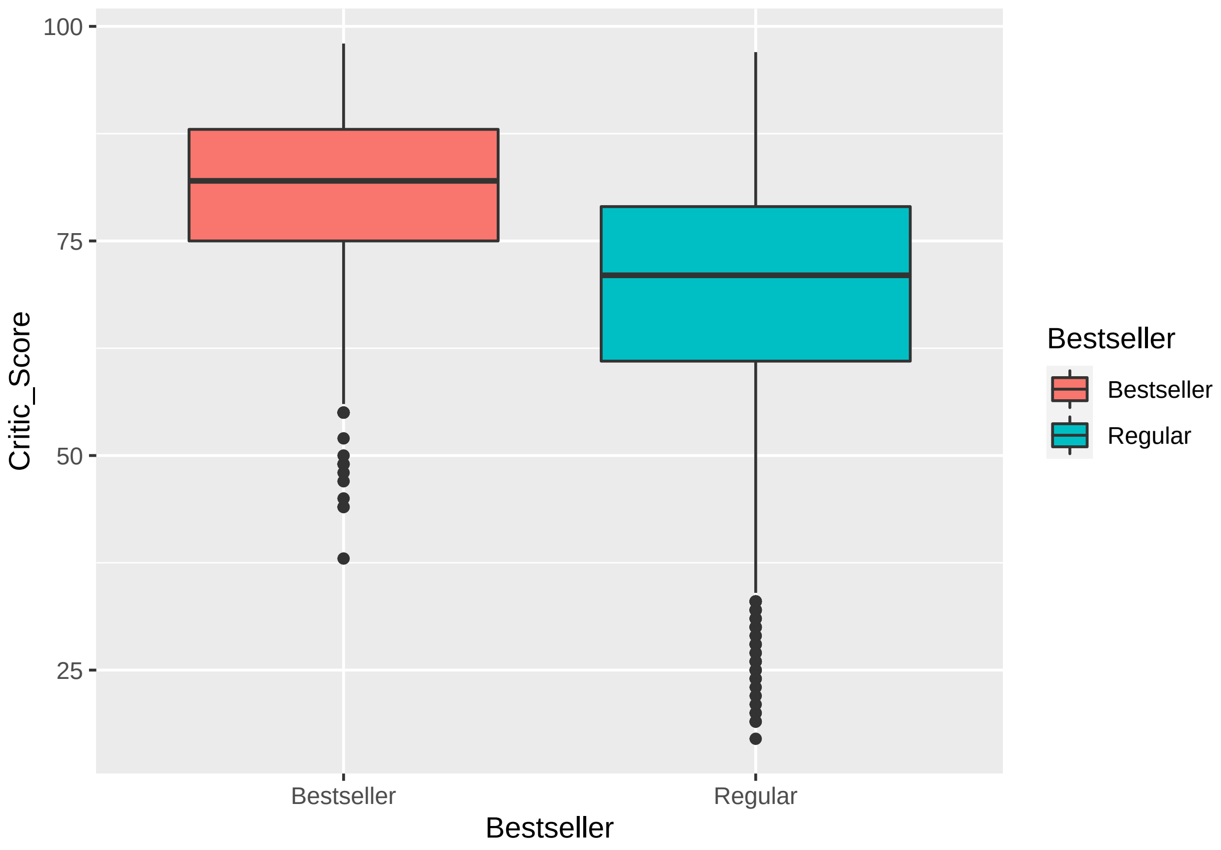


It will bias the models, so we will remove it.

```
train_set <- train_set %>% filter(Sales < 20)
```

Critical Score Data

Our intuition says that users must rely on the critics' opinion when bying games. Let's check if it's true.



There are indeed more higher scores among game bestsellers. Yet because of the intersection we cannot build a decision rule based only on the Critic Score.

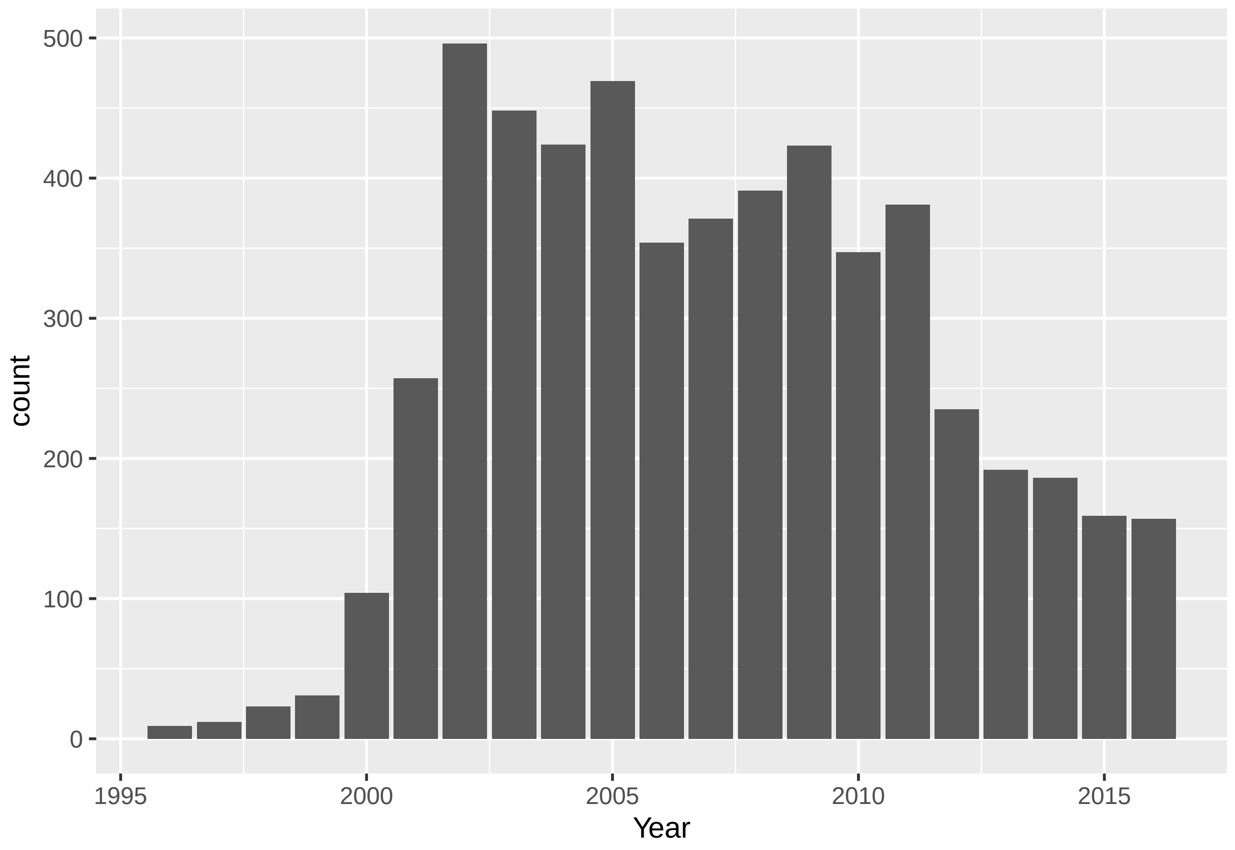
Year Data

Let's have a look at the table of years:

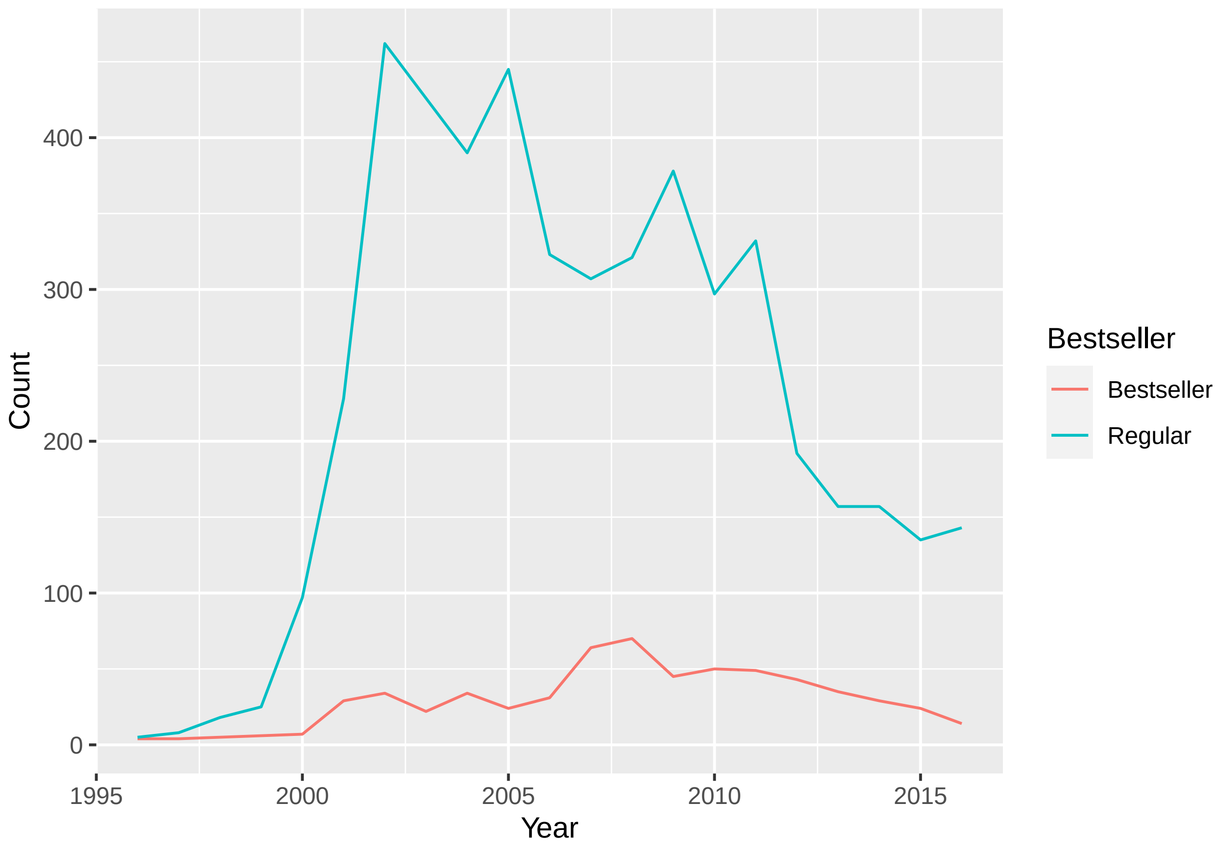
```
##
## 1984 1988 1991 1994 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007
##    1    2    1    1    9   12   23   31  104  257  496  448  424  469  354  371
## 2008 2009 2010 2011 2012 2013 2014 2015 2016
##  391  423  347  381  235  192  186  159  157
```

There are very few games before 1996. We will disregard them as outliers to avoid bias.

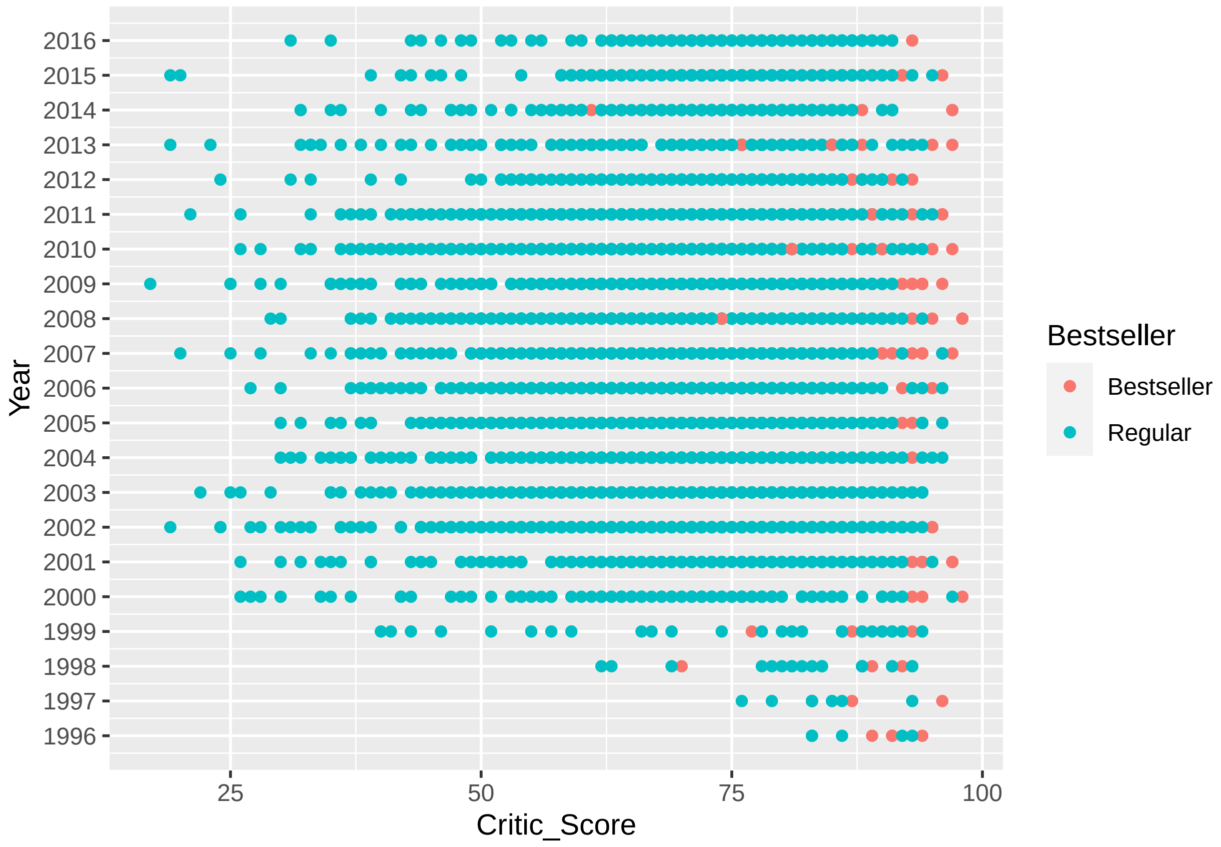
The Year - Game distribution is now as follows.



There are more games in the recent years, but are there more Bestsellers?



Most bestsellers happened in 2006-2010. Let's see if Critic_Score and Year together would be enough for prediction.

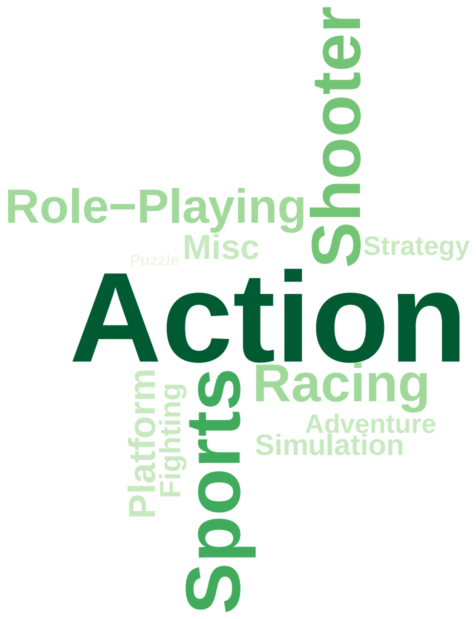


Generally, the higher the Critic score, the more chances that the game is a bestseller. And there are more bestsellers after 2007. This is a solid insight, but we have such a small train set that it can prove wrong on any bigger data. We will try to use more parameters.

Genre Data

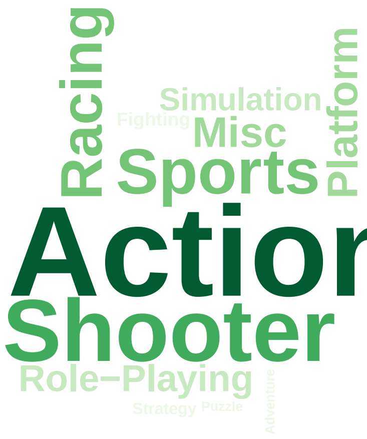
Let's compare wordclouds for genres:

All Genres by Count



There are more Action games and fewer Puzzles.

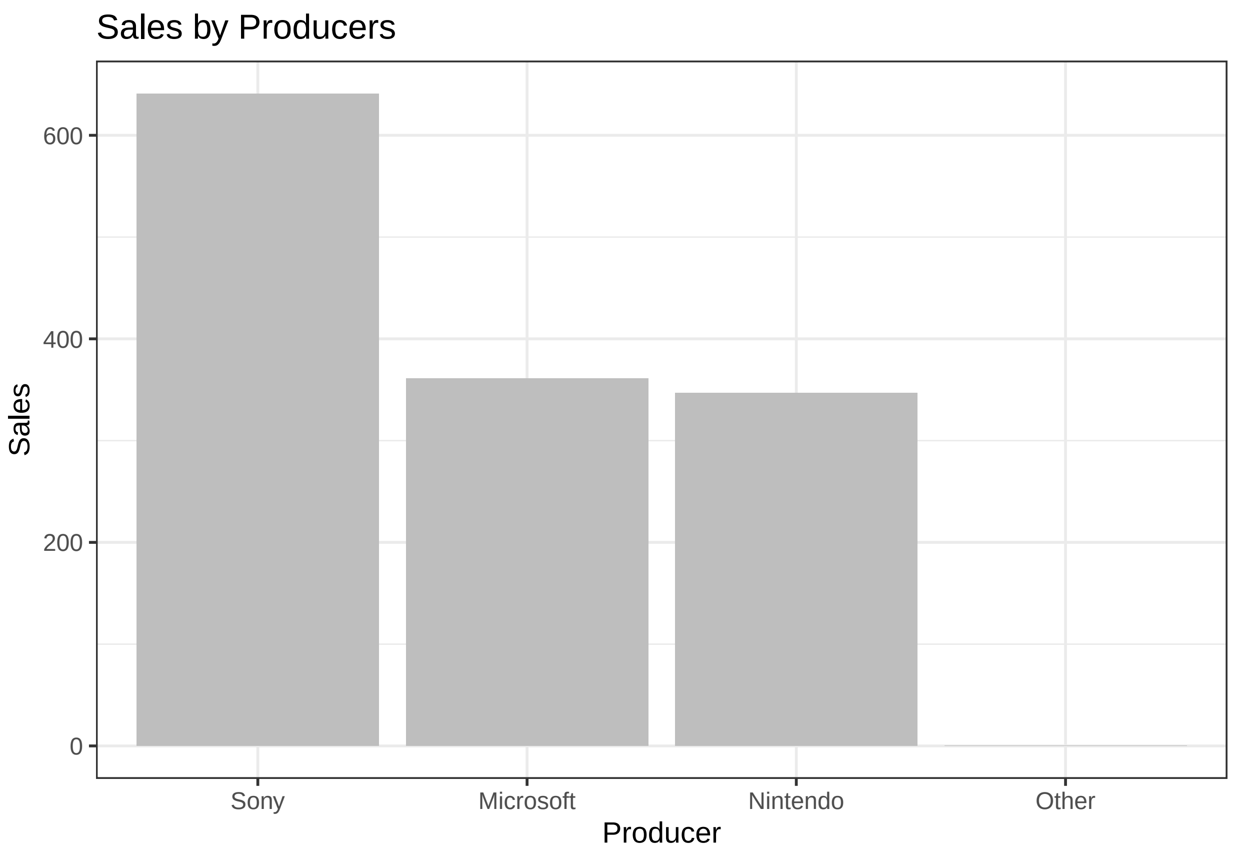
Bestseller Genres by Count



Puzzles don't seem of any importance whatsoever. The wordclouds are overall very similar. We'll check it with the models, but it seems that Genres don't drive a significant distinction between bestsellers and regular sales.

Producer Data

The Producer - Sales bar plot shows that Sony Platforms are more popular. Microsoft and Nintendo ones are about the same in terms of sales.



We will get rid of Other Producers due to the lack of data on them.

Models

We will use the caret package for fitting all models, so it's better to have factor outcomes between 1 and 0. Plus it's easier to calculate Accuracies as means.

Logistic Regression Model

The regression model adapted to logical outcomes is the simpliest one. We will start with it and see if other models will beat its results.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1208  145
##          1    5   11
##
##                Accuracy : 0.8904
##                  95% CI : (0.8727, 0.9065)
##     No Information Rate : 0.886
##     P-Value [Acc > NIR] : 0.323
##
##                   Kappa : 0.109
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99588
##             Specificity : 0.07051
##          Pos Pred Value : 0.89283
##          Neg Pred Value : 0.68750
##              Prevalence : 0.88605
##          Detection Rate : 0.88240
##    Detection Prevalence : 0.98831
##       Balanced Accuracy : 0.53320
##
##        'Positive' Class : 0
##
```

So the accuracy of the model is

```
## Accuracy
## 0.890431
```

which is quite high, but the F1 Score is almost in the middle:

```
## Balanced Accuracy
##         0.5331954
```

This model has very low Specificity.

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

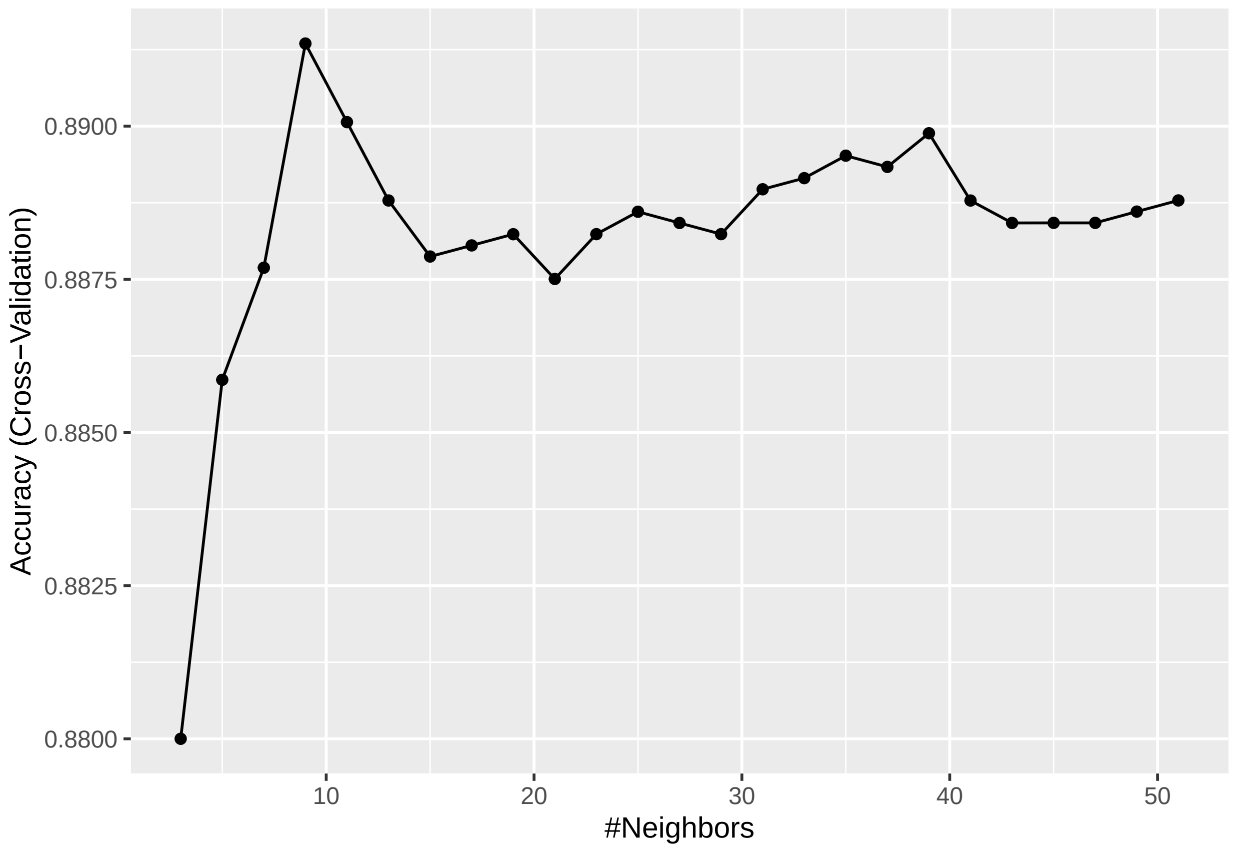
kNN model

For a kNN model, we compute the distance between each observation. This computation can take long time, so we will adjust it with 10-fold cross validation (through trainControl). Thus, we will have we will have 10 samples using 10% of the observations each.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

Best k is:

```
##   k
## 4 9
```



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1196  132
##          1   17   24
##
##                Accuracy : 0.8912
##                  95% CI : (0.8735, 0.9072)
##     No Information Rate : 0.886
##     P-Value [Acc > NIR] : 0.2928
##
##                   Kappa : 0.206
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9860
##             Specificity : 0.1538
##          Pos Pred Value : 0.9006
##          Neg Pred Value : 0.5854
##              Prevalence : 0.8860
##          Detection Rate : 0.8736
##    Detection Prevalence : 0.9701
##       Balanced Accuracy : 0.5699
##
##        'Positive' Class : 0
##
```

Let's compare the models:

```
## # A tibble: 2 x 3
##   Model                     Accuracy F1_Score
##   <chr>                     <chr>    <chr>
## 1 Generalised Linear Model 0.890431 0.533195
## 2 kNN Model                0.891161 0.569916
```

So far so good. With the kNN Model we have better Accuracy and better F1 Score.

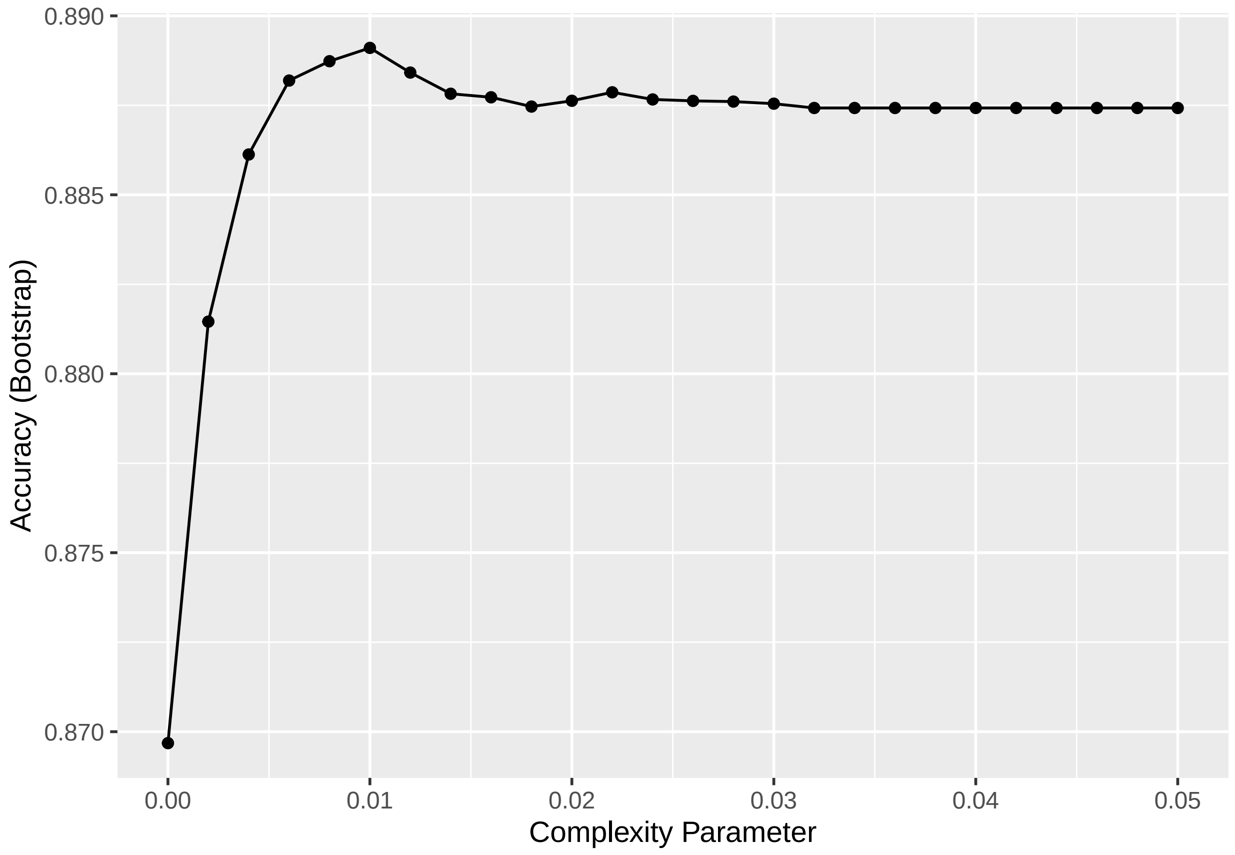
Classification Tree Model

We will apply cross-validation here to avoid too many partitions so that the model would adapt to the training data.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

The optimal value of cp is:

```
##     cp
## 6 0.01
```



Let's have a look at the final decision tree:

```
## n= 5467
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 5467 623 0 (0.88604353 0.11395647)
##    2) Critic_Score< 78.5 3802 212 0 (0.94423987 0.05576013) *
##    3) Critic_Score>=78.5 1665 411 0 (0.75315315 0.24684685)
##      6) Critic_Score< 92.5 1559 348 0 (0.77677999 0.22322001) *
##      7) Critic_Score>=92.5 106  43 1 (0.40566038 0.59433962)
##       14) Year< 2006.5 55  24 0 (0.56363636 0.43636364) *
##       15) Year>=2006.5 51  12 1 (0.23529412 0.76470588) *
```

The most important (root) variable here is Critic Score:

```
## rpart variable importance
##
##                    Overall
## Critic_Score        100.000
## Year                 32.963
## ProducerSony         20.323
## GenreShooter          7.111
## GenreSports           5.860
## GenreAdventure        2.027
## GenrePuzzle           0.000
## GenreSimulation       0.000
## GenrePlatform         0.000
## GenreStrategy         0.000
## GenreRacing           0.000
## ProducerNintendo      0.000
## GenreFighting         0.000
## GenreMisc             0.000
## `GenreRole-Playing`   0.000
```

And Genre (as we suspected) does not have much weight. The Confusion Matrix along with the results are as follows:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1210  151
##          1    3    5
##
##                Accuracy : 0.8875
##                  95% CI : (0.8696, 0.9038)
##     No Information Rate : 0.886
##     P-Value [Acc > NIR] : 0.4535
##
##                   Kappa : 0.0504
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99753
##             Specificity : 0.03205
##          Pos Pred Value : 0.88905
##          Neg Pred Value : 0.62500
##              Prevalence : 0.88605
##          Detection Rate : 0.88386
##    Detection Prevalence : 0.99416
##       Balanced Accuracy : 0.51479
##
```

```
##        'Positive' Class : 0
##
```

Let's check the results:

```
## # A tibble: 3 x 3
##   Model                     Accuracy F1_Score
##   <chr>                     <chr>    <chr>
## 1 Generalised Linear Model  0.890431 0.533195
## 2 kNN Model                 0.891161 0.569916
## 3 Classification Tree Model 0.887509 0.514789
```

Classification Tree Model turned out to be worse than kNN. Can we improve it with Random Forest?

Random Forest Model

For Random Forest Models fitting is the slowest part, so we will only 5-fold cross validation. We will also try to maximise accuracy with mtry. Like with the Classification Tree Model, we will restrict growth by small nodesize. Plus the ntree parameter restricts the number of trees for the sake of faster computation. So the model is as follows:

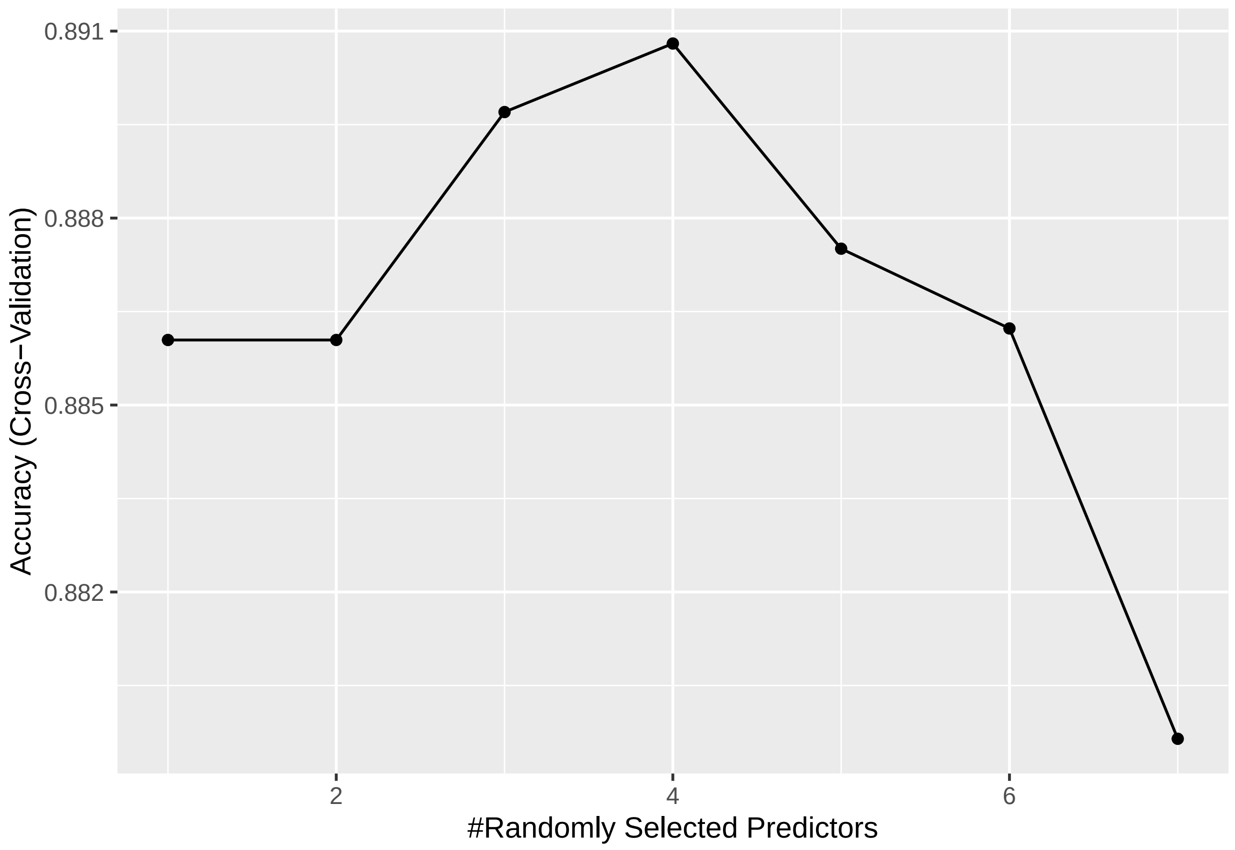
```r
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
control <- trainControl(method="cv", number = 5)
fit_rf <- train(Bestseller ~ Critic_Score + Genre + Year + Producer, data = train_set,
                method = "rf",
                ntree = 150, nodesize = 1,
                tuneGrid = data.frame(mtry = seq(1:7)),
                trControl = control)
```

The best mtry turns out:

```
##   mtry
## 4    4
```



Veriable importance:

```
## rf variable importance
##
##                      Overall
## Critic_Score      100.0000
## Year               43.3958
## ProducerSony        7.5437
## GenreMisc           4.6833
## GenreShooter        3.8216
## GenreSports         3.7516
## ProducerNintendo    3.4432
## GenreRacing         3.4183
## GenrePlatform       3.0050
## GenreRole-Playing   2.3076
## GenreSimulation     1.7015
## GenreFighting       1.2738
## GenreStrategy       0.4776
## GenreAdventure      0.4276
## GenrePuzzle         0.0000
```

And the Confusion Matrix:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1202  141
##          1   11   15
```

```
##
##                Accuracy : 0.889
##                  95% CI : (0.8711, 0.9051)
##     No Information Rate : 0.886
##     P-Value [Acc > NIR] : 0.3868
##
##                   Kappa : 0.1367
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99093
##             Specificity : 0.09615
##          Pos Pred Value : 0.89501
##          Neg Pred Value : 0.57692
##              Prevalence : 0.88605
##          Detection Rate : 0.87801
##    Detection Prevalence : 0.98101
##       Balanced Accuracy : 0.54354
##
##        'Positive' Class : 0
##
```

The overall result:

```
## # A tibble: 4 x 3
##   Model                     Accuracy F1_Score
##   <chr>                     <chr>    <chr>
## 1 Generalised Linear Model  0.890431 0.533195
## 2 kNN Model                 0.891161 0.569916
## 3 Classification Tree Model 0.887509 0.514789
## 4 Random Forest Model       0.888970 0.543543
```

The accuracy is very good. Will Ensemble be able to improve it?

Ensemble

After comparing different combinations of all 4 models, the best result was achieved by combining KNN and Random Forest models (as they are both have the highest accuracy). The final result is as follows:

```r
p_knn <- predict(fit_knn, test_set, type = "prob")
p_rf <- predict(fit_rf, test_set, type = "prob")
p_rf <- p_rf/rowSums(p_rf)
p <- (p_knn + p_rf)/2
pred <- factor(apply(p, 1, which.max)-1)
cm <- confusionMatrix(pred, as.factor(test_set$Bestseller))
acc <- cm$overall["Accuracy"]
bacc <- cm$byClass["Balanced Accuracy"]
results <- bind_rows(results,
                     data_frame(Model = "KNN + RF Ensemble",
                                Accuracy = format(round(acc, 6), nsmall = 6),
                                F1_Score = format(round(bacc, 6), nsmall = 6)))
results
```

```
## # A tibble: 5 x 3
##   Model                     Accuracy F1_Score
##   <chr>                     <chr>    <chr>
## 1 Generalised Linear Model  0.890431 0.533195
## 2 kNN Model                 0.891161 0.569916
## 3 Classification Tree Model 0.887509 0.514789
## 4 Random Forest Model       0.888970 0.543543
## 5 KNN + RF Ensemble         0.893353 0.554395
```

Results

Not surprisingly, the best accuracy is achieved with ensemble of the two models out of the three best performing ones. The balanced Precision-Recall accuracy is at its highest in the kNN Model, and it didn't come much closer to 1. At the same time, F1 Score was used for monitoring.

Our dataset is disbalanced by its nature, so it's quite predictable that there could be many cases when a Regular video game will be called a Bestseller (there are many Developers and Publishers who would love to believe in it:))

Conclusion

By the Year-Critic_Score plot it was more or less predictable that the logistic regression could give a good result (it somehow resembles a regression line). With the best result achieved using kNN and Ensemble, it's worth trying other models. Yet, using any other more advanced machine learning approach one should be aware of the complexity of computations.

In this project the classification was limited to using only Producer, Year, Genre, and Critic Score. There could be further enhancements of the models with adding Rating, Developer, and Publisher values. Definitely, exploring User Score would give more insights, but here it was excluded on purpose, as we cannot have these data for the games before their release.

As we saw, before 1996 there were very few observations. We excluded it because we didn't want to have the unnesesssary shift. In the random forest model I was experimenting with adding more weight to recent years, but it didn't work out for the better accuracy.

Also, if there were more observations in the dataset the result would be better. Empty Critic Scores could be filled with a mean imputation (for instance, using Genre and/or Producer/Platform). Rating could potentially have some effect (as the video games for broader audience have more chances to become a bestseller).

Futhermore, the data of the initial data file cuts off at 2016. Perhaps, for the time being more data can be scrapped.