Video Games Sales ML

Nik Pevnev

2025-03-25

Intro

Video Games Sales with Ratings project is based on creating a predictive machine learning algorithm trained on provided video games sales data and RMSE tested with train set to predict game's global sales. Final deliverables of the project will be: .Rmd file, .DF document, and an R script. Data is based on ML friendly Kaggle data source: https://www.kaggle.com/datasets/rush4ratio/video-game-sales-with-ratings

Datasets

I am going to perform data set checks to make sure data is pre-processed correctly

```
# (1) Check raw data from Kaggle
glimpse(games)
```

```
## Rows: 16,719
## Columns: 16
## $ Name            <chr> "Wii Sports", "Super Mario Bros.", "Mario Kart Wii", "~
## $ Platform        <chr> "Wii", "NES", "Wii", "Wii", "GB", "GB", "DS", "Wii", "~
## $ Year_of_Release <chr> "2006", "1985", "2008", "2009", "1996", "1989", "2006"~
## $ Genre           <chr> "Sports", "Platform", "Racing", "Sports", "Role-Playin~
## $ Publisher       <chr> "Nintendo", "Nintendo", "Nintendo", "Nintendo", "Ninte~
## $ NA_Sales        <dbl> 41.36, 29.08, 15.68, 15.61, 11.27, 23.20, 11.28, 13.96~
## $ EU_Sales        <dbl> 28.96, 3.58, 12.76, 10.93, 8.89, 2.26, 9.14, 9.18, 6.9~
## $ JP_Sales        <dbl> 3.77, 6.81, 3.79, 3.28, 10.22, 4.22, 6.50, 2.93, 4.70,~
## $ Other_Sales     <dbl> 8.45, 0.77, 3.29, 2.95, 1.00, 0.58, 2.88, 2.84, 2.24, ~
## $ Global_Sales    <dbl> 82.53, 40.24, 35.52, 32.77, 31.37, 30.26, 29.80, 28.92~
## $ Critic_Score    <int> 76, NA, 82, 80, NA, NA, 89, 58, 87, NA, NA, 91, NA, 80~
## $ Critic_Count    <int> 51, NA, 73, 73, NA, NA, 65, 41, 80, NA, NA, 64, NA, 63~
## $ User_Score      <chr> "8", "", "8.3", "8", "", "", "8.5", "6.6", "8.4", "", ~
## $ User_Count      <int> 322, NA, 709, 192, NA, NA, 431, 129, 594, NA, NA, 464,~
## $ Developer       <chr> "Nintendo", "", "Nintendo", "Nintendo", "", "", "Ninte~
## $ Rating          <chr> "E", "", "E", "E", "", "", "E", "E", "E", "", "", "E",~
```

```
# We have 16,719 rows and 16 columns across dataset as noted in glimpse(games)

# (2) Review stats for each field
summary(games)
```

```
##      Name              Platform         Year_of_Release       Genre
##  Length:16719       Length:16719       Length:16719       Length:16719
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   Publisher            NA_Sales          EU_Sales         JP_Sales
##  Length:16719       Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  Class :character   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Mode  :character   Median : 0.0800   Median : 0.020   Median : 0.0000
##                     Mean   : 0.2633   Mean   : 0.145   Mean   : 0.0776
##                     3rd Qu.: 0.2400   3rd Qu.: 0.110   3rd Qu.: 0.0400
##                     Max.   :41.3600   Max.   :28.960   Max.   :10.2200
##
##   Other_Sales        Global_Sales      Critic_Score    Critic_Count
##  Min.   : 0.00000   Min.   : 0.0100   Min.   :13.00   Min.   :  3.00
##  1st Qu.: 0.00000   1st Qu.: 0.0600   1st Qu.:60.00   1st Qu.: 12.00
##  Median : 0.01000   Median : 0.1700   Median :71.00   Median : 21.00
##  Mean   : 0.04733   Mean   : 0.5335   Mean   :68.97   Mean   : 26.36
##  3rd Qu.: 0.03000   3rd Qu.: 0.4700   3rd Qu.:79.00   3rd Qu.: 36.00
##  Max.   :10.57000   Max.   :82.5300   Max.   :98.00   Max.   :113.00
##                                       NA's   :8582    NA's   :8582
##   User_Score          User_Count       Developer            Rating
##  Length:16719       Min.   :    4.0   Length:16719       Length:16719
##  Class :character   1st Qu.:   10.0   Class :character   Class :character
##  Mode  :character   Median :   24.0   Mode  :character   Mode  :character
##                     Mean   :  162.2
##                     3rd Qu.:   81.0
##                     Max.   :10665.0
##                     NA's   :9129
```

```
# Summary of raw dataset reveals a big number of NAs in Critic_Score,
# Critic_Count, and User_Count columns that need to pre-processed
# for successful ML run

# (3) We also want to research :character fields, in particular Platforms
# on which games are built.
games %>%
  group_by(Name, Publisher, Platform) %>%
  count(sort = TRUE)
```

```
## # A tibble: 16,715 x 4
## # Groups:   Name, Publisher, Platform [16,715]
##    Name                           Publisher                  Platform     n
##    <chr>                          <chr>                      <chr>    <int>
##  1 ""                             Acclaim Entertainment      GEN          2
##  2 "Madden NFL 13"                Electronic Arts            PS3          2
##  3 "Need for Speed: Most Wanted"  Electronic Arts            PC           2
##  4 "Need for Speed: Most Wanted"  Electronic Arts            X360         2
##  5 " Beyblade Burst"              FuRyu                      3DS          1
##  6 " Fire Emblem Fates"           Nintendo                   3DS          1
##  7 " Frozen: Olaf's Quest"        Disney Interactive Studios 3DS          1
```

```
##  8 " Frozen: Olaf's Quest"       Disney Interactive Studios DS           1
##  9 " Haikyu!! Cross Team Match!" Namco Bandai Games         3DS          1
## 10 " Tales of Xillia 2"          Namco Bandai Games         PS3          1
## # i 16,705 more rows
```

```
# We see that GG and PCFX are only counted once, meaning training data set might
# not have all of the feature values, hence we can remove these 2 rows
# This is to avoid issues with train / test data separation

# (4) Check User_Score field values to see what distinct values it has
games %>%
  distinct(User_Score) %>%
  filter(!grepl("^\\d*\\.?\\d+$", as.character(User_Score))) %>%
  arrange(User_Score)
```

```
##   User_Score
## 1
## 2        tbd
```

```
# We need to be careful with "" and "tbd" values in the dataset
```

Data Cleaning

I am going to perform data cleaning exercises to prepare data sets to have relevant, complete, accurate data for ML to traing and predict correctly

```
# (1) Data cleaning and preparation where character fields are transformed into
# a factor and numeric values are continuous
games_clean <- games %>%
  mutate(
    Year_of_Release = as.numeric(ifelse(grepl("^[0-9]+$"
                                      , Year_of_Release)
                                      , Year_of_Release, NA)),
    User_Score =
        as.numeric(ifelse(User_Score
        %in% c("", "NA", "tbd")
        , NA, User_Score)), # Handle other non-numeric values
    Rating = factor(Rating),
    Genre = factor(Genre),
    Platform = factor(Platform)
  ) %>%
  filter(!is.na(Global_Sales)) %>%  # Remove rows with missing target variable
  filter(!Platform %in% c("GG", "PCFX")) %>% # Remove unwanted platforms
  filter(!Rating %in% c("AO", "RP", "K-A")) # Remove unwanted rating

# (2) Remove rows with NA in clean data set
games_clean <- na.omit(games_clean)

# (3) Update rare publishers to "Other" if they have <= 25 games
publisher_count <- table(games_clean$Publisher)
rare_publishers <- names(publisher_count[publisher_count <= 25])
```

```r
games_clean <- games_clean %>%
  mutate(Publisher = ifelse(Publisher %in% rare_publishers, "Other", Publisher))
games_clean$Publisher <- as.factor(games_clean$Publisher) # Convert to factor

# (4) Check Rating field values to see what distinct values it has
games_clean %>%
  group_by(Rating) %>%
  count(sort = TRUE)
```

```
## # A tibble: 5 x 2
## # Groups:   Rating [5]
##   Rating     n
##   <fct>  <int>
## 1 "T"     2378
## 2 "E"     2082
## 3 "M"     1433
## 4 "E10+"   930
## 5 ""        68
```

```r
# =============== CLEAN DATA ===============
games_clean <- games_clean %>%
  filter(!Rating %in% c("AO", "RP", "K-A")) # Remove unwanted rating
# We see that AO, K-A, RP are only counted once, meaning training data set might
# not have all of the feature values, hence we can remove these ratings
# This is to avoid issues with train / test data separation

# (5) Check the new data structure of the cleaned data
glimpse(games_clean)
```

```
## Rows: 6,891
## Columns: 16
## $ Name            <chr> "Wii Sports", "Mario Kart Wii", "Wii Sports Resort", "~
## $ Platform        <fct> Wii, Wii, Wii, DS, Wii, Wii, DS, Wii, X360, Wii, PS3, ~
## $ Year_of_Release <dbl> 2006, 2008, 2009, 2006, 2006, 2009, 2005, 2007, 2010, ~
## $ Genre           <fct> Sports, Racing, Sports, Platform, Misc, Platform, Raci~
## $ Publisher       <fct> Nintendo, Nintendo, Nintendo, Nintendo, Nintendo, Nint~
## $ NA_Sales        <dbl> 41.36, 15.68, 15.61, 11.28, 13.96, 14.44, 9.71, 8.92, ~
## $ EU_Sales        <dbl> 28.96, 12.76, 10.93, 9.14, 9.18, 6.94, 7.47, 8.03, 4.8~
## $ JP_Sales        <dbl> 3.77, 3.79, 3.28, 6.50, 2.93, 4.70, 4.13, 3.60, 0.24, ~
## $ Other_Sales     <dbl> 8.45, 3.29, 2.95, 2.88, 2.84, 2.24, 1.90, 2.15, 1.69, ~
## $ Global_Sales    <dbl> 82.53, 35.52, 32.77, 29.80, 28.92, 28.32, 23.21, 22.70~
## $ Critic_Score    <int> 76, 82, 80, 89, 58, 87, 91, 80, 61, 80, 97, 95, 77, 97~
## $ Critic_Count    <int> 51, 73, 73, 65, 41, 80, 64, 63, 45, 33, 50, 80, 58, 58~
## $ User_Score      <dbl> 8.0, 8.3, 8.0, 8.5, 6.6, 8.4, 8.6, 7.7, 6.3, 7.4, 8.2,~
## $ User_Count      <int> 322, 709, 192, 431, 129, 594, 464, 146, 106, 52, 3994,~
## $ Developer       <chr> "Nintendo", "Nintendo", "Nintendo", "Nintendo", "Ninte~
## $ Rating          <fct> E, E, E, E, E, E, E, E, E, E, M, M, E, M, M, E, E, M, ~
```

```r
summary(games_clean)
```

```
##      Name              Platform    Year_of_Release          Genre
##  Length:6891        PS2    :1140   Min.   :1985    Action      :1643
```

```
##  Class :character   X360   : 861   1st Qu.:2004    Sports      : 951
##  Mode  :character   PS3    : 775   Median :2007    Shooter     : 868
##                     PC     : 687   Mean   :2007    Role-Playing: 715
##                     XB     : 565   3rd Qu.:2011    Racing      : 586
##                     Wii    : 480   Max.   :2016    Platform    : 403
##                     (Other):2383                   (Other)     :1725
##                        Publisher       NA_Sales          EU_Sales
##  Electronic Arts            : 945   Min.   : 0.0000   Min.   : 0.0000
##  Other                      : 941   1st Qu.: 0.0600   1st Qu.: 0.0200
##  Ubisoft                    : 498   Median : 0.1500   Median : 0.0600
##  Activision                 : 492   Mean   : 0.3909   Mean   : 0.2345
##  Sony Computer Entertainment: 315   3rd Qu.: 0.3900   3rd Qu.: 0.2100
##  THQ                        : 307   Max.   :41.3600   Max.   :28.9600
##  (Other)                    :3393
##     JP_Sales        Other_Sales        Global_Sales      Critic_Score
##  Min.   :0.00000   Min.   : 0.00000   Min.   : 0.0100   Min.   :13.00
##  1st Qu.:0.00000   1st Qu.: 0.01000   1st Qu.: 0.1100   1st Qu.:62.00
##  Median :0.00000   Median : 0.02000   Median : 0.2900   Median :72.00
##  Mean   :0.06368   Mean   : 0.08202   Mean   : 0.7713   Mean   :70.25
##  3rd Qu.:0.01000   3rd Qu.: 0.07000   3rd Qu.: 0.7500   3rd Qu.:80.00
##  Max.   :6.50000   Max.   :10.57000   Max.   :82.5300   Max.   :98.00
##
##   Critic_Count      User_Score      User_Count       Developer
##  Min.   :  3.00   Min.   :0.500   Min.   :    4.0   Length:6891
##  1st Qu.: 14.00   1st Qu.:6.500   1st Qu.:   11.0   Class :character
##  Median : 24.00   Median :7.500   Median :   27.0   Mode  :character
##  Mean   : 28.84   Mean   :7.184   Mean   :  174.4
##  3rd Qu.: 39.00   3rd Qu.:8.200   3rd Qu.:   89.0
##  Max.   :113.00   Max.   :9.600   Max.   :10665.0
##
##      Rating
##  T      :2378
##  E      :2082
##  M      :1433
##  E10+   : 930
##         :  68
##  AO     :   0
##  (Other):   0
```

```r
# Now data looks better for ML processing :)
```

Data Preparation

I am going to separate data into train and test data sets

```r
# (1) Split while maintaining both distributions for Global Sales
set.seed(123)
trainIndex <- createDataPartition(games_clean$Global_Sales, p = 0.8
                                  , list = FALSE)
train_data <- games_clean[trainIndex, ]
test_data <- games_clean[-trainIndex, ]
```

```r
# (2) Review ML data distributions
glimpse(train_data) # 5515 rows and 16 columns
```

```
## Rows: 5,515
## Columns: 16
## $ Name            <chr> "Wii Sports", "Mario Kart Wii", "Wii Play", "New Super~
## $ Platform        <fct> Wii, Wii, Wii, Wii, DS, Wii, X360, Wii, DS, DS, PS2, X~
## $ Year_of_Release <dbl> 2006, 2008, 2006, 2009, 2005, 2007, 2010, 2009, 2005, ~
## $ Genre           <fct> Sports, Racing, Misc, Platform, Racing, Sports, Misc, ~
## $ Publisher       <fct> Nintendo, Nintendo, Nintendo, Nintendo, Nintendo, Nint~
## $ NA_Sales        <dbl> 41.36, 15.68, 13.96, 14.44, 9.71, 8.92, 15.00, 9.01, 4~
## $ EU_Sales        <dbl> 28.96, 12.76, 9.18, 6.94, 7.47, 8.03, 4.89, 8.49, 9.20~
## $ JP_Sales        <dbl> 3.77, 3.79, 2.93, 4.70, 4.13, 3.60, 0.24, 2.53, 4.16, ~
## $ Other_Sales     <dbl> 8.45, 3.29, 2.84, 2.24, 1.90, 2.15, 1.69, 1.77, 2.04, ~
## $ Global_Sales    <dbl> 82.53, 35.52, 28.92, 28.32, 23.21, 22.70, 21.81, 21.79~
## $ Critic_Score    <int> 76, 82, 58, 87, 91, 80, 61, 80, 77, 77, 95, 88, 83, 83~
## $ Critic_Count    <int> 51, 73, 41, 80, 64, 63, 45, 33, 58, 37, 54, 81, 21, 73~
## $ User_Score      <dbl> 8.0, 8.3, 6.6, 8.4, 8.6, 7.7, 6.3, 7.4, 7.9, 7.1, 8.4,~
## $ User_Count      <int> 322, 709, 129, 594, 464, 146, 106, 52, 50, 19, 314, 87~
## $ Developer       <chr> "Nintendo", "Nintendo", "Nintendo", "Nintendo", "Ninte~
## $ Rating          <fct> E, E, E, E, E, E, E, E, E, E, E, M, M, M, M, T, E, M, ~
```

```r
glimpse(test_data) # 1376 rows and 16 columns
```

```
## Rows: 1,376
## Columns: 16
## $ Name            <chr> "Wii Sports Resort", "New Super Mario Bros.", "Grand T~
## $ Platform        <fct> Wii, DS, PS3, PS2, X360, PS2, X360, X360, PS2, PS4, X3~
## $ Year_of_Release <dbl> 2009, 2006, 2013, 2004, 2013, 2002, 2010, 2009, 2001, ~
## $ Genre           <fct> Sports, Platform, Action, Action, Action, Action, Shoo~
## $ Publisher       <fct> Nintendo, Nintendo, Take-Two Interactive, Take-Two Int~
## $ NA_Sales        <dbl> 15.61, 11.28, 7.02, 9.43, 9.66, 8.41, 9.70, 8.52, 6.99~
## $ EU_Sales        <dbl> 10.93, 9.14, 9.09, 0.40, 5.14, 5.49, 3.68, 3.59, 4.51,~
## $ JP_Sales        <dbl> 3.28, 6.50, 0.98, 0.41, 0.06, 0.47, 0.11, 0.08, 0.30, ~
## $ Other_Sales     <dbl> 2.95, 2.88, 3.96, 10.57, 1.41, 1.78, 1.13, 1.28, 1.30,~
## $ Global_Sales    <dbl> 32.77, 29.80, 21.04, 20.81, 16.27, 16.15, 14.61, 13.47~
## $ Critic_Score    <int> 80, 89, 97, 95, 97, 95, 87, 94, 97, 97, 98, 92, 74, 88~
## $ Critic_Count    <int> 73, 65, 50, 80, 58, 62, 89, 100, 56, 66, 86, 20, 24, 7~
## $ User_Score      <dbl> 8.0, 8.5, 8.2, 9.0, 8.1, 8.7, 6.3, 6.3, 8.5, 8.3, 7.9,~
## $ User_Count      <int> 192, 431, 3994, 1588, 3711, 730, 1454, 2698, 664, 2899~
## $ Developer       <chr> "Nintendo", "Nintendo", "Rockstar North", "Rockstar No~
## $ Rating          <fct> E, E, M, M, M, M, M, M, M, M, M, T, E10+, E, E, M, M, ~
```

```r
# Now data looks good to be processed by ML algorithm
```

Machine Learning

I am going to use most efficient library in R I have found so far - h20. I use it to predict global sales of a game based on features like Genre, Rating, Platform, Publisher, and other factors. h2o is an extremely fast, distributed ML that recommends best algorithms such as Random Forests, Gradient Boosting Machines, and Deep Learning models.

```r
# (1) Initialize h2O to use all cores
h2o.init(nthreads = -1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 hours 1 minutes
##     H2O cluster timezone:       Etc/UTC
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.44.0.3
##     H2O cluster version age:    1 year, 3 months and 4 days
##     H2O cluster name:           H2O_started_from_R_vboxuser_vfs439
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.08 GB
##     H2O cluster total cores:    6
##     H2O cluster allowed cores:  6
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.4.3 (2025-02-28)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is (1 year, 3 months and 4 days) old. There may be a newer version available
## Please download and install the latest version from: https://h2o-release.s3.amazonaws.com/h2o/latest
```

```r
# (2) Convert train and test data sets to h2o frame
train_h2o <- as.h2o(train_data)
```

```
##   |                                                                      |
```

```r
valid_h2o <- as.h2o(test_data)
```

```
##   |                                                                      |
```

```r
# (3) Run AutoML to test multiple ML models quickly
aml <- h2o.automl(
  y = "Global_Sales",
  training_frame = train_h2o,
  max_runtime_secs = 300,  # 5 minute timeout
  seed = 123
)
```

```
##   |                                                                      |
## 15:07:54.612: _train param, Dropping bad and constant columns: [Developer, Name]  |
## 15:07:57.105: _train param, Dropping bad and constant columns: [Developer, Name]
## 15:07:57.201: _train param, Dropping bad and constant columns: [Developer, Name]  |
## 15:07:59.471: _train param, Dropping unused columns: [Developer, Name]
## 15:07:59.586: _train param, Dropping bad and constant columns: [Developer, Name]  |
## 15:08:01.492: _train param, Dropping bad and constant columns: [Developer, Name]  |
```

```
## 15:08:05.447: _train param, Dropping bad and constant columns: [Developer, Name]
## 15:08:06.215: _train param, Dropping bad and constant columns: [Developer, Name]  |
## 15:08:07.30: _train param, Dropping bad and constant columns: [Developer, Name]
## 15:08:08.156: _train param, Dropping unused columns: [Developer, Name]
## 15:08:08.281: _train param, Dropping unused columns: [Developer, Name]
## 15:08:08.399: _train param, Dropping bad and constant columns: [Developer, Name]  |
## 15:08:09.532: _train param, Dropping bad and constant columns: [Developer, Name]  |
## 15:08:14.549: _train param, Dropping bad and constant columns: [Developer, Name]  |
## 15:08:15.666: _train param, Dropping bad and constant columns: [Developer, Name]
## 15:08:16.284: _train param, Dropping unused columns: [Developer, Name]
## 15:08:16.399: _train param, Dropping unused columns: [Developer, Name]  |
```

```
# (4) Get best model from h20 analysis
best_model <- aml@leader

# (5) Predict data from test set based on ML Algorithm
h2o_pred <- predict(best_model, valid_h2o)
```

```
##   |                                                                              |
```

```
# (6) Calculate predictions based on test_data dataset and assess its RMSE
h2o_rmse <- RMSE(as.vector(h2o_pred), test_data$Global_Sales)
h2o_rmse # 0.005859286
```

```
## [1] 0.005859286
```

```
# RMSE returns 0.0059, which is good considered we are working with global sales
# prediction that are ranging in double digits, and mostly 0-1 range.
# This is a great prediction result for a data set from Kaggle to predict global
# sales of video games efficiently

##########################################################
######################### END ############################
##########################################################
```

Final RMSE of the project: 0.005859286