Predict Video Game Ratings - Project 1

Ahmed Tanveer

2024-02-11

What we doing?

In this project, I wanted to see if I could make an application that could predict what rating a game would get, based upon certain inputs. I would create this model prediction using Machine Learning in R by using example data to train a model. I want to preface that I have not used Machine Learning techniques before so there will be a lot of experimenting that will happen here today.

Let's Begin

First things first, let's import our data. I found the data we'll be using on Kaggle. I got inspiration for doing something relating to video game ratings from the sample projects provided and happened to stumble upon this dataset that would **hopefully** provide a beginner to Machine Learning like me good clean data to begin with. This dataset contains multiple columns with binary values about whether a game does or does not contain certain aspects (ex. hasDrugUse, hasAlchohol, hasSuggestiveThemes, hasBlood, etc.). It also lists the Title and actual ESRP rating for the games. With this information we should be able to train a model using Machine Learning to be able to accurately predict what rating a game will get based upon giving a model these inputs.

```
test_dataset <- read.csv("/Users/ahmed/Documents/DAT 301/Project 1/test_esrb.csv")
```

Choosing a Model?

It was a bit overwhelming to choose a Machine Learning Model with all the different possible options so I decided to start with the simplest sounding one, and change it if it didn't meet the requirements that I needed for this project. For me, it's easiest to jump in and try out different methods and see what works.

I began with Logistical Regression... Which I quickly learned would not work with my model. The issue here, was that Logistical Regression seemed to require the data values to be all binary; however, my *test_dataset$esrp_ratings* was a character output. I could make it into a binary value, but with having 3 different options for ESRP ratings, it seemed the better choice would be to choose a different model.

I then choose to use a Random Forest Model. I choose this model because of it's ease to implement and it appeared to be a degree more sophisticated in terms of what it could compute compared to Logistical Regression. In order to use this in R, I had to import the ***randomForest*** library. I also imported a couple other libraries that would prove to be useful in either the computations or analysis.

```
library(caret)
library(dplyr)
library(ggplot2)
library(randomForest)
```

Random Forest Modeling

To begin a Random Forest model, we prove the randomForest function the desired output to be evaluated, how many trees we want to compute (the more the better, but eats into computation time), and the dataset we have.

```
model <- randomForest(esrb_rating ~ ., ntree = 10000 ,data = test_dataset)
```

However, this threw many errors. There appeared to be messy data that for the most part got cleaned up by the following code. Additionally, we will be setting the seed here to account for variability and to access our code better.

```
set.seed(12345)

test_dataset$esrb_rating <- as.factor(test_dataset$esrb_rating)

model <- randomForest(esrb_rating ~ ., ntree = 1000 ,data = test_dataset)
```

From this we have created a model to base future predictions upon. But before we go into that, let's analyze what this model can tell us so far.

```
print(model)
```

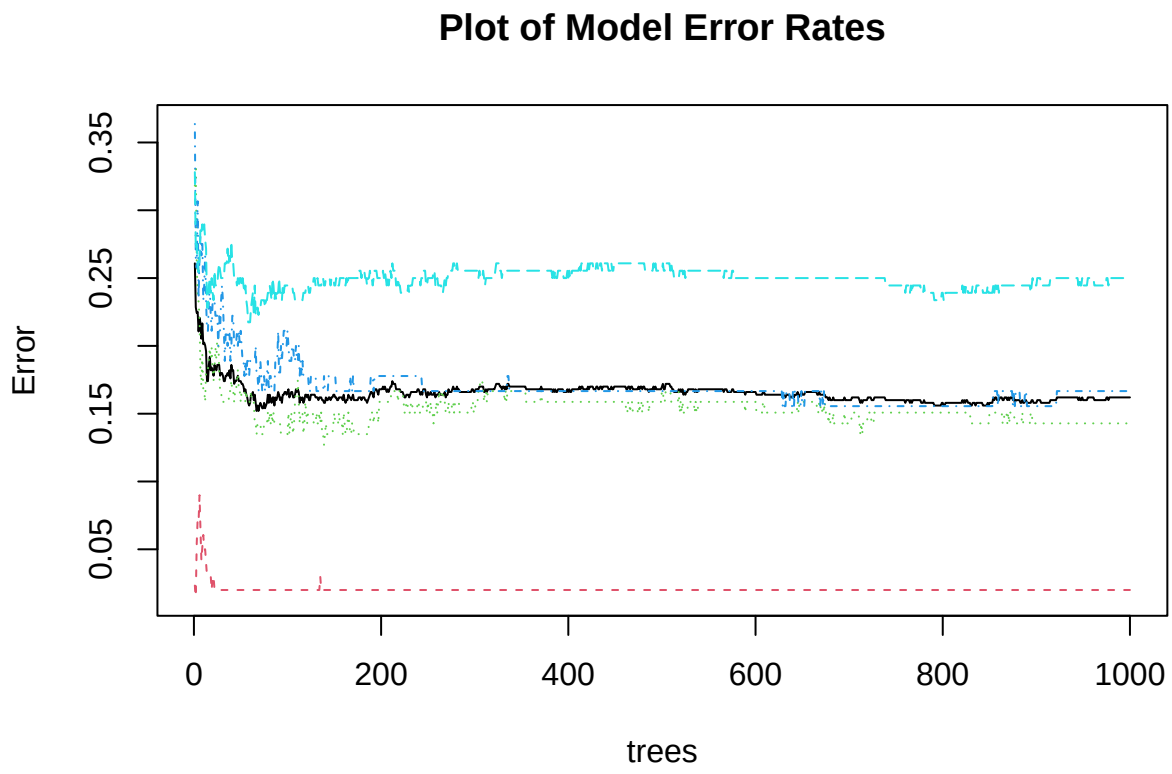
```
##
## Call:
##  randomForest(formula = esrb_rating ~ ., data = test_dataset,      ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 16.2%
## Confusion matrix:
##     E  ET  M   T class.error
## E  98   1  0   1   0.0200000
## ET 11 108  0   7   0.1428571
## M   1   1 75  13   0.1666667
## T   5  25 16 138   0.2500000
```

The most important parts of this model are the OOB estimate and the generated Confusion Matrix.

We can see that the OOB estimate of Error Rate is below 17% which is generally seen as pretty good; however, we can further extrapolate this data from the Confusion Matrix. While ***Confusing*** at first (good one ahmed), the Confusion Matrix shows us the actual values in the rows, and the predicted values from the columns, with the % error in an additonal fifth column. We can see that the model is highly accurate with the E rating with only a 2% error rate; however, gets most of the T ratings wrong with a large 25% error rate.

We can also then plot this model to visualize this data again

```
plot(model, main ="Plot of Model Error Rates")
```



From this graph, we can see the graphical version of the Confusion Matrix, but doesn't really tell us too much more about what we want to see. Rather than going into this plot, let's create some better information to analyze.

Instead of plotting the Error Rates, let's see what factors were most and least important in figuring out what ESRP rating a game will get.

```
#From the model, extract the importance values
imp_df <- as.data.frame(importance(model))

#Add rowNames as columnNames for plotting
imp_df$RowNames <- rownames(importance(model))

#Arrange for formatting
imp_df <- arrange(imp_df, desc(MeanDecreaseGini))

#Top 10 most important values in factoring ESRP Rating
head(imp_df, 10)
```

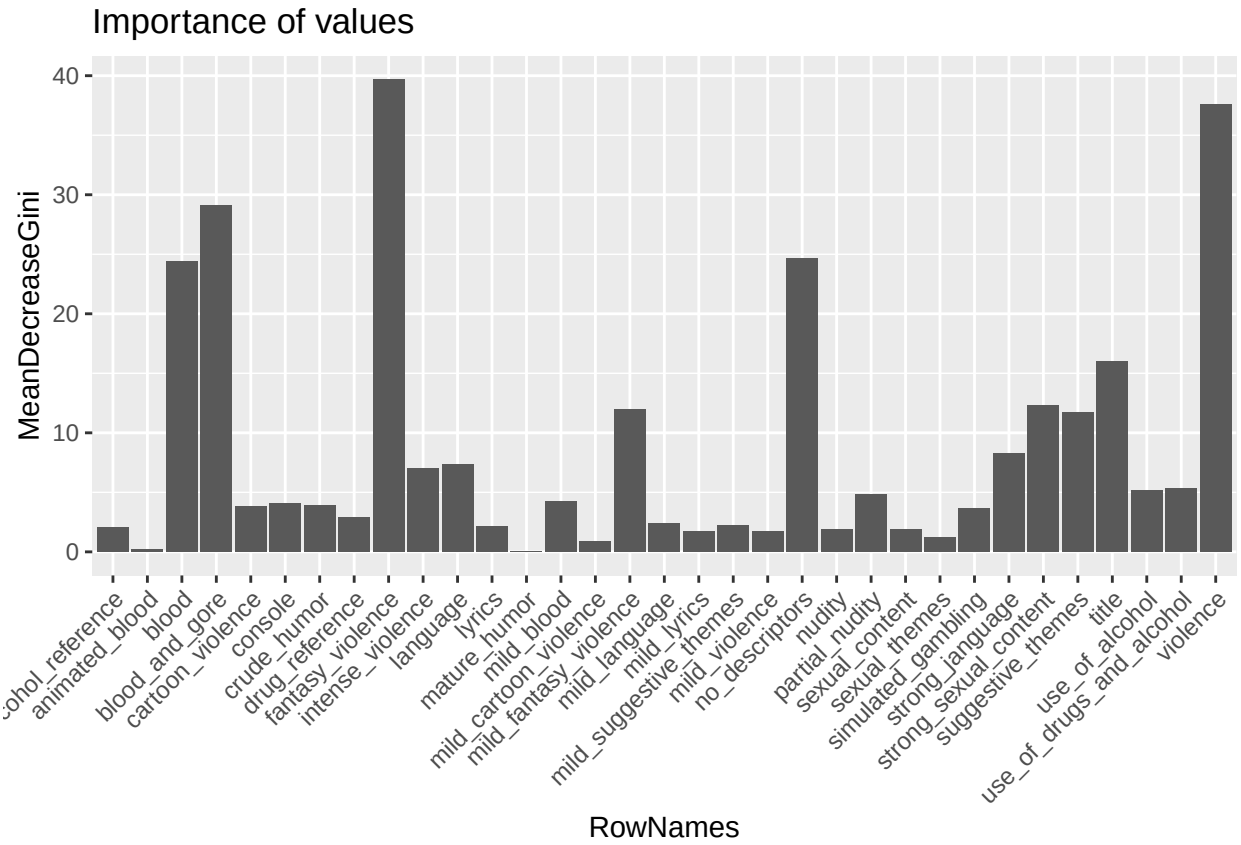
```
##                      MeanDecreaseGini             RowNames
## fantasy_violence             39.693096      fantasy_violence
## violence                     37.594688              violence
## blood_and_gore               29.071602        blood_and_gore
## no_descriptors               24.620709        no_descriptors
```

```
## blood                        24.439889                blood
## title                        16.032422                title
## strong_sexual_content        12.293297 strong_sexual_content
## mild_fantasy_violence        12.011175 mild_fantasy_violence
## suggestive_themes            11.743521     suggestive_themes
## strong_janguage               8.245413       strong_janguage
```

```
#Top 10 least important values in factoring ESRP Rating
tail(imp_df,10)
```

```
##                       MeanDecreaseGini              RowNames
## lyrics                       2.1273214                lyrics
## alcohol_reference            2.0279201     alcohol_reference
## nudity                       1.9232756                nudity
## sexual_content               1.8667463        sexual_content
## mild_violence                1.7066074         mild_violence
## mild_lyrics                  1.6940734           mild_lyrics
## sexual_themes                1.2515737         sexual_themes
## mild_cartoon_violence        0.8611140 mild_cartoon_violence
## animated_blood               0.2214638        animated_blood
## mature_humor                 0.0000000          mature_humor
```

```
#Plot a barplot
ggplot(data = imp_df, aes(x=RowNames, y=MeanDecreaseGini)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Importance of values")
```



Here, we can see actually useful data where now we can see which values were most important in determining what ratings each game would get. Now we don't know from here what rating were most attributed to certain categories. But we can see which ones we're the strongest. From this data, we can see that violence seems to be at the top of the rankings. Whereas, sexual themes and mature humor we closer to the bottom.

This may pose the question to ourselves. Shouldn't mature humor be an indicator for a more mature game? While this may be true, we need to keep in mind that every game has multiple tags for them. A game with mature humor is likely to also contain sexual themes, but these sexual themes could also be categorized into strong sexual content as well. Essentially meaning, that because of these overlaps, there ends up being a few tags that, upon first analysis, look like they are the majority deciding factors on a games rating, when actually, it is that because of overlapping tags, that tags we would assume should immediately indicate that a game is mature seem to have lower power in deciding a games rating.

Running this Model on more data

From the same Kaggle dataset, we also have a set of data to then run this model on to see how accurate our model really is.

```
new_data <- read.csv("/Users/ahmed/Documents/DAT 301/Project 1/Video_games_esrb_rating.csv")

predictions <- predict(model, newdata = new_data)
confusionMatrix(predictions, as.factor(new_data$esrb_rating))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   E  ET   M   T
##         E  413  65  10  51
##         ET   3 295   7 177
##         M    0   4 329 113
##         T    0  39  41 348
##
## Overall Statistics
##
##                Accuracy : 0.7309
##                  95% CI : (0.7103, 0.7507)
##     No Information Rate : 0.3636
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6427
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: E Class: ET Class: M Class: T
## Sensitivity            0.9928    0.7320   0.8501   0.5051
## Specificity            0.9148    0.8747   0.9224   0.9337
## Pos Pred Value         0.7662    0.6120   0.7377   0.8131
## Neg Pred Value         0.9978    0.9236   0.9600   0.7676
## Prevalence             0.2195    0.2127   0.2042   0.3636
## Detection Rate         0.2179    0.1557   0.1736   0.1836
## Detection Prevalence   0.2844    0.2544   0.2354   0.2259
## Balanced Accuracy      0.9538    0.8033   0.8863   0.7194
```

```
print("Actual Number: ")
```

```
## [1] "Actual Number: "
```

```
table(new_data$esrb_rating)
```

```
##
##   E  ET   M   T
## 416 403 387 689
```

```
print("Predicted Number: ")
```

```
## [1] "Predicted Number: "
```

```
summary(predictions)
```

```
##   E  ET   M   T
## 539 482 446 428
```

This outputs a great deal of information, and essentially tells us that the model is pretty accurate at finding what games are rated E and M, the two edges of the given ratings. However, the model seems to have more trouble pinpointing what exactly makes a game rated ET or especially rated T.

Anyway to clean up the data more?

There were a couple of methods to try to make my results a big more accurate. The two most significant methods of getting accurate results are to change factors in the machine learning process of the code, or to clean up the inputted data. The main method that I utilized was to clean up the inputted data. I choose this method primarily, as I believed that the since the dataset was relatively simple to work with, compared to how complicated I knew some data sets could be, I could pick and choose which columns to input into the model with the newfound knowledge from the previous step of analyzing the output importance data.

To begin, let me list out the column names of the beginning data set.

```
colnames(test_dataset)
```

```
##  [1] "title"                    "console"
##  [3] "alcohol_reference"        "animated_blood"
##  [5] "blood"                    "blood_and_gore"
##  [7] "cartoon_violence"         "crude_humor"
##  [9] "drug_reference"           "fantasy_violence"
## [11] "intense_violence"         "language"
## [13] "lyrics"                   "mature_humor"
## [15] "mild_blood"               "mild_cartoon_violence"
## [17] "mild_fantasy_violence"    "mild_language"
## [19] "mild_lyrics"              "mild_suggestive_themes"
## [21] "mild_violence"            "no_descriptors"
## [23] "nudity"                   "partial_nudity"
## [25] "sexual_content"           "sexual_themes"
## [27] "simulated_gambling"       "strong_janguage"
## [29] "strong_sexual_content"    "suggestive_themes"
## [31] "use_of_alcohol"           "use_of_drugs_and_alcohol"
## [33] "violence"                 "esrb_rating"
```

I then went through and tried remove any tags that had overlaps with each other, leaving us with

```
##  [1] "alcohol_reference"        "animated_blood"
##  [3] "blood"                    "blood_and_gore"
##  [5] "cartoon_violence"         "crude_humor"
##  [7] "drug_reference"           "fantasy_violence"
##  [9] "intense_violence"         "language"
## [11] "lyrics"                   "mature_humor"
## [13] "mild_blood"               "mild_cartoon_violence"
## [15] "mild_fantasy_violence"    "mild_language"
## [17] "mild_lyrics"              "mild_suggestive_themes"
## [19] "mild_violence"            "nudity"
## [21] "partial_nudity"           "sexual_content"
## [23] "sexual_themes"            "simulated_gambling"
## [25] "strong_janguage"          "strong_sexual_content"
## [27] "suggestive_themes"        "use_of_alcohol"
## [29] "use_of_drugs_and_alcohol" "violence"
## [31] "esrb_rating"
```

With these much limited columns, I then re-ran all the tests before to create the new model with this condensed data set.

```r
newTest_Data$esrb_rating <- as.factor(newTest_Data$esrb_rating)

model <- randomForest(esrb_rating ~ ., ntree = 1000 ,data = newTest_Data)

print(model)
```

```
##
## Call:
##  randomForest(formula = esrb_rating ~ ., data = newTest_Data,      ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 16.2%
## Confusion matrix:
##     E  ET  M   T class.error
## E  98   1  0   1   0.0200000
## ET 11 106  0   9   0.1587302
## M   1   1 74  14   0.1777778
## T   2  27 14 141   0.2336957
```

```r
predictions <- predict(model, newdata = new_data)
confusionMatrix(predictions, as.factor(new_data$esrb_rating))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   E  ET   M   T
##         E  413  46  17  37
##         ET   3 301   7 182
##         M    0   5 324  93
##         T    0  51  39 377
##
## Overall Statistics
##
##                Accuracy : 0.7467
##                  95% CI : (0.7265, 0.7662)
##     No Information Rate : 0.3636
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6623
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: E Class: ET Class: M Class: T
## Sensitivity            0.9928    0.7469   0.8372   0.5472
## Specificity            0.9324    0.8713   0.9350   0.9254
## Pos Pred Value         0.8051    0.6105   0.7678   0.8073
## Neg Pred Value         0.9978    0.9272   0.9572   0.7815
## Prevalence             0.2195    0.2127   0.2042   0.3636
```

```
## Detection Rate         0.2179    0.1588   0.1710   0.1989
## Detection Prevalence   0.2707    0.2602   0.2227   0.2464
## Balanced Accuracy      0.9626    0.8091   0.8861   0.7363
```

```
print("Actual Number: ")
```

```
## [1] "Actual Number: "
```

```
table(new_data$esrb_rating)
```

```
##
##   E  ET   M   T
## 416 403 387 689
```

```
print("Predicted Number: ")
```

```
## [1] "Predicted Number: "
```

```
summary(predictions)
```

```
##   E  ET   M   T
## 513 493 422 467
```

Overall, the same level of Error Rates were occurring on this new sample set, so I decided to dive into the actual raw data to see if I could come up with any reasonings as to why this was. After looking at this data, I found a significant amount of games which has the same markings for multiple tags and outputted different results. Attempting to filter these tags down even more than we had previously led to even with Error Rates, with the model getting 100% of the E rated games **incorrect**. Thus with the OOD estimate of error rate being 16.2%, I believe that the Random Forest Model allows for us to judge what ratings games would get using Machine Learning with a Training Data Set.