Capstone Project: Video Games Sales with Ratings

Rahasya Chandan

Introduction

The goal of this report is to apply machine learning and data science techniques on a public dataset to study and solve any problem of our choice. The dataset used in this report is " Video Game Sales with Rating ". The dataset contains data of several video games with scores from critics, user scores, sales, ratings and other fields. This dataset has more than 6900 video games rating done by both critics and users. This project is trying to answer if we can predict ratings based on any of the variables in the dataset. An algorithm is written to analyze the dataset and the results are used to train and test the data set to check the accuracy of our predictions. Data splitting, data cleaning, data scaling, data summarization, and visualization are used in this report to analyze the dataset and predict the video game sales ratings.

Read dataset

```
vgames <- read.csv("https://raw.githubusercontent.com/rahasyac/Data_Science_Capstone_Project/main/Data%
```

Data Cleaning

The first step performed after reading the dataset is data cleaning. We go through the process of dropping the irrelevant columns such as "Name", "Year of Release", "Developer", and "Publisher". Then we remove the not applicable data from the dataset to maintain a clean working directory.

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
dropcolumns <- c("Name","Year_of_Release","Developer","Publisher")# drop columns
vgamesdf<- vgames[,!(names(vgames) %in% dropcolumns)]
vgamesdf <- vgamesdf %>% mutate_all(na_if,"")

head(vgamesdf)# print data
```

```
##   Platform        Genre NA_Sales EU_Sales JP_Sales Other_Sales Global_Sales
## 1      Wii       Sports    41.36    28.96     3.77        8.45        82.53
## 2      NES     Platform    29.08     3.58     6.81        0.77        40.24
## 3      Wii       Racing    15.68    12.76     3.79        3.29        35.52
## 4      Wii       Sports    15.61    10.93     3.28        2.95        32.77
## 5       GB Role-Playing    11.27     8.89    10.22        1.00        31.37
## 6       GB       Puzzle    23.20     2.26     4.22        0.58        30.26
##   Critic_Score Critic_Count User_Score User_Count Rating
## 1           76           51          8        322      E
## 2           NA           NA       <NA>         NA   <NA>
## 3           82           73        8.3        709      E
## 4           80           73          8        192      E
## 5           NA           NA       <NA>         NA   <NA>
## 6           NA           NA       <NA>         NA   <NA>
```

```r
# drop NA values/rows
vgamesdf <- vgamesdf %>% drop_na()
```

Data Summarization

The next task performed on the dataset is data summarization to analyze the data. It helps us understand the summary of generated data in an easy and informative manner.

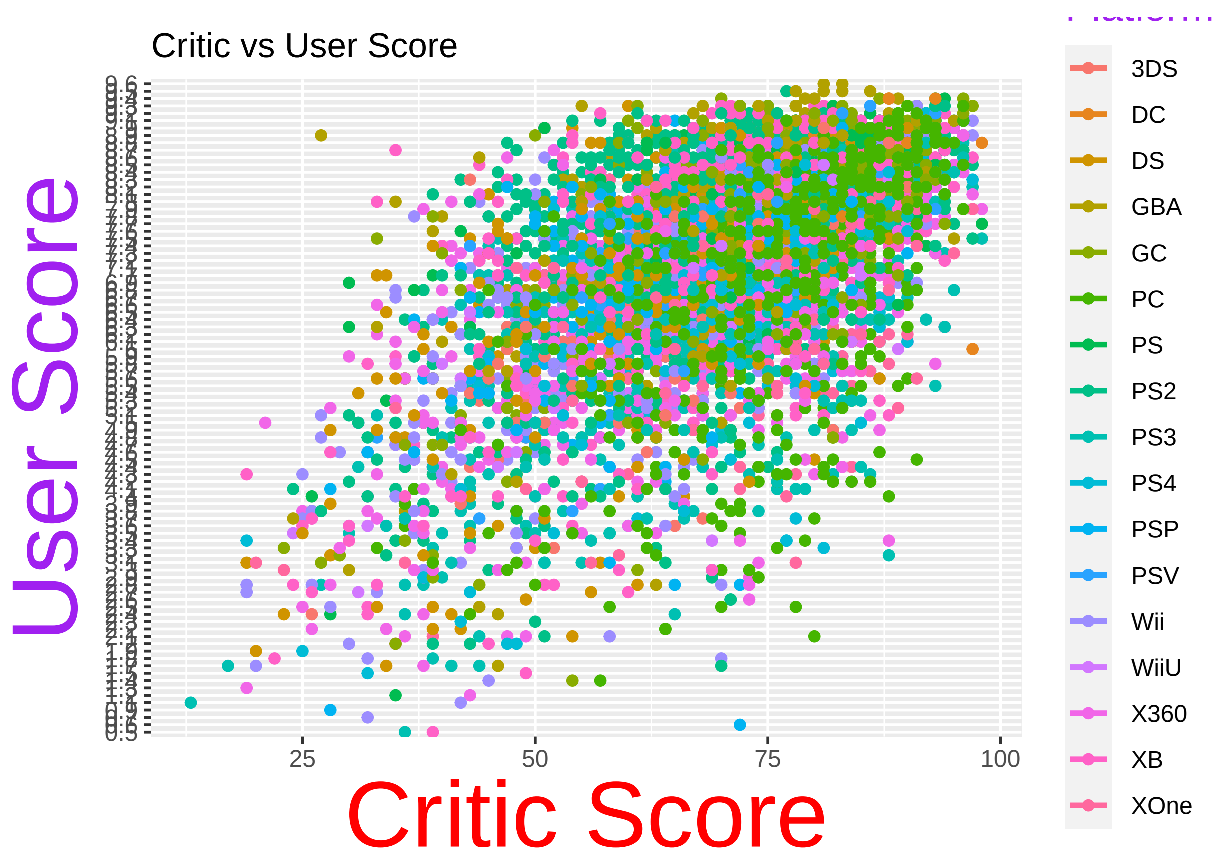
```r
# data summary
summary(vgamesdf)
```

```
##    Platform            Genre              NA_Sales          EU_Sales      
##  Length:6947        Length:6947        Min.   : 0.0000   Min.   : 0.0000  
##  Class :character   Class :character   1st Qu.: 0.0600   1st Qu.: 0.0200  
##  Mode  :character   Mode  :character   Median : 0.1500   Median : 0.0600  
##                                        Mean   : 0.3928   Mean   : 0.2346  
##                                        3rd Qu.: 0.3900   3rd Qu.: 0.2100  
##                                        Max.   :41.3600   Max.   :28.9600  
##     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   :0.06324   Mean   : 0.08219   Mean   : 0.7731   Mean   :70.26  
##  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         Rating         
##  Min.   :  3.00   Length:6947        Min.   :    4.0   Length:6947       
##  1st Qu.: 14.00   Class :character   1st Qu.:   11.0   Class :character  
##  Median : 24.00   Mode  :character   Median :   27.0   Mode  :character  
##  Mean   : 28.87                      Mean   :  173.8                     
##  3rd Qu.: 39.00                      3rd Qu.:   88.0                     
##  Max.   :113.00                      Max.   :10665.0                     
```

Data Visualization

We apply data visualization in the form of a scatter plot by plotting the relationship between "Critic Score" and "User Score" for each video game platform.

```r
library(ggplot2)
plots <-ggplot(data = vgamesdf,aes(x = Critic_Score,y=User_Score,color = Platform))
plots + geom_point() + geom_smooth(fill = NA) + ggtitle("Critic vs User Score")+
  xlab("Critic Score")+
  ylab("User Score")+
  theme(axis.title.x = element_text(color="red",size = 35),
        axis.title.y = element_text(color="purple",size = 35),
        legend.title = element_text(color="purple",size = 15))
```




```r
vgamesdf %>% count(Rating)# count observation for each Rating category
```

```
##   Rating    n
## 1     AO    1
## 2      E 2118
## 3   E10+  946
## 4    K-A    1
## 5      M 1459
## 6     RP    2
## 7      T 2420
```

```r
# the results shown that
#Rating    n
#1     AO    1
#2      E 2118
```

```
#3   E10+  946
#4    K-A    1
#5      M 1459
#6     RP    2
#7      T 2420
# therefore, AO, K-A and RP instances are removed because the machine learning model will be bias as th
vgamesdf <- filter(vgamesdf, Rating != "AO")
vgamesdf <- filter(vgamesdf, Rating != "K-A")
vgamesdf <- filter(vgamesdf, Rating != "RP")

# convert variables to factor and numeric
vgamesdf$Platform <- as.factor(vgamesdf$Platform)
vgamesdf$Genre <- as.factor(vgamesdf$Genre)
vgamesdf$Rating <- as.factor(vgamesdf$Rating)
vgamesdf$User_Score <- as.numeric(vgamesdf$User_Score)
vgamesdf$Critic_Score <- as.numeric(vgamesdf$Critic_Score)
vgamesdf$Critic_Count <- as.numeric(vgamesdf$Critic_Count)
vgamesdf$User_Count <- as.numeric(vgamesdf$User_Count)
```

```
str(vgamesdf)
```

```
## 'data.frame':    6943 obs. of  12 variables:
##  $ Platform    : Factor w/ 17 levels "3DS","DC","DS",..: 13 13 13 3 13 13 3 13 15 13 ...
##  $ Genre       : Factor w/ 12 levels "Action","Adventure",..: 11 7 11 5 4 5 7 11 4 11 ...
##  $ NA_Sales    : num  41.4 15.7 15.6 11.3 14 ...
##  $ EU_Sales    : num  28.96 12.76 10.93 9.14 9.18 ...
##  $ JP_Sales    : num  3.77 3.79 3.28 6.5 2.93 4.7 4.13 3.6 0.24 2.53 ...
##  $ Other_Sales : num  8.45 3.29 2.95 2.88 2.84 2.24 1.9 2.15 1.69 1.77 ...
##  $ Global_Sales: num  82.5 35.5 32.8 29.8 28.9 ...
##  $ Critic_Score: num  76 82 80 89 58 87 91 80 61 80 ...
##  $ Critic_Count: num  51 73 73 65 41 80 64 63 45 33 ...
##  $ User_Score  : num  8 8.3 8 8.5 6.6 8.4 8.6 7.7 6.3 7.4 ...
##  $ User_Count  : num  322 709 192 431 129 594 464 146 106 52 ...
##  $ Rating      : Factor w/ 4 levels "E","E10+","M",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# convert string variables to numeric representation
df <- vgamesdf %>% mutate_if(is.factor, as.numeric)
df <- df[sample(nrow(df)),]# randomly shuffle dataset
```

Data Splitting

I have utilized data splitting to split the dataset into train and test sets. So that I can use the train set to develop a predictive model and then use the test set to test the model's performance.

```
# split dataset
#install.packages('caTools')
library(caTools)

set.seed(12345)
split = sample.split(df$Rating, SplitRatio = 0.70)
```

```
training_set = subset(df, split == TRUE)
test_set = subset(df, split == FALSE)
```

Data Scaling

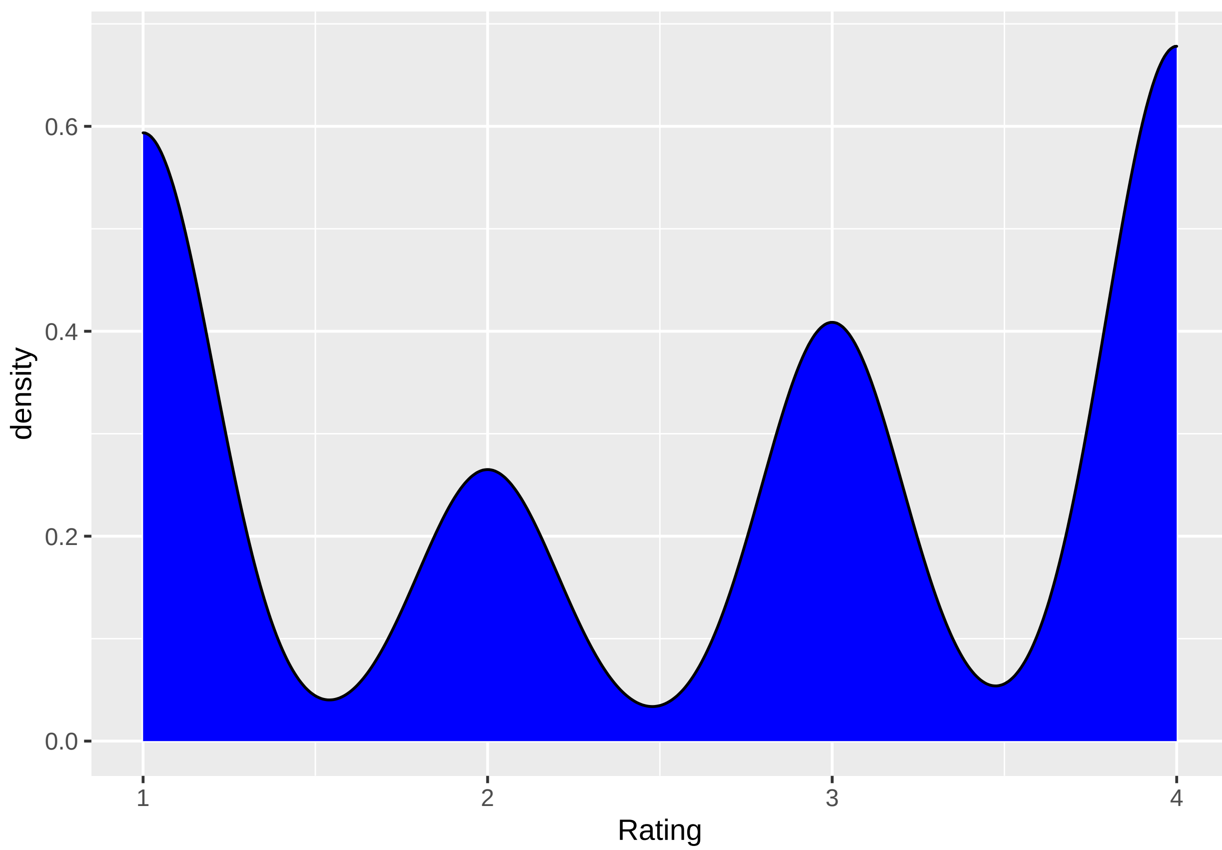
The final data science technique I have performed is data scaling. We need to scale the data before performing the algorithms to transform the data and avoid attributes in the greater numeric ranges as they might cause numerical errors. We need to apply the same method of data scaling to both testing and training sets.

```
# data scaling
training_set[-12] = scale(training_set[-12])
test_set[-12] = scale(test_set[-12])
```

Multiple Regression Model

The first model I am utilizing is multiple linear regression as it will help us to show and predict the relationship between ratings and the other variables in the dataset.

```
# plot the distribution of the variable 'Rating'
ggplot(training_set, aes(Rating)) + geom_density(fill="blue")
```

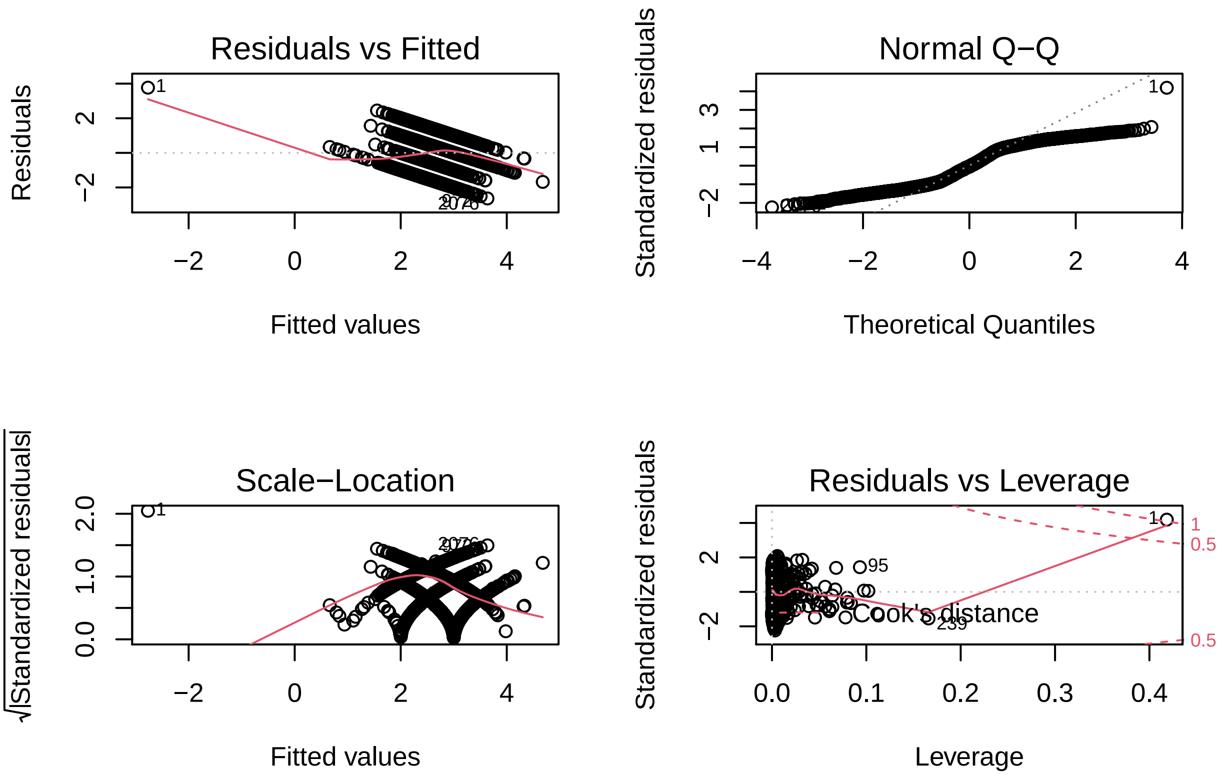


```r
# Take all the inputs to make a multiple regression model
model1 = lm(Rating~., data=training_set)

#data summary
summary(model1)
```

```
##
## Call:
## lm(formula = Rating ~ ., data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6354 -1.1206 -0.0468  1.1432  3.7672
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.60206    0.01691 153.882  < 2e-16 ***
## Platform      0.09864    0.01783   5.531 3.34e-08 ***
## Genre        -0.22118    0.01723 -12.834  < 2e-16 ***
## NA_Sales     -1.96572    2.87849  -0.683    0.495
## EU_Sales     -1.44739    2.14401  -0.675    0.500
## JP_Sales     -0.57502    0.84305  -0.682    0.495
## Other_Sales  -0.37819    0.67441  -0.561    0.575
## Global_Sales  3.83415    5.95038   0.644    0.519
## Critic_Score -0.14186    0.02343  -6.054 1.52e-09 ***
## Critic_Count  0.25027    0.02029  12.336  < 2e-16 ***
## User_Score    0.13279    0.02155   6.162 7.76e-10 ***
## User_Count    0.09421    0.01922   4.903 9.77e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.179 on 4848 degrees of freedom
## Multiple R-squared:  0.104,  Adjusted R-squared:  0.1019
## F-statistic: 51.13 on 11 and 4848 DF,  p-value: < 2.2e-16
```

```r
# plot the model
par(mfrow=c(2,2))
plot(model1)
```



Prediction of Multiple Regression model

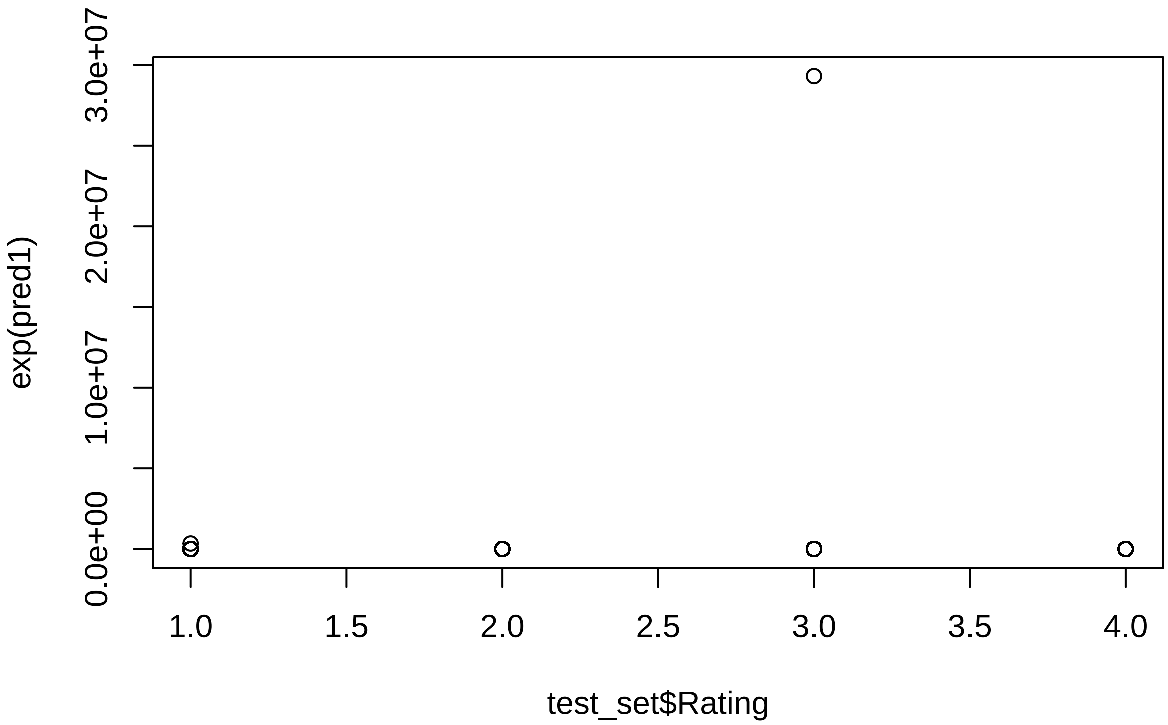
```
# prediction based on model 1
pred1 <- predict(model1, newdata = test_set)

# calculate RSME and R^^2
rmse <- sqrt(sum((exp(pred1) - test_set$Rating)^2)/length(test_set$Rating))

# result for Multiple regression model
result1 <- c(RMSE = rmse, R2=summary(model1)$r.squared)
result1
```

```
##         RMSE           R2
## 6.422662e+05 1.039519e-01
```

```
# plot the model
par(mfrow=c(1,1))
plot(test_set$Rating, exp(pred1))
```



Support Vector Machine model

Since this is not a very large dataset and I don't know a lot of information about the data, the second modeling approach I have chosen is SVM. The SVM algorithm is stable and few changes in data will not show any major changes in the result. Since the SVM model is a classification algorithm, we can use a confusion matrix to easily get the accuracy and performance of the model. Another reason to choose the SVM model is because we know there is a non-linear relationship between ratings and the other variables after the performance of the multiple regression algorithm. An advantage of SVM is that we can use the kernel to make non-linear algorithms.

```
#install.packages('e1071')
library(e1071)

# support vector machine model
classifier = svm(formula = Rating ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')
```

Prediction of the Support Vector Machine model

```
y_pred = predict(classifier, newdata = test_set[-12])
```

```r
# confusion matrix
cm = table(test_set[, 12], y_pred)
cm
```

```
##    y_pred
##       1   2   3   4
##   1 402   0   5 228
##   2 102   0   9 173
##   3  60   0  85 293
##   4 203   0  22 501
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
result2 <- confusionMatrix(cm)
```

Result

In the multiple regression approach, we see that in the model F = 1.18, which is very close to 1. This means that there is a negative relationship between ratings and the other variables in the dataset. We can see from the plot that the relationship is non-linear. An RSME of 13.34 was found for this model. Based on the Multiple R-squared value which is 0.102, we know the model is not such a good fit. In the SVM model approach, we observe there is improvement in performance of this model. Using a confusion matrix we find the accuracy for the model is 0.4666. The accuracy is low and it is not a good model as both True Positive Rate and True Negative Rate are high.

```r
# result for Multiple regression model
result1
```

```
##         RMSE           R2
## 6.422662e+05 1.039519e-01
```

```r
# result for support vector machine
result2
```

```
## Confusion Matrix and Statistics
##
##    y_pred
##       1   2   3   4
##   1 402   0   5 228
##   2 102   0   9 173
##   3  60   0  85 293
##   4 203   0  22 501
##
## Overall Statistics
##
##                Accuracy : 0.4743
##                  95% CI : (0.4527, 0.496)
```

```
##     No Information Rate : 0.5737
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2219
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            0.5241       NA  0.70248   0.4192
## Specificity            0.8229   0.8637  0.82008   0.7466
## Pos Pred Value         0.6331       NA  0.19406   0.6901
## Neg Pred Value         0.7479       NA  0.97812   0.4886
## Prevalence             0.3682   0.0000  0.05809   0.5737
## Detection Rate         0.1930   0.0000  0.04081   0.2405
## Detection Prevalence   0.3048   0.1363  0.21027   0.3485
## Balanced Accuracy      0.6735       NA  0.76128   0.5829
```

Conculsion

The main task of the project is to develop a model to see if we can predict ratings based on the other variables. We can answer the question by observing the performance of the multiple regression model. It shows that the RSME is high for this model. We can conclude that the average predicted value will be off by 13.34 and that the ratings do not have any significant positive correlation with the other variables. Similarly the performance of the Support Vector Machine model has a low accuracy at 46.66%, so ratings are a poor classifier for the other variables in the dataset. By observing both performances of the model, it tells us that we cannot predict ratings based on the other variables in the dataset.