SVM Classification - Neo Zhao & Andrew Sen - CS4375

Linear Models

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.3

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.1.3

## Warning: package 'tidyr' was built under R version 4.1.3

## Warning: package 'readr' was built under R version 4.1.3

## Warning: package 'purrr' was built under R version 4.1.3

## Warning: package 'dplyr' was built under R version 4.1.3

## Warning: package 'forcats' was built under R version 4.1.3

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

```
library(dplyr)
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.1.3
```

```
library(mccr)
```

```
## Warning: package 'mccr' was built under R version 4.1.3
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.3
```

```r
library(rpart)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.1.3
```

```r
# Source: https://www.kaggle.com/datasets/budnyak/wine-rating-and-price?select=Red.csv

# Red, White, Rose, and Sparkling wine are all from the same dataset; however, separated by type

# Red Total: 8666
Red <- read.csv("Red.csv")

# White Total: 3764
White <- read.csv("White.csv")

# Rose Total: 397
Rose <- read.csv("Rose.csv")

# Sparkling Total: 1007
Sparkling <- read.csv("Sparkling.csv")

# Combine the datasets together, Total: 13058
totalWine <- rbind(data = Red, data = White, data = Rose, data = Sparkling)

# Rename ï..Name to just Name
names(totalWine)[1] <- "Name"

# Omit Names, Winery, & Region Column
totalWine <- subset(totalWine, select = -c(Name, Winery, Region))

# Omit all records where Year = N.V.
totalWine <- subset(totalWine, totalWine$Year != "N.V.")

# Omit all records where Rating = 3, Total: 12398
totalWine <- subset(totalWine, totalWine$Rating != 3)
```

```r
# Omit all records before 2000s
totalWine <- subset(totalWine, totalWine$Year >= 2000)

# Make the Year from chr -> num
totalWine$Year <- as.numeric(totalWine$Year)

# Set Country from chr -> factor
totalWine$Country <- as.factor(totalWine$Country)

# Setting Ratings on a scale of 1 to 10
totalWine$Rating <- round(totalWine$Rating / 0.5) * 0.5

totalWine$Rating[totalWine$Rating == 0.5] <- 1
totalWine$Rating[totalWine$Rating == 1] <- 2
totalWine$Rating[totalWine$Rating == 1.5] <- 3
totalWine$Rating[totalWine$Rating == 2] <- 4
totalWine$Rating[totalWine$Rating == 2.5] <- 5
totalWine$Rating[totalWine$Rating == 3] <- 6
totalWine$Rating[totalWine$Rating == 3.5] <- 7
totalWine$Rating[totalWine$Rating == 4] <- 8
totalWine$Rating[totalWine$Rating == 4.5] <- 9
totalWine$Rating[totalWine$Rating == 5] <- 10

# Reorder Columns
totalWine <- totalWine[,c(1,2,3,5,4)]
```

```r
# Split to 80/20 Train/Test
set.seed(512)
i <- sample(1 : nrow(totalWine), nrow(totalWine) * 0.75, replace = FALSE)
train <- totalWine[i,]
test <- totalWine[-i,]

# Reducing size to 500
i <- sample(nrow(train), size = 500, replace = FALSE)
train_sample <- train[i,]
```

Data Exploration

```r
# 1) summary()
summary(train)
```

```
##           Country         Rating       NumberOfRatings        Year
##  Italy         :2711   Min.   : 6.000   Min.   :   25.0   Min.   :2000
##  France        :2318   1st Qu.: 7.000   1st Qu.:   55.0   1st Qu.:2015
##  Spain         :1131   Median : 8.000   Median :  122.0   Median :2016
##  Germany       : 875   Mean   : 7.739   Mean   :  332.6   Mean   :2016
##  South Africa  : 638   3rd Qu.: 8.000   3rd Qu.:  312.0   3rd Qu.:2018
##  United States: 393   Max.   :10.000   Max.   :20293.0   Max.   :2020
##  (Other)       :1684
##      Price
```

```
##  Min.   :   3.55
##  1st Qu.:   9.90
##  Median :  15.90
##  Mean   :  32.45
##  3rd Qu.:  32.00
##  Max.   :1599.95
##
```

```r
# 2) is.na()
colSums(is.na(train))
```

```
##         Country          Rating NumberOfRatings            Year           Price
##               0               0               0               0               0
```

```r
colSums(is.na(test))
```

```
##         Country          Rating NumberOfRatings            Year           Price
##               0               0               0               0               0
```

```r
# 3) str()
str(train)
```

```
## 'data.frame':    9750 obs. of  5 variables:
##  $ Country        : Factor w/ 33 levels "Argentina","Australia",..: 3 17 17 11 11 13 32 32 28 27 ...
##  $ Rating         : num  7 8 7 8 9 8 9 8 7 8 ...
##  $ NumberOfRatings: num  30 28 375 40 290 42 791 110 193 282 ...
##  $ Year           : num  2017 2018 2018 2017 2008 ...
##  $ Price          : num  13.2 12.2 14.4 9.5 69.4 ...
```

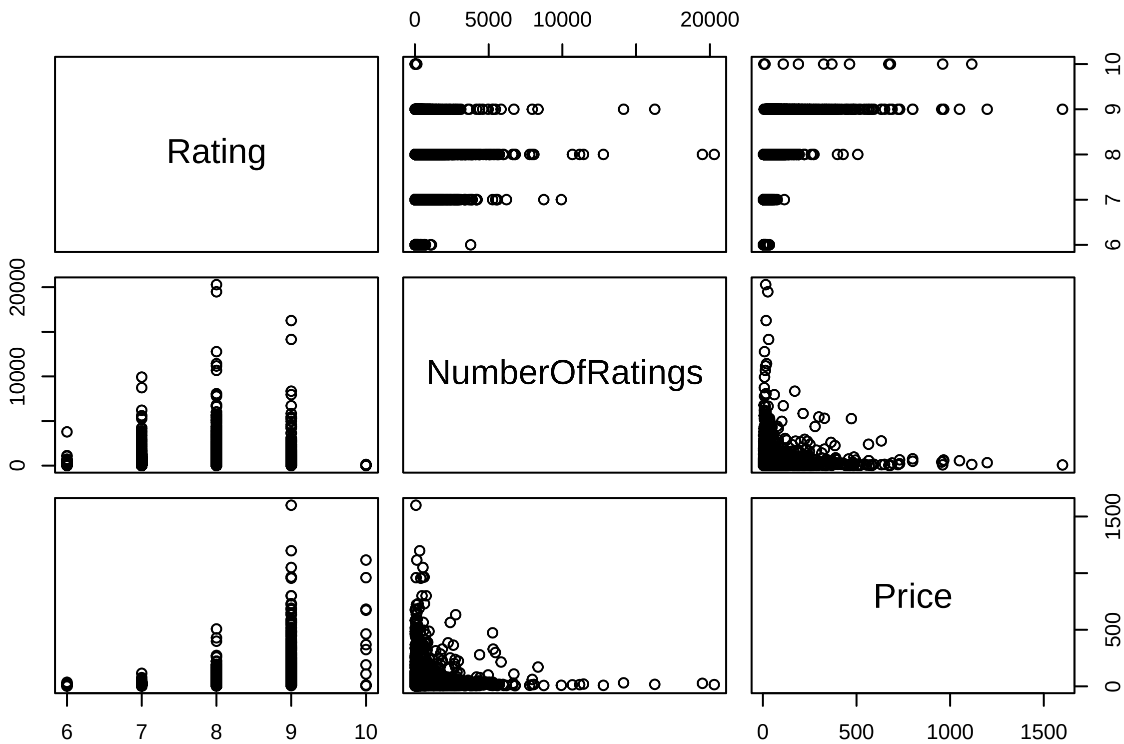
```r
# 4) head() functions
head(train)
```

```
##             Country Rating NumberOfRatings Year Price
## data.28810 Austria      7              30 2017 13.24
## data.19110   Italy      8              28 2018 12.18
## data.33841   Italy      7             375 2018 14.45
## data.84110  France      8              40 2017  9.50
## data.53111  France      9             290 2008 69.36
## data.5227  Germany      8              42 2017 22.00
```

```r
# 5) cor() and pairs()
cor(train[,c(-1, -4)])
```

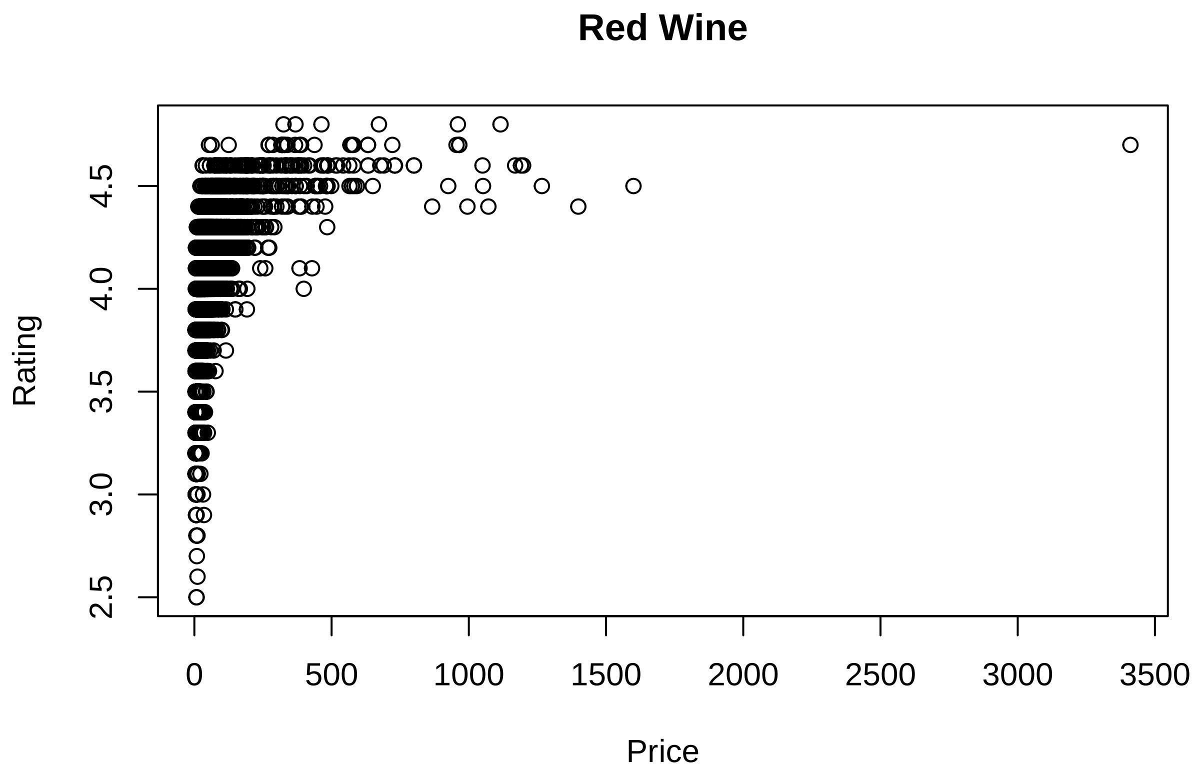
```
##                     Rating NumberOfRatings      Price
## Rating          1.00000000      0.07951796 0.43743631
## NumberOfRatings 0.07951796      1.00000000 0.03004398
## Price           0.43743631      0.03004398 1.00000000
```

```
pairs(train[,c(-1, -4)])
```

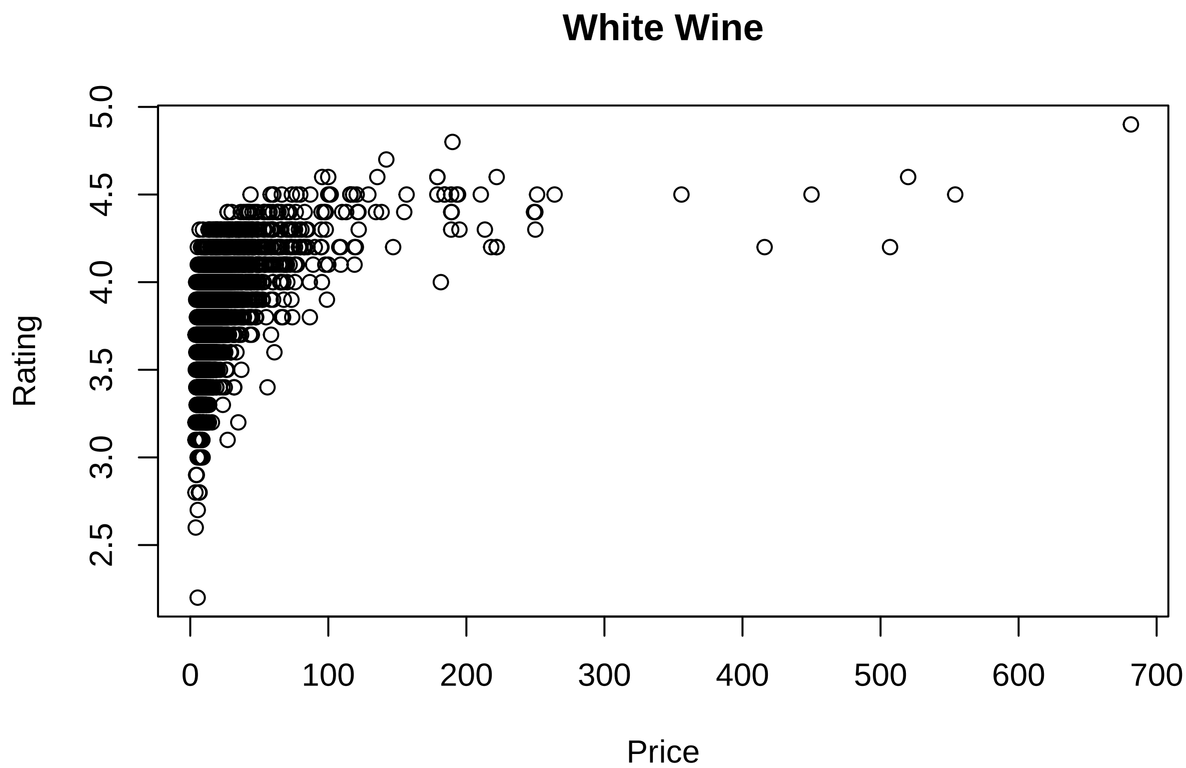


Informative Graphs

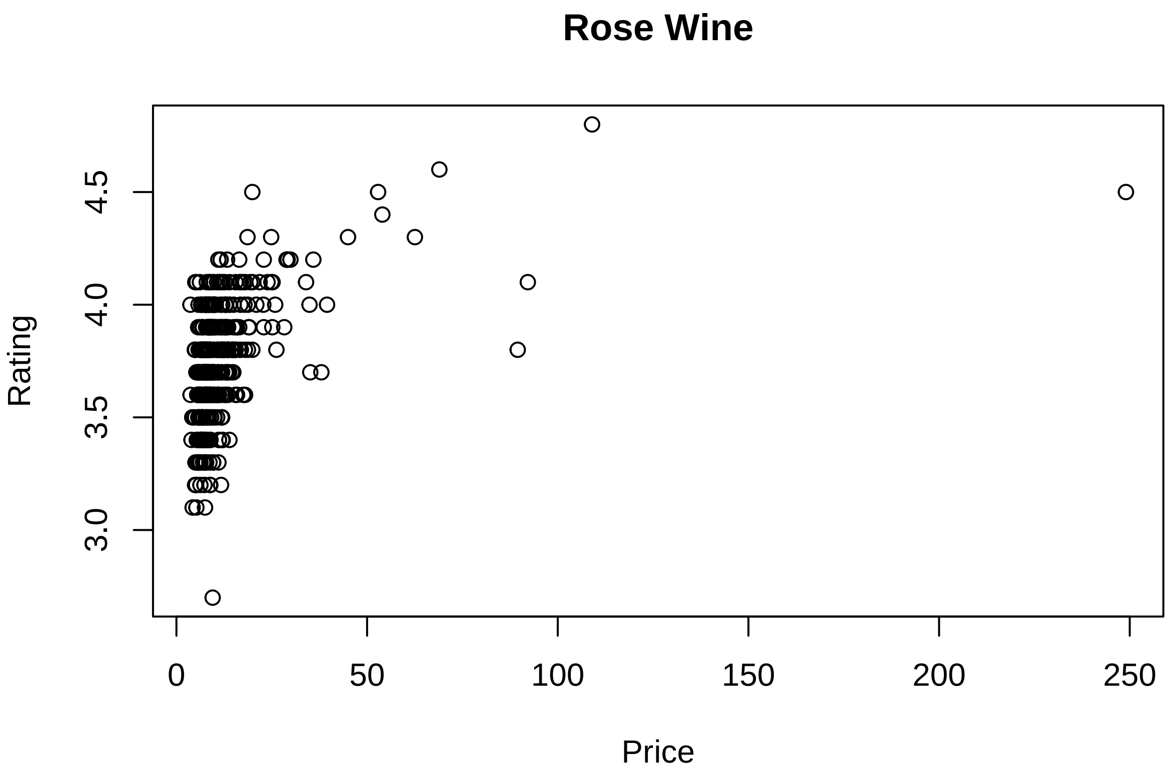
```
# Red
plot(Rating ~ Price, data = Red, main = "Red Wine", xlab = "Price", ylab = "Rating")
```



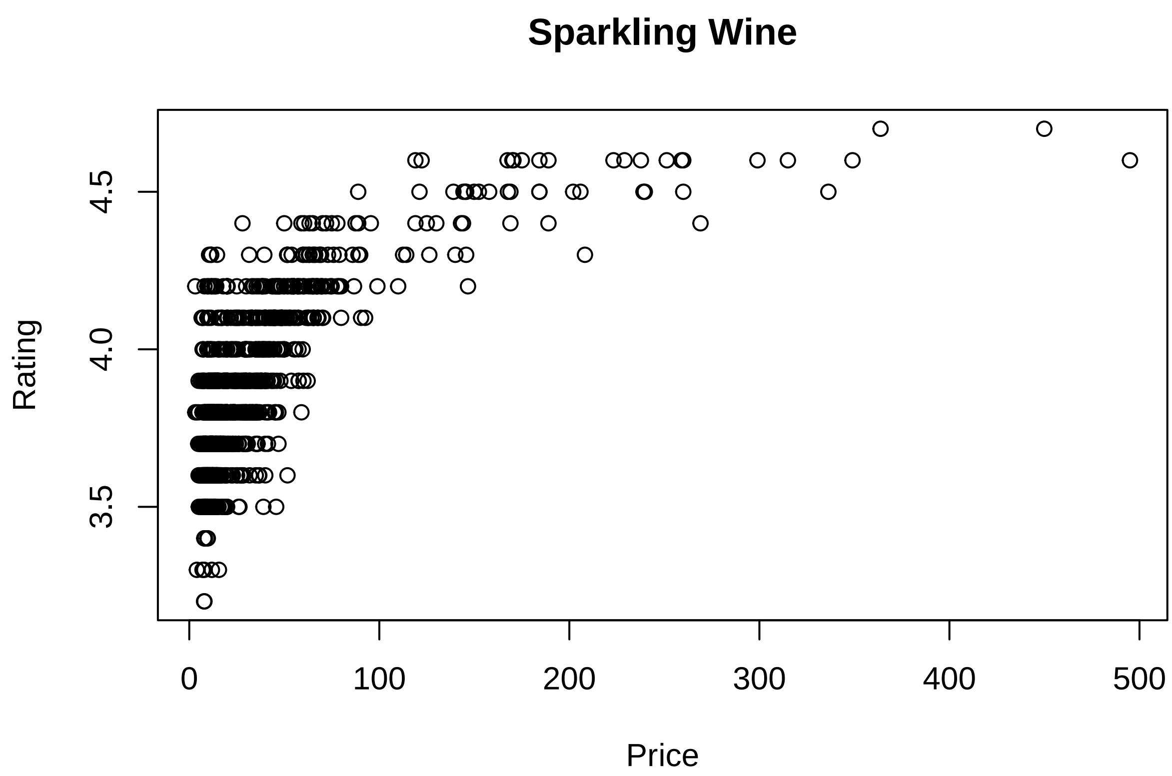

```r
# White
plot(Rating ~ Price, data = White, main = "White Wine", xlab = "Price", ylab = "Rating")
```




```
# Rose
plot(Rating ~ Price, data = Rose, main = "Rose Wine", xlab = "Price", ylab = "Rating")
```




```r
# Sparkling
plot(Rating ~ Price, data = Sparkling, main = "Sparkling Wine", xlab = "Price", ylab = "Rating")
```



SVM Regression - Linear

```
svm1 <- tune.svm(Rating ~., data = train_sample, kernel = "linear",
 cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)
)$best.model

summary(svm1)
```

```
##
## Call:
## best.svm(x = Rating ~ ., data = train_sample, cost = c(0.001, 0.01,
##     0.1, 1, 5, 10, 100), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.02777778
##     epsilon:  0.1
##
##
## Number of Support Vectors:  374
```

```r
# Evaluate
pred <- predict(svm1, newdata = test)
head(table(pred, test$Rating))
```

```
##
## pred               6 7 8 9
##   7.84657044012305 0 0 1 0
##   7.84887152963468 0 0 1 0
##   7.84892574208456 0 1 0 0
##   7.84938101100588 0 1 0 0
##   7.84980150602325 0 1 0 0
##   7.84994275957158 0 0 1 0
```

```r
cor_svm1 <- cor(pred, test$Rating)
mse_svm1 <- mean((pred - test$Rating)^2)
print(paste('Correlation: ', cor_svm1))
```

```
## [1] "Correlation:  0.37736801591945"
```

```r
print(paste('MSE: ', mse_svm1))
```

```
## [1] "MSE:  0.413973714003012"
```

Polynomial

```r
svm2 <- tune.svm(Rating ~ ., data = train_sample, kernel = "polynomial",
 cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)
)$best.model

summary(svm2)
```

```
##
## Call:
## best.svm(x = Rating ~ ., data = train_sample, cost = c(0.001, 0.01,
##     0.1, 1, 5, 10, 100), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  5
##      degree:  3
##       gamma:  0.02777778
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  367
```

```r
# Evaluate
pred <- predict(svm2, newdata = test)
head(table(pred, test$Rating))
```

```
##
## pred                 6 7 8 9
##   -11.7130654277082 0 0 0 1
##   6.10039663247557  0 0 1 0
##   7.16373129658659  0 0 1 0
##   7.60154255954019  0 1 0 0
##   7.63906563664734  0 0 1 0
##   7.6392597172939   0 1 0 0
```

```r
cor_svm2 <- cor(pred, test$Rating)
mse_svm2 <- mean((pred - test$Rating)^2)
print(paste('Correlation: ', cor_svm2))
```

```
## [1] "Correlation:  0.0329731709294317"
```

```r
print(paste('MSE: ', mse_svm2))
```

```
## [1] "MSE:  0.571978128535805"
```

Radial Kernel

```r
svm3 <- tune.svm(Rating ~ ., data = train_sample, kernel = "radial",
 cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100), gamma = c(0.001, 0.01, 0.1, 1, 5, 10, 100)
)$best.model

summary(svm3)
```

```
##
## Call:
## best.svm(x = Rating ~ ., data = train_sample, gamma = c(0.001, 0.01,
##     0.1, 1, 5, 10, 100), cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  453
```

```r
# Evaluate
pred <- predict(svm3, newdata = test)
head(table(pred, test$Rating))
```

```
##
## pred               6 7 8 9
##   6.8038510510069  0 1 0 0
##   6.81342140197694 1 0 0 0
##   6.82246101791359 0 1 0 0
##   6.83440524633374 0 1 0 0
##   6.84274183128588 0 1 0 0
##   6.8582147520239  0 1 0 0
```

```r
cor_svm3 <- cor(pred, test$Rating)
mse_svm3 <- mean((pred - test$Rating)^2)
print(paste('Correlation: ', cor_svm3))
```

```
## [1] "Correlation:  0.567100629958411"
```

```r
print(paste('MSE: ', mse_svm3))
```

```
## [1] "MSE:  0.289725420614269"
```

Conclusion

- Linear Kernel has the most middle performance out of all three; however, it's probably still not the best fit for this dataset as the correlation is still quite low. Therefore, Linear Kernel probably isn't any better than Simple Linear Regression, and we would get a similar Correlation value.
- Polynomial Kernel has poor performance out of the three. It's possible that the model has overfitted due to tuning.
- Radial Kernel has the best performance as it is the most generalized form of kernelization.