SVM Regression

Meinhard Capucao, Khang Thai

Introduction

For this project, I used the diamond.csv data set from previous projects to determine the svm linear regression. We used the depth, table, x (length in mm), y (width in mm), and z (height in mm) to determine the carat value.

Read in the data

The dataset is around 53k dataset and we decided to lower the dataset to 10k so that the amount of time it takes to run svm is a lot less.

```
library(e1071)
DiamondDataset <- read.csv("diamonds.csv", header=TRUE)
DiamondDataset <- DiamondDataset[1:10000,c(2,6,7,8,9,10,11)]
str(DiamondDataset)
```

```
## 'data.frame':    10000 obs. of  7 variables:
##  $ carat: num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ depth: num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table: num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price: int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x    : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y    : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z    : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

Spliting the data into Test, Train, and Validate

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
set.seed(1234)
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(DiamondDataset), nrow(DiamondDataset)*cumsum(c(0,spec)), labels = names(s
pec)))

train <- DiamondDataset[i == "train",]
test <- DiamondDataset[i == "test",]
vald <- DiamondDataset[i =="validate",]
```

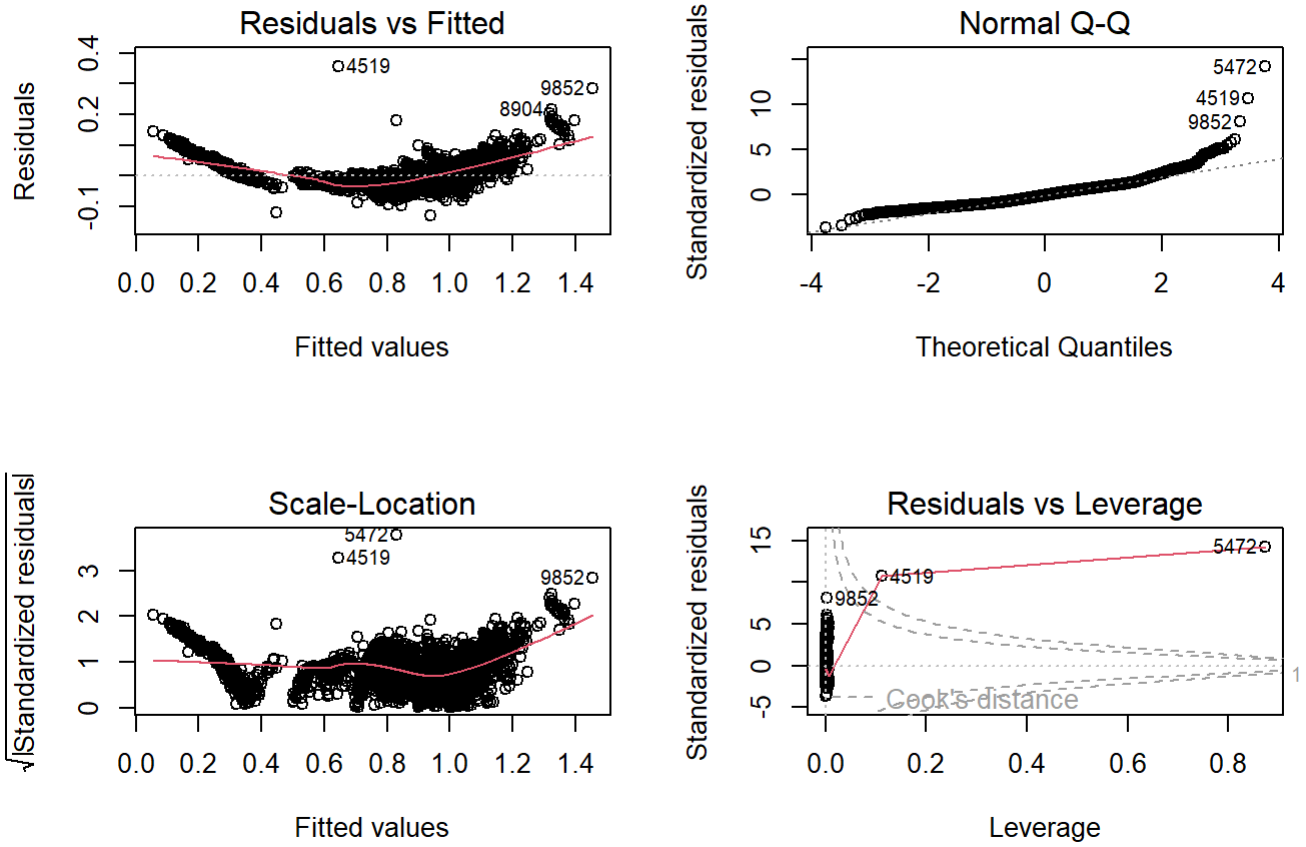
```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.4.1
## ✓ readr   2.1.2      ✓ forcats 0.5.2
## ✓ purrr   0.3.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

```
lm1 <- lm(carat ~., data = train)
summary(lm1)
```

```
##
## Call:
## lm(formula = carat ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12890 -0.02674 -0.00230  0.02173  0.35612
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.713e+00  4.119e-02 -65.872  < 2e-16 ***
## depth        1.737e-02  5.976e-04  29.071  < 2e-16 ***
## table        4.559e-03  2.087e-04  21.843  < 2e-16 ***
## price       -2.228e-05  9.760e-07 -22.824  < 2e-16 ***
## x            2.618e-01  8.915e-03  29.370  < 2e-16 ***
## y            9.813e-02  9.351e-03  10.494  < 2e-16 ***
## z            3.829e-02  8.567e-03   4.469 7.99e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0353 on 5993 degrees of freedom
## Multiple R-squared:  0.9775, Adjusted R-squared:  0.9775
## F-statistic: 4.343e+04 on 6 and 5993 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lm1)
```



Based on the graphs, the data has some outliers but there is a general pattern with the data when trying to predict the karat.

```
pred <- predict(lm1, newdata = test)
cor_lm <- cor(pred, test$carat)
mse_lm <- mean((pred - test$carat)^2)
print(paste("Cor = ", cor_lm))
```

```
## [1] "Cor =  0.987444017043556"
```

```
print(paste("MSE = ", mse_lm))
```

```
## [1] "MSE =  0.00128712386678939"
```

Here we can see that the correlation between the carat and all the predictors is around 98.74% which is really good. This indicates that ## SVM Linear

```
svm1 <- svm(carat~., data=train, kernel = "linear", cost = 10, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = carat ~ ., data = train, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.1666667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  3084
```

```
pred <- predict(svm1, newdata = test)
cor_svm1 <- cor(pred, test$carat)
mse_svm1 <- mean((pred = test$carat)^2)
print(paste("Cor_svm1 = ", cor_svm1))
```

```
## [1] "Cor_svm1 =  0.987513429677802"
```

```
print(paste("MSE_svm1 = ", mse_svm1))
```

```
## [1] "MSE_svm1 =  0.7830374"
```

Using Linear SVM we were able to achieve a similar results. Here we have a 98.75% correlation using SVM with a cost of 10.

Tune

```
tune_svm1 <- tune(svm, carat~., data=vald, kernel="linear", ranges=list(cost=c(0.001, 0.01, 0.1,
1, 5, 10, 100)))
summary(tune_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.001628987
##
## - Detailed performance results:
##    cost       error   dispersion
## 1 1e-03 0.001991230 0.0008725479
## 2 1e-02 0.001628987 0.0010528465
## 3 1e-01 0.001914192 0.0018212007
## 4 1e+00 0.001972603 0.0020297293
## 5 5e+00 0.001974020 0.0020343154
## 6 1e+01 0.001975103 0.0020377706
## 7 1e+02 0.001984071 0.0020663544
```

Evaluate Best Linear SVM

```
pred <- predict(tune_svm1$best.model, newdata=test)
cor_svm1_tune <- cor(pred, test$carat)
mse_svm1_tune <- mean((pred = test$carat)^2)
print(paste("Cor_svm1_tune = ", cor_svm1_tune))
```

```
## [1] "Cor_svm1_tune =  0.982346473476382"
```

```
print(paste("MSE_svm1_tune = ", mse_svm1_tune))
```

```
## [1] "MSE_svm1_tune =  0.7830374"
```

Here we used the best model which was using a cost of .001 in order to find the best correlation between the predictors and the karat. However, it seems that using a cost of .001 did not give us the best correlation as it falls less than the correlation of a cost of 10. Our error and dispersion is also really low so there will be little to no difference between the costs.

Polynomial Kernel

```
svm2 <- svm(carat~., data=train, kernel="polynomial", cost=10, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = carat ~ ., data = train, kernel = "polynomial", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##       gamma:  0.1666667
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  4801
```

```
pred <- predict(svm2, newdata=test)
cor_svm2 <- cor(pred, test$carat)
mse_svm2 <- mean((pred = test$carat)^2)
print(paste("Cor_svm2 = ", cor_svm2))
```

```
## [1] "Cor_svm2 =  0.442051308462712"
```

```
print(paste("MSE_svm2 = ", mse_svm2))
```

```
## [1] "MSE_svm2 =  0.7830374"
```

Here we can see that the correlation of the Polynomial kernel shows that there is only a 44% correlation between the predictors and the karat. We didn't bother testing different cost because there will probably not be a big difference from 44%. Lets move on to Radial kernel and see if it is better than linear. ## Radial Kernel

```
svm3 <- svm(carat~., data=train, kernel="radial", cost=10,gamma=1, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = carat ~ ., data = train, kernel = "radial", cost = 10,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  808
```

```
pred <- predict(svm3, newdata=test)
cor_svm3 <- cor(pred, test$carat)
mse_svm3 <- mean((pred = test$carat)^2)
print(paste("Cor_svm3 = ", cor_svm3))
```

```
## [1] "Cor_svm3 =  0.990946812730196"
```

```
print(paste("MSE_svm3 = ", mse_svm3))
```

```
## [1] "MSE_svm3 =  0.7830374"
```

Amazing! As we can see, using a radial kernel with a cost of 10 resulted in a 99% correlation between the predictors and the karat. This is the best kernel to use to predict the karat based on the predictors. ## Tune Hyperperameters

```
set.seed(1234)
tune.out <- tune(svm, carat~., data = vald, kernel="radial", ranges = list(cost=c(0.1,1,10,100,1
000), gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.001180913
##
## - Detailed performance results:
##     cost gamma       error   dispersion
## 1  1e-01   0.5 0.003788921 0.0017945726
## 2  1e+00   0.5 0.001459705 0.0010655936
## 3  1e+01   0.5 0.001180913 0.0008773303
## 4  1e+02   0.5 0.001223831 0.0008953720
## 5  1e+03   0.5 0.001270969 0.0008961678
## 6  1e-01   1.0 0.006467793 0.0024053446
## 7  1e+00   1.0 0.002412183 0.0013891960
## 8  1e+01   1.0 0.002031266 0.0012252831
## 9  1e+02   1.0 0.002060250 0.0012363858
## 10 1e+03   1.0 0.002081367 0.0012473030
## 11 1e-01   2.0 0.011961581 0.0037831081
## 12 1e+00   2.0 0.004317290 0.0017441987
## 13 1e+01   2.0 0.003512969 0.0015213137
## 14 1e+02   2.0 0.003516127 0.0015206435
## 15 1e+03   2.0 0.003516127 0.0015206435
## 16 1e-01   3.0 0.017251441 0.0048796341
## 17 1e+00   3.0 0.006027340 0.0020738529
## 18 1e+01   3.0 0.005040672 0.0017826969
## 19 1e+02   3.0 0.005040672 0.0017826969
## 20 1e+03   3.0 0.005040672 0.0017826969
## 21 1e-01   4.0 0.022625267 0.0056722606
## 22 1e+00   4.0 0.007739162 0.0024546721
## 23 1e+01   4.0 0.006587752 0.0020892594
## 24 1e+02   4.0 0.006587752 0.0020892594
## 25 1e+03   4.0 0.006587752 0.0020892594
```

Because we know that the radial kernel is the best kernel, we can start trying to tune the gamma to try to find the best gamma.

```
svm4 <- svm(carat~., data = train, kernel = "radial", cost = 100, gamma = 1, scale=TRUE)
summary(svm4)
```

```
##
## Call:
## svm(formula = carat ~ ., data = train, kernel = "radial", cost = 100,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  100
##       gamma:  1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  891
```

```
pred <- predict(svm4, newdata=test)
cor_svm4 <- cor(pred,test$carat)
mse_svm4 <- mean((pred - test$carat)^2)
print(paste("Cor_svm4 = ", cor_svm4))
```

```
## [1] "Cor_svm4 =  0.990721399637288"
```

```
print(paste("MSE_svm4 = ", mse_svm4))
```

```
## [1] "MSE_svm4 =  0.000961828825936622"
```

Here we tested the results if we used a gamma of 1 and got a 99% correlation which is pretty good but we can get a better correlation.

```
svm4 <- svm(carat~., data = train, kernel = "radial", cost = 100, gamma = 0.5, scale=TRUE)
summary(svm4)
```

```
##
## Call:
## svm(formula = carat ~ ., data = train, kernel = "radial", cost = 100,
##     gamma = 0.5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  100
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  766
```

```
pred <- predict(svm4, newdata=test)
cor_svm4 <- cor(pred,test$carat)
mse_svm4 <- mean((pred - test$carat)^2)
print(paste("Cor_svm4 = ", cor_svm4))
```

```
## [1] "Cor_svm4 =  0.993858184309726"
```

```
print(paste("MSE_svm4 = ", mse_svm4))
```

```
## [1] "MSE_svm4 =  0.000631884086905907"
```

Here we can see that having a higher Cost and lower gamma will results in a higher correlation. We were able to get a 99.38% correlation with a gamma of 0.5 and a cost of 100. This is the best result for finding the the karat based on the predictors.