SVM_Regression

2022-10-22

Dataset

It was hard to find good dataset for ML.Therefore I will use Diamond dataset from ggplot2

```
library(ggplot2)
data("diamonds")
df=diamonds
colnames(df)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

```
str(df)
```

```
## tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

50k dataset, so i will cut 10k row for faster run.

```
df <- df[1:10000,]
str(df)
```

```
## tibble [10,000 x 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:10000] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num [1:10000] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num [1:10000] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int [1:10000] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:10000] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num [1:10000] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num [1:10000] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

divide train/test

Use 80% of df as train set and 20% for test set. Also, for faster svm control, i am gonna use another dataset with sampled 100 row of df, since 1k+ row data crashes during tuning in my computer.

```
set.seed(1234)
i <- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
set.seed(1235)
j <- sample(1:nrow(df), 0.01*nrow(df), replace=FALSE)
t_sample<-df[j,]
str(t_sample)
```

```
## tibble [100 x 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:100] 0.79 1.12 1.11 1.01 0.96 1.05 0.7 0.72 1.02 0.84 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 4 1 3 5 4 5 2 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 4 5 4 2 6 4 2 5 2 2 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 3 2 2 2 2 2 7 7 2 3 ...
##  $ depth  : num [1:100] 62.3 62.4 58.8 57.9 62.7 59.6 61.7 62.5 64.2 63.4 ...
##  $ table  : num [1:100] 57 58 59 57 58 57 59 57 59 55 ...
##  $ price  : int [1:100] 2898 4065 4692 3916 3286 3299 3265 2946 4414 2813 ...
##  $ x      : num [1:100] 5.9 6.58 6.8 6.51 6.2 6.58 5.67 5.7 6.26 6 ...
##  $ y      : num [1:100] 5.85 6.63 6.76 6.55 6.24 6.53 5.71 5.73 6.2 5.95 ...
##  $ z      : num [1:100] 3.66 4.12 3.99 3.78 3.9 3.91 3.51 3.57 4 3.79 ...
```

Data exploration

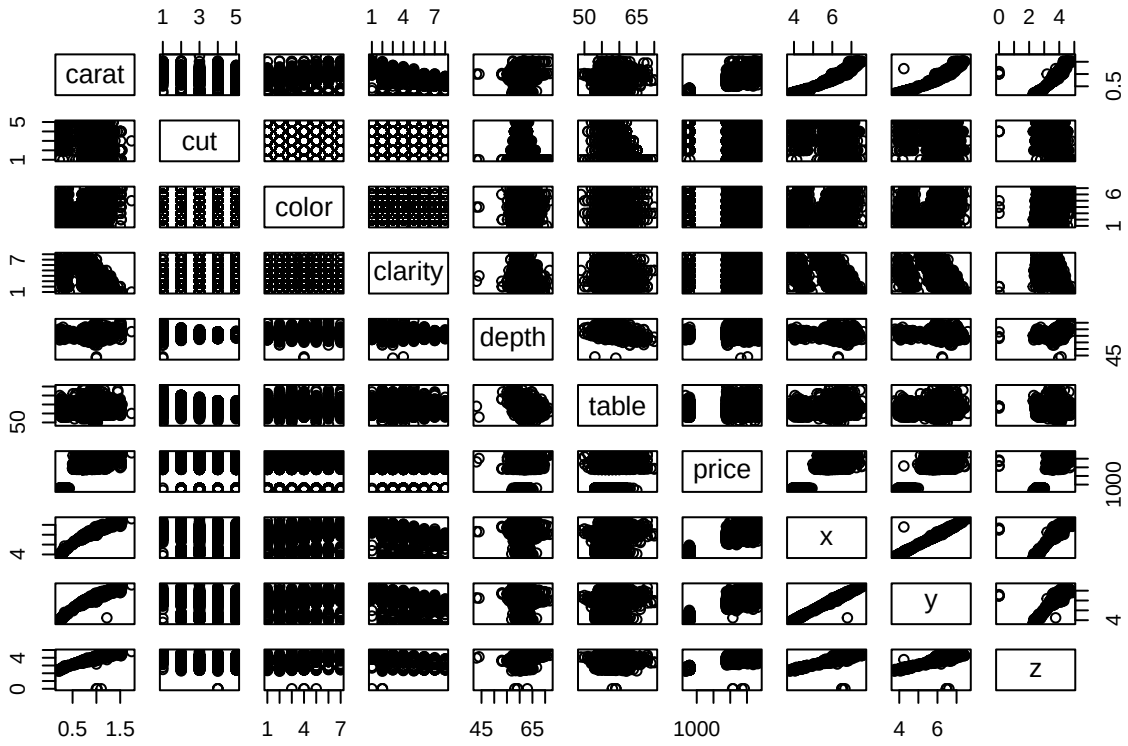
1. look at the training data statistically

```
summary(train)
```

```
##      carat               cut        color        clarity          depth      
##  Min.   :0.2000   Fair      : 415   D:1066   SI2    :2522   Min.   :43.00  
##  1st Qu.:0.7200   Good      :1063   E:1434   SI1    :2158   1st Qu.:61.00  
##  Median :0.9000   Very Good:2029   F:1476   VS2    :1322   Median :61.90  
##  Mean   :0.8474   Premium   :1983   G:1380   VS1    : 958   Mean   :61.87  
##  3rd Qu.:1.0100   Ideal     :2510   H:1308   VVS2   : 421   3rd Qu.:62.80  
##  Max.   :1.7400                    I: 855   VVS1   : 307   Max.   :71.80  
##                                    J: 481   (Other): 312                  
##      table           price            x                y              z       
##  Min.   :49.00   Min.   : 326   Min.   :3.790   Min.   :3.750   Min.   :0.000  
##  1st Qu.:56.00   1st Qu.:3032   1st Qu.:5.780   1st Qu.:5.790   1st Qu.:3.560  
##  Median :58.00   Median :3629   Median :6.130   Median :6.140   Median :3.820  
##  Mean   :57.83   Mean   :3415   Mean   :5.991   Mean   :5.991   Mean   :3.705  
##  3rd Qu.:59.00   3rd Qu.:4206   3rd Qu.:6.400   3rd Qu.:6.390   3rd Qu.:3.970  
##  Max.   :70.00   Max.   :4704   Max.   :7.620   Max.   :7.590   Max.   :4.870  
##
```

2. look at the training data graphically

```
pairs(train)
```



SVM Regression

This time, i will predict price from carat, depth, table, x,y,z.

##build svm model

tune parameters according to kernel. Here, i will use t_sample dataset from above to reduce tuning time.

```
library(e1071)
model1 <- tune.svm(price~carat+depth+table+x+y+z,data=t_sample, kernel="radial",gamma=2^(-5:0), cost=2^
model2 <- tune.svm(price~carat+depth+table+x+y+z,data=t_sample,kernel="linear",gamma=2^(-5:0),cost=2^(0
model3 <- tune.svm(price~carat+depth+table+x+y+z,data=t_sample, kernel="polynomial",gamma=2^(-5:0),cost=

model1$best.parameters
```

```
##     gamma cost
## 1 0.03125    1
```

```
model2$best.parameters
```

```
##      gamma cost
## 13 0.03125    4
```

```
model3$best.parameters
```

```
##     gamma cost
## 1 0.03125    1
```

so, using these best parameters for each kernel, I can model svm.

```
svm1<-svm(price~carat+depth+table+x+y+z,data=train, kernel="radial",gamma=0.03125, cost=1)
svm2<-svm(price~carat+depth+table+x+y+z,data=train, kernel="linear",gamma=0.03125, cost=4)
svm3<-svm(price~carat+depth+table+x+y+z,data=train, kernel="polynomial", gamma=0.03125, cost =1)
```

check result of each cases

```
summary(svm1)
```

```
##
## Call:
## svm(formula = price ~ carat + depth + table + x + y + z, data = train,
##     kernel = "radial", gamma = 0.03125, cost = 1)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.03125
##     epsilon:  0.1
##
##
## Number of Support Vectors:  6068
```

```
summary(svm2)
```

```
##
## Call:
## svm(formula = price ~ carat + depth + table + x + y + z, data = train,
##     kernel = "linear", gamma = 0.03125, cost = 4)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  4
##       gamma:  0.03125
##     epsilon:  0.1
##
##
## Number of Support Vectors:  6442
```

```
summary(svm3)
```

```
##
## Call:
## svm(formula = price ~ carat + depth + table + x + y + z, data = train,
##     kernel = "polynomial", gamma = 0.03125, cost = 1)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##       gamma:  0.03125
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  7030
```

from the model, predict accuracy with test dataset and visualize.

```
svmpredict1<-predict(object = svm1, newdata= test)
svmpredict2<-predict(object = svm2, newdata= test)
svmpredict3<-predict(object = svm3, newdata= test)
#svm1
corsvm1<- cor(svmpredict1,test$price)
msesvm1<- mean((svmpreict1=test$price)^2)
rmsesvm1<- sqrt(msesvm1)
print(paste('correlation:', corsvm1))
```

```
## [1] "correlation: 0.922938630010805"
```

```
print(paste('mse:', msesvm1))
```

```
## [1] "mse: 12648001.4225"
```

```
print(paste('rmse:', rmsesvm1))
```

```
## [1] "rmse: 3556.40287685465"
```

```
#svm2
corsvm2<- cor(svmpredict2,test$price)
msesvm2<- mean((svmpreict2=test$price)^2)
rmsesvm2<- sqrt(msesvm2)
print(paste('correlation:', corsvm2))
```

```
## [1] "correlation: 0.916359733472687"
```

```
print(paste('mse:', msesvm2))
```

```
## [1] "mse: 12648001.4225"
```

```
print(paste('rmse:', rmsesvm2))
```

```
## [1] "rmse: 3556.40287685465"
```

```
#svm3
corsvm3<- cor(svmpredict3,test$price)
msesvm3<- mean((svmpreict3=test$price)^2)
rmsesvm3<- sqrt(msesvm3)
print(paste('correlation:', corsvm3))
```

```
## [1] "correlation: 0.856428582070373"
```

```
print(paste('mse:', msesvm3))
```

```
## [1] "mse: 12648001.4225"
```

```
print(paste('rmse:', rmsesvm3))
```

```
## [1] "rmse: 3556.40287685465"
```

Model	Correlation	RMSE
Radial	0.922938630010805	3556.40287685465
Linear	0.916359733472687	3556.40287685465
Polynomial	0.856428582070373	3556.40287685465

Radial kerner showed the best accuracy and polynomial the worst.

what if we don't use tuned parameter?

```
dsvm1<-svm(price~carat+depth+table+x+y+z,data=train, kernel="radial")
dsvm2<-svm(price~carat+depth+table+x+y+z,data=train, kernel="linear")
dsvm3<-svm(price~carat+depth+table+x+y+z,data=train, kernel="polynomial")
summary(dsvm1)
```

```
##
## Call:
## svm(formula = price ~ carat + depth + table + x + y + z, data = train,
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1666667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  5843
```

```
summary(dsvm2)
```

```
##
## Call:
## svm(formula = price ~ carat + depth + table + x + y + z, data = train,
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.1666667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  6447
```

```
summary(dsvm3)
```

```
##
## Call:
## svm(formula = price ~ carat + depth + table + x + y + z, data = train,
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##       gamma:  0.1666667
##      coef.0:  0
```

```
##     epsilon:  0.1
##
##
## Number of Support Vectors:  6884
```

```
dsvmpredict1<-predict(object = dsvm1, newdata= test)
dsvmpredict2<-predict(object = dsvm2, newdata= test)
dsvmpredict3<-predict(object = dsvm3, newdata= test)
#svm1
cordsvm1<- cor(dsvmpredict1,test$price)
msedsvm1<- mean((dsvmpreict1=test$price)^2)
rmsedsvm1<- sqrt(msedsvm1)
print(paste('correlation:', cordsvm1))
```

```
## [1] "correlation: 0.930808002142208"
```

```
print(paste('mse:', msedsvm1))
```

```
## [1] "mse: 12648001.4225"
```

```
print(paste('rmse:', rmsedsvm1))
```

```
## [1] "rmse: 3556.40287685465"
```

```
#svm2
cordsvm2<- cor(dsvmpredict2,test$price)
msedsvm2<- mean((dsvmpreict2=test$price)^2)
rmsedsvm2<- sqrt(msedsvm2)
print(paste('correlation:', cordsvm2))
```

```
## [1] "correlation: 0.916340292383689"
```

```
print(paste('mse:', msedsvm2))
```

```
## [1] "mse: 12648001.4225"
```

```
print(paste('rmse:', rmsedsvm2))
```

```
## [1] "rmse: 3556.40287685465"
```

```
#svm3
cordsvm3<- cor(dsvmpredict3,test$price)
msedsvm3<- mean((dsvmpreict3=test$price)^2)
rmsedsvm3<- sqrt(msedsvm3)
print(paste('correlation:', cordsvm3))
```

```
## [1] "correlation: 0.883782251440509"
```

```
print(paste('mse:', msedsvm3))
```

```
## [1] "mse: 12648001.4225"
```

```
print(paste('rmse:', rmsedsvm3))
```

```
## [1] "rmse: 3556.40287685465"
```

Model	Correlation	RMSE
Radial	0.922938630010805	3556.40287685465
Radial default	0.930808002142208	3556.40287685465
Linear	0.916359733472687	3556.40287685465

Model	Correlation	RMSE
Linear default	0.916340292383689	3556.40287685465
Polynomial	0.856428582070373	3556.40287685465
Polynomial default	0.883782251440509	3556.40287685465

From default value, accuracy of radial kernel increased 1%p and polynomial kernel increased 3%p. From this result, we can say that there was a problem of tuning with 100 sample that 100 samples cannot represent the full 10k data or underfiting had occurred this time.

However, the result of accuracy rank didn't change. This is because radial svm is for higher(over 4D) dimension dataset. Polynomial kernel uses (x,y) -> (xy$2^{0.5,x}$2,y^2) to calculate 2D data as 3D data. Since we used 7 features from Diamond dataset, polynomial function should have been weak.