SVM_Classification

2022-10-23

Dataset

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

```
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 ...
```

Data Cleaning

Using classification, i will find cut from carat, depth, table, x,y,z values.

Before classification, for better and faster result, i am going to factor dataset a bit.

```
help("diamonds")
```

From description, we can find that cut has 5 level-fair, good, very good, ideal and premium. Here, I will divide level into 2 level, premium+ideal vs fair,good,very good.

```
df$cut_status<-NA
df$cut_status[df$cut=='Fair'|df$cut=='Good'|df$cut=='Very Good']=0
df$cut_status[df$cut=='Premium'|df$cut=='Ideal']=1
df$cut_status=as.factor(df$cut_status)
```

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 11] (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 ...
##  $ cut_status: Factor w/ 2 levels "0","1": 2 2 2 1 1 2 2 2 1 1 ...
```

Data exploration

1. look at the training data statistically

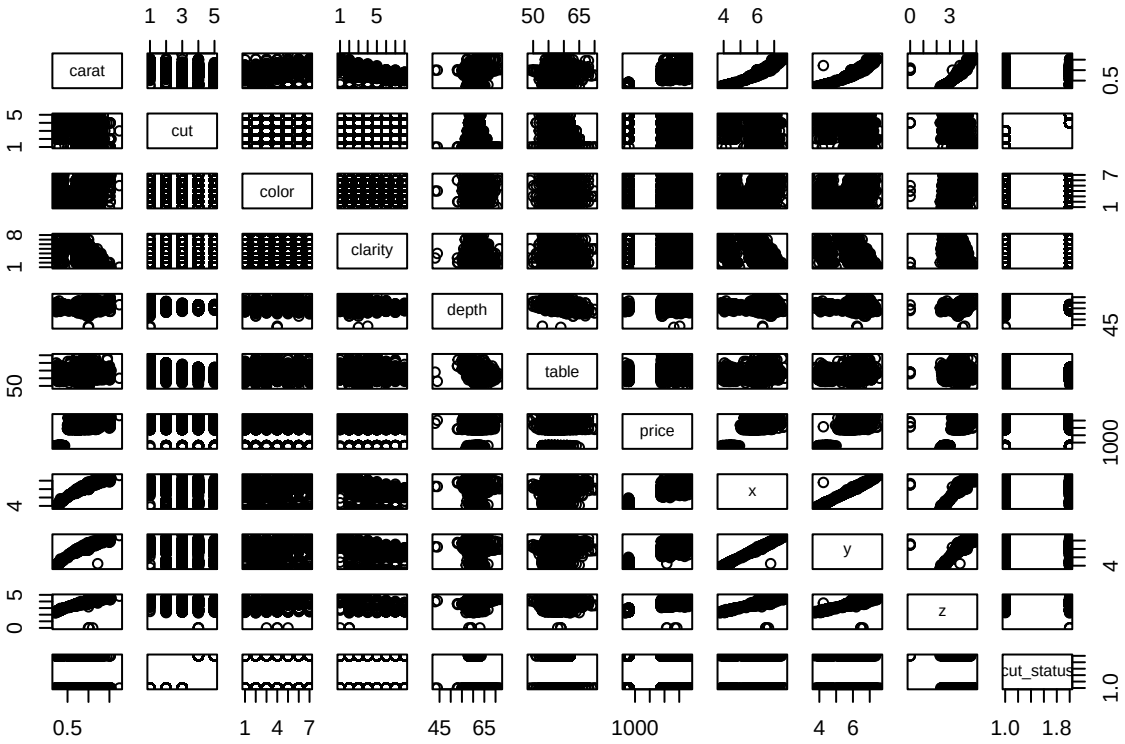
```
summary(train)
```

```
##      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
##
##  cut_status
##  0:3507
##  1:4493
##
##
##
##
##
```

2. look at the training data graphically

```
pairs(train)
```



SVM Classification

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

model1$best.parameters
```

```
##    gamma cost
## 16  0.25    4
```

```
model2$best.parameters
```

```
##      gamma cost
## 25 0.03125   16
```

```
model3$best.parameters
```

```
##    gamma cost
## 12     1    2
```

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

```
svm1<-svm(cut_status~carat+depth+table+x+y+z,data=train, kernel="radial",gamma=0.25, cost=4)
svm2<-svm(cut_status~carat+depth+table+x+y+z,data=train, kernel="linear",gamma=0.03125, cost=16)
svm3<-svm(cut_status~carat+depth+table+x+y+z,data=train, kernel="polynomial", gamma=1, cost =2)
```

check result of each cases

```
summary(svm1)
```

```
##
## Call:
## svm(formula = cut_status ~ carat + depth + table + x + y + z, data = train,
##     kernel = "radial", gamma = 0.25, cost = 4)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  4
##
## Number of Support Vectors:  3066
##
##  ( 1534 1532 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
summary(svm2)
```

```
##
## Call:
## svm(formula = cut_status ~ carat + depth + table + x + y + z, data = train,
##     kernel = "linear", gamma = 0.03125, cost = 16)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  16
##
## Number of Support Vectors:  4787
##
##  ( 2394 2393 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
summary(svm3)
```

```
##
## Call:
## svm(formula = cut_status ~ carat + depth + table + x + y + z, data = train,
##     kernel = "polynomial", gamma = 1, cost = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  2
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  4382
##
##  ( 2193 2189 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

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
table(svmpredict1,test$cut_status)
```

```
##
## svmpredict1    0    1
##           0  674  103
##           1  158 1065
```

```
mean(svmpredict1==test$cut_status)
```

```
## [1] 0.8695
```

```
#svm2
table(svmpredict2,test$cut_status)
```

```
##
## svmpredict2   0   1
##           0 518 185
##           1 314 983
```

```
mean(svmpredict2==test$cut_status)
```

```
## [1] 0.7505
```

```
#svm3
table(svmpredict3,test$cut_status)
```

```
##
## svmpredict3    0    1
##           0  468   86
##           1  364 1082
```

```
mean(svmpredict3==test$cut_status)
```

```
## [1] 0.775
```

Model	Accuracy
Radial	0.8695
Linear	0.7505
Polynomial	0.775

Radial kerner showed the best accuracy and Linear the worst.

what if we don't use tuned parameter?

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

```
##
## Call:
## svm(formula = cut_status ~ carat + depth + table + x + y + z, data = train,
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  3690
##
##  ( 1843 1847 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
summary(dsvm2)
```

```
##
## Call:
## svm(formula = cut_status ~ carat + depth + table + x + y + z, data = train,
```

```
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  4911
##
##  ( 2454 2457 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
summary(dsvm3)
```

```
##
## Call:
## svm(formula = cut_status ~ carat + depth + table + x + y + z, data = train,
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  5619
##
##  ( 2811 2808 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
dsvmpredict1<-predict(object = dsvm1, newdata= test)
dsvmpredict2<-predict(object = dsvm2, newdata= test)
dsvmpredict3<-predict(object = dsvm3, newdata= test)
#dsvm1
table(dsvmpredict1,test$cut_status)
```

```
##
## dsvmpredict1    0    1
##            0  615   76
##            1  217 1092
```

```
mean(dsvmpredict1==test$cut_status)
```

```
## [1] 0.8535
```

```
#dsvm2
table(dsvmpredict2,test$cut_status)
```

```
##
## dsvmpredict2   0   1
##            0 510 182
##            1 322 986
```

```
mean(dsvmpredict2==test$cut_status)
```

```
## [1] 0.748
```

```
#dsvm3
table(dsvmpredict3,test$cut_status)
```

```
##
## dsvmpredict3    0    1
##            0  290   55
##            1  542 1113
```

```
mean(dsvmpredict3==test$cut_status)
```

```
## [1] 0.7015
```

Model	Accuracy
Radial	0.8695
Radial default	0.8535
Linear	0.7505
Linear default	0.748
Polynomial	0.775
Polynomial default	0.7015

This time, models using best parameter of sampled dataset showed better result. Unlike regression, classification predicts descrete data therefore it was easier to represent original dataset with sampled dataset.

As we saw from SVM regression, radial model made the best prediction. this is because we used 6features to predict 1 feature, or the dataset is 7D. radial svm is for higher(over 4D) dimension dataset. Polynomial kernel uses (x,y) -> (xy$2^{0.5,x}$2,yˆ2) to calculate 2D data as 3D data, and this was the reason that polynomial model was hard to predict 7D data.