SVM

Qi Yuchen, yq2279; Jiafei Li, jl5548; Gaotong Liu,

2020/5/16

Introduction

This final project aims to find an optimal model in order to better predict red wine quality based on physicochemical tests. The original dataset contains 1599 observations of 11 covariates from physicochemical test, and 1 response variable (wine quality), describing features of the Portuguese red wine "Vinho Verde".

Data Preparation

First, we identify the missing values in the dataset. As a result, there is no variable containing missing data.

```
df.raw = read_csv("winequality-red.csv")
# check NA data
df.na = is.na(df.raw)
var.na = colSums(df.na)
```

Then we clean the dataset, and get the training and test data.

```
df = df.raw %>%
  janitor::clean_names() %>%
  mutate(quality = factor(quality, labels = c("q3", "q4", "q5", "q6", "q7", "q8"))) %>%
  mutate(quality = fct_collapse(quality,
                                poor = c("q3", "q4", "q5"),
                                good = c("q6", "q7", "q8")))

set.seed(1)
rowTrain <- createDataPartition(y = df$quality,
                                p = 2/3,
                                list = FALSE)
df.train = df[rowTrain,]
x = model.matrix(quality~., df.train)[,-1]
y = df.train$quality
df.test = df[-rowTrain,]

levels(df$quality) # 2 levels: poor and good
```

```
## [1] "poor" "good"
```

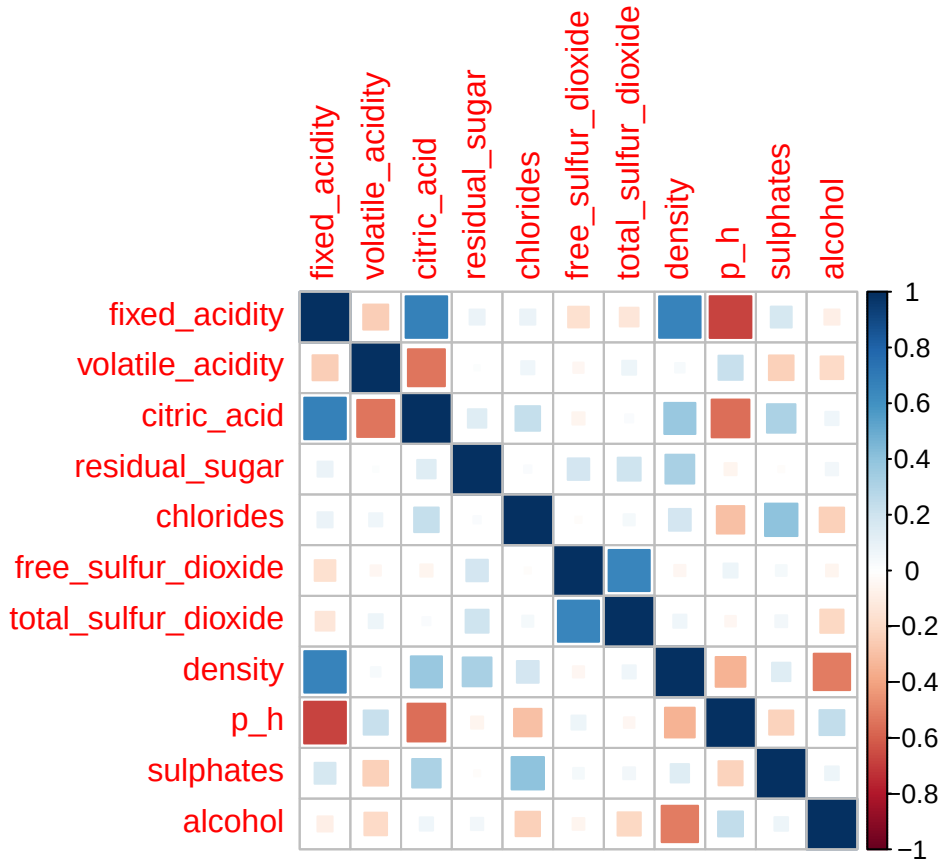
Exploratory analysis

Correlations

11 predictors are all numerical variables. There is no strong correlation (>0.7) between them.

```
var.numerical = df.train %>% dplyr::select_if(is.numeric) %>% as.matrix()
var.cor = cor(var.numerical)
```

```
corrplot(cor(var.numerical), method = "square", type = "full")
```



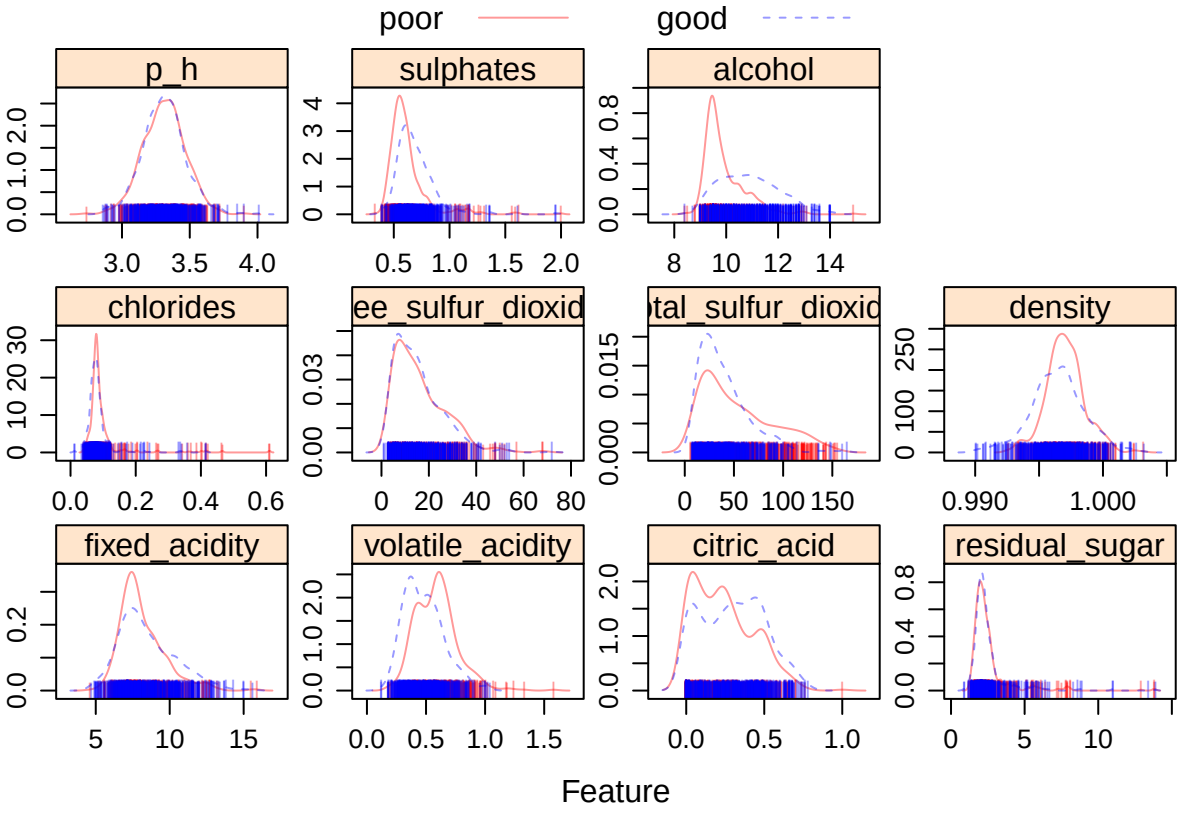

```
which((var.cor > 0.7 & var.cor < 1), arr.ind = TRUE)
```

```
##      row col
```

Scatter plot

```
theme1 <- transparentTheme(trans = .4)
trellis.par.set(theme1)

featurePlot(x,
            y,
            scales = list(x=list(relation="free"),
                          y=list(relation="free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```

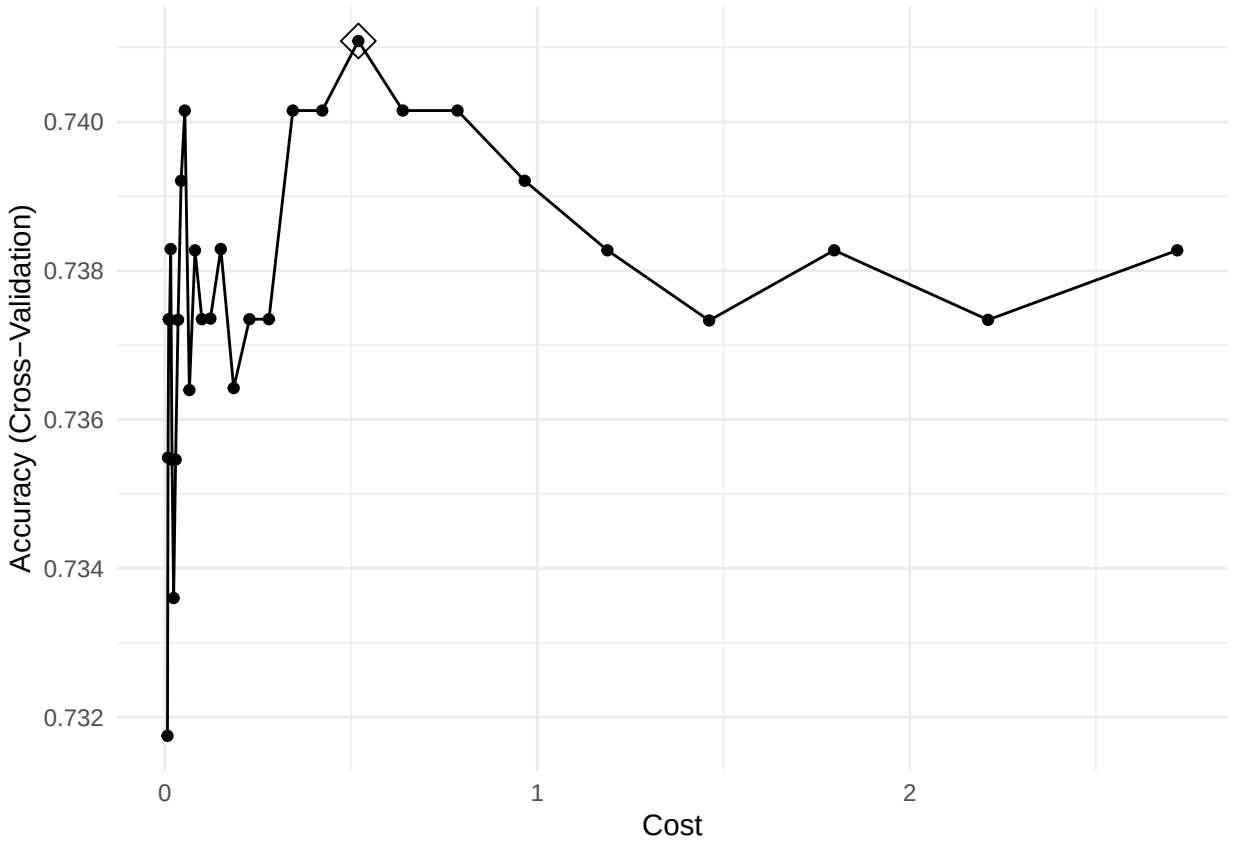


Models

Support Vector Machine

Linear kernel

```
ctrl <- trainControl(method = "cv")
set.seed(1)
svmlinear.fit <- train(quality~.,
                   data = df.train,
                   method = "svmLinear2",
                   preProcess = c("center", "scale"),
                   tuneGrid = data.frame(cost = exp(seq(-5, 1, len=30))),
                   trControl = ctrl)
ggplot(svmlinear.fit, highlight = TRUE)
```




```
svmlinear.fit$bestTune
```

```
##         cost
## 22 0.5193525
```

```
# train error
pred.svmlinear.train <- predict(svmlinear.fit$finalModel, data = df.train)
confusionMatrix(data = pred.svmlinear.train,
                reference = df$quality[rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction poor good
##       poor  380  155
##       good  116  415
##
##                Accuracy : 0.7458
##                  95% CI : (0.7185, 0.7717)
##     No Information Rate : 0.5347
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4917
##
##  Mcnemar's Test P-Value : 0.02098
##
##             Sensitivity : 0.7661
##             Specificity : 0.7281
```

```
##          Pos Pred Value : 0.7103
##          Neg Pred Value : 0.7815
##              Prevalence : 0.4653
##          Detection Rate : 0.3565
##    Detection Prevalence : 0.5019
##       Balanced Accuracy : 0.7471
##
##        'Positive' Class : poor
##
```

```
# test error
pred.svmlinear.test <- predict(svmlinear.fit, newdata = df.test)
confusionMatrix(data = pred.svmlinear.test,
                reference = df$quality[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction poor good
##       poor  191   80
##       good   57  205
##
##                Accuracy : 0.743
##                  95% CI : (0.7036, 0.7796)
##     No Information Rate : 0.5347
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4865
##
##  Mcnemar's Test P-Value : 0.06016
##
##             Sensitivity : 0.7702
##             Specificity : 0.7193
##          Pos Pred Value : 0.7048
##          Neg Pred Value : 0.7824
##              Prevalence : 0.4653
##          Detection Rate : 0.3583
##    Detection Prevalence : 0.5084
##       Balanced Accuracy : 0.7447
##
##        'Positive' Class : poor
##
```

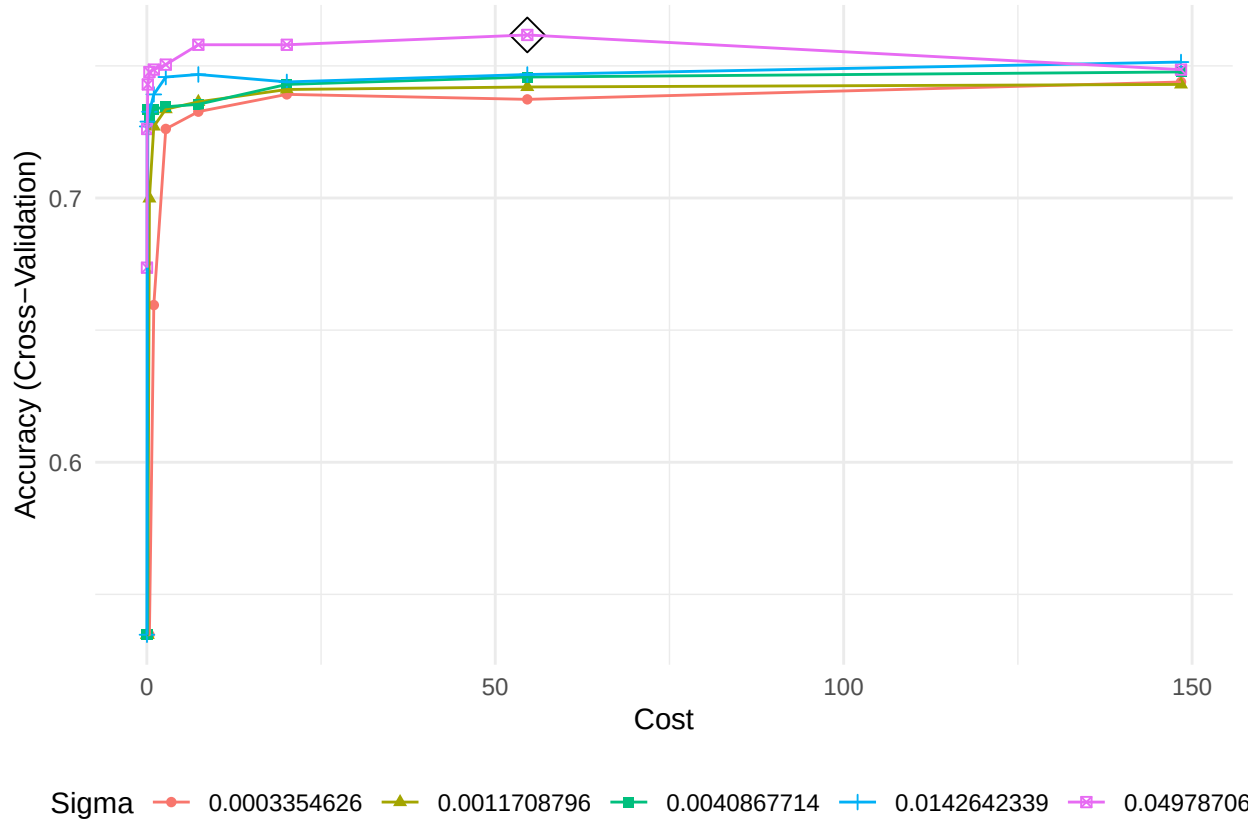
Comment: By 10 fold cross validation using `train()` function from caret package, the best cost tuning parameter is 0.519. Then train misclassification error of this linear SVM on entire train dataset is 1-0.7458 = 0.2542. The test missclassification error is 1-0.743 = 0.257. The test error is slightly greater than train error, so our model seems to be a good fit.

Radial kernel

Different from the linear kernel, radial kernel can construct nonlinear classification boundaries.

```
svmr.grid <- expand.grid(C = exp(seq(-4, 5,len=10)),
                         sigma = exp(seq(-8,-3,len=5)))
set.seed(1)
```

```
svmradial.fit <- train(quality~.,
                       data = df,
                       subset = rowTrain,
                       method = "svmRadial",
                       preProcess = c("center", "scale"),
                       tuneGrid = svmr.grid,
                       trControl = ctrl)
ggplot(svmradial.fit, highlight = TRUE)
```




```
svmradial.fit$bestTune
```

```
##         sigma        C
## 45 0.04978707 54.59815
```

```
# train error
pred.svmradial.train <- predict(svmradial.fit, newdata = df.train)
confusionMatrix(data = pred.svmradial.train,
                reference = df$quality[rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction poor good
##       poor  429   86
##       good   67  484
##
##                Accuracy : 0.8565
##                  95% CI : (0.834, 0.877)
##     No Information Rate : 0.5347
```

```
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7123
##
##  Mcnemar's Test P-Value : 0.1456
##
##             Sensitivity : 0.8649
##             Specificity : 0.8491
##          Pos Pred Value : 0.8330
##          Neg Pred Value : 0.8784
##              Prevalence : 0.4653
##          Detection Rate : 0.4024
##    Detection Prevalence : 0.4831
##       Balanced Accuracy : 0.8570
##
##        'Positive' Class : poor
##
```

```
# test error
pred.svmradial.test <- predict(svmradial.fit, newdata = df.test)
confusionMatrix(data = pred.svmradial.test,
                reference = df$quality[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction poor good
##       poor  187   67
##       good   61  218
##
##                Accuracy : 0.7598
##                  95% CI : (0.7213, 0.7955)
##     No Information Rate : 0.5347
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5181
##
##  Mcnemar's Test P-Value : 0.6585
##
##             Sensitivity : 0.7540
##             Specificity : 0.7649
##          Pos Pred Value : 0.7362
##          Neg Pred Value : 0.7814
##              Prevalence : 0.4653
##          Detection Rate : 0.3508
##    Detection Prevalence : 0.4765
##       Balanced Accuracy : 0.7595
##
##        'Positive' Class : poor
##
```

Comment: The best tuning parameter is sigma = 0.050, C = 54.598, the train error is 1 - 0.8565 = 0.1435, and the test error is 1- 0.7598 = 0.2402. Both the train and test errors are smaller than the linear kernel SVM.

Conclusion