Untitled

Yue Lai

5/15/2020

import data

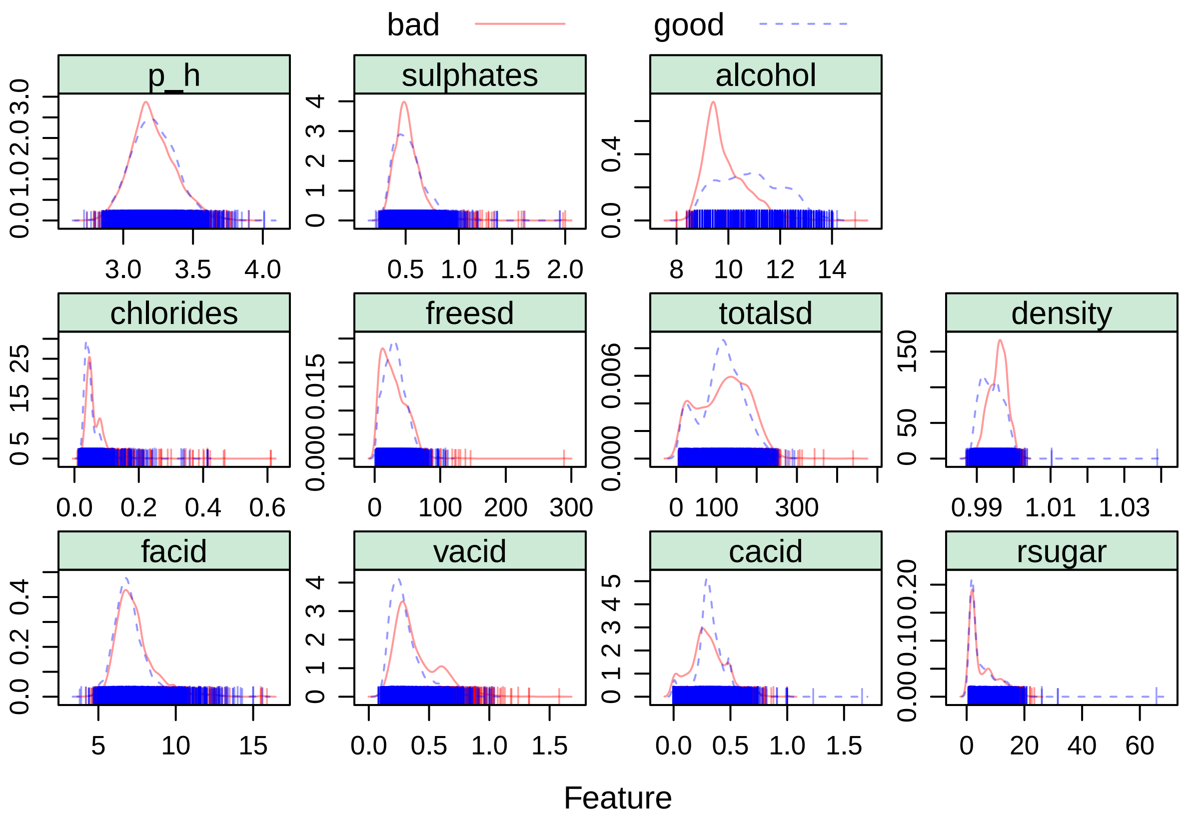
```
red = read_excel(path = "./data/wine.xlsx", sheet = "red") %>%
  janitor::clean_names()
```

```
## Warning in FUN(X[[i]], ...): strings not representable in native encoding will
## be translated to UTF-8
```

```
white = read_excel(path = "./data/wine.xlsx", sheet = "white") %>%
  janitor::clean_names()

wine = data.frame(rbind(red, white))

wine = wine %>%
  mutate(quality = as.factor(ifelse(quality > 5, "good", "bad")))
```

The data contains 6497 observations and 11 variables. The outcome is the binary variable category. We start from some simple visualization of the data.

```
theme1 = transparentTheme(trans = .4)
theme1$strip.background$col = rgb(.0, .6, .2, .2)
trellis.par.set(theme1)

featurePlot(x = wine[,1:11],
            y = wine$quality,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



Divide the data into two part (training and test)

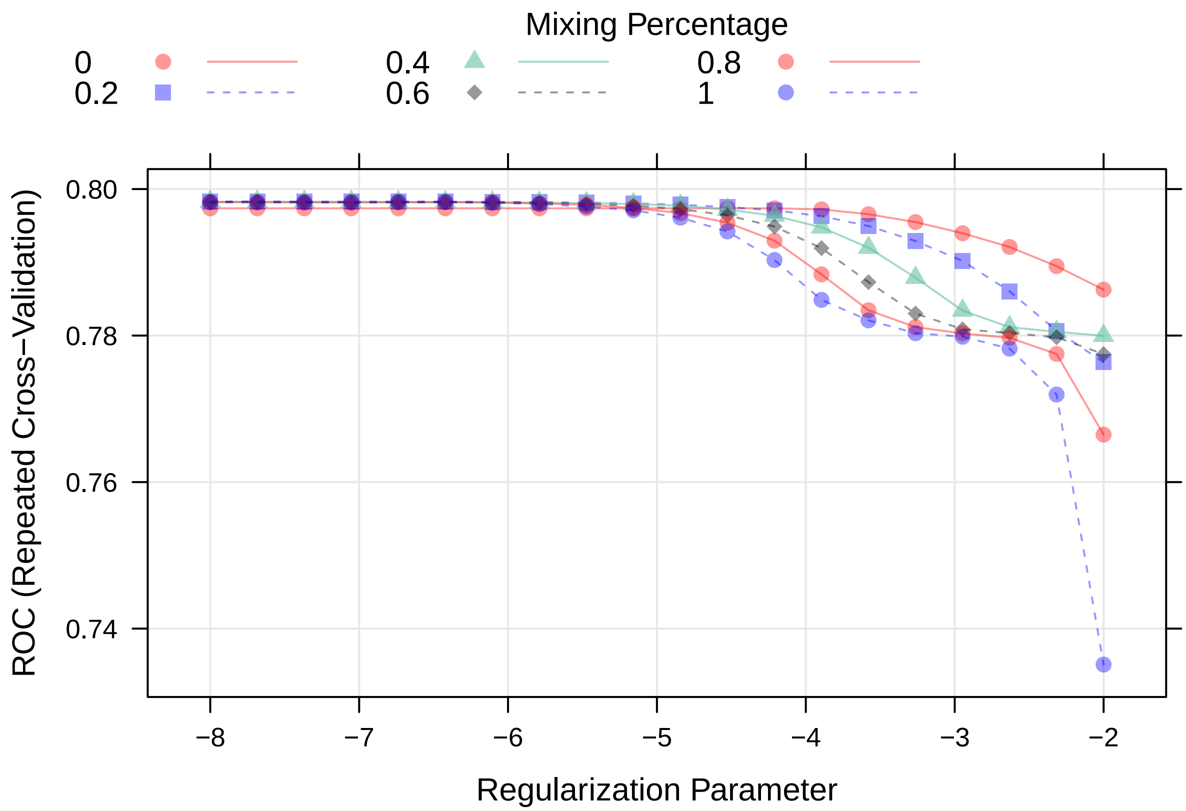
```
rowtrain = createDataPartition(y = wine$quality,
                               p = 2/3,
                               list = FALSE)

ctrl = trainControl(method = "repeatedcv",
                    repeats = 5,
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE)
```

glm

```
set.seed(666)
model.glm = train(x = wine[rowtrain, 1:11],
                  y = wine$quality[rowtrain],
                  method = "glm",
                  metric = "ROC",
                  trControl = ctrl)
```

glmn

```r
set.seed(666)
glmngrid = expand.grid(.alpha = seq(0,1,length = 6),
                       .lambda = exp(seq(-8,-2, length = 20)))

model.glmn = train(x = wine[rowtrain, 1:11],
                   y = wine$quality[rowtrain],
                   method = "glmnet",
                   metric = "ROC",
                   tuneGrid = glmngrid,
                   trControl = ctrl)
plot(model.glmn, xTrans = function(x) log(x))
```




```r
model.glmn$bestTune
```

```
##    alpha      lambda
## 25   0.2 0.001186388
```

LDA

```r
set.seed(666)
model.lda = train(x = wine[rowtrain, 1:11],
                  y = wine$quality[rowtrain],
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl)
```

QDA

```r
set.seed(666)
model.qda = train(x = wine[rowtrain, 1:11],
                  y = wine$quality[rowtrain],
                  method = "qda",
                  metric = "ROC",
                  trControl = ctrl)
```

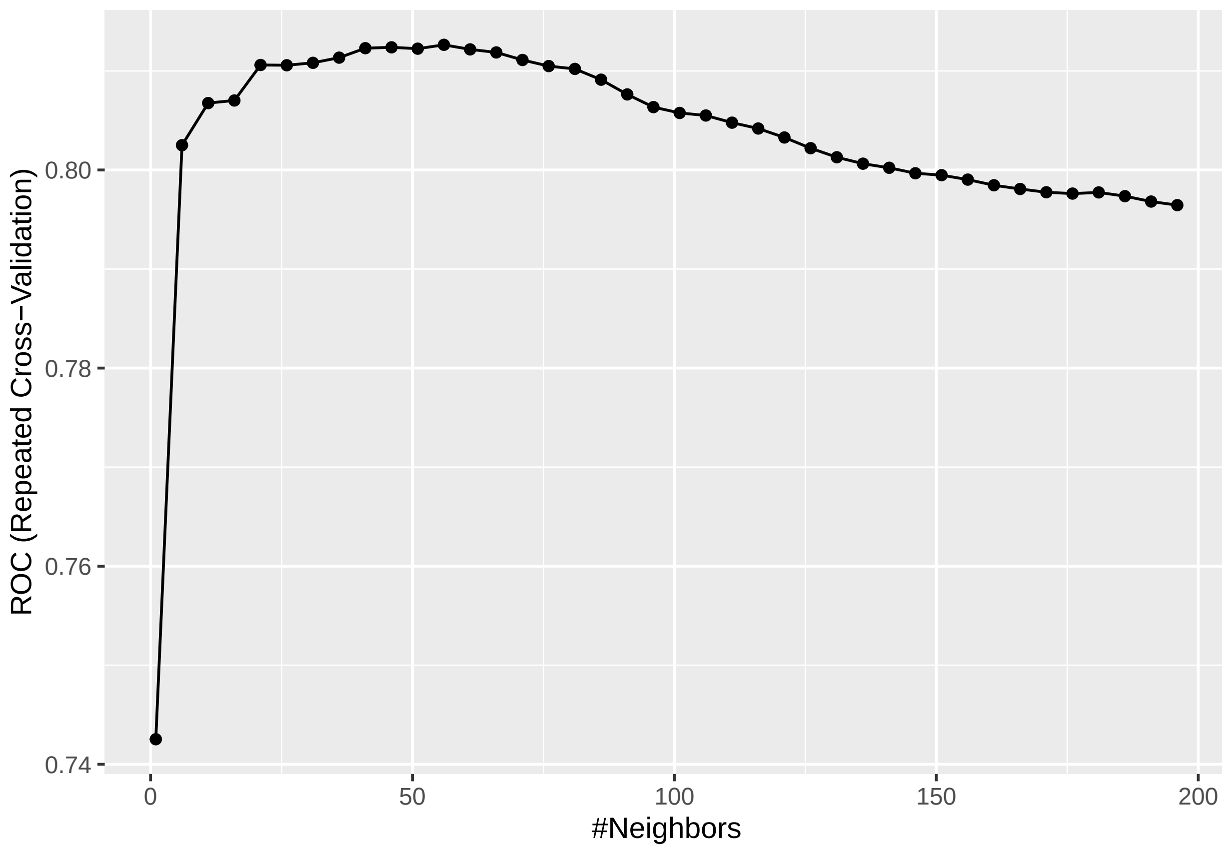
Naive Bayes

KNN

```r
set.seed(666)
model.knn = train(x = wine[rowtrain, 1:11],
                  y = wine$quality[rowtrain],
                  method = "knn",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(k = seq(1, 200, by = 5)),
                  trControl = ctrl)
```

```
## Warning in train.default(x = wine[rowtrain, 1:11], y = wine$quality[rowtrain], :
## The metric "Accuracy" was not in the result set. ROC will be used instead.
```

```r
ggplot(model.knn)
```



summary

```r
res = resamples(list(GLM = model.glm,
                     GLMNET = model.glmn,
                     LDA = model.lda,
                     DQA = model.qda,
#                     NB = model.nb,
                     KNN = model.knn))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMNET, LDA, DQA, KNN
## Number of resamples: 50
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7526052 0.7834629 0.7988679 0.7982395 0.8143995 0.8399440    0
## GLMNET 0.7535923 0.7839088 0.7986507 0.7983034 0.8148295 0.8400817    0
## LDA    0.7496213 0.7841838 0.7993512 0.7981896 0.8144793 0.8398063    0
## DQA    0.7272185 0.7750715 0.7928775 0.7888367 0.8046814 0.8277326    0
```

```
## KNN    0.7577813 0.8012346 0.8175506 0.8126347 0.8247802 0.8557935    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.4716981 0.5408805 0.5660377 0.5651572 0.5959119 0.6603774    0
## GLMNET 0.4716981 0.5408805 0.5691824 0.5655346 0.5911950 0.6540881    0
## LDA    0.4591195 0.5298742 0.5628931 0.5603774 0.5911950 0.6540881    0
## DQA    0.3647799 0.4528302 0.4685535 0.4740881 0.4968553 0.5723270    0
## KNN    0.4591195 0.5110063 0.5440252 0.5417610 0.5691824 0.6289308    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7846715 0.8116490 0.8378898 0.8363204 0.8531022 0.8868613    0
## GLMNET 0.7846715 0.8140378 0.8378898 0.8370490 0.8540146 0.8868613    0
## LDA    0.7810219 0.8189283 0.8394161 0.8379979 0.8531022 0.8905109    0
## DQA    0.8138686 0.8540146 0.8686131 0.8671732 0.8798905 0.9087591    0
## KNN    0.8138686 0.8512774 0.8759124 0.8707506 0.8868613 0.9160584    0
```

test set performance

```r
glm.pred = predict(model.glm, newdata = wine[-rowtrain,], type = "prob")[,2]
glmn.pred = predict(model.glmn, newdata = wine[-rowtrain,], type = "prob")[,2]
lda.pred = predict(model.lda, newdata = wine[-rowtrain,], type = "prob")[,2]
qda.pred = predict(model.qda, newdata = wine[-rowtrain,], type = "prob")[,2]
#nb.pred = predict(model.nb, newdata = wine[-rowtrain,], type = "prob")[,2]
knn.pred = predict(model.knn, newdata = wine[-rowtrain,], type = "prob")[,2]

roc.glm = roc(wine$quality[-rowtrain], glm.pred)
```

```
## Setting levels: control = bad, case = good
```

```
## Setting direction: controls < cases
```

```r
roc.glmn = roc(wine$quality[-rowtrain], glmn.pred)
```

```
## Setting levels: control = bad, case = good
## Setting direction: controls < cases
```

```r
roc.lda = roc(wine$quality[-rowtrain], lda.pred)
```

```
## Setting levels: control = bad, case = good
## Setting direction: controls < cases
```

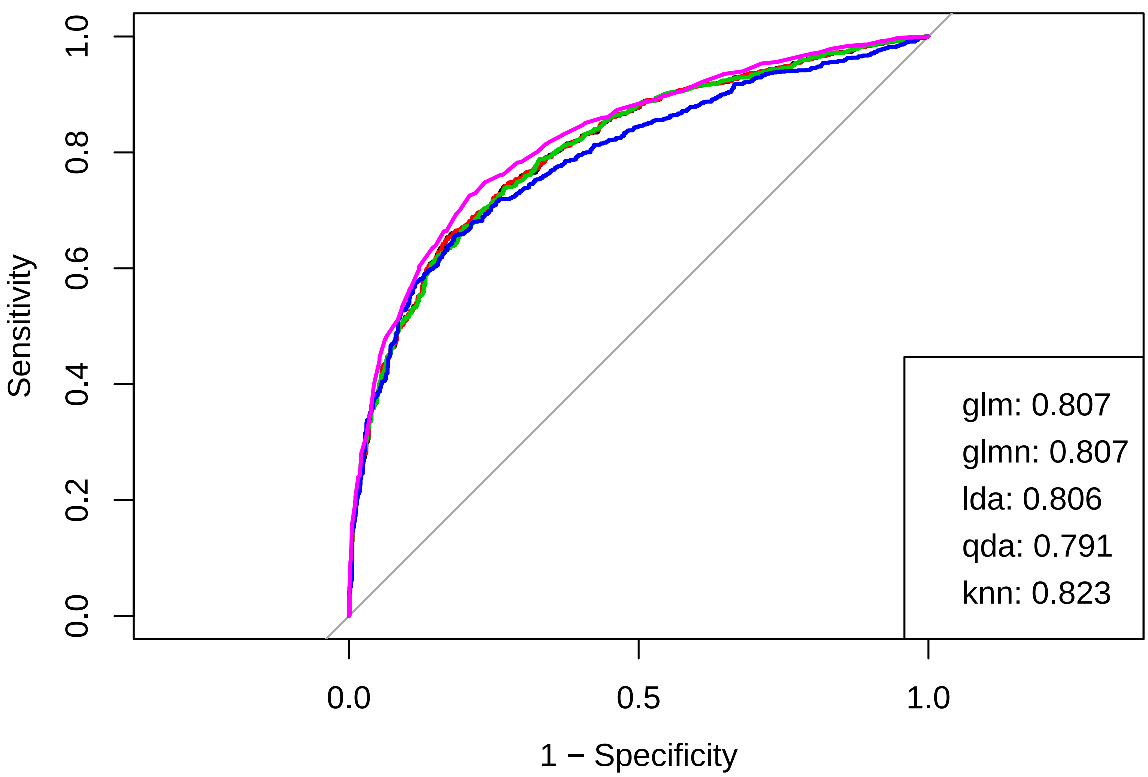
```r
roc.qda = roc(wine$quality[-rowtrain], qda.pred)
```

```
## Setting levels: control = bad, case = good
## Setting direction: controls < cases
```

```r
#roc.nb = roc(wine$quality[-rowtrain], nb.pred)
roc.knn = roc(wine$quality[-rowtrain], knn.pred)
```

```
## Setting levels: control = bad, case = good
## Setting direction: controls < cases
```

```r
auc = c(roc.glm$auc[1], roc.glmn$auc[1], roc.lda$auc[1], roc.qda$auc[1],
#        roc.nb$auc[1],
        roc.knn$auc[1])

plot(roc.glm, legacy.axes = TRUE)
plot(roc.glmn, col = 2, add = TRUE)
plot(roc.lda, col = 3, add = TRUE)
plot(roc.qda, col = 4, add = TRUE)
#plot(roc.nb, col = 5, add = TRUE)
plot(roc.knn, col = 6, add = TRUE)
modelNames = c("glm", "glmn", "lda", "qda",
#               "nb",
               "knn")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc, 3)))
```




```r
glm=1-sum(predict(model.glm, newdata = wine[-rowtrain,])==wine[-rowtrain,12])/nrow(wine[-rowtrain,])
knn=1-sum(predict(model.knn, newdata = wine[-rowtrain,])==wine[-rowtrain,12])/nrow(wine[-rowtrain,])
glmn=1-sum(predict(model.glmn, newdata = wine[-rowtrain,])==wine[-rowtrain,12])/nrow(wine[-rowtrain,])
```

```
lda =1-sum(predict(model.lda, newdata = wine[-rowtrain,])==wine[-rowtrain,12])/nrow(wine[-rowtrain,])
qda =1-sum(predict(model.qda, newdata = wine[-rowtrain,])==wine[-rowtrain,12])/nrow(wine[-rowtrain,])

T1=data.frame(model=c("GLM","GLMN","LDA","QDA","KNN"),Error_rate=c(glm,glmn,lda,qda,knn)) %>% knitr::kal
T1
```

model	Error_rate
GLM	0.2609700
GLMN	0.2586605
LDA	0.2572748
QDA	0.2840647
KNN	0.2508083