STAT 8330 FALL 2015 ASSIGNMENT 1

Peng Shao

September 6, 2015

▶ **Exercises 2.5. Solution.**

(1).

- advantage: can fit many different functional forms; low bias; usually predict more accurately
- disadvantage: overfitting problem; sually hard to interpret; high variance

(2). If our goal is to predict more accurately, it will usually be best to choose a more flexible approach.

(3). If our goal is to make some inferences, we prefer choosing a less flexible approach because the relation between response and predictor is more explicit.

▶ **Exercises 2.6. Solution.**

(1). The essential difference between parametric and non-parametric approach is that, the parametric make an assumption of the form of f, which can reduce problem of estimating f down to one of estimating a set of parameter, but non-parametric do not make explicit assumptions about the functional form of f.

(2).

- advantage: it is easier to estimate parameter; the relation between response and predictor is more explicit;
- disadvantage: the model we choose will usually not match the true unknown form of f; sometimes need more assumption.

▶ **Exercises 2.10. Solution.**

▶ **Exercises 3.5. Solution.**

▶ **Exercises 3.15. Solution.**

▶ **Exercises 4.3. Solution.**

We know that we classify X into kth class based on Bayes' classifier if

$$p_k(x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

is largest among all $p_l(x)$, $l = 1, 2, ..., K$. For 1 dimension, the density of x from kth class is

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$

In comparing two classes k and l, it is sufficient to look at the log-ratio, and we see that

$$\begin{aligned}
\log\left(\frac{p_k(x)}{p_l(x)}\right) &= \log\left(\frac{\pi_k}{\pi_l}\right) + \log\left(\frac{f_k(x)}{f_l(x)}\right) \\
&= \log\left(\frac{\pi_k}{\pi_l}\right) + \log\left(\frac{\sigma_l}{\sigma_k}\right) - \frac{(x-\mu_k)^2}{2\sigma_k^2} + \frac{(x-\mu_l)^2}{2\sigma_l^2} \\
&= \left(-\frac{(x-\mu_k)^2}{2\sigma_k^2} - \log\sigma_k + \log\pi_k\right) - \left(-\frac{(x-\mu_l)^2}{2\sigma_l^2} - \log\sigma_l + \log\pi_l\right) \\
&= \delta_k(x) - \delta_l(x)
\end{aligned}$$

Then the Beyes' classifier can be be defined as

$$C(x) = \arg\max_k \delta_k(x)$$

where $\delta_k(x) = -\frac{(x-\mu_k)^2}{2\sigma_k^2} - \log\sigma_k + \log\pi_k$.

It is obvious that the decision boundary between each pair of classes k and l is described by a quadratic equation $\{x : \delta_k(x) = \delta_l(x)\}$.

▶ **Exercises 4.10. Solution.**

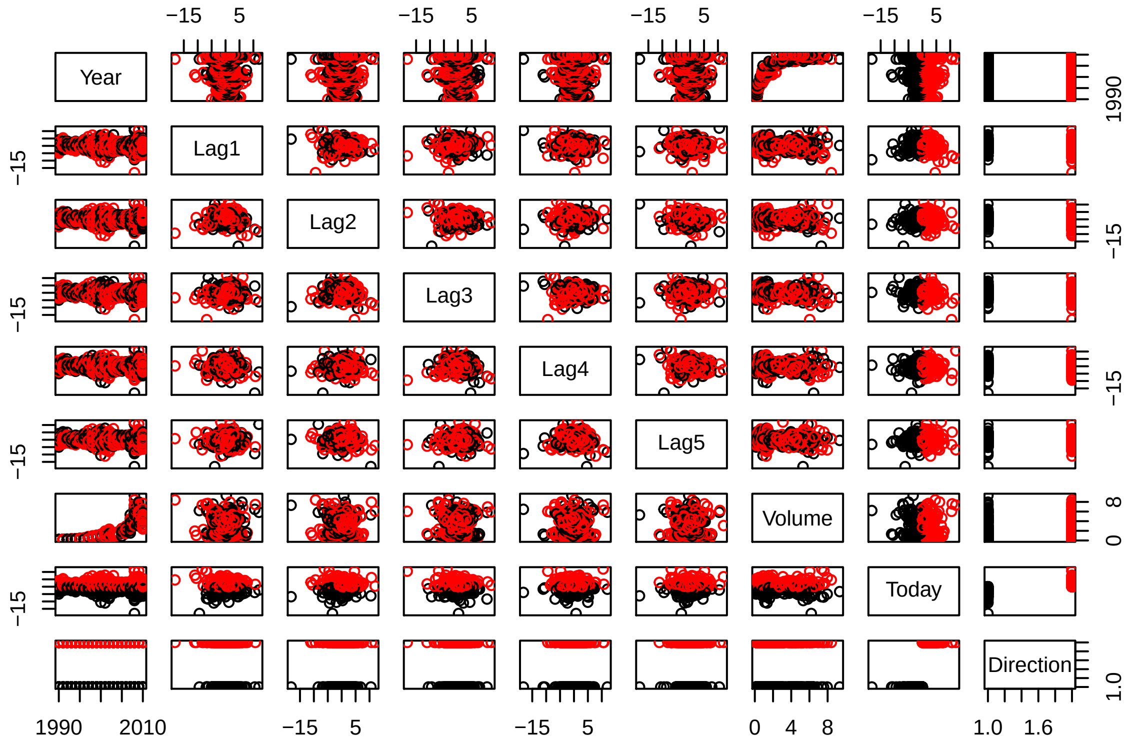
(a). From the output, we can see that (1) the variable Volume is increased as the Year increased, and the increase rates become larger and larger; (2) the variable Today is highly, but not complete, correlated with the indicator variable Direction, so we may guess that Direction is transformed from Today. Except this two pairs, no other pairs show any obvious patterns.

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
cor(Weekly[, -9])
```

```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

```
pairs(Weekly, col = Direction)
```



(b). Fitting the model as below, the summary result shows that only intercept and coefficient of Lag2 is significant.

```
logit.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                 family = binomial, data = Weekly)
summary(logit.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
```

```
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

(c). Using threshold = 0.5, we can get the confusion table as below. At a first glance, the prediction accuracy is 56.11% and the error rate is 43.89%, which is not so good but acceptable, since the prediction of trend of stock index is so difficult. The true positive rate is so good as 92.07%, but the false positive is also too high (almost 90%), which is catastrophic. So we should review the accuracy and error rate. Suppose we have a trivial classifier which always predict "UP". Then the error rate of this classifier is $484/1089 \times 100\% = 44.44\%$, which is just slightly worse than the logistic regression! So, the logsitic regression using all variables as predictors is almost useless for this case.

```
glm.probs <- predict(logit.fit,type="response")
glm.pred <- rep("Down",nrow(Weekly))
glm.pred[glm.probs > 0.50] <- "Up"
ct <- table(glm.pred, Direction)
ct
```

```
##         Direction
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```
ConfusionTable(ct)
```

```
## $Accuracy
## [1] 0.5610652
##
## $`True Positive Rate`
## [1] 0.9206612
##
## $`False Posistive Rate`
## [1] 0.8884298
##
## $Precision
## [1] 0.5643364
##
## $`Total Error Rate`
## [1] 0.4389348
```

(d). The overall fraction of correct predictions is the accuracy of the classifier, which is 62.5%.

```
train <- Year <= 2008
Weekly.test <- Weekly[!train, ]
logit.fit <- glm(Direction ~ Lag2, family = binomial, data = Weekly, subset = train)
glm.probs <- predict(logit.fit, Weekly.test, type="response")
glm.pred <- rep("Down", nrow(Weekly.test))
glm.pred[glm.probs > 0.50] <- "Up"
ct <- table(glm.pred, Weekly.test$Direction)
ct
```

```
##
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```r
ConfusionTable(ct)$Accuracy
```

```
## [1] 0.625
```

(e). The overall fraction of correct predictions is 62.5%.

```r
lda.fit <- lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.class <- predict(lda.fit, Weekly.test)$class
ct <- table(lda.class,Weekly.test$Direction)
ct
```

```
##
## lda.class Down Up
##      Down    9  5
##      Up     34 56
```

```r
ConfusionTable(ct)$Accuracy
```

```
## [1] 0.625
```

(f).The overall fraction of correct predictions is 58.65%.

```r
qda.fit <- qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.class <- predict(qda.fit, Weekly.test)$class
ct <- table(qda.class, Weekly.test$Direction)
ct
```

```
##
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```r
ConfusionTable(ct)$Accuracy
```

```
## [1] 0.5865385
```

(g). The overall fraction of correct predictions is 50%.

```r
train.X <- matrix(Lag2[train])
test.X <- matrix(Lag2[!train])
train.Direction <- Direction[train]
test.Direction <- Direction[!train]
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)
ct <- table(knn.pred, test.Direction)
ct
```

```
##         test.Direction
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```r
ConfusionTable(ct)$Accuracy
```

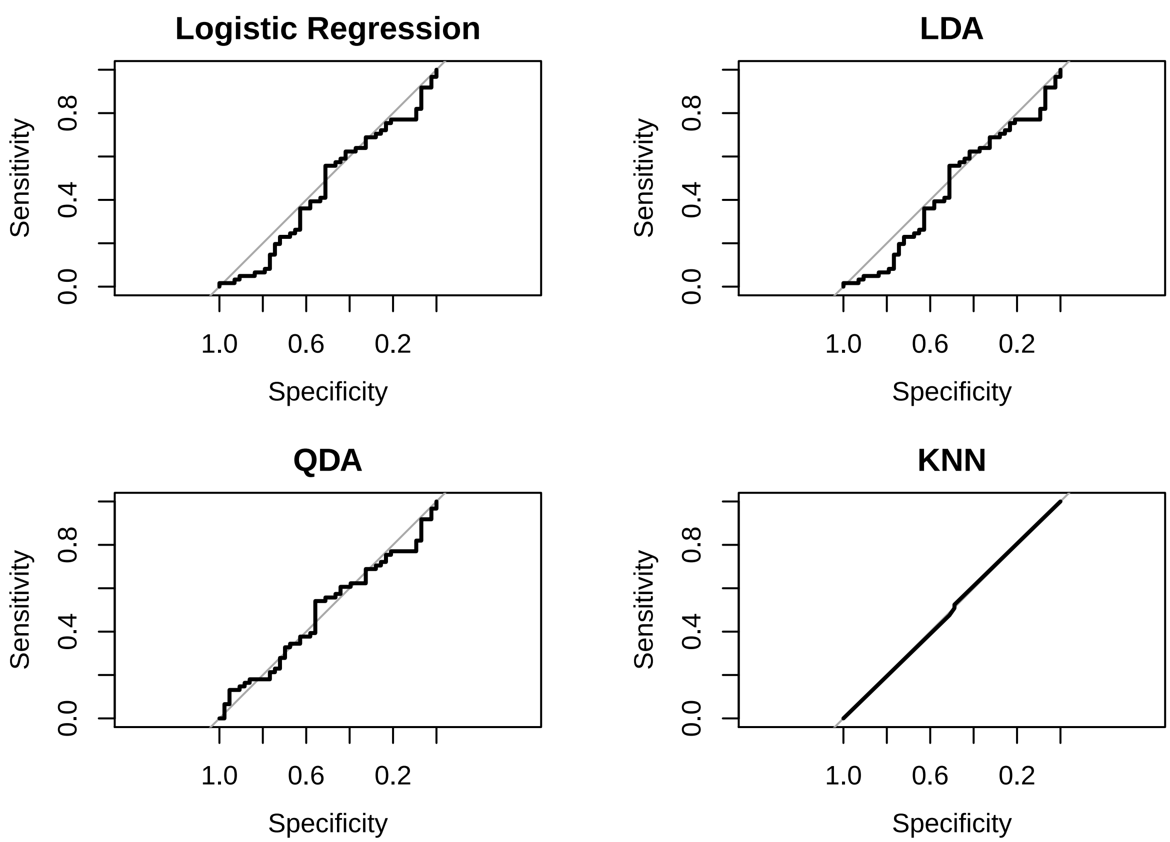
```
## [1] 0.5
```

(h) To compare the these four methods, the simplest way is to plot the ROC curve of each method. Plots are list below, ans R can also provide the AUCs of four methods. We can see that QDA (the third one) has the largest AUC, which is 0.4914. Thus, we can say that QDA may be the best classifier among these four methods for this dataset.

```
##
## Call:
## roc.formula(formula = Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (Direction Down) > 61 cases (Direction Up).
## Area under the curve: 0.4537

##
## Call:
## roc.formula(formula = Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (Direction Down) > 61 cases (Direction Up).
## Area under the curve: 0.4537

##
## Call:
## roc.formula(formula = Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (Direction Down) > 61 cases (Direction Up).
## Area under the curve: 0.4914
```




```
##
## Call:
## roc.formula(formula = test.Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (test.Direction Down) < 61 cases (test.Direction Up).
## Area under the curve: 0.4998
```

▶ **Exercises 4.13. Solution.**

▶ Appendices

Code of function ConfusionTable()

```
ConfusionTable <- function(ct){
    accuracy <- (ct[1, 1] + ct[2, 2]) / (sum(ct))
    TPr <- ct[2, 2] / (ct[1, 2] + ct[2, 2])
    FPr <- ct[2, 1] / (ct[1, 1] + ct[2, 1])
    precision <- ct[2, 2] / (ct[2, 1] + ct[2, 2])
    error <- 1 - accuracy
    result <- list(accuracy, TPr, FPr, precision, error)
    names(result) <- c("Accuracy", "True Positive Rate",
                       "False Posistive Rate", "Precision",
                       "Total Error Rate")
    return(result)
}
```