MSSC 6250 Machine Learning Homework 3

Bayesian Regression, Logistic Regression, Generative Models

Henri Medeiros Dos Reis

- Deadline: **Friday, April 7 11:59 PM**
- Homework presentation date: **Tuesday, April 11**
- Please submit your work in **one PDF** file to **D2L > Assessments > Dropbox**. *Multiple files or a file that is not in pdf format are not allowed.*
- Any relevant code should be attached.
- Read **ISL** Chapter 4.

Exercises required for all students

1. **ISL** Sec. 4.8: 2

Solution:

4.17 since all σ are equal gives

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{i=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)}$$

and 4.18

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

Given $p_k(x)$ is normal. We need to classify an observation to class k, maximizing $p_k(x)$. We can take the log, since it is equivalent to the function when maximizing

$$log(p_k(x)) = log\left(\frac{\pi_k \exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{i=1}^{K} \pi_l \exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)}\right) =$$

$$log(\pi_k \exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)) - log(\sum_{i=1}^{K} \pi_l \exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)) =$$

$$log(\pi_k) + (-\frac{1}{2\sigma^2}(x-\mu_k)^2)) - log(\sum_{i=1}^{K} \pi_l \exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2))$$

Since the third term is just a constant, it would disappear when we take the derivative with respect to k, since it is a sum over K. Therefore, maximizing the first two terms is equivalent to maximizing the whole equation. Then

$$\Rightarrow log(\pi_k) + (-\frac{1}{2\sigma^2}(x-\mu_k)^2)) = log(\pi_k) - \frac{x^2 - 2\mu_k x + \mu_k^2}{2\sigma^2}$$

$$= log(\pi_k) - \frac{\mu_k^2}{2\sigma^2} + \frac{\mu_k x}{2\sigma^2} - \frac{x^2}{2\sigma^2}$$

And $\frac{x^2}{2\sigma^2}$ is a constant

$$\Rightarrow log(\pi_k) - \frac{\mu_k^2}{2\sigma^2} + \frac{\mu_k x}{2\sigma^2} = x\frac{\mu_k}{2\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

Which is exactly the same as 4.18. Therefore, maximizing $p_k(x)$ is the same as maximizing $\delta_k(x)$

2. **ISL** Sec. 4.8: 3

Solution:

Now we follow the same steps as last problem, but with different σs

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} \exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)}{\sum_{i=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} \exp(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2)}$$

$$= \frac{\frac{\pi_k}{\sigma_k} exp(-\frac{1}{2\sigma_k^2})(x-\mu_k)^2}{\sum_{l=1}^{K} \frac{\pi_l}{\sigma_l} exp(-\frac{1}{2\sigma_k^2})(x-\mu_k)^2}$$

$$\Rightarrow log(p_k(x)) = log\left(\frac{\frac{\pi_k}{\sigma_k} exp(-\frac{1}{2\sigma_k^2})(x-\mu_k)^2}{\sum_{l=1}^{K} \frac{\pi_l}{\sigma_l} exp(-\frac{1}{2\sigma_k^2})(x-\mu_k)^2}\right)$$

$$= log(\frac{\pi_k}{\sigma_k} exp(-\frac{1}{2\sigma_k^2})(x-\mu_k)^2) - log(\sum_{l=1}^{K} \frac{\pi_l}{\sigma_l} exp(-\frac{1}{2\sigma_k^2})(x-\mu_k)^2)$$

and the term involving σ_l is a constant

$$\Rightarrow log(\pi_k) - \frac{1}{2\sigma_k^2}(x-\mu_k^2) - log(\sigma_k)$$

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

As we can see, we have terms involving x that are squared. Therefore, not linear, and in fact, it is quadratic

3. **ISL** Sec. 4.8: 5

Solution:

a-) QDA would perform better on the training data set, because of its flexibility. But the opposite should happen in the test data set, LDA would be better, because QDA would overfit.

b-) QDA would perform better in both training and testing data sets in most cases. Since the flexibility would capture the non-linearity better in most cases. However it highly depends on the nature of the non-linearity.

c-) The test predict accuracy of QDA wuld improve compared to LDA because of the flexibility advantage of QDA.

d-) False, with a small enough sample size, the QDA model will easily overfit, leading to a larger test error compared to LDA.

4. **ISL** Sec. 4.8: 7

Solution:

$$f_{yes}(x) = \frac{e^{\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

$$f_{yes}(4) = \frac{e^{\frac{-36}{72}}}{\sqrt{72\pi}}$$

$$f_{no}(x) = \frac{e^{\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

$$f_{no}(4) = \frac{e^{\frac{-2}{9}}}{\sqrt{72\pi}}$$

$$P_r(Y = yes|X = 4) = \frac{\pi_{yes} f_{yes}(4)}{\pi_{no} f_{no}(4) + \pi_{yes} f_{yes}(4)}$$

$$= \frac{0.8\frac{e^{\frac{-36}{72}}}{\sqrt{72\pi}}}{0.2\frac{e^{\frac{-2}{9}}}{\sqrt{72\pi}} + 0.8\frac{e^{\frac{-36}{72}}}{\sqrt{72\pi}}}$$

5. **ISL** Sec. 4.8: 10

Solution:

$$
\begin{aligned}
log\left(\frac{P_r(Y=k|X=x)}{P_r(Y=K|X=x)}\right) = log\left(\frac{\pi_k f_k(x)}{\pi_K f_K(x)}\right) \\
= log(\frac{\pi_k}{\pi_K}) + log\left(\frac{\frac{1}{\sqrt{2\pi\sigma}}exp(\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\frac{1}{\sqrt{2\pi\sigma}}exp(\frac{1}{2\sigma^2}(x-\mu_K)^2)}\right) \\
= log(\frac{\pi_k}{\pi_K}) - \frac{(x-\mu_k)^2}{2\sigma^2} + \frac{(x-\mu_K)^2}{2\sigma^2} \\
= log(\frac{\pi_k}{\pi_K}) - \frac{x^2}{2\sigma^2} + \frac{\mu_k x}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \frac{x^2}{2\sigma^2} - \frac{\mu_K x}{\sigma^2} + \frac{\mu_K^2}{2\sigma^2} \\
= log(\frac{\pi_k}{\pi_K}) - \frac{\mu_k^2}{2\sigma^2} + \frac{\mu_K^2}{2\sigma^2} + \frac{\mu_k x}{\sigma^2} - \frac{\mu_K x}{\sigma^2} \\
= log(\frac{\pi_k}{\pi_K}) + \frac{\mu_K^2 - \mu_k^2}{2\sigma^2} + x(\frac{\mu_k - \mu_K}{\sigma^2}) \\
= a_k + xb_{kj}
\end{aligned}
$$

Where $a_k = log(\frac{\pi_k}{\pi_K}) + \frac{\mu_K^2 - \mu_k^2}{2\sigma^2}$ and $b_{kj} = \frac{\mu_k - \mu_j}{\sigma^2}$

6. **ISL** Sec. 4.8: 13

Solution:

a-)

```
library(ISLR2)
```

```
Warning: package 'ISLR2' was built under R version 4.2.2
```

```
head(Weekly)
```

```
  Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```
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
```

```r
summary(Weekly)
```

```
      Year           Lag1               Lag2               Lag3
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
      Lag4               Lag5              Volume             Today
 Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
 1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
 Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
 Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
 3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
 Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
 Direction
 Down:484
 Up  :605
```

Year and volume appear to be correlated. From the summary statistics, we can observe that all the Lag variables are somewhat similar to each other and 'Today'.

b-)

```r
logit_fit = glm(Direction ~ Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
                      data=Weekly, family=binomial)
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
```

Looking at the p values, only lag 2 appears to be significant.

c-)

```r
pred_prob <- predict(logit_fit, type = "response")
(res<-table(pred_prob > 0.5, Weekly$Direction))
```

```
        Down  Up
  FALSE   54  48
  TRUE   430 557
```

```r
correct <- res[1,1]+res[2,2]
total <- sum(res)
correct/total
```

```
[1] 0.5610652
```

Only 56% of the predictions was correct. Considering that a random guess would give around 50%, this model is not much better.

d-)

```r
train <- Weekly[Weekly$Year <= 2008, ]
test <- Weekly[Weekly$Year > 2008, ]

logit_fit_lag2 <- glm(Direction ~ Lag2, data = train, family = "binomial")

pred_prob <- factor(ifelse(
  predict(logit_fit_lag2, newdata = test, type = "response") < 0.5, "Down", "Up")

caret::confusionMatrix(pred_prob, test$Direction, positive = "Up")
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    9  5
      Up     34 56

               Accuracy : 0.625
                 95% CI : (0.5247, 0.718)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.2439

                  Kappa : 0.1414

 Mcnemar's Test P-Value : 7.34e-06

            Sensitivity : 0.9180
            Specificity : 0.2093
         Pos Pred Value : 0.6222
         Neg Pred Value : 0.6429
             Prevalence : 0.5865
         Detection Rate : 0.5385
   Detection Prevalence : 0.8654
```

```
      Balanced Accuracy : 0.5637

       'Positive' Class : Up
```

This model is an improvement from the previous one, but not by much. The accuracy is at 62.5%.

e-)

```r
lda_fit_lag2 <- MASS::lda(Direction ~ Lag2, data = train, family = "binomial")

pred_prob_lda <- predict(lda_fit_lag2, newdata = test, type = "response")

caret::confusionMatrix(data=pred_prob_lda$class, test$Direction, positive = "Up")
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    9  5
      Up     34 56

               Accuracy : 0.625
                 95% CI : (0.5247, 0.718)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.2439

                  Kappa : 0.1414

 Mcnemar's Test P-Value : 7.34e-06

            Sensitivity : 0.9180
            Specificity : 0.2093
         Pos Pred Value : 0.6222
         Neg Pred Value : 0.6429
             Prevalence : 0.5865
         Detection Rate : 0.5385
   Detection Prevalence : 0.8654
      Balanced Accuracy : 0.5637

       'Positive' Class : Up
```

This model is an improvement from the first model, but give pretty much the same results as the previous one. The accuracy is at 62.5%.

f-)

```
qda_fit_lag2 <- MASS::qda(Direction ~ Lag2, data = train, family = "binomial")

pred_prob_qda <- predict(qda_fit_lag2, newdata = test, type = "response")

caret::confusionMatrix(data=pred_prob_qda$class, test$Direction, positive = "Up")
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    0  0
      Up     43 61

               Accuracy : 0.5865
                 95% CI : (0.4858, 0.6823)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.5419

                  Kappa : 0

 Mcnemar's Test P-Value : 1.504e-10

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.5865
         Neg Pred Value :    NaN
             Prevalence : 0.5865
         Detection Rate : 0.5865
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Up
```

We get an Accuracy of 58.6%.

g-)

```
set.seed(1)
predicted_knn <- class::knn(train = data.frame(Lag2 = train$Lag2),
                  test = data.frame(Lag2 = test$Lag2),
                  cl = train$Direction,
                  k = 1,
```

```r
                    prob = T)

caret::confusionMatrix(data = predicted_knn,
                  reference = test$Direction,
                  positive = "Up")
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down   21 30
      Up     22 31

               Accuracy : 0.5
                 95% CI : (0.4003, 0.5997)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.9700

                  Kappa : -0.0033

 Mcnemar's Test P-Value : 0.3317

            Sensitivity : 0.5082
            Specificity : 0.4884
         Pos Pred Value : 0.5849
         Neg Pred Value : 0.4118
             Prevalence : 0.5865
         Detection Rate : 0.2981
   Detection Prevalence : 0.5096
      Balanced Accuracy : 0.4983

       'Positive' Class : Up
```

This model gave an accuracy of 50% which is basically the same as guessing.

h-)

```r
nb_fit <- e1071::naiveBayes(Direction~Lag2 ,data=train)

nb_pred <- predict(nb_fit, test)

caret::confusionMatrix(data = nb_pred,
                  reference = test$Direction,
```

```
                     positive = "Up")
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    0  0
      Up     43 61

               Accuracy : 0.5865
                 95% CI : (0.4858, 0.6823)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.5419

                  Kappa : 0

 Mcnemar's Test P-Value : 1.504e-10

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.5865
         Neg Pred Value :    NaN
             Prevalence : 0.5865
         Detection Rate : 0.5865
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Up
```

We get an Accuracy of 58.6% which is the same as we got with the QDA.

i-)

LDA & Logistic Regression get the same test accuracy of 0.625, so these two are tied as the best model.

j-)

```
predicted_knn6 <- class::knn(train = data.frame(Lag2 = train$Lag2),
                    test = data.frame(Lag2 = test$Lag2),
                    cl = train$Direction,
                    k = 6,
                    prob = T)
```

```r
cm <- caret::confusionMatrix(data = predicted_knn6,
                reference = test$Direction,
                positive = "Up")
cm$overall[1]
```

```
 Accuracy
0.5769231
```

Trying KNN with 6 nearest neighbors gives an accuracy of 56.7%

```r
predicted_knn9 <- class::knn(train = data.frame(Lag2 = train$Lag2),
                  test = data.frame(Lag2 = test$Lag2),
                  cl = train$Direction,
                  k = 9,
                  prob = T)

cm <- caret::confusionMatrix(data = predicted_knn9,
                reference = test$Direction,
                positive = "Up")
cm$overall[1]
```

```
 Accuracy
0.5576923
```

Trying KNN with 9 nearest neighbors gives an accuracy of 55.7%

```r
predicted_knn15 <- class::knn(train = data.frame(Lag2 = train$Lag2),
                  test = data.frame(Lag2 = test$Lag2),
                  cl = train$Direction,
                  k = 15,
                  prob = T)

cm <- caret::confusionMatrix(data = predicted_knn15,
                reference = test$Direction,
                positive = "Up")
cm$overall[1]
```

```
 Accuracy
0.5865385
```

Trying KNN with 15 nearest neighbors gives an accuracy of 58.7%

The results seem to get better as we increase the number of nearest neighbors. Which is likely going to stop at some point and start to get worse as it increases.

7. **ISL** Sec. 4.8: 14

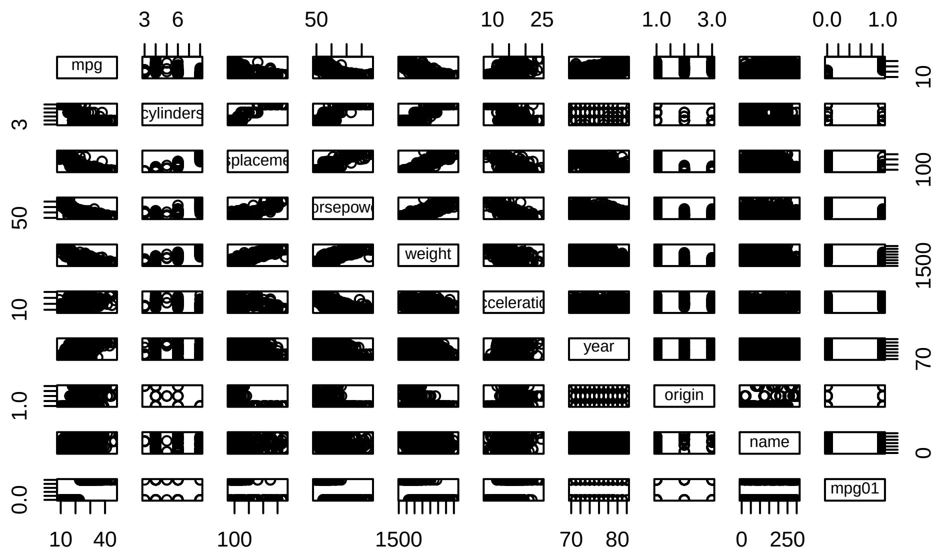
Solution:

a-)

```
mpg01 = rep(0, length(Auto$mpg))
mpg01[Auto$mpg>median(Auto$mpg)] = 1
Auto = data.frame(Auto, mpg01)
```

b-)

```
pairs(Auto)
```



We can see that there is a positive correlation between mpg and mpg01, and a negative correlation between cylinders, displacement, weight, horsepower and mpg01.

c-)

```
index <- caret::createDataPartition(y = Auto$mpg01, p = 0.8, list = F)

train1 <- Auto[index, ]
test1 <- Auto[-index, ]
```

d-)

```r
lda_fit_mpg <- MASS::lda(mpg01~cylinders+weight+displacement
                         +horsepower,
                          data = train1, family = "binomial")

pred_prob_lda_mpg <- predict(lda_fit_mpg, newdata = test1,
                             type = "response")

res1 <- table(pred_prob_lda_mpg$class, test1$mpg01)
correct <- res1[1,1]+res1[2,2]
total <- sum(res1)
1-correct/total
```

```
[1] 0.1025641
```

The test error obtained using a LDA model is 8.97%.

e-)

```r
qda_fit_mpg <- MASS::qda(mpg01~cylinders+weight+displacement
                         +horsepower,
                          data = train1, family = "binomial")

pred_prob_qda_mpg <- predict(qda_fit_mpg, newdata = test1,
                             type = "response")

res_qda <- table(pred_prob_qda_mpg$class, test1$mpg01)
correct_qda <- res_qda[1,1]+res_qda[2,2]
total_qda <- sum(res_qda)
1-correct_qda/total_qda
```

```
[1] 0.1282051
```

The test error obtained using a QDA model is 7.69%.

f-)

```r
logit_fit_mpg <- glm(mpg01~cylinders+weight+displacement
                     +horsepower,
                          data = train1, family = "binomial")

pred_prob_logit_mpg <- predict(logit_fit_mpg, newdata = test1,
                               type = "response")
pred_prob_logit_mpg[pred_prob_logit_mpg>0.5]<-1
pred_prob_logit_mpg[pred_prob_logit_mpg<0.5]<-0
```

```r
res_logit <- table(pred_prob_logit_mpg, test1$mpg01)
correct_logit <- res_logit[1,1]+res_logit[2,2]
total_logit <- sum(res_logit)
1-correct_logit/total_logit
```

```
[1] 0.1153846
```

The test error obtained is using the logistic model is 8.97%.

g-)

```r
nb_fit_mpg <- e1071::naiveBayes(mpg01~cylinders+weight
                                +displacement+horsepower,
                          data = train1, family = "binomial")

pred_prob_nb_mpg <- predict(nb_fit_mpg,test1)

res_nb <- table(pred_prob_nb_mpg, test1$mpg01)
correct_nb <- res_nb[1,1]+res_nb[2,2]
total_nb <- sum(res_nb)
1-correct_nb/total_nb
```

```
[1] 0.1025641
```

The test error obtained is using the naive Bayes model is 6.41%.

h-)

```r
tr1_matrix = data.matrix(train1[,c("cylinders","displacement",
                                   "weight","horsepower")])
te1_matrix = data.matrix(test1[,c("cylinders","displacement",
                                  "weight","horsepower")])
tr1_y = data.matrix(train1$mpg01)
te1_y = data.matrix(test1$mpg01)

predicted_knn_mpg1 <- class::knn(tr1_matrix,
                          test = te1_matrix,
                          cl = tr1_y,
                          k = 1,
                          prob = T)

res_knn1 <- table(predicted_knn_mpg1, te1_y)
correct_knn1 <- res_knn1[1,1]+res_knn1[2,2]
```

```r
total_knn1 <- sum(res_knn1)
1-correct_knn1/total_knn1
```

```
[1] 0.1410256
```

The test error obtained is using KNN with K=1 is 11.54%.

```r
predicted_knn_mpg5 <- class::knn(tr1_matrix,
                    test = te1_matrix,
                    cl = tr1_y,
                    k = 5,
                    prob = T)

res_knn5 <- table(predicted_knn_mpg5, te1_y)
correct_knn5 <- res_knn5[1,1]+res_knn5[2,2]
total_knn5 <- sum(res_knn5)
1-correct_knn5/total_knn5
```

```
[1] 0.1153846
```

The test error obtained is using KNN with K=5 is 10.27%.

```r
predicted_knn_mpg15 <- class::knn(tr1_matrix,
                    test = te1_matrix,
                    cl = tr1_y,
                    k = 15,
                    prob = T)

res_knn15 <- table(predicted_knn_mpg15, te1_y)
correct_knn15 <- res_knn15[1,1]+res_knn15[2,2]
total_knn15 <- sum(res_knn15)
1-correct_knn15/total_knn15
```

```
[1] 0.1410256
```

The test error obtained is using KNN with K=15 is 12.82%.

```r
predicted_knn_mpg10 <- class::knn(tr1_matrix,
                    test = te1_matrix,
                    cl = tr1_y,
                    k = 10,
                    prob = T)
```

```
res_knn10 <- table(predicted_knn_mpg10, te1_y)
correct_knn10<- res_knn10[1,1]+res_knn10[2,2]
total_knn10 <- sum(res_knn10)
1-correct_knn10/total_knn10
```

```
[1] 0.1153846
```

The test error obtained is using KNN with K=10 is 12.82%.

The best number of nearest neighbors seems to be 5.

Exercises required for MSSC PhD students

1. **ISL** Sec. 4.8: 4

Solution:

a-) Since we are assuming an uniform distribution, and assuming $x \in [0.05, 0.95]$, then intervals: $[x - 0.05, x + 0.05]$, so length$= 0.1$.Which means that On average 10% of the observations would be available.

b-) Using $x \in [0.05, 0.95]$, $x1_{length} \times x2_{length} = 0.01$. Therefore, only 1% of the available observations would be used to make a prediction.

c-) Using $p = 100$, gives $0.1^{100} \times 100\%$ of the observations will be available.

d-)

Using the answers from parts a)-c), We notice very quickly that, as dimensionality grows, the likelihood of there being training observations 'like' or 'near' the test observation X across all p dimensions approaches zero.

Due to the lack of training observations that are "close" across all p dimensions, the K nearest neighbors selected in vast datasets will really not be particularly close. *e-)* $p = 1; d(length) = 0.1^{1/1} = 0.1$ $p = 2; d(length) = 0.1^{1/2} = 0.32$ $p = 100; d(length) = 0.1^{1/100} = 0.977$

The side length converges to 1 as p increasing, indicating that the hypercube with only 10% of the test observation at its center must be almost the same size as the hypercube with all of the observations. Additionally, it demonstrates that when p rises, observations are concentrated close to the hypercube's edge and move "further" away from a test observation.

2. **ISL** Sec. 4.8: 11

Solution:

$$log(\frac{P_r(Y=k|X=x)}{P_r(Y=K|X=x)}) = log(\frac{\pi_k\delta_k(x)}{\pi_K\delta_K(x)})$$

$$= log(\frac{\pi_k}{\pi_K}) + \pi_k(\frac{-1}{2}log(\Sigma_k) - \frac{-1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + log(\pi_k))$$

$$+\pi_K(\frac{-1}{2}log(\Sigma_K) - \frac{-1}{2}(x-\mu_K)^T\Sigma_K^{-1}(x-\mu_K) + log(\pi_K))$$

Then we can separate into constant, linear and quadratic terms

$$\Rightarrow log(\frac{\pi_k}{\pi_K}) - \frac{\pi_k}{2}log(\Sigma_k) + \pi_k log(\pi_k) + \frac{\pi_K}{2}log(\Sigma_K) - \pi_K log(\pi_K) - \frac{\pi_k}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{\pi_K}{2}\mu_K^T\Sigma_K^{-1}\mu_K$$

$$+\pi_k x^T\Sigma_k^{-1}\mu_k + \pi_K x^T\Sigma_K^{-1}\mu_K$$

$$-\frac{\pi_k}{2}x^T\Sigma_k^{-1}x - \frac{\pi_K}{2}x^T\Sigma_K^{-1}x$$

$$\Rightarrow log(\frac{\pi_k}{\pi_K}) - \frac{\pi_k}{2}log(\Sigma_k) + \pi_k log(\pi_k) + \frac{\pi_K}{2}log(\Sigma_K) - \pi_K log(\pi_K) - \frac{\pi_k}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{\pi_K}{2}\mu_K^T\Sigma_K^{-1}\mu_K$$

$$+x^T(\pi_k\Sigma_k^{-1}\mu_k + \pi_K\Sigma_K^{-1}\mu_K) + x^T(-\frac{\pi_k}{2}\Sigma_k^{-1} - \frac{\pi_K}{2}\Sigma_K^{-1})x$$

$$= a_k + x^T b + x^T c x$$

Where

$$a = log(\frac{\pi_k}{\pi_K}) - \frac{\pi_k}{2}log(\Sigma_k) + \pi_k log(\pi_k) + \frac{\pi_K}{2}log(\Sigma_K) - \pi_K log(\pi_K) - \frac{\pi_k}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{\pi_K}{2}\mu_K^T\Sigma_K^{-1}\mu_K$$

,

$$b = (\pi_k\Sigma_k^{-1}\mu_k + \pi_K\Sigma_K^{-1}\mu_K)$$

, and

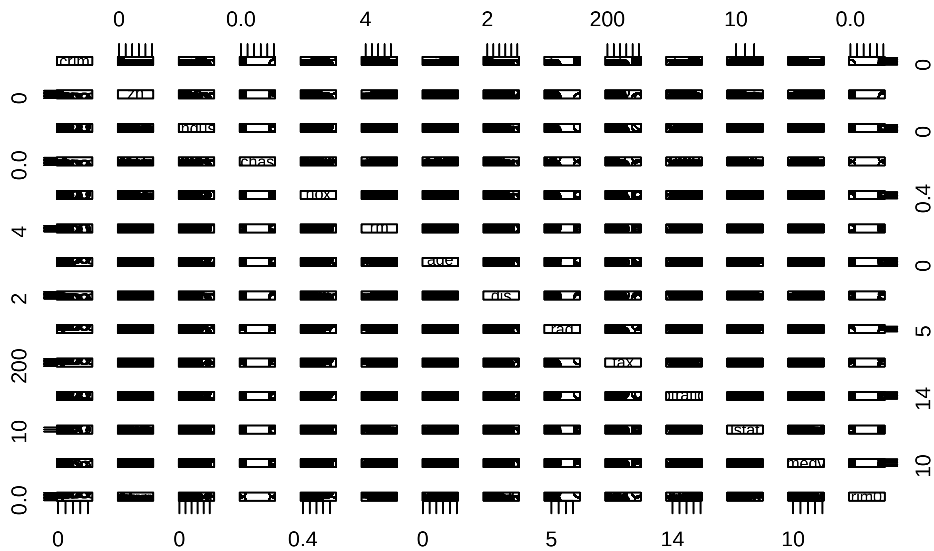
$$c = -\frac{\pi_k}{2}\Sigma_k^{-1} - \frac{\pi_K}{2}\Sigma_K^{-1}$$

3. **ISL** Sec. 4.8: 16

First, lets start by generating the binary variable.

```
b_df <- Boston
#Add 1 to column if CRIM > median and 0 otherwise
median_crim <- median(Boston$crim)
b_df$crim01 <- with(ifelse(crim>median_crim, 1, 0), data=Boston)
```

```
pairs(b_df)
```



```
cor(b_df$crim01,b_df)
```

```
          crim        zn     indus       chas       nox         rm       age
[1,] 0.4093955 -0.436151 0.6032602 0.07009677 0.7232348 -0.1563718 0.6139399
            dis       rad       tax   ptratio     lstat       medv crim01
[1,] -0.6163416 0.6197862 0.6087413 0.2535684 0.4532627 -0.2630167      1
```

It seems to be correlated with everything but the variable “chas”

Let’s separate the data into training and testing.

```
b_sample <- caret::createDataPartition(b_df$crim01, p = 0.8, list = F)
b_train <- b_df[index,]
b_test <- b_df[-index,]
```

Now, let’s see how the models fit.

```
lda_fit_boston <- MASS::lda(crim01 ~.-chas-crim,
                            data = b_train, family = "binomial")

pred_prob_lda_boston <- predict(lda_fit_boston, newdata = b_test,
                             type = "response")

res_b_lda <- table(pred_prob_lda_boston$class, b_test$crim01)
correct_b_lda <- res_b_lda[1,1]+res_b_lda[2,2]
total_b_lda <- sum(res_b_lda)
```

```
1-correct_b_lda/total_b_lda
```

```
[1] 0.1197917
```

The test error using LDA is 12%

```
qda_fit_boston <- MASS::qda(crim01 ~.-chas-crim,
                              data = b_train, family = "binomial")

pred_prob_qda_boston <- predict(qda_fit_boston, newdata = b_test,
                                 type = "response")

res_b_qda <- table(pred_prob_qda_boston$class, b_test$crim01)
correct_b_qda <- res_b_qda[1,1]+res_b_qda[2,2]
total_b_qda <- sum(res_b_qda)
1-correct_b_qda/total_b_qda
```

```
[1] 0.06770833
```

The test error using QDA is 6.8%, which is better than LDA.

```
logit_fit_boston <- glm(crim01 ~.-chas-crim,
                              data = b_train, family = "binomial")

pred_prob_logit_boston <- predict(logit_fit_boston, newdata = b_test,
                                   type = "response")
pred_prob_logit_boston[pred_prob_logit_boston>0.5]<-1
pred_prob_logit_boston[pred_prob_logit_boston<0.5]<-0

res_logit_b <- table(pred_prob_logit_boston, b_test$crim01)
correct_logit_b <- res_logit_b[1,1]+res_logit_b[2,2]
total_logit_b <- sum(res_logit_b)
1-correct_logit_b/total_logit_b
```

```
[1] 0.08854167
```

The test error using logistic regression is 8.9%, which is better than LDA, but worse than QDA.

```
nb_fit_boston <- e1071::naiveBayes(crim01 ~.-chas-crim,
                              data = b_train, family = "binomial")

pred_prob_nb_boston <- predict(nb_fit_boston,b_test)
```

```r
res_nb_boston <- table(pred_prob_nb_boston, b_test$crim01)
correct_nb_boston <- res_nb_boston[1,1]+res_nb_boston[2,2]
total_nb_boston <- sum(res_nb_boston)
1-correct_nb_boston/total_nb_boston
```

```
[1] 0.1302083
```

The test error using naive Bayes is 13%, which is worst performing model so far.

```r
b_train_m = data.matrix(subset(b_train,select=-c(crim,chas)))
b_test_m = data.matrix(subset(b_test,select=-c(crim,chas)))
train_y = data.matrix(b_train[,14])
test_y = data.matrix(b_test[,14])
```

```r
predicted_knn_boston1 <- class::knn(b_train_m,
                  test = b_test_m,
                  cl = train_y,
                  k = 1,
                  prob = T)

res_knn_b1 <- table(predicted_knn_boston1, test_y)
correct_knn_b1 <- res_knn_b1[1,1]+res_knn_b1[2,2]
total_knn_b1 <- sum(res_knn_b1)
1-correct_knn_b1/total_knn_b1
```

```
[1] 0.07291667
```

The test error using KNN with k=1 is 7.3%, which is the best so far.

```r
predicted_knn_boston3 <- class::knn(b_train_m,
                  test = b_test_m,
                  cl = train_y,
                  k = 3,
                  prob = T)

res_knn_b3 <- table(predicted_knn_boston3, test_y)
correct_knn_b3 <- res_knn_b3[1,1]+res_knn_b3[2,2]
total_knn_b3 <- sum(res_knn_b3)
1-correct_knn_b3/total_knn_b3
```

```
[1] 0.06770833
```

The test error using KNN with k=3 is 6.8%, which is better than k=1.

```r
predicted_knn_boston5 <- class::knn(b_train_m,
                  test = b_test_m,
                  cl = train_y,
                  k = 5,
                  prob = T)

res_knn_b5 <- table(predicted_knn_boston5, test_y)
correct_knn_b5 <- res_knn_b5[1,1]+res_knn_b5[2,2]
total_knn_b5<- sum(res_knn_b5)
1-correct_knn_b5/total_knn_b5
```

```
[1] 0.0625
```

The test error using KNN with k=5 is 6.2%, which is better than the previous one.

```r
predicted_knn_boston10 <- class::knn(b_train_m,
                  test = b_test_m,
                  cl = train_y,
                  k = 10,
                  prob = T)

res_knn_b10 <- table(predicted_knn_boston10, test_y)
correct_knn_b10 <- res_knn_b10[1,1]+res_knn_b10[2,2]
total_knn_b10<- sum(res_knn_b10)
1-correct_knn_b10/total_knn_b10
```

```
[1] 0.06770833
```

The test error using KNN with k=10 is 6.8%, which is a little worse than k=5.

In conclusion, it is possible to see that KNN, with k=5, is the best models among the tested ones.

4. In 250 words, summarize of what you learned in the deep learning workshop.

During the deep learning workshop, I was able to gain insight into a few key topics within this vast and rapidly growing field. These included computer vision, natural language processing, and hands-on experience with deep learning frameworks.

We started by learning a little bit about computer vision, which is an application of deep learning that has become increasingly relevant in recent years. This topic covered how deep learning algorithms can be trained to detect and classify objects within images or videos. We also discussed some of the challenges associated with computer vision, including the interpretation of complex visual data, and how deep learning can help overcome these obstacles.

We then moved on to natural language processing, another essential application of deep learning. This topic covered how deep learning algorithms can be trained to process and analyze natural language data, such as text or speech. We explored how this technology has been used to develop chatbots, language translation software, and other language-based applications.

Finally, the workshop provided hands-on experience with deep learning frameworks such as TensorFlow, and Keras. Through this practical application of the concepts we learned, we were able to develop a deeper understanding of how deep learning works in practice.

In conclusion, the workshop provided a comprehensive overview of deep learning, including its applications in computer vision and natural language processing, and practical experience with deep learning frameworks. Getting to see the challenges and solutions when working with deep learning was valuable.

No more optional exercises because no one tried any in the previous two homework sets. Reach out to me if you love to study more machine learning problems.