hw4_shuangyu_zhao

shuangyu_zhao

2023-02-20

1.

```
auto <- read.csv("/Users/apple/Desktop/STT811 appl_stat_model/data/Auto.csv")
auto$mpg01 <- ifelse(auto$mpg > median(auto$mpg), 1, 0)
auto$horsepower <- as.numeric(auto$horsepower)
```

```
## Warning: NAs introduced by coercion
```

```
which(is.na(auto))
```

```
## [1] 1224 1318 1522 1528 1546
```

```
auto <- na.omit(auto)
```

```
# train-test split
split_pro <- 0.75
n <- length(auto$mpg)*split_pro
row_samp <- sample(1:length(auto$mpg), n, replace = FALSE)
train1 <- auto[row_samp,]
test1 <- auto[-row_samp,]
```

a. perform naive bayes on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
library(e1071)
mod1 <- naiveBayes(data = train1, mpg01 ~ displacement + horsepower + weight + acceleration + year+ cyli
```

```
# train set
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
train_predict1 <- predict(mod1, train1)
train1nb_ma  <- confusionMatrix(data = as.factor(train_predict1), reference = as.factor(train1$mpg01))
train1nb_ma
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 124   7
##          1  24 139
##
##                Accuracy : 0.8946
##                  95% CI : (0.8537, 0.9272)
##     No Information Rate : 0.5034
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7893
##
##  Mcnemar's Test P-Value : 0.004057
##
##             Sensitivity : 0.8378
##             Specificity : 0.9521
##          Pos Pred Value : 0.9466
##          Neg Pred Value : 0.8528
##              Prevalence : 0.5034
##          Detection Rate : 0.4218
##    Detection Prevalence : 0.4456
##       Balanced Accuracy : 0.8949
##
##        'Positive' Class : 0
##
```

```r
# test set
test_predict1 <- predict(mod1, test1)
test1nb_ma <- confusionMatrix(data = as.factor(test_predict1), reference = as.factor(test1$mpg01))
test1nb_ma
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  3
##          1  9 38
##
##                Accuracy : 0.8776
##                  95% CI : (0.7959, 0.9351)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 1.654e-10
##
##                   Kappa : 0.7535
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.8421
##             Specificity : 0.9268
##          Pos Pred Value : 0.9412
##          Neg Pred Value : 0.8085
##              Prevalence : 0.5816
```

```
##          Detection Rate : 0.4898
##    Detection Prevalence : 0.5204
##       Balanced Accuracy : 0.8845
##
##        'Positive' Class : 0
##
```

the test error is

```r
1 - as.numeric(test1nb_ma$overall["Accuracy"])
```

```
## [1] 0.122449
```

b. Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). what is the test error of the model obtained?

```r
library(MASS)
mod2 <- lda(data = train1, mpg01 ~ displacement  + weight + acceleration + horsepower + year+ cylinders
```

```r
test_predict2 <- predict(mod2, test1)
test1lda_ma <- confusionMatrix(data = as.factor(test_predict2$class), reference = as.factor(test1$mpg01)
test1lda_ma
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  2
##          1  9 39
##
##                Accuracy : 0.8878
##                  95% CI : (0.808, 0.9426)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 3.105e-11
##
##                   Kappa : 0.7748
##
##  Mcnemar's Test P-Value : 0.07044
##
##             Sensitivity : 0.8421
##             Specificity : 0.9512
##          Pos Pred Value : 0.9600
##          Neg Pred Value : 0.8125
##              Prevalence : 0.5816
##          Detection Rate : 0.4898
##    Detection Prevalence : 0.5102
##       Balanced Accuracy : 0.8967
##
##        'Positive' Class : 0
##
```

the test error is

```r
1 - as.numeric(test1lda_ma$overall["Accuracy"])
```

```
## [1] 0.1122449
```

c. perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in b. What test errors do you obtain? which value of K seems to perform the best on this data set?

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## x dplyr::select() masks MASS::select()
```

```r
train1_knn <- scale(select_if(train1[, 2:8], is.numeric))
train1_knn_y <- train1$mpg01
test1_knn <- scale(select_if(test1[, 2:8], is.numeric))
test1_knn_y <- test1$mpg01
```

```r
library(class)
knn_mod1_5 <- knn(train1_knn, test1_knn, cl = train1_knn_y , k = 5, prob = TRUE)
test1knn_mo_5 <- confusionMatrix(knn_mod1_5 , reference = as.factor(test1_knn_y))
test1knn_mo_5
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52  3
##          1  5 38
##
##                Accuracy : 0.9184
##                  95% CI : (0.8455, 0.9641)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 1.105e-13
##
##                   Kappa : 0.8334
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9123
##             Specificity : 0.9268
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.8837
```

```
##              Prevalence : 0.5816
##          Detection Rate : 0.5306
##    Detection Prevalence : 0.5612
##       Balanced Accuracy : 0.9196
##
##        'Positive' Class : 0
##
```

```
knn_mod1_7 <- knn(train1_knn, test1_knn, cl = train1_knn_y , k = 7, prob = TRUE)
test1knn_mo_7 <- confusionMatrix(knn_mod1_7 , reference = as.factor(test1_knn_y))
test1knn_mo_7
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52  3
##          1  5 38
##
##                Accuracy : 0.9184
##                  95% CI : (0.8455, 0.9641)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 1.105e-13
##
##                   Kappa : 0.8334
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9123
##             Specificity : 0.9268
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.8837
##              Prevalence : 0.5816
##          Detection Rate : 0.5306
##    Detection Prevalence : 0.5612
##       Balanced Accuracy : 0.9196
##
##        'Positive' Class : 0
##
```

```
knn_mod1_9 <- knn(train1_knn, test1_knn, cl = train1_knn_y , k = 9, prob = TRUE)
test1knn_mo_9 <- confusionMatrix(knn_mod1_9 , reference = as.factor(test1_knn_y))
test1knn_mo_9
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 51  2
##          1  6 39
##
##                Accuracy : 0.9184
##                  95% CI : (0.8455, 0.9641)
```

```
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 1.105e-13
##
##                   Kappa : 0.8345
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.8947
##             Specificity : 0.9512
##          Pos Pred Value : 0.9623
##          Neg Pred Value : 0.8667
##              Prevalence : 0.5816
##          Detection Rate : 0.5204
##    Detection Prevalence : 0.5408
##       Balanced Accuracy : 0.9230
##
##        'Positive' Class : 0
##
```

the best test error is

```
1 - as.numeric(test1knn_mo_7$overall["Accuracy"])
```

```
## [1] 0.08163265
```

d. redo the naive bayes calculation from first principles(ie, without any package, by calculating the class means and standard deviation)

```
mean1_weight <-  mean(filter(auto, mpg01 == 1)$weight)
mean0_weight <-  mean(filter(auto, mpg01 == 0)$weight)
sd1_weight <- sd(filter(auto, mpg01 == 1)$weight)
sd0_weight <- sd(filter(auto, mpg01 == 0)$weight)
mean1_weight
```

```
## [1] 2315.23
```

```
mean0_weight
```

```
## [1] 3581.78
```

```
sd1_weight
```

```
## [1] 382.3683
```

```
sd0_weight
```

```
## [1] 693.213
```

```r
frac1 <- sum(auto$mpg01 == 1)/nrow(auto)
```

```r
LDA_pred <- frac1 * dnorm(auto$weight, mean1_weight, sd1_weight)/(frac1 * dnorm(auto$weight, mean1_weig

summary(LDA_pred)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.005113 0.578819 0.483381 0.916064 0.947947
```

```r
prediction_nb_hand <- ifelse(LDA_pred<=0.5, 0, 1)
confusionMatrix(data = as.factor(prediction_nb_hand), reference = as.factor(auto$mpg01))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 172  17
##          1  33 170
##
##                Accuracy : 0.8724
##                  95% CI : (0.8353, 0.9038)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7453
##
##  Mcnemar's Test P-Value : 0.03389
##
##             Sensitivity : 0.8390
##             Specificity : 0.9091
##          Pos Pred Value : 0.9101
##          Neg Pred Value : 0.8374
##              Prevalence : 0.5230
##          Detection Rate : 0.4388
##    Detection Prevalence : 0.4821
##       Balanced Accuracy : 0.8741
##
##        'Positive' Class : 0
##
```

e. do a modified naive bayes model(2 numerical X's) which takes into account the class covariances betweent the X's

```r
# choose year and weight
sigma <- cov(auto[, c(5, 7)])
sigma
```

```
##              weight       year
## weight 721484.7090 -967.22846
## year     -967.2285   13.56991
```

```r
mean1_year <-  mean(filter(auto, mpg01 == 1)$year)
mean0_year <-  mean(filter(auto, mpg01 == 0)$year)
sd1_year <- sd(filter(auto, mpg01 == 1)$year)
sd0_year <- sd(filter(auto, mpg01 == 0)$year)
mean1_year
```

```
## [1] 77.72727
```

```r
mean0_year
```

```
## [1] 74.38537
```

```r
sd1_year
```

```
## [1] 3.625693
```

```r
sd0_year
```

```
## [1] 2.944383
```

```r
library(mvtnorm)
LDA_pred_cov <- frac1 * dmvnorm(auto[, c(5, 7)], c(mean1_weight, mean1_year), sigma)/(frac1 * dmvnorm(au

summary(LDA_pred_cov)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01445 0.21645 0.55471 0.49182 0.75022 0.92064
```

```r
prediction_nb_cov <- ifelse(LDA_pred_cov<=0.5, 0, 1)
confusionMatrix(data = as.factor(prediction_nb_cov), reference = as.factor(auto$mpg01))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 175   9
##          1  30 178
##
##                Accuracy : 0.9005
##                  95% CI : (0.8665, 0.9283)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8016
##
##  Mcnemar's Test P-Value : 0.001362
##
##             Sensitivity : 0.8537
##             Specificity : 0.9519
```

```
##          Pos Pred Value : 0.9511
##          Neg Pred Value : 0.8558
##              Prevalence : 0.5230
##          Detection Rate : 0.4464
##    Detection Prevalence : 0.4694
##       Balanced Accuracy : 0.9028
##
##        'Positive' Class : 0
##
```

f. create confusion matrices and compute the overall accuracy for the 5 models(test dataset). Compare how the model did

```r
# model1 naive bayes in packages
test1nb_ma
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  3
##          1  9 38
##
##                Accuracy : 0.8776
##                  95% CI : (0.7959, 0.9351)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 1.654e-10
##
##                   Kappa : 0.7535
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.8421
##             Specificity : 0.9268
##          Pos Pred Value : 0.9412
##          Neg Pred Value : 0.8085
##              Prevalence : 0.5816
##          Detection Rate : 0.4898
##    Detection Prevalence : 0.5204
##       Balanced Accuracy : 0.8845
##
##        'Positive' Class : 0
##
```

```r
# model2 LDA
test1lda_ma
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  2
##          1  9 39
```

```
##
##                Accuracy : 0.8878
##                  95% CI : (0.808, 0.9426)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 3.105e-11
##
##                   Kappa : 0.7748
##
##  Mcnemar's Test P-Value : 0.07044
##
##             Sensitivity : 0.8421
##             Specificity : 0.9512
##          Pos Pred Value : 0.9600
##          Neg Pred Value : 0.8125
##              Prevalence : 0.5816
##          Detection Rate : 0.4898
##    Detection Prevalence : 0.5102
##       Balanced Accuracy : 0.8967
##
##        'Positive' Class : 0
##
```

```
# model3 KNN
test1knn_mo_7
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52  3
##          1  5 38
##
##                Accuracy : 0.9184
##                  95% CI : (0.8455, 0.9641)
##     No Information Rate : 0.5816
##     P-Value [Acc > NIR] : 1.105e-13
##
##                   Kappa : 0.8334
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9123
##             Specificity : 0.9268
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.8837
##              Prevalence : 0.5816
##          Detection Rate : 0.5306
##    Detection Prevalence : 0.5612
##       Balanced Accuracy : 0.9196
##
##        'Positive' Class : 0
##
```

```r
# model4 naive bayes written according to theory witout pachkage
confusionMatrix(data = as.factor(prediction_nb_hand), reference = as.factor(auto$mpg01))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 172  17
##          1  33 170
##
##                Accuracy : 0.8724
##                  95% CI : (0.8353, 0.9038)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7453
##
##  Mcnemar's Test P-Value : 0.03389
##
##             Sensitivity : 0.8390
##             Specificity : 0.9091
##          Pos Pred Value : 0.9101
##          Neg Pred Value : 0.8374
##              Prevalence : 0.5230
##          Detection Rate : 0.4388
##    Detection Prevalence : 0.4821
##       Balanced Accuracy : 0.8741
##
##        'Positive' Class : 0
##
```

```r
# model5 modified model4 by covariance with 2 X
confusionMatrix(data = as.factor(prediction_nb_cov), reference = as.factor(auto$mpg01))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 175   9
##          1  30 178
##
##                Accuracy : 0.9005
##                  95% CI : (0.8665, 0.9283)
##     No Information Rate : 0.523
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8016
##
##  Mcnemar's Test P-Value : 0.001362
##
##             Sensitivity : 0.8537
##             Specificity : 0.9519
##          Pos Pred Value : 0.9511
```

```
##          Neg Pred Value : 0.8558
##              Prevalence : 0.5230
##          Detection Rate : 0.4464
##    Detection Prevalence : 0.4694
##       Balanced Accuracy : 0.9028
##
##        'Positive' Class : 0
##
```

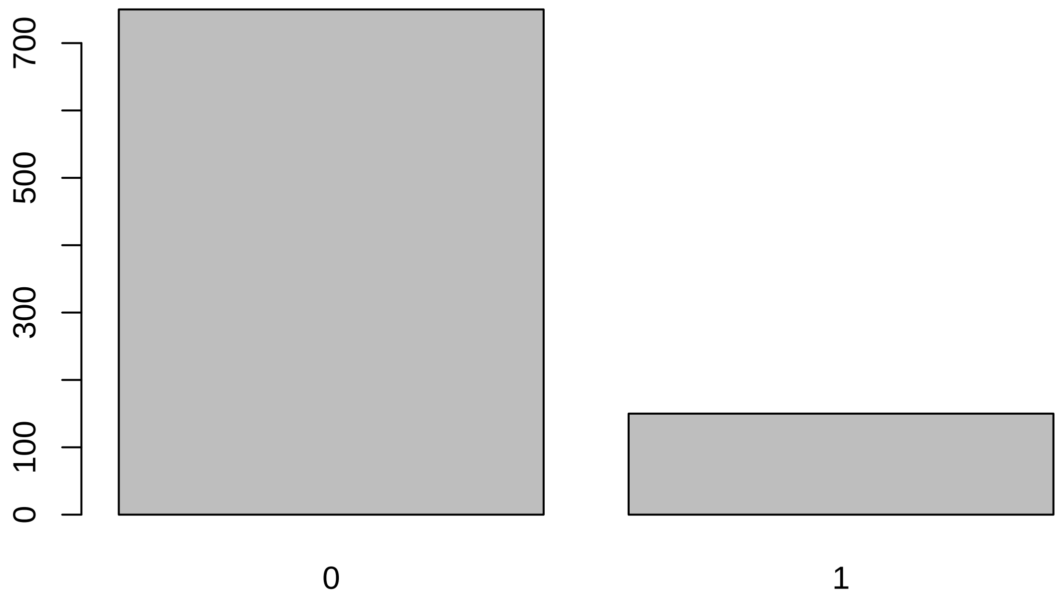
The accuracy of model 3 is the highest, and the accuracy of model 5 is the second highest, and model 4 are th lowest. So model 3 is the best.

2.

```
churn <- read.csv("/Users/apple/Desktop/STT811 appl_stat_model/data/customer_churn.csv")
churn_model <- churn[, c(2:6, 10)]
head(churn_model)
```

```
##   Age Total_Purchase Account_Manager Years Num_Sites Churn
## 1  42       11066.80               0  7.22         8     1
## 2  41       11916.22               0  6.50        11     1
## 3  38       12884.75               0  6.67        12     1
## 4  42        8010.76               0  6.71        10     1
## 5  37        9191.58               0  5.56         9     1
## 6  48       10356.02               0  5.12         8     1
```

```
barplot(table(churn_model$Churn))
```




```
# train-test split
split_pro <- 0.5
n <- length(churn_model$Churn)*split_pro
row_samp <- sample(1:length(churn_model$Churn), n, replace = FALSE)
train2 <- churn_model[row_samp,]
test2 <- churn_model[-row_samp,]
```

a. Create a naïve Bayes model to predict churn.

```
mod3 <- naiveBayes(data = train2, Churn ~ Age + Total_Purchase + Account_Manager + Years + Num_Sites)
test_predict2 <- predict(mod3, test2)
test2nb_ma <- confusionMatrix(data = as.factor(test_predict2), reference = as.factor(test2$Churn))
test2nb_ma
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 353  46
##          1  11  40
##
##                Accuracy : 0.8733
##                  95% CI : (0.839, 0.9026)
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.0001749
```

```
##
##                   Kappa : 0.5149
##
##  Mcnemar's Test P-Value : 6.687e-06
##
##             Sensitivity : 0.9698
##             Specificity : 0.4651
##          Pos Pred Value : 0.8847
##          Neg Pred Value : 0.7843
##              Prevalence : 0.8089
##          Detection Rate : 0.7844
##    Detection Prevalence : 0.8867
##       Balanced Accuracy : 0.7174
##
##        'Positive' Class : 0
##
```

the test error is

```
1 - as.numeric(test2nb_ma$overall["Accuracy"])
```

```
## [1] 0.1266667
```

b. Create a KNN neighbors model to predict churn. Vary K from 4 to 10 and find out which K has the highest accuracy.

```
train2knn_y <- train2$Churn
test2knn_y <- test2$Churn
Account_Manager <- train2$Account_Manager
train2knn <- cbind(scale(train2[, c(1, 2, 4, 5)]), Account_Manager)
Account_Manager <- test2$Account_Manager
test2knn <- cbind(scale(test2[, c(1, 2, 4, 5)]), Account_Manager)
```

```
# k=4
knn_mod2_4 <- knn(train2knn, test2knn, cl = train2knn_y , k = 4, prob = TRUE)
test2knn_mo_4 <- confusionMatrix(knn_mod2_4 , reference = as.factor(test2knn_y))
test2knn_mo_4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 350  45
##          1  14  41
##
##                Accuracy : 0.8689
##                  95% CI : (0.8342, 0.8987)
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.0004661
##
##                   Kappa : 0.5082
##
```

```
##  Mcnemar's Test P-Value : 9.397e-05
##
##             Sensitivity : 0.9615
##             Specificity : 0.4767
##          Pos Pred Value : 0.8861
##          Neg Pred Value : 0.7455
##              Prevalence : 0.8089
##          Detection Rate : 0.7778
##    Detection Prevalence : 0.8778
##       Balanced Accuracy : 0.7191
##
##        'Positive' Class : 0
##
```

```r
# k = 5
knn_mod2_5 <- knn(train2knn, test2knn, cl = train2knn_y , k = 5, prob = TRUE)
test2knn_mo_5 <- confusionMatrix(knn_mod2_5 , reference = as.factor(test2knn_y))
test2knn_mo_5
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 352  51
##          1  12  35
##
##                Accuracy : 0.86
##                  95% CI : (0.8245, 0.8907)
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.002659
##
##                   Kappa : 0.4523
##
##  Mcnemar's Test P-Value : 1.688e-06
##
##             Sensitivity : 0.9670
##             Specificity : 0.4070
##          Pos Pred Value : 0.8734
##          Neg Pred Value : 0.7447
##              Prevalence : 0.8089
##          Detection Rate : 0.7822
##    Detection Prevalence : 0.8956
##       Balanced Accuracy : 0.6870
##
##        'Positive' Class : 0
##
```

```r
# k=6
knn_mod2_6 <- knn(train2knn, test2knn, cl = train2knn_y , k = 6, prob = TRUE)
test2knn_mo_6 <- confusionMatrix(knn_mod2_6 , reference = as.factor(test2knn_y))
test2knn_mo_6
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 355  57
##          1   9  29
##
##                Accuracy : 0.8533
##                  95% CI : (0.8172, 0.8847)
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.008186
##
##                   Kappa : 0.3971
##
##  Mcnemar's Test P-Value : 7.238e-09
##
##             Sensitivity : 0.9753
##             Specificity : 0.3372
##          Pos Pred Value : 0.8617
##          Neg Pred Value : 0.7632
##              Prevalence : 0.8089
##          Detection Rate : 0.7889
##    Detection Prevalence : 0.9156
##       Balanced Accuracy : 0.6562
##
##        'Positive' Class : 0
##
```

```r
# k = 7
knn_mod2_7 <- knn(train2knn, test2knn, cl = train2knn_y , k = 7, prob = TRUE)
test2knn_mo_7 <- confusionMatrix(knn_mod2_7 , reference = as.factor(test2knn_y))
test2knn_mo_7
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 358  58
##          1   6  28
##
##                Accuracy : 0.8578
##                  95% CI : (0.822, 0.8887)
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.003933
##
##                   Kappa : 0.4019
##
##  Mcnemar's Test P-Value : 1.83e-10
##
##             Sensitivity : 0.9835
##             Specificity : 0.3256
##          Pos Pred Value : 0.8606
##          Neg Pred Value : 0.8235
##              Prevalence : 0.8089
##          Detection Rate : 0.7956
```

```
##    Detection Prevalence : 0.9244
##       Balanced Accuracy : 0.6545
##
##        'Positive' Class : 0
##
```

```r
# k=8
knn_mod2_8 <- knn(train2knn, test2knn, cl = train2knn_y , k = 8, prob = TRUE)
test2knn_mo_8 <- confusionMatrix(knn_mod2_8 , reference = as.factor(test2knn_y))
test2knn_mo_8
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 360  58
##          1   4  28
##
##                Accuracy : 0.8622
##                  95% CI : (0.8269, 0.8927)
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.001767
##
##                   Kappa : 0.4138
##
##  Mcnemar's Test P-Value : 1.685e-11
##
##             Sensitivity : 0.9890
##             Specificity : 0.3256
##          Pos Pred Value : 0.8612
##          Neg Pred Value : 0.8750
##              Prevalence : 0.8089
##          Detection Rate : 0.8000
##    Detection Prevalence : 0.9289
##       Balanced Accuracy : 0.6573
##
##        'Positive' Class : 0
##
```

```r
# k = 9
knn_mod2_9 <- knn(train2knn, test2knn, cl = train2knn_y , k = 9, prob = TRUE)
test2knn_mo_9 <- confusionMatrix(knn_mod2_9 , reference = as.factor(test2knn_y))
test2knn_mo_9
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 360  61
##          1   4  25
##
##                Accuracy : 0.8556
##                  95% CI : (0.8196, 0.8867)
```

```
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.005722
##
##                   Kappa : 0.3745
##
##  Mcnemar's Test P-Value : 3.759e-12
##
##             Sensitivity : 0.9890
##             Specificity : 0.2907
##          Pos Pred Value : 0.8551
##          Neg Pred Value : 0.8621
##              Prevalence : 0.8089
##          Detection Rate : 0.8000
##    Detection Prevalence : 0.9356
##       Balanced Accuracy : 0.6399
##
##        'Positive' Class : 0
##
```

```
# k = 10
knn_mod2_10 <- knn(train2knn, test2knn, cl = train2knn_y , k = 10, prob = TRUE)
test2knn_mo_10 <- confusionMatrix(knn_mod2_10 , reference = as.factor(test2knn_y))
test2knn_mo_10
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 362  59
##          1   2  27
##
##                Accuracy : 0.8644
##                  95% CI : (0.8293, 0.8947)
##     No Information Rate : 0.8089
##     P-Value [Acc > NIR] : 0.001154
##
##                   Kappa : 0.413
##
##  Mcnemar's Test P-Value : 7.496e-13
##
##             Sensitivity : 0.9945
##             Specificity : 0.3140
##          Pos Pred Value : 0.8599
##          Neg Pred Value : 0.9310
##              Prevalence : 0.8089
##          Detection Rate : 0.8044
##    Detection Prevalence : 0.9356
##       Balanced Accuracy : 0.6542
##
##        'Positive' Class : 0
##
```

k=10 is the best

c. Create the confusion matrices for the test dataset for each of these and compare the models' performance.

```
test2knn_mo_4$table
```

```
##           Reference
## Prediction   0   1
##          0 350  45
##          1  14  41
```

```
test2knn_mo_5$table
```

```
##           Reference
## Prediction   0   1
##          0 352  51
##          1  12  35
```

```
test2knn_mo_6$table
```

```
##           Reference
## Prediction   0   1
##          0 355  57
##          1   9  29
```

```
test2knn_mo_7$table
```

```
##           Reference
## Prediction   0   1
##          0 358  58
##          1   6  28
```

```
test2knn_mo_8$table
```

```
##           Reference
## Prediction   0   1
##          0 360  58
##          1   4  28
```

```
test2knn_mo_9$table
```

```
##           Reference
## Prediction   0   1
##          0 360  61
##          1   4  25
```

```
test2knn_mo_10$table
```

```
##           Reference
## Prediction   0   1
##          0 362  59
##          1   2  27
```

when k=10, the result of confusion matrix is the best, with highest true negative.

Most of the time, the number of true negative with k=2t-1 is higher than one with k=2t, and the number of true positive with k=2t-1 is lower than one with k=2t.

For the accuracy, one with k=2t is the same with one with k=2t+1.