3. Question 16 in Section 4.8

(Only use logistic regression and KNN models for this question)

Using the Boston data set, fit classification models in order to predict whether a given census tract has a crime rate above or below the me- dian. Explore logistic regression, LDA, naive Bayes, and KNN models using various subsets of the predictors. Describe your findings. Hint: You will have to create the response variable yourself, using the variables that are contained in the Boston data set

Grab our data, one-hot encode the crime variable such that it tells us if its ABOVE or BELOW the average crime rate:

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

```
attach(Boston)
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 4.1.2
```

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

```
set.seed(1)

# one-hot encode Crime Variable. 1=above median, 0=below median
Boston$crim <- factor(ifelse(Boston$crim > median(Boston$crim), 1, 0))
head(Boston)
```

```
##   crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1    0 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2    0  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3    0  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4    0  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5    0  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6    0  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

As you can see the 'crim' column changed from a series of continuous numbers into binary 0/1 based on their relationship to the median 'crim' value.

Split data into testing and training:

```r
sample_size <- floor(0.75 * nrow(Boston))
subset <- sample(seq_len(nrow(Boston)), size = sample_size)
Boston_train <- Boston[subset, ] # get the training set
Boston_test <- Boston[-subset, ] # get the test set

print(paste0("Rows in boston: ", nrow(Boston)))
```

```
## [1] "Rows in boston: 506"
```

```r
print(paste0("Rows in test set: ", nrow(Boston_test)))
```

```
## [1] "Rows in test set: 127"
```

```r
print(paste0("Rows in training set: ", nrow(Boston_train)))
```

```
## [1] "Rows in training set: 379"
```

```r
print(paste0("Rows in Training + Testing: ",sum(nrow(Boston_test),nrow(Boston_train))))
```

```
## [1] "Rows in Training + Testing: 506"
```

Now we have a training set where the training set is 75% of the data and the testing set is 25% of the data.

Logistic Regression

Set up a logistic regression model with all variables:

```r
# Glm = generalized linear model, "family = binomial" means that we consider logistic regression
my_glm <- glm(crim~ . , data = Boston_train, family = "binomial")
summary(my_glm)
```

```
##
## Call:
## glm(formula = crim ~ ., family = "binomial", data = Boston_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1974  -0.1381  -0.0003   0.0030   3.6668
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -47.765298   7.762319  -6.153 7.58e-10 ***
## zn           -0.116529   0.046515  -2.505  0.01224 *
## indus        -0.084808   0.054927  -1.544  0.12259
## chas          0.111021   0.798530   0.139  0.88943
## nox          59.769363  10.154042   5.886 3.95e-09 ***
## rm           -0.312253   0.855574  -0.365  0.71514
## age           0.027033   0.014222   1.901  0.05733 .
## dis           1.234418   0.302872   4.076 4.59e-05 ***
```

```
## rad           0.646595   0.179666   3.599  0.00032 ***
## tax          -0.005976   0.003077  -1.942  0.05209 .
## ptratio       0.281197   0.146327   1.922  0.05464 .
## lstat         0.115201   0.056457   2.041  0.04130 *
## medv          0.195255   0.083889   2.328  0.01994 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 525.40  on 378  degrees of freedom
## Residual deviance: 151.01  on 366  degrees of freedom
## AIC: 177.01
##
## Number of Fisher Scoring iterations: 9
```

Compute the classification accuracy from training data:

```r
# Build logistic regression model
predicted_glm_train <- predict(my_glm, Boston_train, type = "response")
yhat_predict_train <- ifelse(predicted_glm_train > 0.5, 1, 0)

# Create a table
table_Boston_train <- table(y = Boston_train$crim, yhat = yhat_predict_train)
table_Boston_train
```

```
##    yhat
## y     0   1
##   0 178  12
##   1  16 173
```

```r
# Classification accuracy
accuracy_Boston_train <- sum(diag(table_Boston_train))/ sum(table_Boston_train)
accuracy_Boston_train
```

```
## [1] 0.9261214
```

The training Accuracy is 93%.

Compute the classification accuracy from the testing data:

```r
# Compute the classification accuracy from test data
predicted_glm_test <- predict(my_glm, Boston_test, type = "response")
yhat_predict_test <- ifelse(predicted_glm_test > 0.5, 1, 0)

table_Boston_test <- table(y = Boston_test$crim, yhat = yhat_predict_test)
table_Boston_test
```

```
##    yhat
## y    0  1
##   0 51 12
##   1  5 59
```

```
# Classification accuracy from test data
accuracy_Boston_test <- sum(diag(table_Boston_test))/ sum(table_Boston_test)
accuracy_Boston_test
```

```
## [1] 0.8661417
```

The testing Accuracy is 87%.

K nearest Neighbors

Omit NAs from the data and re-split:

```
# Remove missing data from data
Boston <- na.omit(Boston)

sample_size <- floor(0.75 * nrow(Boston))
subset <- sample(seq_len(nrow(Boston)), size = sample_size)
Boston_train <- Boston[subset, ] # get the training set
Boston_test <- Boston[-subset, ] # get the test set

print(paste0("Rows in boston: ", nrow(Boston)))
```

```
## [1] "Rows in boston: 506"
```

```
print(paste0("Rows in test set: ", nrow(Boston_test)))
```

```
## [1] "Rows in test set: 127"
```

```
print(paste0("Rows in training set: ", nrow(Boston_train)))
```

```
## [1] "Rows in training set: 379"
```

```
print(paste0("Rows in Training + Testing: ",sum(nrow(Boston_test),nrow(Boston_train))))
```

```
## [1] "Rows in Training + Testing: 506"
```

Fit a KNN Classifier with K=1

```
# We first try with K = 1
my_knn <- knn(Boston_train[, -1],
              Boston_test[,-1],
              Boston_train$crim,
              k = 1)
table_knn <- table(my_knn, Boston_test$crim)

accuracy_Boston_test <- sum(diag(table_knn))/ sum(table_knn)
accuracy_Boston_test
```

```
## [1] 0.9448819
```

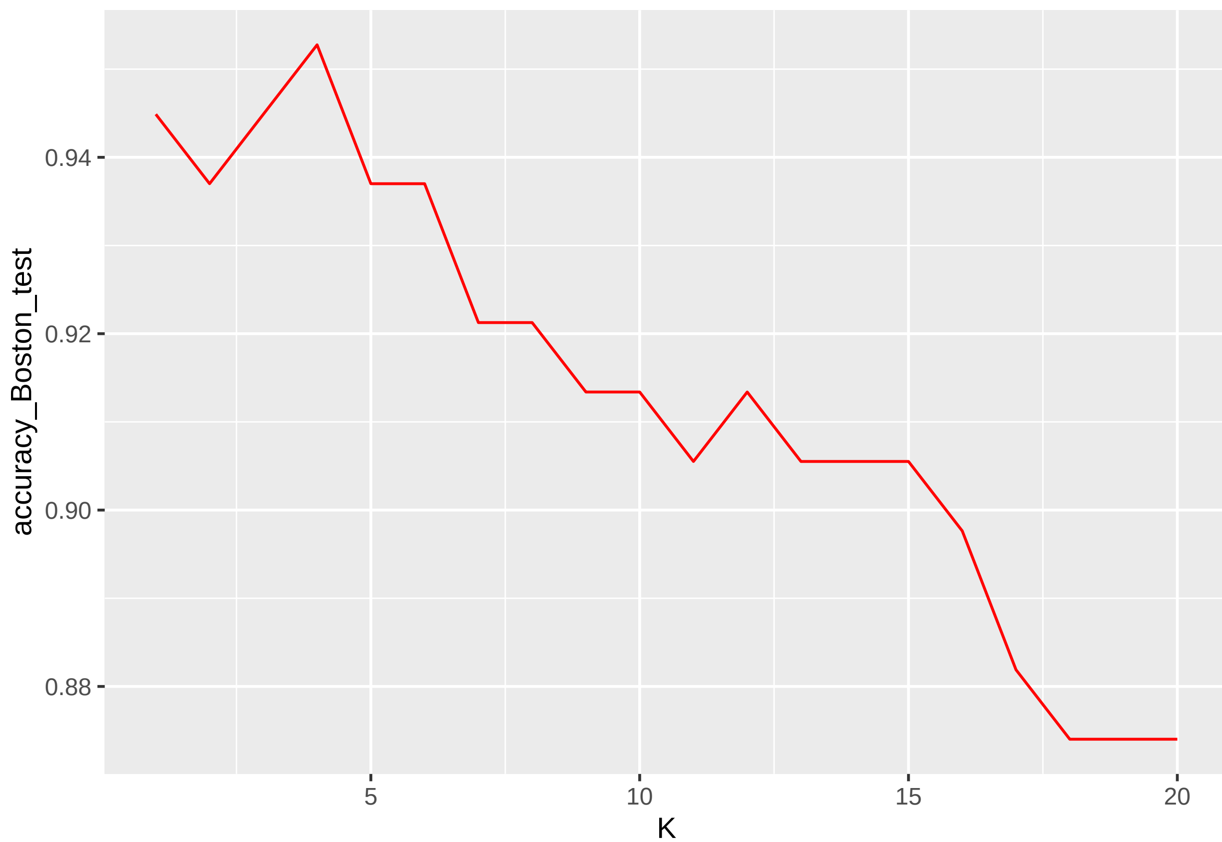
Our test accuracy is 90%.

Now lets see which K will give us the highest testing accuracy:

```
# We plot the accuracy for different values of K
M = 20 # the number of possible values of K
K <- seq(1, M)

accuracy_Boston_test <- rep(0,M)

for (i in 1:M)
{
  my_knn <- knn(Boston_train[,-1], Boston_test[,-1], Boston_train$crim, k = K[i])
  table_knn <- table(my_knn, Boston_test$crim)

  accuracy_Boston_test[i] <- sum(diag(table_knn))/ sum(table_knn)
}
max(accuracy_Boston_test)
```

```
## [1] 0.9527559
```

```
accuracy_Boston_test <- as.data.frame(accuracy_Boston_test)

ggplot(accuracy_Boston_test,
       aes(x = K,
           y = accuracy_Boston_test)) +
      geom_line(col = "red")
```



The best value for K is around 1 or 2!

```
accuracy_Boston_test[1,]
```

```
## [1] 0.9448819
```

```
accuracy_Boston_test[2,]
```

```
## [1] 0.9370079
```

The test accuracy of K=1 and K=2 is 90%.