SDM_Assignment4_2

Sri Balaji Muruganandam

14/11/2021

Use the same Boston dataset that you used in Question 1. Fit classification models in order to predict whether a given census tract has a crime rate avoce or below the median. Explore logistic regression, LDA, knn and CART. Describe your findings.

```
rm(list = ls())
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.1
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.1.1
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.1
```

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

```
## Warning: package 'ggplot2' was built under R version 4.1.1
```

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

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.1.1
```

```
#Library(MMST)
data(Boston)
dim(Boston)
```

```
## [1] 506  14
```

```
head(Boston,10)
```

```
##       crim   zn indus chas   nox    rm   age    dis rad tax ptratio  black
## 1  0.00632 18.0  2.31    0 0.538 6.575  65.2 4.0900   1 296    15.3 396.90
## 2  0.02731  0.0  7.07    0 0.469 6.421  78.9 4.9671   2 242    17.8 396.90
## 3  0.02729  0.0  7.07    0 0.469 7.185  61.1 4.9671   2 242    17.8 392.83
## 4  0.03237  0.0  2.18    0 0.458 6.998  45.8 6.0622   3 222    18.7 394.63
## 5  0.06905  0.0  2.18    0 0.458 7.147  54.2 6.0622   3 222    18.7 396.90
## 6  0.02985  0.0  2.18    0 0.458 6.430  58.7 6.0622   3 222    18.7 394.12
## 7  0.08829 12.5  7.87    0 0.524 6.012  66.6 5.5605   5 311    15.2 395.60
## 8  0.14455 12.5  7.87    0 0.524 6.172  96.1 5.9505   5 311    15.2 396.90
## 9  0.21124 12.5  7.87    0 0.524 5.631 100.0 6.0821   5 311    15.2 386.63
## 10 0.17004 12.5  7.87    0 0.524 6.004  85.9 6.5921   5 311    15.2 386.71
##    lstat medv
## 1   4.98 24.0
## 2   9.14 21.6
## 3   4.03 34.7
## 4   2.94 33.4
## 5   5.33 36.2
## 6   5.21 28.7
## 7  12.43 22.9
## 8  19.15 27.1
## 9  29.93 16.5
## 10 17.10 18.9
```

Creating the crim_class column based on the median of the crim

```
crim_medi = median(Boston$crim)
data = data.frame(Boston)

data$crim_class[data$crim>crim_medi] <- 1
data$crim_class[data$crim<=crim_medi] <- 0
tail(data,13)
```

```
##        crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 494 0.17331  0  9.69    0 0.585 5.707 54.0 2.3817   6 391    19.2 396.90 12.01
## 495 0.27957  0  9.69    0 0.585 5.926 42.6 2.3817   6 391    19.2 396.90 13.59
## 496 0.17899  0  9.69    0 0.585 5.670 28.8 2.7986   6 391    19.2 393.29 17.60
## 497 0.28960  0  9.69    0 0.585 5.390 72.9 2.7986   6 391    19.2 396.90 21.14
## 498 0.26838  0  9.69    0 0.585 5.794 70.6 2.8927   6 391    19.2 396.90 14.10
## 499 0.23912  0  9.69    0 0.585 6.019 65.3 2.4091   6 391    19.2 396.90 12.92
## 500 0.17783  0  9.69    0 0.585 5.569 73.5 2.3999   6 391    19.2 395.77 15.10
## 501 0.22438  0  9.69    0 0.585 6.027 79.7 2.4982   6 391    19.2 396.90 14.33
## 502 0.06263  0 11.93    0 0.573 6.593 69.1 2.4786   1 273    21.0 391.99  9.67
## 503 0.04527  0 11.93    0 0.573 6.120 76.7 2.2875   1 273    21.0 396.90  9.08
## 504 0.06076  0 11.93    0 0.573 6.976 91.0 2.1675   1 273    21.0 396.90  5.64
## 505 0.10959  0 11.93    0 0.573 6.794 89.3 2.3889   1 273    21.0 393.45  6.48
## 506 0.04741  0 11.93    0 0.573 6.030 80.8 2.5050   1 273    21.0 396.90  7.88
##     medv crim_class
## 494 21.8          0
## 495 24.5          1
## 496 23.1          0
## 497 19.7          1
## 498 18.3          1
## 499 21.2          0
## 500 17.5          0
## 501 16.8          0
## 502 22.4          0
## 503 20.6          0
## 504 23.9          0
## 505 22.0          0
## 506 11.9          0
```

Splitting the data into training and the test data

```
set.seed(23)
random_index = sample(c(1:nrow(data)), size = round(8/10 * nrow(data)), replace = FALSE)
train_data <- data[random_index,]
test_data <- data[-random_index,]

y_train_data = train_data$crim_class
y_test_data = test_data$crim_class

dim(train_data)
```

```
## [1] 405  15
```

```
dim(test_data)
```

```
## [1] 101  15
```

Modelling Logistic Regression

```
model = glm(crim_class~., data = train_data, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model)
```

```
##
## Call:
## glm(formula = crim_class ~ ., family = "binomial", data = train_data)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.280e-03  -2.000e-08  -2.000e-08   2.000e-08   2.477e-03
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.170e+02  2.365e+05  -0.001    0.999
## crim         1.070e+03  6.509e+04   0.016    0.987
## zn           1.991e+00  2.796e+02   0.007    0.994
## indus       -7.083e-01  1.610e+03   0.000    1.000
## chas        -3.340e+01  1.492e+04  -0.002    0.998
## nox         -4.993e+02  5.117e+05  -0.001    0.999
## rm           2.798e+01  7.261e+03   0.004    0.997
## age          3.608e-02  2.717e+02   0.000    1.000
## dis         -1.861e+01  1.022e+04  -0.002    0.999
## rad          1.959e+00  5.831e+03   0.000    1.000
## tax          4.201e-02  5.412e+01   0.001    0.999
## ptratio      1.011e+01  5.224e+03   0.002    0.998
## black        1.716e-01  1.036e+02   0.002    0.999
## lstat       -3.424e-02  1.964e+03   0.000    1.000
## medv        -3.238e+00  2.006e+03  -0.002    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5.6125e+02  on 404  degrees of freedom
## Residual deviance: 1.8344e-05  on 390  degrees of freedom
## AIC: 30
##
## Number of Fisher Scoring iterations: 25
```

Predicting for new values

```
test_predict = predict(model, newdata = test_data, type = "response")
y_predict_test = round(test_predict)
```

Calculating the error

```
test_error <- sum(abs(y_predict_test- y_test_data))/length(y_test_data)
print(test_error)
```

```
## [1] 0.01980198
```

Computing the confusion matrix

```
conf <- confusionMatrix(as.factor(y_predict_test), as.factor(y_test_data))
names(conf)
```

```
## [1] "positive" "table"    "overall"  "byClass"  "mode"     "dots"
```

```
conf$table
```

```
##           Reference
## Prediction  0  1
##          0 44  0
##          1  2 55
```

```
conf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.801980e-01   9.599365e-01   9.302949e-01   9.975928e-01   5.445545e-01
## AccuracyPValue  McnemarPValue
##   7.918404e-24   4.795001e-01
```

The model has predicted all of the test errors correctly

Calculating the accuracy using formula

```
false_result = which(y_predict_test==y_test_data)
accuracy=length(false_result) / length(y_test_data)
print(accuracy)
```

```
## [1] 0.980198
```

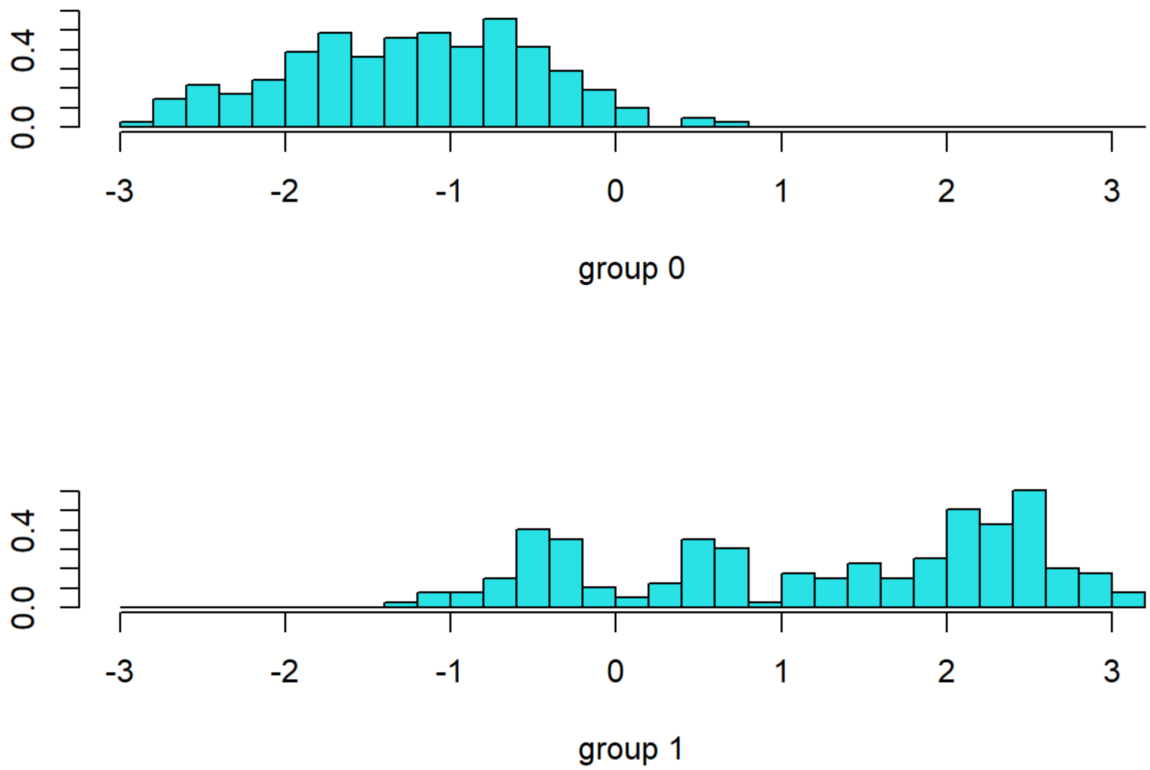
Modelling LDA

```
lda_model = lda(crim_class~., data = train_data)
summary(lda_model)
```

```
##         Length Class  Mode
## prior    2     -none- numeric
## counts   2     -none- numeric
## means   28     -none- numeric
## scaling 14     -none- numeric
## lev      2     -none- character
## svd      1     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

Plotting the values of LDA

```
plot(lda_model)
```



Predicting for test data

```
lda_predict_train = predict(lda_model, newdata = train_data)
lda_predict_test = predict(lda_model, newdata = test_data)
res_train = lda_predict_train$class
res_test = lda_predict_test$class
```

Calculating the train and test errors

```
lda_result_train = which(res_train!=y_train_data)
lda_train_error=length(lda_result_train) / length(y_train_data)
print(lda_train_error)
```

```
## [1] 0.1308642
```

```
lda_result_test = which(res_test!=y_test_data)
lda_test_error=length(lda_result_test) / length(y_test_data)
print(lda_test_error)
```

```
## [1] 0.1782178
```

The train error is 0.1308642 and the test error is 0.1782178 when predicted using LDA.

Modelling KNN with the values k = 1,3,5,7,9,11,13,15

```
require(class)
```

```
## Loading required package: class
```

```
kvalues = c(1,3,5,7,9,11,13,15)
model_knn = c()
error_knn = c()
knn_accuracy = c()
for(i in 1:8) {
  predict_knn <-knn(train_data[,-1],test_data[,-1],y_train_data,k=kvalues[i])
  #model_knn[i] = predict_knn
  error_knn[i]<- mean(predict_knn != y_test_data)

  false_result_knn = which(predict_knn == y_test_data)
  knn_accuracy[i] = length(false_result_knn)/length(y_test_data)
}
```

Error at each k value

```
print(error_knn)
```

```
## [1] 0.07920792 0.06930693 0.04950495 0.06930693 0.07920792 0.08910891 0.09900990
## [8] 0.09900990
```

```
print(min(error_knn))
```

```
## [1] 0.04950495
```

Accuracy at each k value

```
print(knn_accuracy)
```

```
## [1] 0.9207921 0.9306931 0.9504950 0.9306931 0.9207921 0.9108911 0.9009901
## [8] 0.9009901
```

```
print(max(knn_accuracy))
```

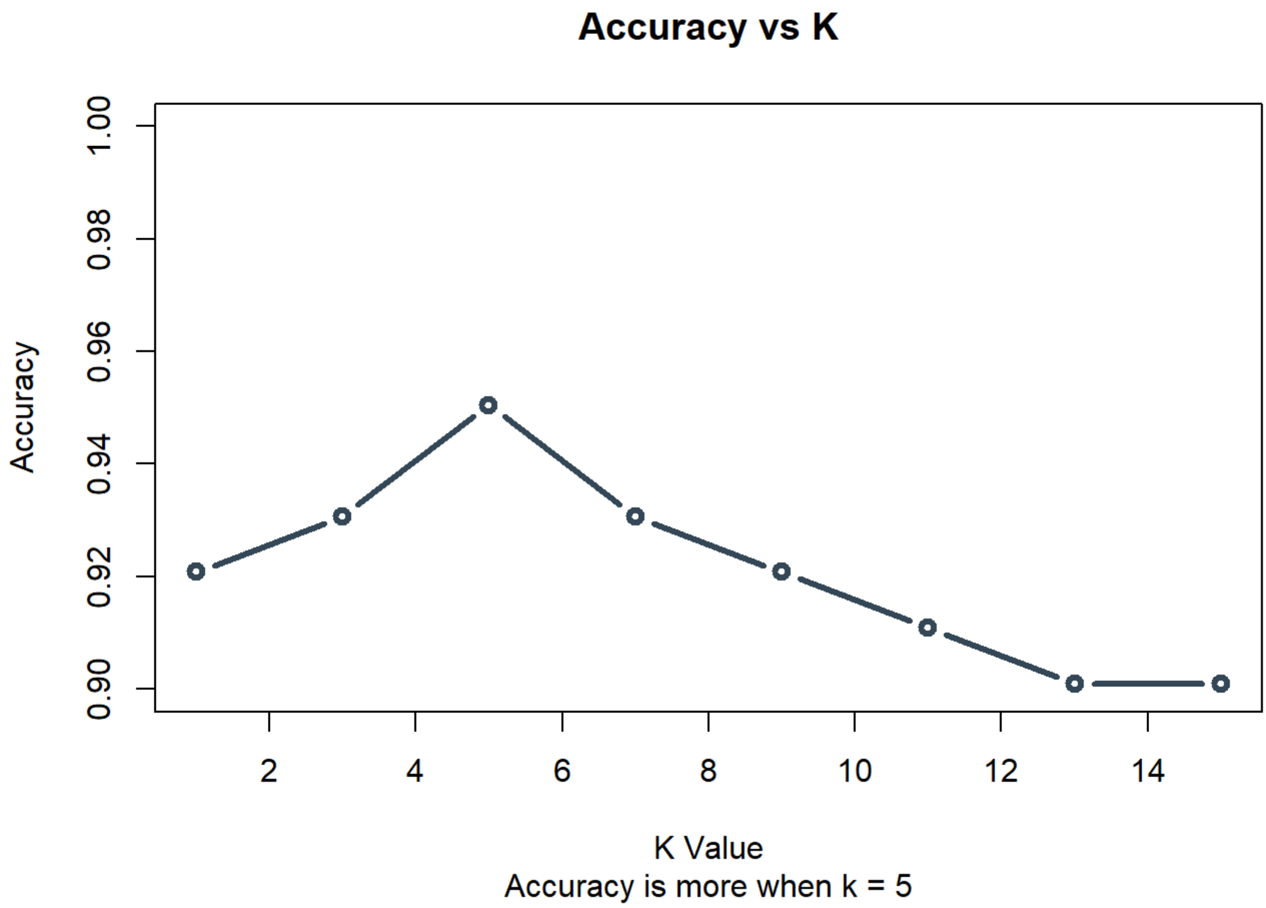
```
## [1] 0.950495
```

From the given k values k = 1,3,5,7,9,11,13,15, k =3 gives better accuracy

When K= 5 the error is 0.04950495

Plotting K and Accuracy

```
plot(kvalues,knn_accuracy, col="#334756", type = "b", xlab = "K Value", ylab = "Accuracy", yl
im = c(0.90,1.0), main = "Accuracy vs K", sub="Accuracy is more when k = 5", lwd = 3.0)
```



Modelling CART

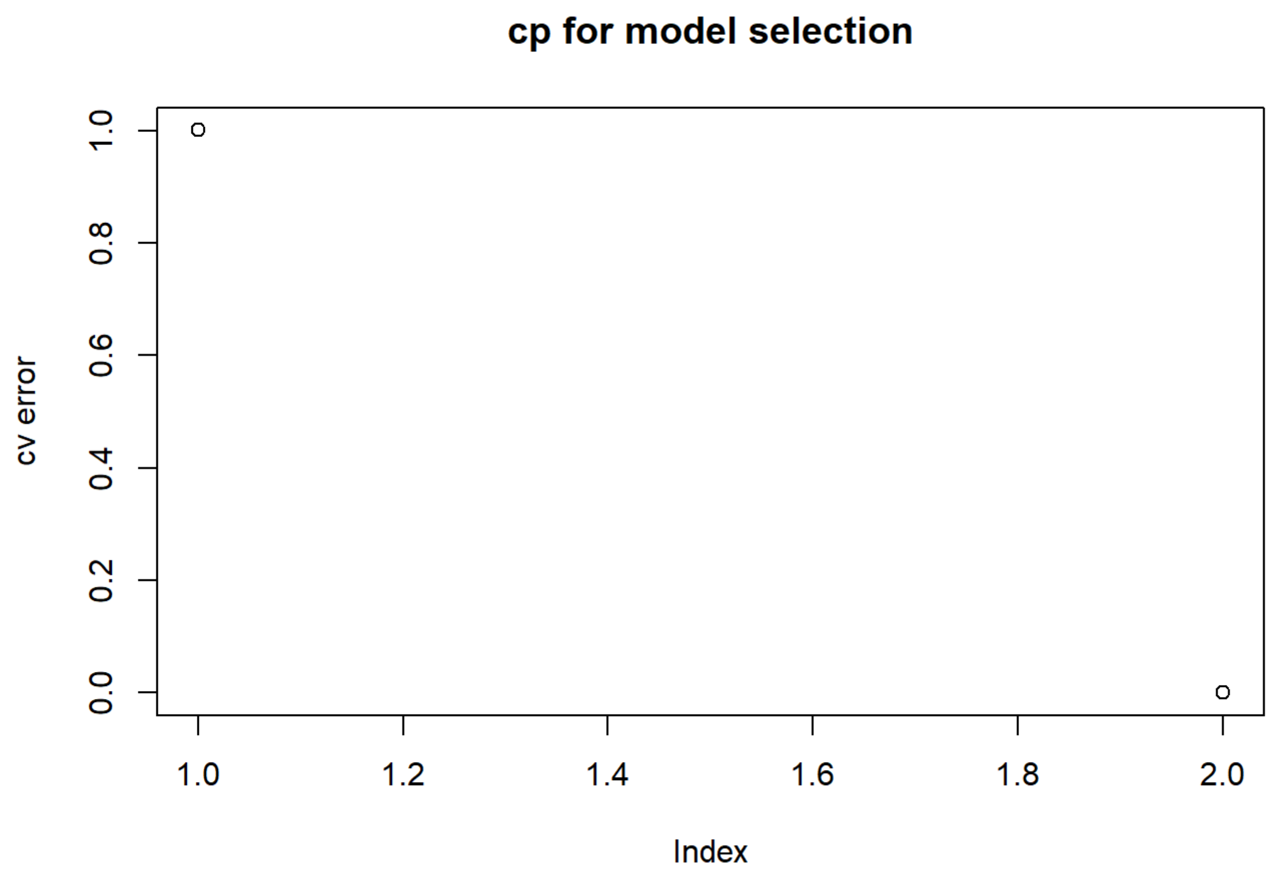
```
model.controls = rpart.control(minsplit = 10, minbucket = 3, xval=5,cp=0)
fit_boston = rpart(crim_class~., data = data, control = model.controls)
```

Finding mininmum cp

```
min_cp = which.min(fit_boston$cptable[,4])
print(min_cp)
```

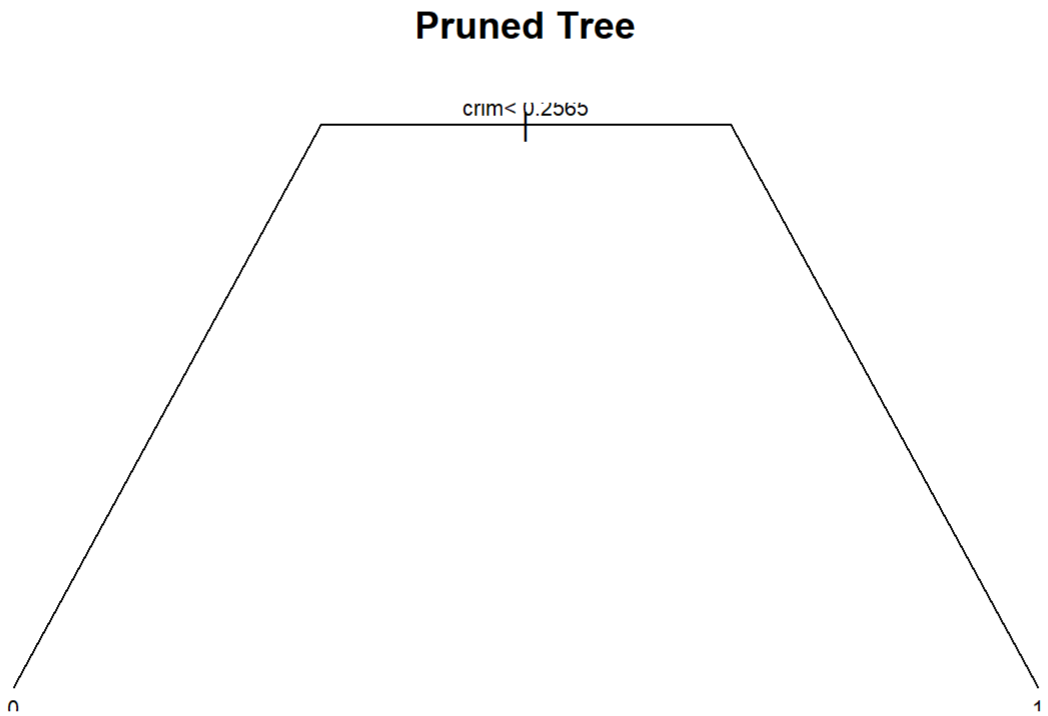
```
## 2
## 2
```

```
pruned_fit = prune(fit_boston, cp = fit_boston$cptable[min_cp,1])
plot(fit_boston$cptable[,4], main = "cp for model selection", ylab="cv error")
```



Pruned Tree

```
plot(pruned_fit, branch=0.4,compress=T, main="Pruned Tree")
text(pruned_fit,cex=0.7)
```



Full Tree

```
plot(fit_boston, branch=0.3,compress=T, main="Full Tree")
text(fit_boston,cex=0.7)
```

Full Tree