Week 6 - Tree Based Methods - Decision Trees

Dr. Liwan Liyanage

School of Computer, Data and Mathematical Sciences

WESTERN SYDNEY UNIVERSITY

Tree-based Methods

- Here we describe tree-based methods for regression and classifcation.
- These involve stratifying or segmenting the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods.

Pros and Cons

Pros

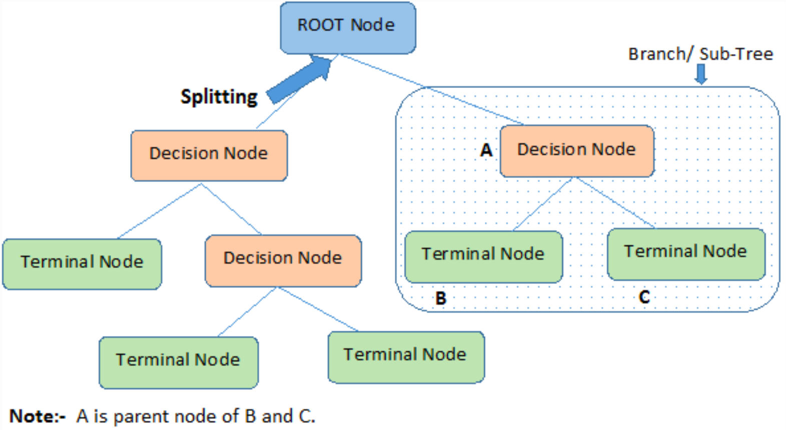
- Easy to explain, most people can understand them.
- Easily represented as visualisation and interpretable.
- Qualitative predictors are easily handled

Cons

- Do not have the same level of predictive accuracy compared to some other approaches
- Non-robust: a small change in the data can cause a large change in the final estimated tree

WESTERN SYDNEY UNIVERSITY

The Basics of Decision Trees



source:https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

WESTERN SYDNEY UNIVERSITY

- Decision trees can be applied to both regression and classification problems.
- We first consider classification problems, and then move on to regression.

Classification Tree

Classification Trees with Carseats data

- View the variables
- Is the Variable Sales Continuous or Categorical?
- Can we do the classification for Sales?

Upload data and view the data

Note that library(tree) is needed to create decision trees

to install the package :

```
install.packages("tree")
```

```r
library(ISLR)
library(tree)
attach(Carseats)
```

Explore variable names and data snapshot

```r
dim(Carseats)
```

```
## [1] 400  11
```

```r
head(Carseats)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

Transfer Sales variable from a Continuous variable to a Categorical variable

- Is the Variable Sales Continuous or Categorical?
- Can we do the classification for Sales?

```
HighSales=ifelse(Sales<=8,"No","Yes")
str(HighSales)
```

```
##  chr [1:400] "Yes" "Yes" "Yes" "No" "No" "Yes" "No" "Yes" "No" "No" "Yes" ...
```

```
HighSales=as.factor(HighSales)
str(HighSales)
```

```
##  Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 1 2 1 1 ...
```

```
CarseatsNew=data.frame(Carseats,HighSales)
head(CarseatsNew)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US HighSales
## 1   Yes Yes       Yes
## 2   Yes Yes       Yes
## 3   Yes Yes       Yes
## 4   Yes Yes        No
## 5   Yes  No        No
## 6    No Yes       Yes
```

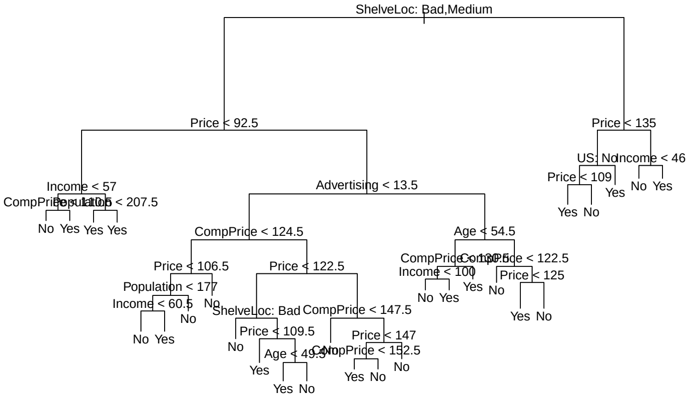
Remove Sales variable and create new data frame with HighSales variable

```
Carseatsnew=CarseatsNew[,-1]
names(Carseatsnew)
```

```
##  [1] "CompPrice"   "Income"      "Advertising" "Population"  "Price"
##  [6] "ShelveLoc"   "Age"         "Education"   "Urban"       "US"
## [11] "HighSales"
```

Fit the tree model for Carseatsnew data

```
tree_model=tree(HighSales~.,Carseatsnew)
plot(tree_model)
text(tree_model,pretty=0)
```


ShelveLoc: Bad,Medium
Price < 92.5
Price < 135
US: No
Income < 46
Price < 109
Yes No
Yes
No Yes
Income < 57
No Yes Yes Yes
Advertising < 13.5
CompPrice < 124.5
Age < 54.5
Price < 106.5
Price < 122.5
Population < 177
Income < 60.5
No
No Yes
ShelveLoc: Bad
CompPrice < 147.5
Price < 109.5
No
Age < 49.5
Yes
Yes No
Price < 147
CompPrice < 152.5
Yes No
No
Income < 100
No Yes
Yes
CompPrice < 122.5
Price < 125
No
Yes No

Check how the model is doing

```
summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = HighSales ~ ., data = Carseatsnew)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

Check how the model is doing ctd. . .

```
tree_pred=predict(tree_model,Carseatsnew,type="class")
table(tree_pred,HighSales)
```

```
##           HighSales
## tree_pred  No Yes
##        No  213  13
##        Yes  23 151
```

WESTERN SYDNEY UNIVERSITY

Calculate the Missclassification rate

```
tab1 <- table(tree_pred,HighSales)
MisclassificationRate <- (tab1[1,2]+tab1[2,1])/sum(tab1)
MisclassificationRate
```

```
## [1] 0.09
```

WESTERN SYDNEY UNIVERSITY

recall

Misclassification rate=(False Positive+ False Negative)/Total

True Positive rate=True Positive/Total Positive

False Positive rate= False Positive/ Total Negative (Type I error)

False Negative rate= False Negative/ Total Positive (Type II error)

Cross Validation using Training and Testing datasets

-Cross validation allows us to check model performance against new observations. (more details Lecture 6)

- Constructing the Training and Testing datasets from the original dataset

```
set.seed(3)
train=sample(1:nrow(Carseatsnew), 200)
Carseats.train=Carseatsnew[train,]
Carseats.test=Carseatsnew[-train,]
dim(Carseats.train)
```

```
## [1] 200  11
```

```
dim(Carseats.test)
```

```
## [1] 200  11
```

```r
head(Carseats.train)
```

```
##     CompPrice Income Advertising Population Price ShelveLoc Age Education Urban
## 261       129    117           8        400   101       Bad  36        10   Yes
## 186       130    100          11        449   107    Medium  64        10   Yes
## 140       146     62          10        310    94    Medium  30        13    No
## 36        131     84          11         29    96    Medium  44        17    No
## 399       100     79           7        284    95       Bad  50        12   Yes
## 363       131     55           0         26   110       Bad  79        12   Yes
##      US HighSales
## 261 Yes        No
## 186 Yes       Yes
## 140 Yes       Yes
## 36  Yes       Yes
## 399 Yes        No
## 363 Yes        No
```

```
head(Carseats.test)
```

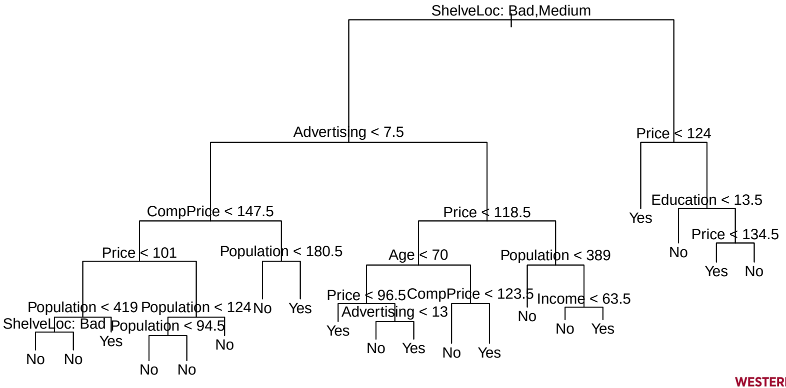
```
##    CompPrice Income Advertising Population Price ShelveLoc Age Education Urban
## 1        138     73          11        276   120       Bad  42        17   Yes
## 3        113     35          10        269    80    Medium  59        12   Yes
## 5        141     64           3        340   128       Bad  38        13   Yes
## 11       121     78           9        150   100       Bad  26        10    No
## 17       118     32           0        284   110      Good  63        13   Yes
## 18       147     74          13        251   131      Good  52        10   Yes
##     US HighSales
## 1  Yes       Yes
## 3  Yes       Yes
## 5   No        No
## 11 Yes       Yes
## 17  No        No
## 18 Yes       Yes
```

Set the HighSales variable (Target Variable) in training and testing data sets

```
HighSales.train=HighSales[train]
HighSales.test=HighSales[-train]
```

Build the tree model for Training data

```
tree_model1=tree(HighSales~.,Carseats.train)
plot(tree_model1)
text(tree_model1,pretty=0)
```



WESTERN SYDNEY UNIVERSITY

Check how the model is doing

```
summary(tree_model1)
```

```
##
## Classification tree:
## tree(formula = HighSales ~ ., data = Carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Advertising" "CompPrice"   "Price"
## [6] "Age"         "Income"      "Education"
## Number of terminal nodes:  20
## Residual mean deviance:  0.3999 = 71.99 / 180
## Misclassification error rate: 0.1 = 20 / 200
```

Check how the model is doing ctd…

Predict the outcomes for Test data using the tree model

```
tree_pred1=predict(tree_model1,Carseats.test,type="class")
table(tree_pred1,HighSales.test)
```

```
##            HighSales.test
## tree_pred1 No Yes
##         No  88  41
##         Yes 23  48
```

Calculate the Misclassification Rate

```
tab2 <- table(tree_pred1,HighSales.test)
misrate <- (tab2[1,2]+tab2[2,1])/sum(tab2)
misrate
```

```
## [1] 0.32
```

Pruning

- When trees are too big and complex, they lead to overfitting (too much variance), which lead to bad prediction
- A smaller tree with fewer splits might lead to lower variance and better interpretation at the cost of a little bias
- Strategy: grow a very large tree, then prune it

Pruning the tree using Cross Validation

```
set.seed(3)
cv.Carseats=cv.tree(tree_model1,FUN=prune.misclass)
names(cv.Carseats)
```

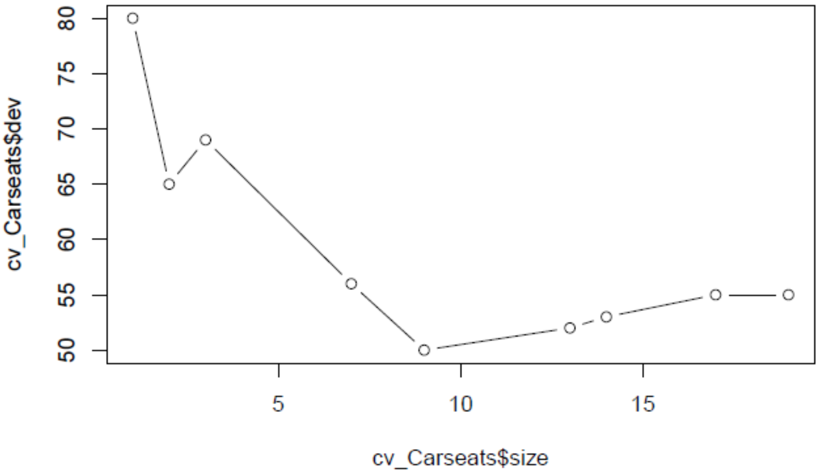
```
## [1] "size"   "dev"    "k"      "method"
```

cv.Carseats

```
## $size
## [1] 19 17 14 13  9  7  3  2  1
##
## $dev
## [1] 55 55 53 52 50 56 69 65 80
##
## $k
## [1]       -Inf  0.0000000  0.6666667  1.0000000  1.7500000  2.0000000
## [7]  4.2500000  5.0000000 23.0000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Figure 1:

```r
plot(cv.Carseats$size, cv.Carseats$dev, type = "b")
```



Pruning the tree

```
prune.Carseats=prune.misclass(tree_model1,best=9)
plot(prune.Carseats)
text(prune.Carseats,pretty=0)
```

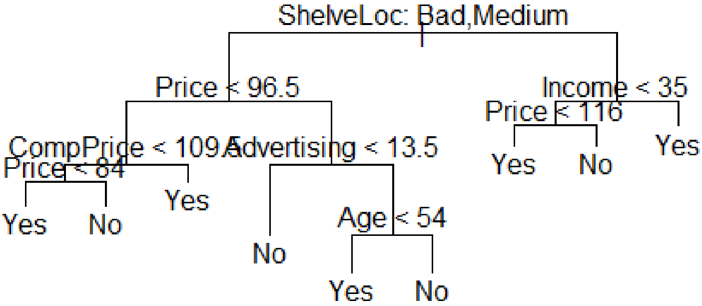

ShelveLoc: Bad,Medium
Price < 96.5
Income < 35
Price < 116
CompPrice < 109.5
Advertising < 13.5
Price < 84
Yes
No
Yes
No
Age < 54
Yes
No
Yes
No
Yes

Figure 3:

Calculate the Misclassification Rate

```
tree.pred2=predict(prune.Carseats,Carseats.test,type='class')
table(tree.pred2,HighSales.test)
```

```
##             HighSales.test
## tree.pred2 No Yes
##        No  92  38
##        Yes 19  51
```

```r
tab3 <- table(tree.pred2,HighSales.test)
mis_rate <- (tab3[1,2]+tab3[2,1])/sum(tab3)
mis_rate
```

```
## [1] 0.285
```

Regression Tree

Regression Trees with Boston data

- View the variables of Boston data with Housing Values in Suburbs of Boston
- Is the Variable "medv", median value of owner-occupied homes in \$1000s. Countinuous or Catagorical?
- Can we do the classification for "medv"?

```
library(MASS)
attach(Boston)
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
```

Variables in Boston data

- crim-per capita crime rate by town.
- zn-proportion of residential land zoned for lots over 25,000 sq.ft.
- indus-proportion of non-retail business acres per town.
- chas-Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- nox-nitrogen oxides concentration (parts per 10 million).
- rm-average number of rooms per dwelling.
- age-proportion of owner-occupied units built prior to 1940.
- dis-weighted mean of distances to five Boston employment entres.
- rad-index of accessibility to radial highways.
- tax-full-value property-tax rate per $10,000.
- ptratio-pupil-teacher ratio by town.
- black-1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- lstat-lower status of the population (percent).
- medv-median value of owner-occupied homes in $1000s.

WESTERN SYDNEY UNIVERSITY

Fit a tree to Train set

```
dim(Boston)
```

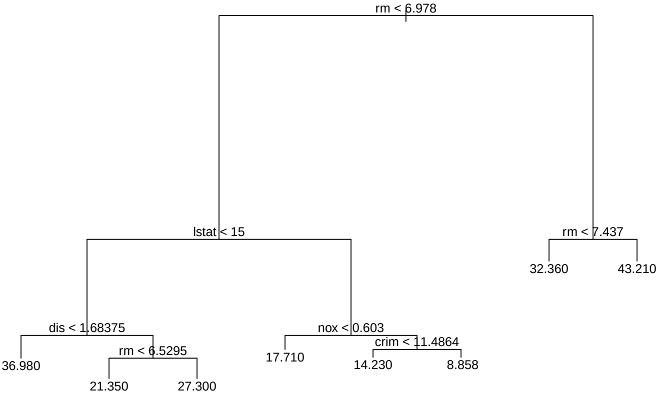
```
## [1] 506  14
```

```
set.seed(5)
train = sample(1:nrow(Boston), nrow(Boston)/2)
tree.boston=tree(medv~.,Boston,subset=train)
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "dis"   "nox"   "crim"
## Number of terminal nodes:  8
## Residual mean deviance:  18.22 = 4463 / 245
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -21.680  -2.009   0.070   0.000   2.146  13.020
```

WESTERN SYDNEY UNIVERSITY

```r
plot(tree.boston)
text(tree.boston,pretty=0)
```

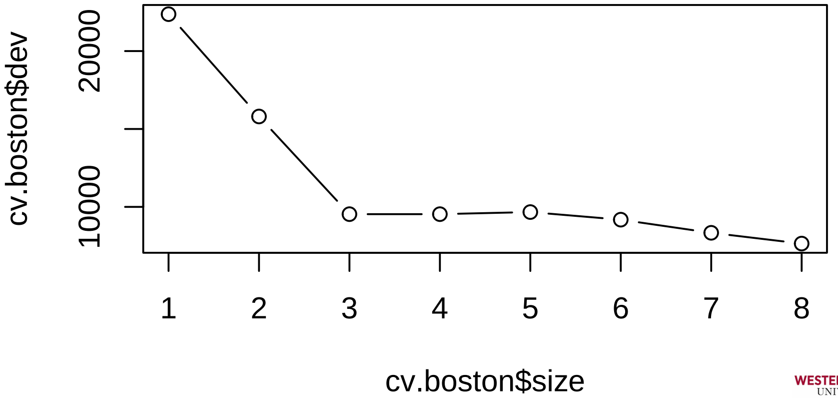


Cross Validation

```
cv.boston=cv.tree(tree.boston)
cv.boston
```

```
## $size
## [1] 8 7 6 5 4 3 2 1
##
## $dev
## [1]  7644.343  8336.673  9179.038  9667.911  9534.281  9533.536 15802.202
## [8] 22368.645
##
## $k
## [1]      -Inf  335.7127  598.9508  863.9841  927.7429  974.3240 4089.7786
## [8] 9528.0219
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.boston$size, cv.boston$dev,type="b")
```

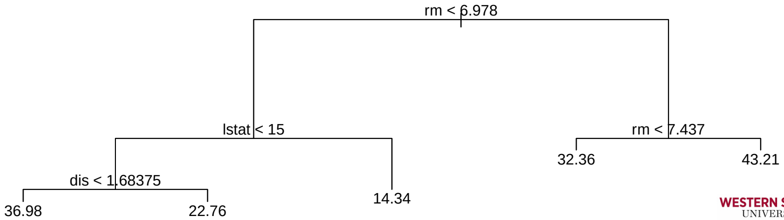


Pruning

Note: Best size is 8 (which is the size of the tree fitted for the training set). Pruning does not improve the model in this situation.

Let's say if the best size is 5, then we can prune it like this.

```
prune.boston=prune.tree(tree.boston,best=5)
#Please note, pruning doesn't improve the moodel in this case
plot(prune.boston)
text(prune.boston,pretty=0)
```



Testing Model Accuracy

- Predict Target variable of the testing data set using the model created by training data set
- Calculate the mean value of the squared errors (MSE)

Testing Model Accuracy

```
yhat=predict(tree.boston,newdata=Boston[-train,])
boston.test=Boston[-train,'medv']
MSE <- mean((yhat-boston.test)^2)
MSE
```

```
## [1] 19.71534
```

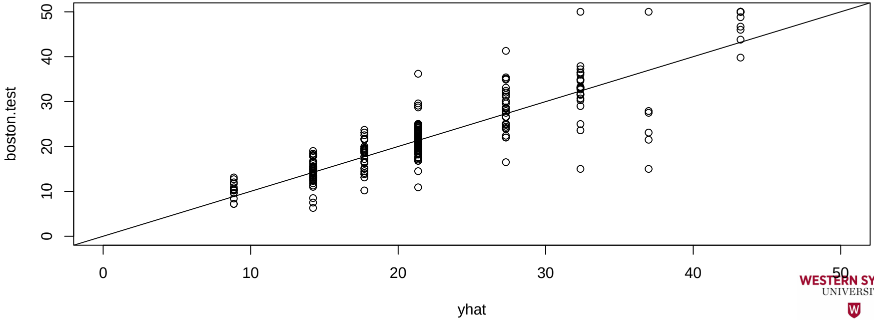
```
RMSE <- sqrt(MSE)
RMSE
```

```
## [1] 4.440196
```

Plot the predicted values against the actual values of the Target variable ("medv") for the testing data set

Insert the abline(0,1)

```
plot(yhat,boston.test, xlim = c(0,50), ylim = c(0,50))
abline(0,1)
```



```
tree.boston
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 253 21780.0 22.340
##    2) rm < 6.978 221 10200.0 20.000
##      4) lstat < 15 140  4313.0 23.270
##        8) dis < 1.68375 5  1281.0 36.980 *
##        9) dis > 1.68375 135  2058.0 22.760
##         18) rm < 6.5295 103   761.9 21.350 *
##         19) rm > 6.5295 32   431.8 27.300 *
##      5) lstat > 15 81  1797.0 14.340
##       10) nox < 0.603 32   453.5 17.710 *
##       11) nox > 0.603 49   744.7 12.150
##         22) crim < 11.4864 30   291.3 14.230 *
##         23) crim > 11.4864 19   117.6  8.858 *
##    3) rm > 6.978 32  2054.0 38.460
##      6) rm < 7.437 14   306.6 32.360 *
##      7) rm > 7.437 18   819.7 43.210 *
```

TEXT BOOK

Lecture notes are based on the textbook.

For further reference refer;

Prescribed Textbook - Chapter 8

– James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R Springer.