#building classification trees with stevens data

### Unit 4 - "Judge, Jury, and Classifier" Lecture

###Using CART method package rpart method=class

#building classification trees with ClaimsData

### Unit 4 - "Keeping an Eye on Healthcare Costs" Lecture

###Using CART method package rpart method=class

###penalty calculation using baseline method

#Regression Trees for Housing Data

### using CART model, rpart() package.

### Cross-validation

# Unit 4 - "Judge, Jury, and Classifier" Lecture

# VIDEO 4

# Read in the data

stevens = read.csv("stevens.csv")

str(stevens)

# Split the data

library(caTools)

set.seed(3000)

spl = sample.split(stevens$Reverse, SplitRatio = 0.7)

Train = subset(stevens, spl==TRUE)

Test = subset(stevens, spl==FALSE)

# Install rpart library

install.packages("rpart")

library(rpart)

install.packages("rpart.plot")

library(rpart.plot)

#In R, a parameter that controls this is minbucket

##The smaller it is, the more splits will be generated

##If it is too small, overfitting will occur

##If it is too large, model will be too simple and accuracywill be poor

# CART model

StevensTree = rpart(Reverse ~ Circuit + Issue + Petitioner + Respondent + LowerCourt + Unconst, data = Train, method="class", minbucket=25)

prp(StevensTree)

# Make predictions

PredictCART = predict(StevensTree, newdata = Test, type = "class")

table(Test$Reverse, PredictCART)

(41+71)/(41+36+22+71)

# ROC curve

library(ROCR)

PredictROC = predict(StevensTree, newdata = Test)

PredictROC

pred = prediction(PredictROC[,2], Test$Reverse)

perf = performance(pred, "tpr", "fpr")

plot(perf)

# VIDEO 5 - Random Forests

# Install randomForest package

install.packages("randomForest")

library(randomForest)

# Build random forest model

# The parameters nodesize🡺 is the number of observations in a subset

# The parameter ntree 🡺 is the number of trees should not be too small , because bagging may ignore some of the observations

StevensForest = randomForest(Reverse ~ Circuit + Issue + Petitioner + Respondent + LowerCourt + Unconst, data = Train, ntree=200, nodesize=25 )

# Convert outcome to factor

Train$Reverse = as.factor(Train$Reverse)

Test$Reverse = as.factor(Test$Reverse)

# Try again

# The parameters nodesize🡺 is the number of observations in a subset

# The parameter ntree 🡺 is the number of trees should not be too small , because bagging may ignore some of the observations

StevensForest = randomForest(Reverse ~ Circuit + Issue + Petitioner + Respondent + LowerCourt + Unconst, data = Train, ntree=200, nodesize=25 )

# Make predictions

PredictForest = predict(StevensForest, newdata = Test)

table(Test$Reverse, PredictForest)

(40+74)/(40+37+19+74)

# VIDEO 6

# Install cross-validation packages

install.packages("caret")

library(caret)

install.packages("e1071")

library(e1071)

# Define cross-validation experiment

numFolds = trainControl( method = "cv", number = 10 )

cpGrid = expand.grid( .cp = seq(0.01,0.5,0.01))

# Perform the cross validation

train(Reverse ~ Circuit + Issue + Petitioner + Respondent + LowerCourt + Unconst, data = Train, method = "rpart", trControl = numFolds, tuneGrid = cpGrid )

# Create a new CART model

StevensTreeCV = rpart(Reverse ~ Circuit + Issue + Petitioner + Respondent + LowerCourt + Unconst, data = Train, method="class", cp = 0.18)

# Make predictions

PredictCV = predict(StevensTreeCV, newdata = Test, type = "class")

table(Test$Reverse, PredictCV)

(59+64)/(59+18+29+64)

#########################################\*

###building trees with ClaimsData

#########################################\*

# Unit 4 - "Keeping an Eye on Healthcare Costs" Lecture

#########################################\*

# VIDEO 6

# Read in the data

Claims = read.csv("ClaimsData.csv")

str(Claims)

# Percentage of patients in each cost bucket

table(Claims$bucket2009)/nrow(Claims)

# Split the data

library(caTools)

set.seed(88)

spl = sample.split(Claims$bucket2009, SplitRatio = 0.6)

ClaimsTrain = subset(Claims, spl==TRUE)

ClaimsTest = subset(Claims, spl==FALSE)

# VIDEO 7

# Baseline method

table(ClaimsTest$bucket2009, ClaimsTest$bucket2008)

(110138 + 10721 + 2774 + 1539 + 104)/nrow(ClaimsTest) #sum up all the diagonal figures divided by totals

# Penalty Matrix

PenaltyMatrix = matrix(c(0,1,2,3,4,2,0,1,2,3,4,2,0,1,2,6,4,2,0,1,8,6,4,2,0), byrow=TRUE, nrow=5)

PenaltyMatrix

# Penalty Error of Baseline Method

as.matrix(table(ClaimsTest$bucket2009, ClaimsTest$bucket2008))\*PenaltyMatrix

sum(as.matrix(table(ClaimsTest$bucket2009, ClaimsTest$bucket2008))\*PenaltyMatrix)/nrow(ClaimsTest)

# VIDEO 8

# Load necessary libraries

library(rpart)

library(rpart.plot)

# CART model

ClaimsTree = rpart(bucket2009 ~ age + alzheimers + arthritis + cancer + copd + depression + diabetes + heart.failure + ihd + kidney + osteoporosis + stroke + bucket2008 + reimbursement2008, data=ClaimsTrain, method="class", cp=0.00005)

prp(ClaimsTree)

# Make predictions

PredictTest = predict(ClaimsTree, newdata = ClaimsTest, type = "class")

table(ClaimsTest$bucket2009, PredictTest)

(114141 + 16102 + 118 + 201 + 0)/nrow(ClaimsTest)

# Penalty Error

as.matrix(table(ClaimsTest$bucket2009, PredictTest))\*PenaltyMatrix

sum(as.matrix(table(ClaimsTest$bucket2009, PredictTest))\*PenaltyMatrix)/nrow(ClaimsTest)

# New CART model with loss matrix

ClaimsTree = rpart(bucket2009 ~ age + alzheimers + arthritis + cancer + copd + depression + diabetes + heart.failure + ihd + kidney + osteoporosis + stroke + bucket2008 + reimbursement2008, data=ClaimsTrain, method="class", cp=0.00005, parms=list(loss=PenaltyMatrix))

# Redo predictions and penalty error

PredictTest = predict(ClaimsTree, newdata = ClaimsTest, type = "class")

table(ClaimsTest$bucket2009, PredictTest)

(94310 + 18942 + 4692 + 636 + 2)/nrow(ClaimsTest)

sum(as.matrix(table(ClaimsTest$bucket2009, PredictTest))\*PenaltyMatrix)/nrow(ClaimsTest)

################################\*

#Regression Trees for Housing Data

################################\*

# Unit 4, Recitation

# VIDEO 2

# Read in data

boston = read.csv("boston.csv")

str(boston)

# Plot observations

plot(boston$LON, boston$LAT)

# Tracts alongside the Charles River

points(boston$LON[boston$CHAS==1], boston$LAT[boston$CHAS==1], col="blue", pch=19)

# Plot MIT

points(boston$LON[boston$TRACT==3531],boston$LAT[boston$TRACT==3531],col="red", pch=20)

# Plot polution

summary(boston$NOX)

points(boston$LON[boston$NOX>=0.55], boston$LAT[boston$NOX>=0.55], col="green", pch=20)

# Plot prices

plot(boston$LON, boston$LAT)

summary(boston$MEDV)

points(boston$LON[boston$MEDV>=21.2], boston$LAT[boston$MEDV>=21.2], col="red", pch=20)

# VIDEO 3

# Linear Regression using LAT and LON

plot(boston$LAT, boston$MEDV)

plot(boston$LON, boston$MEDV)

latlonlm = lm(MEDV ~ LAT + LON, data=boston)

summary(latlonlm)

# Visualize regression output

plot(boston$LON, boston$LAT)

points(boston$LON[boston$MEDV>=21.2], boston$LAT[boston$MEDV>=21.2], col="red", pch=20)

latlonlm$fitted.values

points(boston$LON[latlonlm$fitted.values >= 21.2], boston$LAT[latlonlm$fitted.values >= 21.2], col="blue", pch="$")

# Video 4

# Load CART packages

library(rpart)

library(rpart.plot)

# CART model

latlontree = rpart(MEDV ~ LAT + LON, data=boston)

prp(latlontree)

# Visualize output

plot(boston$LON, boston$LAT)

points(boston$LON[boston$MEDV>=21.2], boston$LAT[boston$MEDV>=21.2], col="red", pch=20)

fittedvalues = predict(latlontree)

points(boston$LON[fittedvalues>21.2], boston$LAT[fittedvalues>=21.2], col="blue", pch="$")

# Simplify tree by increasing minbucket

#In R, a parameter that controls this is minbucket

##The smaller it is, the more splits will be generated

##If it is too small, overfitting will occur

##If it is too large, model will be too simple and accuracy will be poor

latlontree = rpart(MEDV ~ LAT + LON, data=boston, minbucket=50)

plot(latlontree)

text(latlontree)

# Visualize Output

plot(boston$LON,boston$LAT)

abline(v=-71.07)

abline(h=42.21)

abline(h=42.17)

points(boston$LON[boston$MEDV>=21.2], boston$LAT[boston$MEDV>=21.2], col="red", pch=20)

# VIDEO 5

# Let's use all the variables

# Split the data

library(caTools)

set.seed(123)

split = sample.split(boston$MEDV, SplitRatio = 0.7)

train = subset(boston, split==TRUE)

test = subset(boston, split==FALSE)

# Create linear regression

linreg = lm(MEDV ~ LAT + LON + CRIM + ZN + INDUS + CHAS + NOX + RM + AGE + DIS + RAD + TAX + PTRATIO, data=train)

summary(linreg)

# Make predictions

linreg.pred = predict(linreg, newdata=test)

linreg.sse = sum((linreg.pred - test$MEDV)^2)

linreg.sse

# Create a CART model

tree = rpart(MEDV ~ LAT + LON + CRIM + ZN + INDUS + CHAS + NOX + RM + AGE + DIS + RAD + TAX + PTRATIO, data=train)

prp(tree)

# Make predictions

tree.pred = predict(tree, newdata=test)

tree.sse = sum((tree.pred - test$MEDV)^2)

tree.sse

# Video 7

# Load libraries for cross-validation

library(caret)

library(e1071)

# Number of folds

tr.control = trainControl(method = "cv", number = 10)

# cp values

cp.grid = expand.grid( .cp = (0:10)\*0.001)

# What did we just do?

1\*0.001

10\*0.001

0:10

0:10 \* 0.001

# Cross-validation

tr = train(MEDV ~ LAT + LON + CRIM + ZN + INDUS + CHAS + NOX + RM + AGE + DIS + RAD + TAX + PTRATIO, data = train, method = "rpart", trControl = tr.control, tuneGrid = cp.grid)

# Extract tree

best.tree = tr$finalModel

prp(best.tree)

# Make predictions

best.tree.pred = predict(best.tree, newdata=test)

best.tree.sse = sum((best.tree.pred - test$MEDV)^2)

best.tree.sse