**C-4.5 and CART**

install.packages("RWeka")

install.packages("party")

library(RWeka)

library(party)

getwd()

data<-read.csv(file.choose())

data$Outlook<-factor(data$Outlook)

data$Temp<-factor(data$Temp)

data$Humidity<-factor(data$Humidity)

data$Windy<-factor(data$Windy)

data$Play.Golf<-factor(data$Play.Golf)

str(data)

head(data)

summary(data)

fit1 <- J48(Play.Golf ~ Outlook + Temp + Humidity + Windy, data=data)

# summarize the fit

summary(fit1)

install.packages("partykit")

library(partykit)

plot(fit1)

print(fit1)

#-----------------------------------------------------------------------------

#**CART**

data<-read.csv(file.choose())

data$Outlook<-factor(data$Outlook)

data$Temp<-factor(data$Temp)

data$Humidity<-factor(data$Humidity)

data$Windy<-factor(data$Windy)

data$Play.Golf<-factor(data$Play.Golf)

str(data)

head(data)

summary(data)

install.packages("rpart")

library(rpart)

#rpart(formula, data=, method='')

#arguments:

# formula: The function to predict

# data: Specifies the data frame

# method: "class" for a classification tree ,"anova" for a regression tree

fit <- rpart(Play.Golf ~ Outlook + Temp + Humidity + Windy, method="class", data=data,

control=rpart.control(minsplit=1))

summary(fit)

print(fit)

install.packages("rpart.plot")

library(rpart.plot)

rpart.plot(fit, extra = 101)

**Pruning**

install.packages("rpart")

library(rpart)

install.packages("rpart.plot")

library(rpart.plot)

bank.df<-read.csv(choose.files())

str(bank.df)

bank.df <- bank.df[ , -c(1, 5)] # Drop ID and zip code columns.

# partition

set.seed(1)

train.index <- sample(c(1:dim(bank.df)[1]), dim(bank.df)[1]\*0.6)

train.df <- bank.df[train.index, ]

valid.df <- bank.df[-train.index, ]

# classification tree

default.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class")

#personal loan is target variable.method is classification

length(default.ct$frame$var[default.ct$frame$var == "<leaf>"])

#the 8 we are getting here shows the depth of the tree.

# plot tree

prp(default.ct, type =1, split.font = 1, varlen = -10)

# set argument type = "class" in predict() to generate predicted class membership.

default.ct.point.pred.train <- predict(default.ct,train.df,type = "class")

# generate confusion matrix for training data

library(caret)

library(ggplot2)

confusionMatrix(default.ct.point.pred.train, as.factor(train.df$Personal.Loan))

### Tree Pruning

deeper.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class", cp = 0, minsplit = 1)

# count number of leaves

length(deeper.ct$frame$var[deeper.ct$frame$var == "<leaf>"])

# plot tree

prp(deeper.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10,

box.col=ifelse(deeper.ct$frame$var == "<leaf>", 'gray', 'white'))

# argument xval refers to the number of folds to use in rpart's built-in

# cross-validation procedure

# argument cp sets the smallest value for the complexity parameter.

cv.ct <- rpart(Personal.Loan ~ ., data = train.df, method = "class",

cp = 0.00001, minsplit = 5, xval = 5)

# use printcp() to print the table.

printcp(cv.ct)

# prune by lower cp

pruned.ct <- prune(cv.ct,cp = 0.00909)

length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])

prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)

**SVM**

# Importing the dataset

dataset = social

dataset = dataset[3:5]

head(dataset)

str(dataset)

# Encoding the target feature as factor

dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set

library(caTools)

set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Feature Scaling

training\_set[-3] = scale(training\_set[-3])

test\_set[-3] = scale(test\_set[-3])

head(training\_set)

library(e1071)

classifier = svm(formula = Purchased ~ .,

data = training\_set,

type = 'C-classification',

kernel = 'linear')

# Predicting the Test set results

y\_pred = predict(classifier, newdata = test\_set[-3])

y\_pred1<-as.data.frame(y\_pred)

# Making the Confusion Matrix

cm = table(test\_set$Purchased, y\_pred1$y\_pred)

cm

install.packages(ElemStatLearn)

library(ElemStatLearn)

# Plotting the training data set results

set = training\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3],

main = 'SVM (Training set)',

xlab = 'Age', ylab = 'Estimated Salary',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'coral1', 'aquamarine'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

# Test Results

set = test\_set

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

y\_grid = predict(classifier, newdata = grid\_set)

plot(set[, -3], main = 'SVM (Test set)',

xlab = 'Age', ylab = 'Estimated Salary',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'coral1', 'aquamarine'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

################################

set.seed(10111)

x = matrix(rnorm(40), 20, 2)

y = rep(c(-1, 1), c(10, 10))

x[y == 1,] = x[y == 1,] + 1

plot(x, col = y + 3, pch = 19)

dat = data.frame(x, y = as.factor(y))

head(dat)

svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)

print(svmfit)

plot(svmfit, dat)

make.grid = function(x, n = 75) {

grange = apply(x, 2, range)

x1 = seq(from = grange[1,1], to = grange[2,1], length = n)

x2 = seq(from = grange[1,2], to = grange[2,2], length = n)

expand.grid(X1 = x1, X2 = x2)

}

xgrid = make.grid(x)

xgrid[1:10,]

ygrid = predict(svmfit, xgrid)

plot(xgrid, col = c("red","blue")[as.numeric(ygrid)], pch = 20, cex = .2)

points(x, col = y + 3, pch = 19)

points(x[svmfit$index,], pch = 5, cex = 2)

beta = drop(t(svmfit$coefs)%\*%x[svmfit$index,])

beta0 = svmfit$rho

plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = .2)

points(x, col = y + 3, pch = 19)

points(x[svmfit$index,], pch = 5, cex = 2)

abline(beta0 / beta[2], -beta[1] / beta[2])

abline((beta0 - 1) / beta[2], -beta[1] / beta[2], lty = 2)

abline((beta0 + 1) / beta[2], -beta[1] / beta[2], lty = 2)

names(ESL.mixture)

rm(x, y)

attach(ESL.mixture)

plot(x, col = y + 1)

dat = data.frame(y = factor(y), x)

fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial", cost = 5)

xgrid = expand.grid(X1 = px1, X2 = px2)

ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)

points(x, col = y + 1, pch = 19)

func = predict(fit, xgrid, decision.values = TRUE)

func = attributes(func)$decision

xgrid = expand.grid(X1 = px1, X2 = px2)

ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)

points(x, col = y + 1, pch = 19)

contour(px1, px2, matrix(func, 69, 99), level = 0, add = TRUE)

contour(px1, px2, matrix(func, 69, 99), level = 0.5, add = TRUE, col = "blue", lwd = 2) # Best possible classifier

**NN**

df <- read.csv("placement.csv",header=T)

head(df)

# load library

require(neuralnet)

# fit neural network

nn=neuralnet(Placed~.,data=df, hidden=c(2,3),act.fct = "logistic",

linear.output = FALSE)

# plot neural network

plot(nn)

# creating test set

TKS=c(30,40,85,100,20,30,40,60,45,67,90,25,50)

CSS=c(85,50,40,20,80,30,40,75,80,60,30,45,50)

test=data.frame(TKS,CSS)

placed\_test <- c(1,1,0,0,1,0,0,1,1,1,0,0,1)

## Prediction using neural network

Predict=compute(nn,test)

Predict$net.result

# Converting probabilities into binary classes setting threshold level 0.5

prob <- Predict$net.result

pred <- ifelse(prob>0.5, 1, 0)

pred

table(placed\_test,pred)

**Association Analysis**

install.packages("arules")

install.packages("arulesViz")

library(arules)

library(arulesViz)

library(datasets)

# Load the data set

data(Groceries)

head(Groceries)

Grocery<- Groceries@itemInfo

Grocery<- as.data.frame(Grocery)

str(Grocery)

typeof(Grocery)

# Create an item frequency plot for the top 20 items

itemFrequencyPlot(Groceries,topN=20,type="absolute")

###########

#Apriori

#############

# Get the rules

rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))

# Show the top 5 rules, but only 2 digits

options(digits=2)

inspect(rules[1:150])

**Bagging & Boosting**

install.packages("ISLR")

install.packages('psych')

library(psych)

library(ggplot2)

library(lattice)

library(caret)

library(rpart) #for trees

library(rpart.plot) # Enhanced tree plots

library(RColorBrewer) # Color selection for fancy tree plot

library(pROC) #for ROC curves

library(ISLR) #for the Carseat Data

#loading the data

data("Carseats")

#data structure

head(Carseats,10)

str(Carseats)

summary(Carseats)

summary(Carseats$Sales)

#creating new binary variable

Carseats$HighSales=ifelse(Carseats$Sales<=9.32,"No","Yes")

#remove old variable

Carseats$Sales <- NULL

str(Carseats)

#Splitting the data into training and test sets

datasample = sample(dim(Carseats)[1], dim(Carseats)[1]\*0.7)

#create training and test data sets

Carseats.train = Carseats[datasample, ]

Carseats.test = Carseats[-datasample, ]

#check the distribution of HighSales

prop.table(table(Carseats.train$HighSales))

prop.table(table(Carseats.test$HighSales))

# install.packages('e1071')

# define cross-validation (cv) parameters; we'll do 10-fold cross-validation

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

cpGrid = expand.grid( .cp = seq(0.001, to = 0.05, by = 0.001))

#########################################

# Model 1

#########################################

set.seed(10)

train.tree <- train(HighSales ~ .,

data = Carseats.train,

method = "rpart",

control = rpart.control(minsplit = 10),

trControl = numFolds,

tuneGrid = cpGrid

)

train.tree

plot(train.tree)

#pruning

train.tree1 <- rpart(HighSales ~ ., data = Carseats.train, method = "class",

control = rpart.control(minsplit = 10, cp =0.023))

print(train.tree1)

#install.packages("rattle")

library(rattle)

fancyRpartPlot(train.tree1)

#obtaining class predictions

tree.classTrain <- predict(train.tree, type="raw")

head(tree.classTrain)

#computing confusion matrix

confusionMatrix(as.factor(Carseats.train$HighSales),tree.classTrain)

#obtaining class predictions

tree.classTest <- predict(train.tree,

newdata = Carseats.test,

type="raw")

head(tree.classTest)

#computing confusion matrix

confusionMatrix(as.factor(Carseats.test$HighSales),tree.classTest)

#Obtaining predicted probabilites for Test data

tree.probs=predict(train.tree,

newdata=Carseats.test,

type="prob")

head(tree.probs)

#Calculate ROC curve

rocCurve.tree <- roc(Carseats.test$HighSales,tree.probs[,"Yes"])

#plot the ROC curve

plot(rocCurve.tree,col=c(4))

#calculate the area under curve (bigger is better)

auc(rocCurve.tree)

cvcontrol <- trainControl(method="repeatedcv", number = 10,repeats = 10)

train.bagg <- train(as.factor(HighSales) ~ .,

data=Carseats.train,

method="treebag",

trControl=cvcontrol,

nbagg=100,

importance=TRUE)

plot(varImp(train.bagg))

############################################

#Model 3: BOOSTING ADABOOST

###########################################

install.packages("ada")

library(ada)

train.ada <- train(as.factor(HighSales) ~ .,

data=Carseats.train,

method="ada",

trControl=cvcontrol)

train.ada

#obtaining class predictions

ada.classTrain <- predict(train.ada,type="raw")

head(ada.classTrain)

#computing confusion matrix

confusionMatrix(as.factor(Carseats.train$HighSales),ada.classTrain)

#obtaining class predictions

ada.classTest <- predict(train.ada,

newdata = Carseats.test,

type="raw")

head(ada.classTest)

#computing confusion matrix

confusionMatrix(as.factor(Carseats.test$HighSales),ada.classTest)

#Obtaining predicted probabilites for Test data

ada.probs=predict(train.ada,

newdata=Carseats.test,

type="prob")

head(ada.probs)

#Calculate ROC curve

rocCurve.ada <- roc(Carseats.test$HighSales,ada.probs[,"Yes"])

#plot the ROC curve

plot(rocCurve.ada, col=c(3))

#calculate the area under curve (bigger is better)

auc(rocCurve.ada)

#Model 4: Random Forest for classification trees

###############################################

install.packages("randomForest")

library(randomForest)

set.seed(121)

train.rf = randomForest(as.factor(HighSales) ~ ., data=Carseats.train, ntree=100, mtry=3, importance=TRUE)

#ntree= Number of trees to grow, mtry = Number of variables randomly sampled as candidates at each split, importance = Should importance of predictors be assessed?

train.rf

#mean( predict( train.rf ) != Carseats.train$HighSales )

#obtaining class predictions

rf.classTrain <- predict(train.rf, type="response")

head(rf.classTrain)

#computing confusion matrix

confusionMatrix(as.factor(Carseats.train$HighSales),rf.classTrain)

#obtaining class predictions

rf.classTest <- predict(train.rf,

newdata = Carseats.test,

type="class")

head(rf.classTest)

#computing confusion matrix

confusionMatrix(as.factor(Carseats.test$HighSales),rf.classTest)

#Obtaining predicted probabilites for Test data

rf.probs=predict(train.rf,

newdata=Carseats.test,

type="class")

head(rf.probs)

rf.prob1<- as.data.frame(rf.probs)

#Calculate ROC curve

rocCurve.rf <- roc(Carseats.test$HighSales,rf.probs[,"Yes"])

#plot the ROC curve

plot(rocCurve.rf,col=c(1))

#calculate the area under curve (bigger is better)

auc(rocCurve.rf)

plot(rocCurve.tree,col=c(4))

plot(rocCurve.bagg,add=TRUE,col=c(6)) # color magenta is bagg

plot(rocCurve.rf,add=TRUE,col=c(1)) # color black is rf

plot(rocCurve.ada,add=TRUE,col=c(3)) # color green is ada

**Clustering**

#Clustering Code

library(dplyr)

data("USArrests")

str(USArrests)

summary(USArrests)

my\_data <- USArrests %>%

na.omit() %>% # Remove missing values (NA)

scale() # Scale variables

# View the firt 3 rows

head(my\_data, n = 3)

pkgs <- c("factoextra", "NbClust")

install.packages(pkgs)

library(factoextra)

library(NbClust)

# Elbow method

fviz\_nbclust(my\_data, kmeans, method = "wss") +

labs(subtitle = "Elbow method")

set.seed(123)

?kmeans

kmc <- kmeans(my\_data, 4, nstart = 25)

# Print the results

print(kmc)

# Cluster size

kmc$size

# Cluster means

kmc$centers

kmc$betweenss

kmc$totss

kmc$cluster

# Visualize

library("factoextra")

fviz\_cluster(kmc, data = my\_data,

ellipse.type = "convex",

palette = "jco",

ggtheme = theme\_minimal())

#############################################

install.packages("magrittr") # package installations are only needed the first time you use it

install.packages("dplyr") # alternative installation of the %>%

library(magrittr) # needs to be run every time you start R and want to use %>%

library(dplyr) # alternatively, this also loads %>%

# Compute hierarchical clustering

res.hc <- USArrests %>%

scale() %>% # Scale the data

dist(method = "euclidean") %>% # Compute dissimilarity matrix

hclust(method = "complete") # Compute hierachical clustering

# Visualize using factoextra

# Cut in 4 groups and color by groups

fviz\_dend(res.hc, k = 4, # Cut in four groups

cex = 0.5, # label size

k\_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),

color\_labels\_by\_k = TRUE, # color labels by groups

rect = TRUE # Add rectangle around groups

)

##########################################

# Compute hierarchical clustering

res.hc <- USArrests %>%

scale() %>% # Scale the data

dist(method = "euclidean") %>% # Compute dissimilarity matrix

hclust(method = "average") # Compute hierachical clustering

# Visualize using factoextra

# Cut in 4 groups and color by groups

fviz\_dend(res.hc, k = 4, # Cut in four groups

cex = 0.5, # label size

k\_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),

color\_labels\_by\_k = TRUE, # color labels by groups

rect = TRUE # Add rectangle around groups

)

#############

install.packages("fpc")

library("fpc")

set.seed(123)

db <- fpc::dbscan(my\_data, eps = 0.7, MinPts = 3)

# Plot DBSCAN results

library("factoextra")

fviz\_cluster(db, data = my\_data, stand = FALSE,

ellipse = FALSE, show.clust.cent = FALSE,

geom = "point",palette = "jco", ggtheme = theme\_classic())

################################

################################################

install.packages("dbscan")

library(dbscan)

dbscan::kNNdistplot(my\_data, k = 3)