**Random Forest Code**

#Install needed packages for the dataset

install.packages('caret')

install.packages('randomForest')

install.packages("dplyr")

#Load needed packages for the dataset

library(caret)

library(randomForest)

library(dplyr)

#Using Boston Housing dataset for the ISLR package

dfCM <- read.csv("BostonHousing.csv")

summary(dfCM)

#Transform Binary DV to factor

dfCM$CAT..MEDV <- factor(dfCM$CAT..MEDV)

str(dfCM)

#Transform IV CHAS to factor

dfCM$CHAS <- factor(dfCM$CHAS)

#Discarding continous DV

dfCM\_2 <- dfCM %>% select(-c(MEDV))

str(dfCM)

#Data partition using Caret package

set.seed(101)

trainIndexCM\_2 <- createDataPartition(dfCM\_2$CAT..MEDV,

p=0.7,

list=FALSE,

times=1)

dfCM\_2.train <- dfCM\_2[trainIndexCM\_2, ]

dfCM\_2.valid <- dfCM\_2[-trainIndexCM\_2, ]

#Creating default random forest model

rf\_default\_CM <- train(CAT..MEDV~.,

data=dfCM\_2.train,

method='rf',

metric='Accuracy',

ntree=100)

print(rf\_default\_CM)

#Detailed model parameter tuning to search for the best mtry

tuneGrid\_CM <- expand.grid(.mtry=c(1:12))

rf\_mtry\_CM <- train(CAT..MEDV~.,

data=dfCM\_2.train,

method='rf',

metric='Accuracy',

tuneGrid=tuneGrid\_CM,

ntree=100)

print(rf\_mtry\_CM)

plot(rf\_mtry\_CM)

#Evaluating model performance

prediction\_CM <- predict(rf\_mtry\_CM, dfCM\_2.valid)

confusionMatrix(prediction\_CM, dfCM\_2.valid$CAT..MEDV, positive='1')

#Viewing Variable Importance

varImp(rf\_mtry\_CM)

plot(varImp(rf\_mtry\_CM))

**SVM (Linear vs. Radial) Code**

#Load needed packages for the dataset

library(caret)

library(ggplot2)

library(dplyr)

#Loading Boston Housing Dataset

dfCM <- read.csv('BostonHousing.csv')

str(dfCM)

#Transform Binary DV to factor

dfCM$CAT..MEDV <- factor(dfCM$CAT..MEDV)

str(dfCM)

#Transform IV CHAS to factor

dfCM$CHAS <- factor(dfCM$CHAS)

#Discarding continous DV

dfCM\_2 <- dfCM %>% select(-c(MEDV))

str(dfCM\_2)

#Data partition using Caret Package

set.seed(101)

trainIndexCM\_2 <- createDataPartition(dfCM\_2$CAT..MEDV,

p=0.7,

list=FALSE,

times=1)

dfCM\_2.train <- dfCM\_2[trainIndexCM\_2, ]

dfCM\_2.valid <- dfCM\_2[-trainIndexCM\_2, ]

#Create 10-fold cross validation with trainControl() function

trControl\_CM\_2 <- trainControl(method='cv',

number=10,

search='grid')

#SVM Model with linear kernel function

#Pre-processing data with centering and scaling

svm\_linear\_CM\_2 <- train(CAT..MEDV~.,

data=dfCM\_2.train,

method='svmLinear',

trControl=trControl\_CM\_2,

preProcess=c('center', 'scale'))

print(svm\_linear\_CM\_2)

#Evaluating the Linear Model's performance

linear\_pred\_CM\_2 <- predict(svm\_linear\_CM\_2, dfCM\_2.valid)

confusionMatrix(linear\_pred\_CM\_2, dfCM\_2.valid$CAT..MEDV, positive='1')

#SVM Model with Radial Kernel Function

#Pre-processing data with centering and scaling

svm\_radial\_CM\_2 <- train(CAT..MEDV~.,

data=dfCM\_2.train,

method='svmRadial',

trControl=trControl\_CM\_2,

preProcess=c('center', 'scale'))

#Evaluating the Radial Model's performance

radial\_pred\_CM\_2 <- predict(svm\_radial\_CM\_2, dfCM\_2.valid)

confusionMatrix(radial\_pred\_CM\_2, dfCM\_2.valid$CAT..MEDV, positive='1')

#Additional model tuning for the linear SVM model

grid\_linear\_CM\_2 <- expand.grid(C = c(0, 0.25, 0.5, 0.75, 1))

svm\_linear\_tune\_CM\_2 <- train(CAT..MEDV~.,

data=dfCM\_2.train,

method='svmLinear',

trControl=trControl\_CM\_2,

preProcess=c('center', 'scale'),

tuneGrid=grid\_linear\_CM\_2)

print(svm\_linear\_tune\_CM\_2)

#Evaluating the Linear SVM model performance

linear\_tune\_pred\_CM\_2 <- predict(svm\_linear\_tune\_CM\_2, dfCM\_2.valid)

confusionMatrix(linear\_tune\_pred\_CM\_2,dfCM\_2.valid$CAT..MEDV, positive='1')