# k-Fold Cross Validation

#improving model performances by k-fold cross varidation

#it split training set into suppose 10 different fold(mostly take 10) which after iterate one by one on 9 fold and then test on last fold

#then take average of accuracy of different iterated and then calculate variance

#then there is tradeoff between accuracy and biased

#model we use kernal-SVM in part 3 and add k-fold cross validation

# Importing the dataset

dataset = read.csv('Social\_Network\_Ads.csv')

dataset = dataset[3:5]

# 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

# install.packages('caTools')

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])

# Fitting Kernel SVM to the Training set

# install.packages('e1071')

library(e1071)

classifier = svm(formula = Purchased ~ .,

data = training\_set,

type = 'C-classification',

kernel = 'radial')

# Predicting the Test set results

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

# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

# Applying k-Fold Cross Validation

# install.packages('caret')

library(caret)

folds = createFolds(training\_set$Purchased, k = 10)#create 10 dold of training set

cv = lapply(folds, function(x) { #first argument is list where we will apply i.e folds

training\_fold = training\_set[-x, ] #here we withdraw test fold i.e -X because X is each element of fold list here(above) and "," to take all column

test\_fold = training\_set[x, ]

classifier = svm(formula = Purchased ~ ., #then ADD KERNAL SVM classifier

data = training\_fold, #careful here training at training\_fold

type = 'C-classification',

kernel = 'radial')

y\_pred = predict(classifier, newdata = test\_fold[-3]) #to predict test set here(testing performance)

cm = table(test\_fold[, 3], y\_pred) #create confusion matrix for observe test\_fold

accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1]) #to evaluate accuracy//cm[1,1] no of correct prediction in first class,,cm[1,2] no of incorrect prediction in first class

return(accuracy) #type CV here vto find 10 accuracy

})

accuracy = mean(as.numeric(cv))#to find out mean which tell overall accuracy which we got 91%

# Visualising the Training set results

library(ElemStatLearn)

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 = 'Kernel 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, 'springgreen3', 'tomato'))

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

# Visualising the Test set results

library(ElemStatLearn)

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 = 'Kernel 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, 'springgreen3', 'tomato'))

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