# Kernel PCA

#above pca and lda work on linear problem i.e when data is linearly seperable

#kernal pca woprk for non-linear problem which is kernalised version of pca where we map data to higherdimension using kernal trick then from there we extract new principal component

#here we are using logestic regression model from previous data

# Importing the dataset

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

dataset = dataset[, 3:5]

# 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[, 1:2] = scale(training\_set[, 1:2])

test\_set[, 1:2] = scale(test\_set[, 1:2])

# Applying Kernel PCA

# install.packages('kernlab')

library(kernlab)

kpca = kpca(~., data = training\_set[-3], kernel = 'rbfdot', features = 2) #[-3] to remove dependent variable, rbfdot is gaussion kernal,, so in oreder to visulize the 2-D we will keep feature 2(final independent variable)

training\_set\_pca = as.data.frame(predict(kpca, training\_set)) #transform original data into new extracted training set, this will return matrix so add detaframe

training\_set\_pca$Purchased = training\_set$Purchased #add dependent variable into new training set variable of training\_set\_pca

test\_set\_pca = as.data.frame(predict(kpca, test\_set))

test\_set\_pca$Purchased = test\_set$Purchased

# Fitting Logistic Regression to the Training set

classifier = glm(formula = Purchased ~ ., #it have linear classifier

family = binomial,

data = training\_set\_pca) # here data is training\_set\_pca

# Predicting the Test set results

prob\_pred = predict(classifier, type = 'response', newdata = test\_set\_pca[-3])

y\_pred = ifelse(prob\_pred > 0.5, 1, 0)

# Making the Confusion Matrix

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

cm

# Visualising the Training set results

#install.packages('ElemStatLearn')

library(ElemStatLearn)

set = training\_set\_pca

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('V1', 'V2')# 'V1', 'V2' is column name of new training set

prob\_set = predict(classifier, type = 'response', newdata = grid\_set)

y\_grid = ifelse(prob\_set > 0.5, 1, 0)

plot(set[, -3],

main = 'Logistic Regression (Training set)',

xlab = 'PC1', ylab = 'PC2',

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

# install.packages('ElemStatLearn')

library(ElemStatLearn)

set = test\_set\_pca

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('V1', 'V2')# 'V1', 'V2' is column name of new test set

prob\_set = predict(classifier, type = 'response', newdata = grid\_set)

y\_grid = ifelse(prob\_set > 0.5, 1, 0)

plot(set[, -3],

main = 'Logistic Regression (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'))