**Simple linear Regression**

dataset = read.csv('Salary\_Data.csv')

library(caTools)

set.seed(231)

split = sample.split(dataset$Salary, SplitRatio = 2/3)

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

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

regressor = lm(formula = Salary ~ YearsExperience,

data = training\_set)

y\_pred = predict(regressor, newdata = test\_set)

print(y\_pred)

library(ggplot2)

ggplot() +

geom\_point(aes(x = training\_set$YearsExperience, y = training\_set$Salary),

colour = 'red') +

geom\_line(aes(x = training\_set$YearsExperience, y = predict(regressor, newdata = training\_set)),

colour = 'blue') +

geom\_smooth(method = 'lm') +

ggtitle('Salary vs Experience (Training set)') +

xlab('Years of experience') +

ylab('Salary')

ggplot() +

geom\_point(aes(x = test\_set$YearsExperience, y = test\_set$Salary),

colour = 'red') +

geom\_line(aes(x = training\_set$YearsExperience, y = predict(regressor, newdata = training\_set)),

colour = 'blue') +

geom\_smooth(method = 'lm') +

ggtitle('Salary vs Experience (Test set)') +

xlab('Years of experience') +

ylab('Salary')

**Multiple Linear Regression**

df <- read.csv("ToyotaCorolla.csv")

summary(df)

head(df)

Toyota<-df[c("Price","Age\_08\_04","KM","HP","cc","Doors","Gears","Quarterly\_Tax","Weight")]

pairs(Toyota)

library(caret)

set.seed(1234)

trainIndex <- createDataPartition(df$Price, p = .8, list = FALSE)

head(trainIndex)

train\_data <- df[ trainIndex,]

test\_data <- df[-trainIndex,]

fit <- lm(Price ~ Age\_08\_04 + KM + Fuel\_Type + HP + Weight,

data = train\_data)

summary(fit)

fit1 <- lm(Price ~ Age\_08\_04 + KM + Fuel\_Type + HP + Met\_Color + Automatic + cc + Doors + Weight,

data = train\_data)

summary(fit1)

#prediction

# Predict the price on testing set

test\_data$Pred <- predict(fit1, test\_data)

# Plot the price and predicted price

plot(test\_data$Price,test\_data$Pred,xlab="Price",ylab="predicts")

# RMSE (root mean-squared error)

cat("RMSE =", sqrt(mean((test\_data$Price-test\_data$Pred)^2)))

**KNN**

# Installing Packages

install.packages("e1071")

install.packages("caTools")

install.packages("class")

# Loading package

library(e1071)

library(caTools)

library(class)

# Loading data

data(iris)

head(iris)

# Splitting data into train

# and test data

split <- sample.split(iris, SplitRatio = 0.7)

train\_cl <- subset(iris, split == "TRUE")

test\_cl <- subset(iris, split == "FALSE")

# Feature Scaling

train\_scale <- scale(train\_cl[, 1:4])

test\_scale <- scale(test\_cl[, 1:4])

# Fitting KNN Model

# to training dataset

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 1)

classifier\_knn

# Confusiin Matrix

cm <- table(test\_cl$Species, classifier\_knn)

cm

# Model Evaluation - Choosing K

# Calculate out of Sample error

misClassError <- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 3

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 3)

misClassError <- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 5

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 5)

misClassError <- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 7

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 7)

misClassError <- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 15

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 15)

misClassError <- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

# K = 19

classifier\_knn <- knn(train = train\_scale,

test = test\_scale,

cl = train\_cl$Species,

k = 19)

misClassError <- mean(classifier\_knn != test\_cl$Species)

print(paste('Accuracy =', 1-misClassError))

**KMEANS**

# Installing Packages

install.packages("ClusterR")

install.packages("cluster")

# Loading package

library(ClusterR)

library(cluster)

# Removing initial label of

# Species from original dataset

iris\_1 <- iris[, -5]

# Fitting K-Means clustering Model

# to training dataset

set.seed(240) # Setting seed

kmeans.re <- kmeans(iris\_1, centers = 3, nstart = 20)

kmeans.re

# Cluster identification for

# each observation

kmeans.re$cluster

# Confusion Matrix

cm <- table(iris$Species, kmeans.re$cluster)

cm

# Model Evaluation and visualization

plot(iris\_1[c("Sepal.Length", "Sepal.Width")])

plot(iris\_1[c("Sepal.Length", "Sepal.Width")],

col = kmeans.re$cluster)

plot(iris\_1[c("Sepal.Length", "Sepal.Width")],

col = kmeans.re$cluster,

main = "K-means with 3 clusters")

## Plotiing cluster centers

kmeans.re$centers

kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]

# cex is font size, pch is symbol

points(kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")],

col = 1:3, pch = 8, cex = 3)

## Visualizing clusters

y\_kmeans <- kmeans.re$cluster

clusplot(iris\_1[, c("Sepal.Length", "Sepal.Width")],

y\_kmeans,

lines = 0,

shade = TRUE,

color = TRUE,

labels = 2,

plotchar = FALSE,

span = TRUE,

main = paste("Cluster iris"),

xlab = 'Sepal.Length',

ylab = 'Sepal.Width')

**Navie Bayes**

# Installing Packages

install.packages("e1071")

install.packages("caTools")

install.packages("caret")

# Loading package

library(e1071)

library(caTools)

library(caret)

# Splitting data into train

# and test data

split <- sample.split(iris, SplitRatio = 0.7)

train\_cl <- subset(iris, split == "TRUE")

test\_cl <- subset(iris, split == "FALSE")

# Feature Scaling

train\_scale <- scale(train\_cl[, 1:4])

test\_scale <- scale(test\_cl[, 1:4])

# Fitting Naive Bayes Model

# to training dataset

set.seed(120) # Setting Seed

classifier\_cl <- naiveBayes(Species ~ ., data = train\_cl)

classifier\_cl

# Predicting on test data'

y\_pred <- predict(classifier\_cl, newdata = test\_cl)

# Confusion Matrix

cm <- table(test\_cl$Species, y\_pred)

cm

# Model Evaluation

confusionMatrix(cm)