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)
```