Content (Code in R)

1. Linear regression (Chapter 3)
2. Logistic Regression (Chapter 4)
3. Decision Tree (Chapter 5)
4. Random Forest (Chapter 6)
5. Rule base classifier (Chapter 7)
6. Naïve Bayes Classifier (Chapter 8)
7. K-NN (Chapter 9)
8. Support Vector Machine (Chapter 10)
9. K-means Clustering (Chapter 11)
10. PCA (Principal Component Analysis) (Chapter 12)
11. Dimensionality Reduction with t-SNE (Chapter 12)
12. Association Rule Mining (Chapter 13)
13. FP growth (Chapter 13)

**LINEAR REGRESSION**

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is −

* Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
* Create a relationship model using the **lm()** functions in R.
* Find the coefficients from the model created and create the mathematical equation using these
* Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
* To predict the weight of new persons, use the **predict()** function in R.

# Values of height

151, 174, 138, 186, 128, 136, 179, 163, 152, 131

# Values of weight.

63, 81, 56, 91, 47, 57, 76, 72, 62, 48

**Create Relationship Model & get the Coefficients**

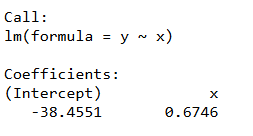
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

**# Apply the lm() function.**

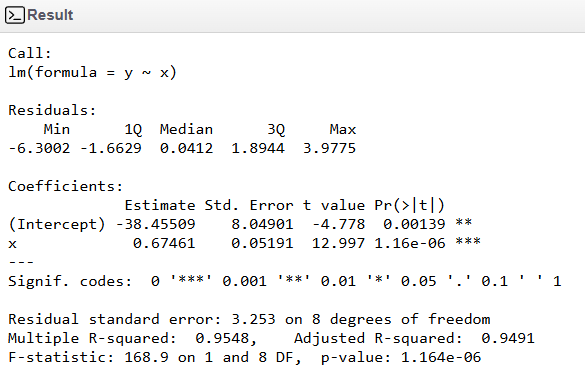
relation <- lm(y~x)

print(relation)



**Get the Summary of the Relationship**

print(summary(relation))



**Predict the weight of new persons**

**# The predictor vector.**

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

**# The resposne vector.**

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

**# Apply the lm() function.**

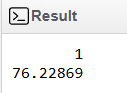
relation <- lm(y~x)

**# Find weight of a person with height 170.**

a <- data.frame(x = 170)

result <- predict(relation,a)

print(result)



**Visualize the Regression Graphically**

**# Create the predictor and response variable.**

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

relation <- lm(y~x)

**# Give the chart file a name.**

png(file = "linearregression.png")

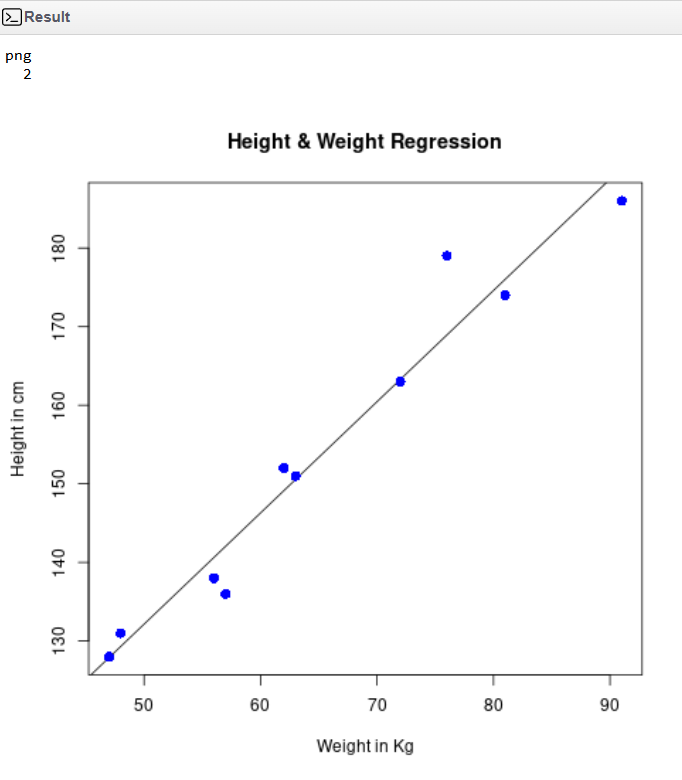
**# Plot the chart.**

plot(y,x,col = "blue",main = "Height & Weight Regression",

abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

**# Save the file.**

dev.off()



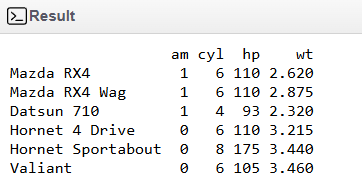
**LOGISTIC REGRESSION**

The in-built data set "mtcars" describes different models of a car with their various engine specifications. In "mtcars" data set, the transmission mode (automatic or manual) is described by the column am which is a binary value (0 or 1). We can create a logistic regression model between the columns "am" and 3 other columns - hp, wt and cyl.

**# Select some columns form mtcars.**

input <- mtcars[,c("am","cyl","hp","wt")]

print(head(input))

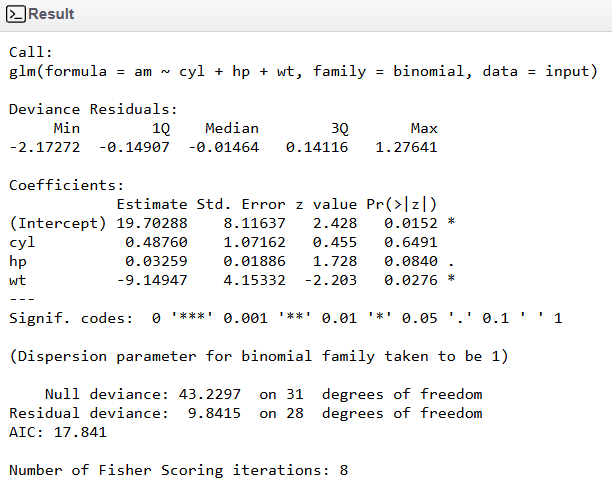


**Create Regression Model**

input <- mtcars[,c("am","cyl","hp","wt")]

am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)

print(summary(am.data))



**DECISION TREE**

Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions. It is mostly used in Machine Learning and Data Mining applications using R.

**Install R Package**

Use the below command in R console to install the package. You also have to install the dependent packages if any.

install.packages("party")

**Input Data**

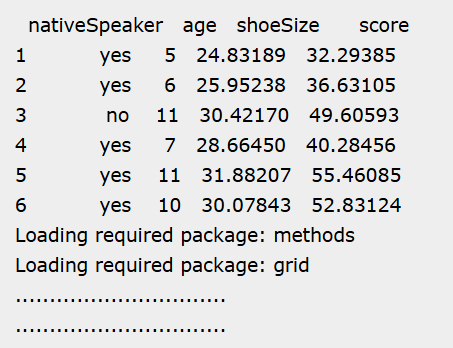
**# Load the party package. It will automatically load other**

**# dependent packages.**

library(party)

**# Print some records from data set readingSkills.**

print(head(readingSkills))

****

**# Load the party package. It will automatically load other**

**# dependent packages.**

library(party)

**# Create the input data frame.**

input.dat <- readingSkills[c(1:105),]

**# Give the chart file a name.**

png(file = "decision\_tree.png")

**# Create the tree.**

output.tree <- ctree(

nativeSpeaker ~ age + shoeSize + score,

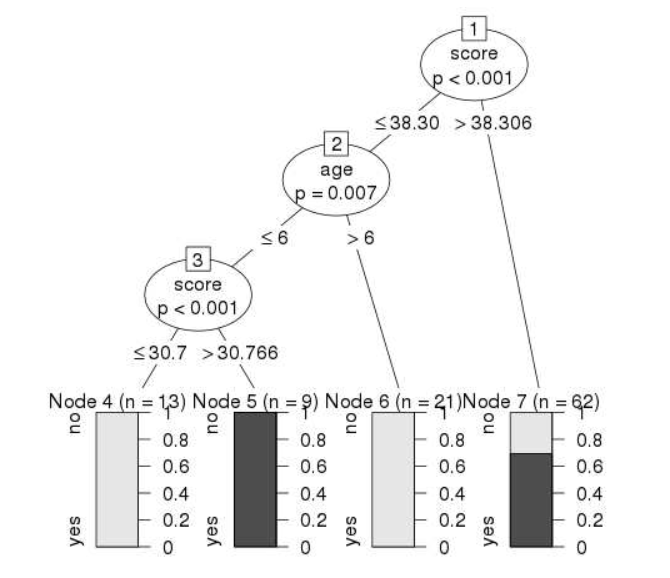
data = input.dat)

**# Plot the tree.**

plot(output.tree)

**# Save the file.**

dev.off()



**RANDOM FOREST**

In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

**# Load the party package. It will automatically load other**

**# required packages.**

library(party)

library(randomForest)

**# Create the forest.**

output.forest <- randomForest(nativeSpeaker ~ age + shoeSize + score,

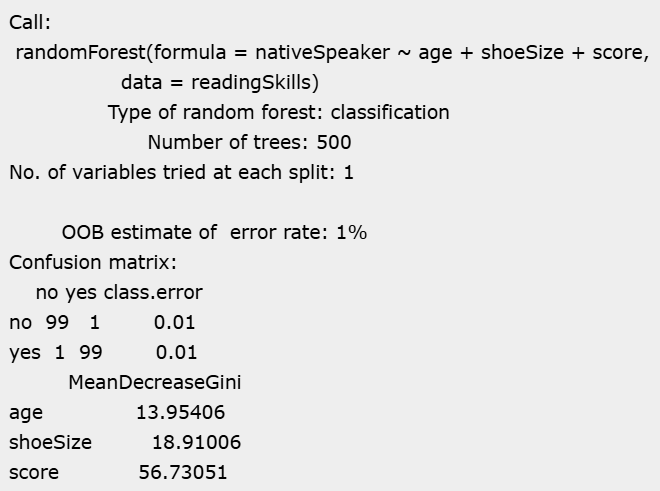
data = readingSkills)

**# View the forest results.**

print(output.forest)

**# Importance of each predictor.**

print(importance(fit,type = 2))



**NAVIE BAYES CLASSFIER**

# Install and load the e1071 package

install.packages("e1071")

library(e1071)

# Load the Iris dataset

data(iris)

# Split the data into training and test sets

set.seed(123) # For reproducibility

sample\_index <- sample(1:nrow(iris), 0.7 \* nrow(iris))

train\_data <- iris[sample\_index, ]

test\_data <- iris[-sample\_index, ]

# Train the Naive Bayes model

model <- naiveBayes(Species ~ ., data = train\_data)

# Make predictions on the test set

predictions <- predict(model, test\_data)

# Evaluate the model

confusion\_matrix <- table(predictions, test\_data$Species)

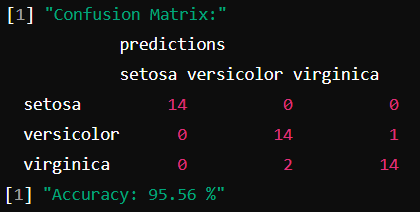
accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)

# Output the results

print("Confusion Matrix:")

print(confusion\_matrix)

print(paste("Accuracy:", round(accuracy \* 100, 2), "%"))



**K NEAREST NEIGHBOR**

# Install and load the class package

install.packages("class")

library(class)

# Load the Iris dataset

data(iris)

# Normalize the data

normalize <- function(x) {

return ((x - min(x)) / (max(x) - min(x)))

}

iris\_norm <- as.data.frame(lapply(iris[, 1:4], normalize))

iris\_norm$Species <- iris$Species

# Split the data into training and test sets

set.seed(123) # For reproducibility

sample\_index <- sample(1:nrow(iris), 0.7 \* nrow(iris))

train\_data <- iris\_norm[sample\_index, ]

test\_data <- iris\_norm[-sample\_index, ]

# Train the KNN model and make predictions on the test set

k <- 3

train\_labels <- train\_data$Species

test\_labels <- test\_data$Species

predictions <- knn(train = train\_data[, -5], test = test\_data[, -5], cl = train\_labels, k = k)

# Evaluate the model

confusion\_matrix <- table(predictions, test\_labels)

accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)

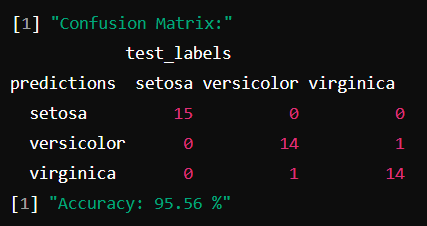
# Output the results

print("Confusion Matrix:")

print(confusion\_matrix)

print(paste("Accuracy:", round(accuracy \* 100, 2), "%"))

**OUTPUT**



**SUPPORT VECTOR MACHINE**

# Install and load necessary packages

install.packages("e1071")

install.packages("caTools")

install.packages("ggplot2")

library(e1071)

library(caTools)

library(ggplot2)

# Load the dataset

social\_data <- read.csv("Social.csv")

# Split the dataset into training and test sets

set.seed(123) # For reproducibility

split <- sample.split(social\_data$Purchased, SplitRatio = 0.75)

train\_data <- subset(social\_data, split == TRUE)

test\_data <- subset(social\_data, split == FALSE)

# Fit the SVM classifier

classifier <- svm(formula = Purchased ~ .,

data = train\_data,

type = 'C-classification',

kernel = 'linear')

# Predict the test results

predictions <- predict(classifier, newdata = test\_data)

# Make the confusion matrix

confusion\_matrix <- table(test\_data$Purchased, predictions)

print(confusion\_matrix)

# Visualize the training set results

set <- train\_data

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)

ggplot() +

geom\_point(data = set, aes(x = Age, y = EstimatedSalary, color = as.factor(Purchased)), shape = 21) +

geom\_contour(data = data.frame(grid\_set, y\_grid), aes(x = Age, y = EstimatedSalary, z = as.numeric(y\_grid)), binwidth = 1, color = 'blue') +

labs(title = 'SVM (Training set)', x = 'Age', y = 'Estimated Salary')

# Visualize the test set results

set <- test\_data

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)

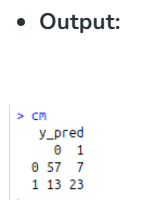
ggplot() +

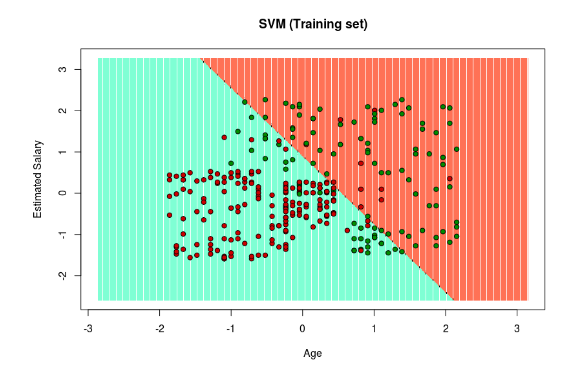
geom\_point(data = set, aes(x = Age, y = EstimatedSalary, color = as.factor(Purchased)), shape = 21) +

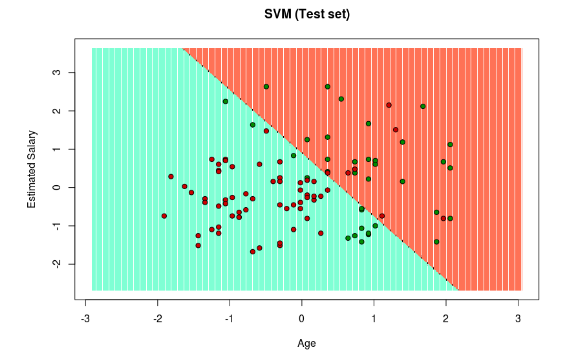
geom\_contour(data = data.frame(grid\_set, y\_grid), aes(x = Age, y = EstimatedSalary, z = as.numeric(y\_grid)), binwidth = 1, color = 'blue') +

labs(title = 'SVM (Test set)', x = 'Age', y = 'Estimated Salary')

**CONFUSION MATRIX**







**K MEANS CLUSTERING**

# Load the Iris dataset

data(iris)

# Run K-means clustering with 3 clusters

set.seed(123) # For reproducibility

kmeans\_result <- kmeans(iris[, 1:4], centers = 3, nstart = 20)

# Create a table to compare the clustering results with actual species

cluster\_assignment <- table(iris$Species, kmeans\_result$cluster)

# Print the table to see the assignment

print(cluster\_assignment)

# Create a confusion matrix by mapping clusters to species

species\_map <- apply(cluster\_assignment, 2, which.max)

mapped\_clusters <- species\_map[kmeans\_result$cluster]

# Create a confusion matrix

confusion\_matrix <- table(iris$Species, mapped\_clusters)

print(confusion\_matrix)

# Plot the K-means clusters for sepal length and sepal width

library(ggplot2)

# Create a data frame for plotting

plot\_data <- iris

plot\_data$Cluster <- as.factor(kmeans\_result$cluster)

# Plot the clusters

ggplot(plot\_data, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster)) +

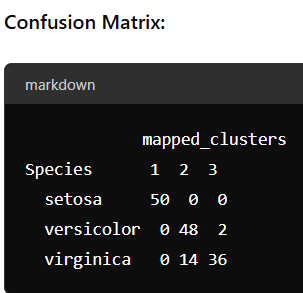
geom\_point(size = 3) +

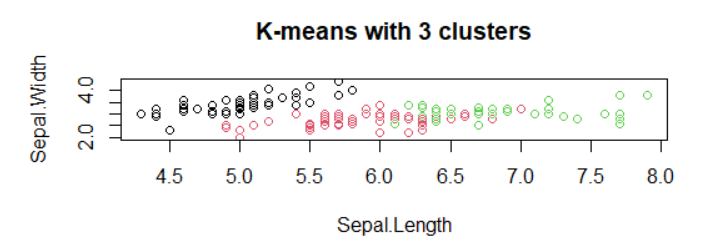
labs(title = "K-means Clustering (k=3) on Iris Dataset",

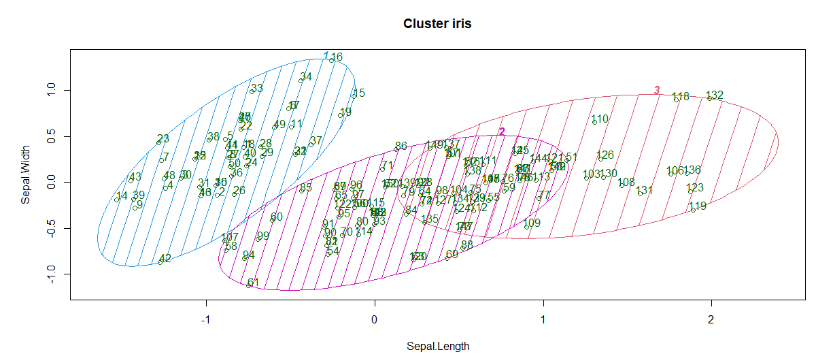
x = "Sepal Length",

y = "Sepal Width") +

theme\_minimal()







**PCA (PRINCIPAL COMPONENT ANALYSIS)**

# Load required libraries

library(tidyverse)

# Load the USArrests dataset

data("USArrests")

# Standardize the variables (mean centering and scaling to unit variance)

df <- scale(USArrests)

# Perform PCA

pca\_result <- prcomp(df, center = TRUE, scale. = TRUE)

# Extract principal component loadings

loadings <- pca\_result$rotation

# Extract principal component scores

scores <- pca\_result$x

# Variance explained by each principal component

variance\_explained <- pca\_result$sdev^2 / sum(pca\_result$sdev^2)

# Create a biplot to visualize PCA results

biplot(pca\_result, scale = 0, cex = 0.8)

# Add labels for variables (arrest categories)

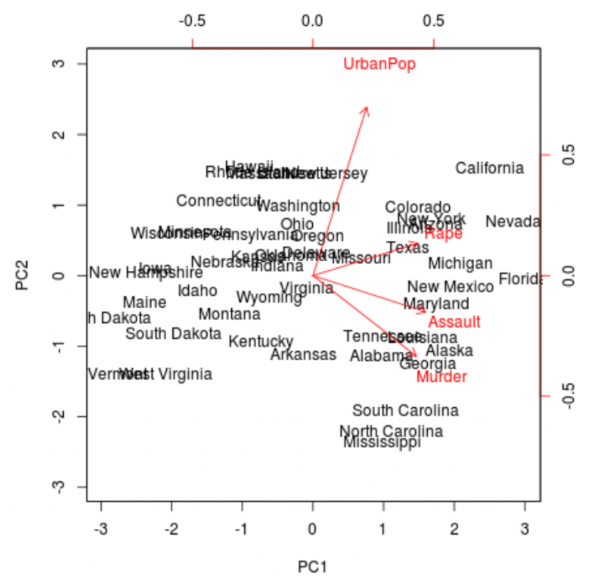
text(pca\_result$rotation[, 1:2], labels = colnames(USArrests), col = "blue")

# Add labels for observations (states)

text(pca\_result$x[, 1:2], labels = rownames(USArrests), col = "red", cex = 0.8)

# Add title and axis labels

title("Biplot of PCA on USArrests Dataset")



**DIMENSIONALITY REDUCTION WITH T-SNE**

# Load necessary libraries

library(datasets) # For iris dataset

library(Rtsne) # For t-SNE implementation

library(ggplot2) # For plotting

# Step 1: Load the IRIS dataset

data(iris)

# Separate features and labels

iris\_data <- iris[, 1:4]

iris\_labels <- iris[, 5]

# Step 2: Scale the features (recommended for t-SNE)

iris\_data\_scaled <- scale(iris\_data)

# Step 3: Apply t-SNE transformation

tsne\_result <- Rtsne(iris\_data\_scaled, dims = 2, pca = TRUE, perplexity = 30, verbose = TRUE, max\_iter = 500)

# Step 4: Create a data frame combining t-SNE results and labels

tsne\_df <- data.frame(X1 = tsne\_result$Y[, 1], X2 = tsne\_result$Y[, 2], Species = iris\_labels)

# Step 5: Plot the t-SNE results using ggplot2

ggplot(tsne\_df, aes(x = X1, y = X2, color = Species)) +

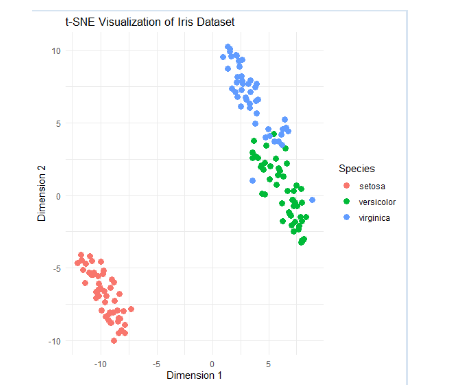
geom\_point(size = 3) +

labs(title = "t-SNE Visualization of IRIS Dataset",

x = "t-SNE Component 1", y = "t-SNE Component 2",

color = "Species") +

theme\_minimal()



**ASSOCIATION RULE MINING**

# Installing Packages

install.packages("arules")

install.packages("arulesViz")

# Loading package

library(arules)

library(arulesViz)

# Fitting model

# Training Apriori on the dataset

set.seed = 220 # Setting seed

associa\_rules = apriori(data = dataset, parameter = list(support = 0.004, confidence = 0.2))

# Plot

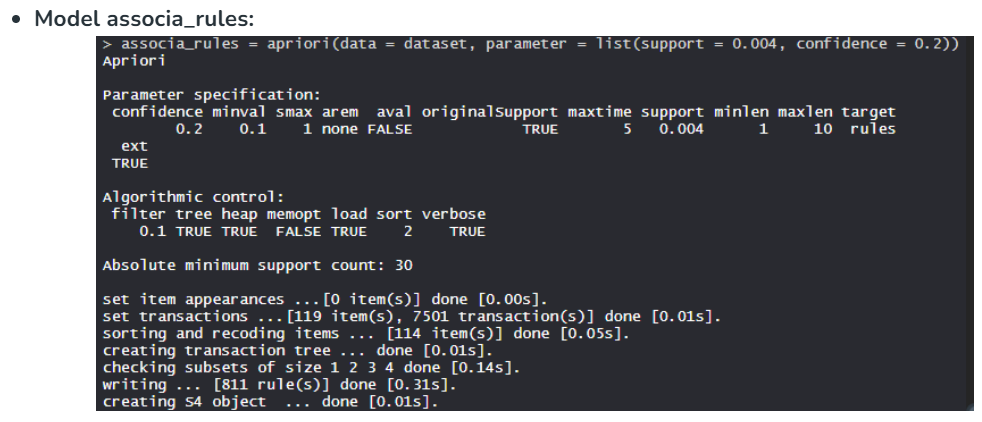
itemFrequencyPlot(dataset, topN = 10)

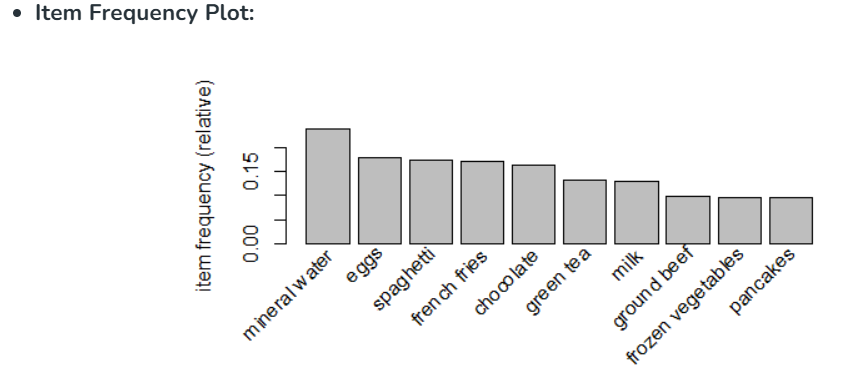
# Visualising the results

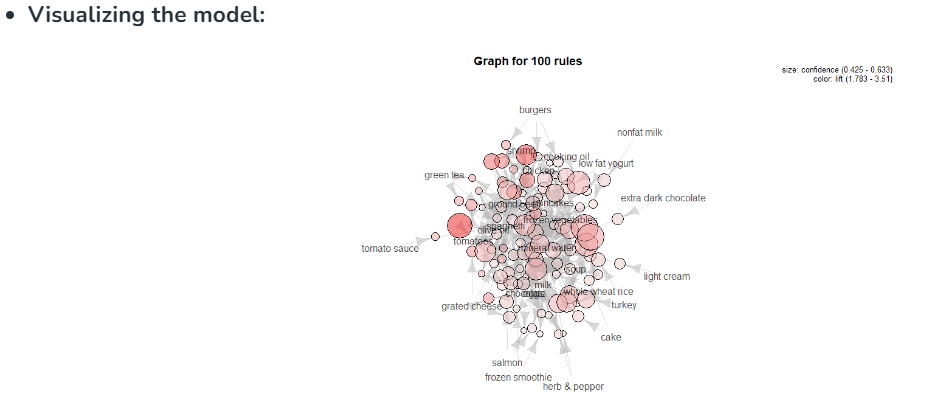
inspect(sort(associa\_rules, by = 'lift')[1:10])

plot(associa\_rules, method = "graph",

measure = "confidence", shading = "lift")







**FP GROWTH**

# Install and load necessary packages

install.packages("arules")

library(arules)

# Example transaction dataset (list format)

transactions <- list(

c("bread", "milk"),

c("bread", "diapers", "beer", "eggs"),

c("milk", "diapers", "beer", "cola"),

c("bread", "milk", "diapers", "beer"),

c("bread", "milk", "diapers", "cola")

)

# Convert transactions to transaction format for arules

trans <- as(transactions, "transactions")

# Mine frequent itemsets using FP-growth

frequent\_itemsets <- fpgrowth(trans, support = 0.4, verbose = TRUE)

# Display the frequent itemsets

inspect(frequent\_itemsets)

