**Ex 3**

AverageMapper.java

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*; import java.io.IOException;

public class AverageMapper extends Mapper <LongWritable, Text, Text, IntWritable>

{

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

{

String line = value.toString(); String year = line.substring(15,19); int temperature;

if (line.charAt(87)=='+')

temperature = Integer.parseInt(line.substring(88, 92));

else

temperature = Integer.parseInt(line.substring(87, 92));

String quality = line.substring(92, 93);

if(temperature != MISSING && quality.matches("[01459]")) context.write(new Text(year),new IntWritable(temperature));

}

}

AverageReducer.java

import org.apache.hadoop.mapreduce.\*;

import java.io.IOException;

public class AverageReducer extends Reducer <Text, IntWritable,Text, IntWritable >

{

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException

{

int max\_temp = 0; int count = 0;

for (IntWritable value : values)

{

max\_temp += value.get(); count+=1;

}

context.write(key, new IntWritable(max\_temp/count));

} }

AverageDriver.java

import org.apache.hadoop.io.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AverageDriver

{

public static void main (String[] args) throws Exception

{

if (args.length != 2)

{

System.err.println("Please Enter the input and output parameters"); System.exit(-1);

}

Job job = new Job(); job.setJarByClass(AverageDriver.class); job.setJobName("Max temperature");

FileInputFormat.addInputPath(job,new Path(args[0])); FileOutputFormat.setOutputPath(job,new Path (args[1]));

job.setMapperClass(AverageMapper.class); job.setReducerClass(AverageReducer.class);

job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);

System.exit(job.waitForCompletion(true)?0:1);

}

}

Ex 4a

X=c(151,174,138,186,128,136,179,163,152,131)

Y=c(63,81,56,91,47,57,76,72,62,48)

plot(X,Y)

relation=lm(Y~X)

print(relation)

print(summary(relation))

a=data.frame(X=170)

result=predict(relation,a)

print(result)

png(file="linearregression.png")

plot(Y,X,col="green",main="Height & Weight Regression",abline(lm(X~Y)),

cex=1.3,pch=16,Xlab="Weight in kg",Ylab="Height in cm")

dev.off()

ex 4b

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

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

print(summary(am.data))

ex 5a

# Load the required library

library(e1071)

# Create a sample dataset

data <- data.frame(

x1 = c(1, 1.5, 2, 3, 3.5, 4, 4.5, 5),

x2 = c(2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5),

class = c(0, 0, 0, 0, 1, 1, 1, 1)

)

# Train an SVM classifier

svm\_model <- svm(class ~ ., data = data, kernel = "linear", cost = 10)

# Summary of the SVM model

summary(svm\_model)

# Make predictions on new data

new\_data <- data.frame(

x1 = c(2.5, 3.5),

x2 = c(3.5, 4.5)

)

predictions <- predict(svm\_model, newdata = new\_data)

# Print the predictions

print(predictions)

Ex 5b

library(party)

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

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

output.tree <- ctree( nativeSpeaker ~ age + shoeSize + score, data = input.dat)

plot(output.tree)

dev.off()

Ex 6a- hierarchical clust

install.packages("factoextra")

install.packages("cluster")

install.packages("magrittr")

library("factoextra")

library("cluster")

library("magrittr")

res.hc <- USArrests %>%

scale() %>%

# Scale the data

dist(method = "euclidean") %>% # Compute dissimilarity matrix

hclust(method = "ward.D2") # Compute hierachical clustering

# Visualize using factoextra

# Cut in 4 groups and color by groups

fviz\_dend(res.hc, k = 4, # Cut in four groups

cex = 0.5, # label size

k\_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),

color\_labels\_by\_k = TRUE, # color labels by groups

rect = TRUE # Add rectangle around groups

)

Chatgpt

# Sample data

set.seed(123)

data <- matrix(rnorm(100), ncol = 2)

# Perform hierarchical clustering

hc <- hclust(dist(data))

# Plot dendrogram

plot(hc, main = "Hierarchical Clustering Dendrogram", xlab = "", sub = "")

# Cut the dendrogram to get clusters

clusters <- cutree(hc, k = 3)

print(clusters)

Ex 6b- partition clust

# Load required packages

library(cluster)

# Load your dataset, or use a built-in dataset for demonstration

data(USArrests)

# Standardize the data (optional but recommended)

scaled\_data <- scale(USArrests)

# Perform k-means clustering with, for example, 3 clusters

num\_clusters <- 3

kmeans\_result <- kmeans(scaled\_data, centers = num\_clusters)

# Extract k-means clustering results

cluster\_centers <- kmeans\_result$centers

cluster\_assignments <- kmeans\_result$cluster

# Visualize the clusters (optional)

library(factoextra) # Install if not already installed

fviz\_cluster(kmeans\_result, data = scaled\_data)

Ex 6c fuzzy clust

# Load required packages

install.packages("cluster") # Install if not already installed

library(cluster)

# Load your dataset, or use a built-in dataset for demonstration

data(USArrests)

# Standardize the data (optional but recommended)

scaled\_data <- scale(USArrests)

# Perform fuzzy clustering with, for example, 3 clusters

num\_clusters <- 3

fuzzy\_result <- fanny(scaled\_data, num\_clusters)

# Extract fuzzy clustering results

membership\_coefficients <- fuzzy\_result$membership

cluster\_centers <- fuzzy\_result$centers

# Visualize the fuzzy clusters (optional)

library(factoextra) # Install if not already installed

fviz\_cluster(fuzzy\_result, data = scaled\_data)

Ex 7a

Chatgpt

Dbscan

# Load required library

library(fpc)

# Generate sample data

set.seed(123)

data <- matrix(rnorm(200), ncol = 2)

# Perform DBSCAN clustering

dbscan\_result <- dbscan(data, eps = 0.5, MinPts = 5)

# Plot clusters

plot(dbscan\_result, data, main = "DBSCAN Clustering")

# Print cluster memberships

print(dbscan\_result$cluster)

record

install.packages("factoextra")

install.packages("magrittr")

install.packages("cluster")

library("factoextra")

library("magrittr")

library("cluster")

install.packages("fpc")

install.packages("dbscan")

install.packages("factoextra")

# Load the data

data("multishapes", package="factoextra")

df<-multishapes[, 1:2]

# Compute DBSCAN using fpc package

library("fpc")

set.seed(123)

db<-fpc::dbscan(df, eps=0.15, MinPts=5)

# Plot DBSCAN results

library("factoextra")

fviz\_cluster(db, data=df, stand=FALSE,

ellipse=FALSE,

show.clust.cent=FALSE,

geom="point",palette="jco", ggtheme=theme\_classic())

Ex 7b

Model based clust

# Load required packages

install.packages("factoextra") # Install if not already installed

library(factoextra)

# Load the diabetes dataset

data("diabetes")

# Perform model-based clustering (e.g., using Gaussian Mixture Models) on the diabetes dataset

model <- Mclust(diabetes)

# Visualize the clustering result

fviz\_cluster(model, data = diabetes, geom = "point", pointsize = 3)

ex 11 b

# Load required libraries

library(quantmod)

# Define the stock symbol and time period

stock\_symbol <- "AAPL"

start\_date <- "2020-01-01"

end\_date <- "2022-12-31"

# Download historical stock price data

getSymbols(stock\_symbol, from = start\_date, to = end\_date)

# Extract adjusted closing prices

stock\_data <- Ad(get(stock\_symbol))

# Create lagged variables for predictors (e.g., lagged closing prices)

lag1 <- Lag(stock\_data, k = 1)

lag2 <- Lag(stock\_data, k = 2)

lag3 <- Lag(stock\_data, k = 3)

# Combine lagged variables into a data frame

predictors <- data.frame(stock\_data, lag1, lag2, lag3)

predictors <- na.omit(predictors)

# Split the data into training and testing sets

train\_size <- 0.8

train\_index <- 1:round(train\_size \* nrow(predictors))

train\_data <- predictors[train\_index, ]

test\_data <- predictors[-train\_index, ]

# Fit a linear regression model

model <- lm(stock\_data ~ lag1 + lag2 + lag3, data = train\_data)

# Make predictions on the test data

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

# Evaluate the model (e.g., calculate RMSE)

RMSE <- sqrt(mean((predictions - test\_data$stock\_data)^2))

print(paste("Root Mean Squared Error (RMSE):", RMSE))