DS Praccs Code Explanation

Practical 1 : EDA

eda_data = read.csv(file.choose(), header = TRUE, sep = ",") → Reads data from csv file. choose() → Opens window to choose file; header = TRUE → First row of file contains column names ; sep = "," → separator used in column

eda_data ⇒ Reads entire dataset from entered csv file.

head(eda_data) ⇒ Shows first six rows

summary(eda_data) ⇒ Shows mean, min, max, etc of each column.

str(eda_data) ⇒ Shows data type of each column.

eda_data[1:8] ⇒ Shows first eight rows

head(eda_data, 3) ⇒ Shows first three rows

head(eda_data, 8) ⇒ Shows first eight rows

tail(eda_data, 8) ⇒ Shows last eight rows.

eda_data[, 1:5] ⇒ Shows first 5 columns of dataset

newdata1 = subset(eda_data, eda_data$Education == "Grad") ⇒ Creates new dataset, where Education column is equal to "Grad".

newdata1 ⇒ Displays the above data set

newdata2 = subset(eda_data, eda_data$Age == "51" & eda_data$Gender == "M") ⇒ Creates dataset, where Age column is "51" & Gender column is "M".

newdata2 ⇒ Displays the above data set

a = eda_data[order(eda_data$Education),] ⇒ Sorts Education column of eda_data in ascending order & store it in "a".

a ⇒ Displays the ascending values of Education Column

a = eda_data[order(eda_data$Education, decreasing = TRUE),] ⇒ Sorts Education column of eda_data in descending order & store it in "a".

a ⇒ Displays the descending values of Education Column

a = colSums(is.na(eda_data)) ⇒ Calculates number of missing values in each column of eda_data & stores it in "a".

a ⇒ Shows number of missing values.

hist(eda_data$Age) ⇒ Histogram of Age Column.

boxplot(eda_data$Age) ⇒ Boxplot of Age Column

mean(eda_data$Age) ⇒ Mean of Age Column

min(eda_data$Age) ⇒ Minimum Value of Age Column

max(eda_data$Age) ⇒ Maximum Value of Age Column

median(eda_data$Age) ⇒ Median of Age column

counts = table(eda_data$Education, eda_data$Gender) ⇒ Generates a table that counts number of occurrences of each combination of Education & Gender

counts ⇒ Displays the table

help("pie") ⇒ Provides information about "pie" function

plot(eda_dataAge, eda_dataSalary) ⇒ Creates Scatter Plot of Age and Salary Column

hist(eda_data$Age, col="green", border="red", xlim=c(20,70), ylim=c(0,70)) ⇒ Creates Histogram of Age Column, with green bars & red border, with x-axis values from 20 to 70 & y-axis values from 0 to 70.

plot(data.data$Salary, type="p", col="red", main="Salary", xlim=c(0,20), ylim=c(0,80)) ⇒ Creates Point Plot of Salary Column, with red points & title Salary x-axis range → 0 to 20 y-axis range → 0 to 80

barplot(Counts, main="Data Distribution by Education vs Gender", ⇒ Creates a Bar Plot
col=c("blue","red"), legend.text=TRUE, xlim=c(0,100))

x=table(data.data$Education) ⇒ Creates Table of Education Levels Counts

per=round(x/sum(x)*100,1) ⇒ Calculates Percentage of Education Level

pie(x, labels=per, main="City Pie Chart", col=c("blue","red")) ⇒ Creates Pie Chart of Education Level.

Practical 2 :- Hypothesis

* 2.1 :- Independent 't' test :-

data1 = read.csv(file.choose(), sep = ",", header = T) ⇒ Reads csv file.

t.test(data1$UK, data1$Germany, alternative = "two.sided", var.equal = FALSE) ⇒ Performs two-tailed t test; alternative = "two.sided" ⇒ Alternative Hypothesis; var.equal = FALSE ⇒ variance of 2 groups are not assumed to be equal.

* 2.2 :- Paired 't' test :-

data2 = read.csv(file.choose(), sep = ",", header = T) ⇒ Reads csv file

t.test(data2$before, data2$after, alternative = "greater", paired = T) ⇒ Performs Paired t test with $H_0: \mu_1 - \mu_2 \leq 0$; $H_a: \mu_1 - \mu_2 > 0$

t.test(data2$before, data2$after, alternative = "less", paired = T) ⇒ Performs Paired t test with $H_0: \mu_1 - \mu_2 \geq 0$; $H_a: \mu_1 - \mu_2 < 0$

t.test(data2$before, data2$after, alternative = "two.sided", paired = T) ⇒ Performs Paired t test with $H_0: \mu_1 - \mu_2 = 0$; $H_a: \mu_1 - \mu_2 \neq 0$

* 2.3 :- Test for Correlation :-

data3 = read.csv(file.choose(), sep = ",", header = T) ⇒ Reads csv file

cor.test(data3$mpg, data3$wt, alternative = "two.sided", method = "pearson") ⇒ Perform Pearson Correlation Test.

* 2.4 :- One Sample 't' test :-

data4 = read.csv(file.choose(), sep = ",", header = T) ⇒ Reads csv file

t.test(data4$IQ, alternative = "greater", mu = 100) ⇒ Performs One Sample t test with alternative hypothesis (H_a) :- True mean is greater than H_0 ($\mu = 100$)

Practical 4:- ANOVA

* Ftest :-

ftest = read.csv(file.choose(), sep=",", header=T) ⇒ Reads csv file

var.test(ftest$Gr1, ftest$Gr2, alternative="two.sided") ⇒ Performs ftest

* One Way ANOVA :-

data1 = read.csv(file.choose(), sep=",", header=T) ⇒ Reads csv file

names(data1) ⇒ Returns names of columns in data1.

summary(data1) ⇒ Gives Max, min, mean, etc of each column in data1.

head(data1) ⇒ Displays first 6 rows of data1.

anv = aov(formula = satindex ~ dept, data = data1) ⇒ Performs ANOVA, with satindex as response variable & dept as predictor variable.

summary(anv) ⇒ Displays summary of ANOVA results.

* Two Way ANOVA :-

data2 = read.csv(file.choose(), sep=",", header=T) ⇒ Reads csv file

names(data2) ⇒ Returns names of columns in data2.

summary(data2) ⇒ Gives Max, min, mean, etc of each column in data2.

head(data2) ⇒ Displays first 6 rows of data2.

anv = aov(formula = satindex ~ dept + exp + dept * exp, data = data2) ⇒ Performs ANOVA, dependent variable → satindex ; Independent variable → Dept. & exp.

summary(anv) ⇒ Displays summary of ANOVA results.

Practical 5:- Time Series Analysis

AirPassengers ⇒ Loads the AirPassengers dataset
view(AirPassengers) ⇒ Opens new window to view dataset.
class(AirPassengers) ⇒ Displays class of dataset, which "ts" for Time Series
str(AirPassengers) ⇒ Displays structure of dataset i.e Dimension & Attribute
start(AirPassengers) ⇒ Displays start date of Time Series
end(AirPassengers) ⇒ Displays end date of Time Series
frequency(AirPassengers) ⇒ Displays no. of observation per unit time
summary(AirPassengers) ⇒ Displays summary of dataset.

plot(AirPassengers) ⇒ Creates plot of Time Series Data
abline(reg = lm(AirPassengers ~ time(AirPassengers))) ⇒ Adds regression line to plot.

cycle(AirPassengers) ⇒ Calculates position in cycle of each observation
plot(aggregate(AirPassengers, FUN = mean)) ⇒ Creates Plot of mean values of time series data
boxplot(AirPassenger ~ cycle(AirPassengers)) ⇒ Creates BoxPlot of time series data, grouped by cycle.

acf(AirPassengers)
acf(log(AirPassengers))
acf(diff(log(AirPassengers)))
acf(diff(AirPassengers))
] Creates Autocorrelation Plot, by applying log & difference in respective methods

pacf(diff(log(AirPassengers)))
plot(diff(log(AirPassengers)))
] Creates Partial Autocorrelation Plot, by applying log & difference & plot it.

fit <- arima(log(AirPassengers), c(0,1,1), seasonal = list(order = c(0,1,1), period = 12)) ⇒ Fits an ARIMA model, with first order difference & seasonal difference of order 1, log is used to make variance stationary
pred = predict(fit, n.ahead = 10*12) ⇒ Predict next 10 years i.e 120 months.
pred1 = round(2.718^pred$pred, 0) ⇒ Rounds predicted values to nearest integers
pred1 ⇒ Displays Predicted values
ts.plot(AirPassengers, pred1, log = "y", lty = c(1,3)) ⇒ Creates plot of Original Time Series data & Predicted values

PAGE No.
DATE / /

Practical 8:- Decision Tree

titanic = read.csv(file.choose(), header = TRUE) ⇒ Reads CSV file

library(rpart) ⇒ loads rpart package used for building decision tree

fit = rpart(Survived ~ Pclass + Gender + Age + SibSp + Parch + Fare + Embarked, data = titanic, method = "class") ⇒ Creates a Decision Tree Model

plot(fit) ⇒ Plots the Created Decision Tree

text(fit) ⇒ Adds text to the plot.

install.packages("Rattle") ⇒ This package provides interface for working with decision trees

install.packages("rpart.plot") ⇒ -11- provides additional plotting capabilities for decision tree.

install.packages("RColorBrewer") ⇒ -11- provides Color Palettes for Data visualization.

library(rattle)
library(rpart.plot)
library(RColorBrewer)
} Loads the respective installed packages.

fancyRpartPlot(fit) ⇒ Creates fancier Plot using "rpart.plot" package.

Prediction = predict(fit, titanic, type = "class") ⇒ Predicts the Survival Status of Titanic passengers, type = "class" predicts class labels instead of probabilities.

Prediction ⇒ Displays the Prediction.

PAGE No.
DATE / /

Practical 9:- Clustering

iris ⇒ Loads the pre-installed iris dataset.

"I think data(iris) is an unnecessary line".

names(iris) ⇒ Displays names of columns in iris dataset.

new_data = subset(iris, select = c(-Species)) ⇒ Creates new dataset containing all columns of iris except Species

new_data ⇒ Displays the dataset of new_data.

cl = kmeans(new_data, 3) ⇒ Performs K-means clustering on new_data with K=3

cl ⇒ Displays the result of K-means clustering.

data = new_data ⇒ Storing new_data into data.

wss = sapply(1:15, function(k){kmeans(data, k)$tot.withinss}) ⇒ Calculates Total within cluster Sum of Squares for "K" values ranging from 1 to 15 using "kmeans()" function

plot(1:15, wss, type = "b", pch=19, frame=FALSE, xlab="Number of Clusters K", ylab="Total Within Clusters Sum of Squares") ⇒ Creates a plot of total within clusters Sum of squares v/s no. of clusters "K".

install.packages("cluster") ⇒ Install "cluster" package for cluster analysis

library(cluster) ⇒ Loads the "cluster" package

clusplot(new_data, cl$cluster, color=TRUE, shade=TRUE, labels=2, lines=0) ⇒ Creates Scatterplot of data points colored by clusters assigned by K-means algorithm from plot

cl$cluster ⇒ Displays cluster assignment for each data point. Removes default line

cl$centers ⇒ Displays coordinates of centroid of each cluster.

"Agglomerative Clustering"

clusters = hclust(dist(iris[, 3:4])) ⇒ Performs Hierarchical Clustering using only "Petal.Length" & "Petal.Width".

plot(clusters) ⇒ Creates dendogram plot of Hierarchical Clustering

clusterCut = cutree(clusters, 3) ⇒ Cuts Hierarchical Clustering Tree into 3 clusters

table(clusterCut, iris$Species) ⇒ Creates contingency table of Species.

Practical 10 :- PCA

data_iris = iris[1:4] ⇒ Assigns first 4 columns of iris dataset to data_iris variable

data_iris ⇒ Displays the assigned first 4 columns

Cov_data = cov(data_iris) ⇒ Calculates covariance matrix of data_iris

Cov_data ⇒ Displays the covariance matrix.

Eigen_data = eigen(Cov_data) ⇒ Calculates Eigenvectors & Eigenvalues of Cov_data

Eigen_data ⇒ Display the Eigenvectors & Eigenvalues

PCA_data = princomp(data_iris, cor = "False") ⇒ Performs PCA using princomp

PCA_data ⇒ Displays the result of PCA.

Eigen_data $values ⇒ Accesses Eigenvalues of Covariance Matrix.

PCA_data $sdev^2 ⇒ Accesses variance of each principal component obtained from PCA.

PCA_data $loadings[,1:4] ⇒ Accesses the loadings (i.e eigenvectors) of first 4 principal components

Eigen_data $vectors ⇒ Accesses eigenvectors of covariance matrix.

summary(PCA_data) ⇒ Provides summary of PCA results

biplot(PCA_data) ⇒ Creates biplot showing loadings & scores of first two principal components

screeplot(PCA_data, type = "lines") ⇒ Creates screeplot that shows variance explained by each principal component.

model2 = PCA_data $loadings[,1] ⇒ Select the principal component for second Model.

model2

model2_scores = as.matrix(data_iris) %*% model2 ⇒ Calculates scores of second model by multiplying data by first principal component loading

model2_scores

library(class)
install.packages("e1071")
library(e1071)

} libraries for naiveBayes model.

mod1 = naiveBayes(iris[, 1:4], iris[, 5]) ⇒ Fits a Naive Bayes model using first 4 columns of iris dataset as predictors & fifth column of iris dataset as response variable and stores in mod1

mod2 = naiveBayes(model2_scores, iris[, 5]) ⇒ Fits a Naive Bayes model using scores of second model as predictor & fifth column of iris dataset as response variable and stores in mod2

table(predict(mod1, iris[, 1:4]), iris[, 5]) ⇒ Calculates accuracy of first model.

table(predict(mod2, model2_scores), iris[, 5]) ⇒ Calculates accuracy of second model.

round(cor(PCA_data, Score)) ⇒ Calculates correlation between scores of principal components