Practical 3: Perform the data classification using classification algorithm using R/Python.

```
rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
# Convert it to a time series object.
rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)
print(rainfall.timeseries)
png(file = "rainfall.png")
plot(rainfall.timeseries)
dev.off()
```

Practical 4: Perform the data clustering using clustering algorithm using R/Python.

```
newiris <- iris
newiris$Species <- NULL
(kc <- kmeans(newiris,3))
table(iris$Species,kc$cluster)
plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)
```

Practical 5: Perform the Linear regression on the given data warehouse data using R/Python.

5ab:

```
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)
print(summary(relation))
a <- data.frame(x=170)
result <- predict(relation,a)
```

```
print(result)
print(result)
```

5c:

```
# Create the predictor (x) and response (y) variables
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)  # Height in cm
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
# Weight in Kg
# Save the chart to a file
png(file = "linearregression.png")
# Plot the data points
plot(x, y,
    col = "blue",
    main = "Height & Weight Regression",
    xlab = "Height in cm",
    ylab = "Weight in Kg",
    pch = 16,
    cex = 1.3)
# Add the regression line
abline(lm(y ~ x), col = "red")
# Close the graphic device
dev.off()
```

Practical 6: Perform the logistic regression on the given data warehouse data using R/Python.

```
# Load necessary dataset
library(datasets)
ir_data <- iris
# View first few rows of the dataset head(ir_data)
# Structure of the dataset
```

```
str(ir_data)
# Check levels of species column
levels(ir_data$Species)
# Check for missing values
sum(is.na(ir_data))
# Select first 100 rows
ir_data <- ir_data[1:100,]
# Set seed for reproducibility
set.seed(100)
# Randomly sample 80 indices from 1 to 100
samp <- sample(1:100, 80)
# Create training (test) and control datasets
ir_test <- ir_data[samp,]
ir_ctrl <- ir_data[-samp,]
# Install and load ggplot2 package
install.packages("ggplot2")
library(ggplot2)
# Install and load GGally package
install.packages("GGally")
library(GGally)
# Create pair plot of test dataset ggpairs(ir_test)
# Define dependent and independent variables
y <- ir_test$Species
x <- ir_test$Sepal.Length
# Fit logistic regression model
glfit <- glm(y ~ x, family = 'binomial')
# Display model summary
summary(glfit)
# Create new data frame for prediction
newdata <- data.frame(x = ir_ctrl$Sepal.Length)
```

```
# Predict values using logistic regression model
predicted_val <- predict(glfit, newdata, type = "response")
# Create a data frame with actual and predicted values
prediction <- data.frame(ir_ctrl$Sepal.Length, ir_ctrl$Species, predicted_val)
# Visualize the predictions
qplot(prediction[,1], round(prediction[,3]), col = prediction[,2],
 xlab = 'Sepal Length', ylab = 'Prediction using Logistic Regression')
```

Practical 7: Write a Python program to read data from a CSV file, perform simple data analysis, and generate basic insights. (Use Pandas is a Python library).

```
import pandas as pd

# Loading data into a DataFrame
data_frame = pd.read_csv('customers.csv')

# Displaying first five rows
print(data_frame.head())

# Displaying last five rows
print(data_frame.tail())

# Display column names
print(list(data_frame.columns))

# Display DataFrame info
data_frame.info()

# Display statistical summary
print(data_frame.describe())
```

```
# Removing missing values (if any)
data_frame = data_frame.dropna()

# Dropping the row at index 4 (5th row) and displaying first few rows
print(data_frame.drop(4).head())

# Renaming columns (Incorrect syntax in original code - needs `inplace=True` or assignment)
data_frame.rename(columns={0: "First", 1: "Second"}, inplace=True)

# Adding a new column with a constant value
data_frame['NewColumn'] = 1
print(data_frame.head())

# Sorting values by Age in descending order
print(data_frame.sort_values(by='Age', ascending=False).head())

# Sorting values by Age and Annual Income (k$)
print(data_frame.sort_values(by=['Age', 'Annual Income (k$)']).head(10))

# Creating first DataFrame
df1 = pd.DataFrame({
   'Name': ['Jeevan', 'Raavan', 'Geeta', 'Bheem'],
   'Age': [25, 24, 52, 40],
   'Qualification': ['MSc', 'MA', 'MCA', 'PhD']
})
print(df1)

# Creating second DataFrame
df2 = pd.DataFrame({
   'Name': ['Jeevan', 'Raavan', 'Geeta', 'Bheem'],
```

```
    'Salary': [100000, 50000, 20000, 40000]
})
print(df2)

# Merging two DataFrames
df = pd.merge(df1, df2, on="Name")  # Ensure merge is done on a common column
print(df)

# Function to classify customer satisfaction based on spending score
def fun(value):
    if value > 70:
        return "Yes"
    else:
        return "No"

# Applying the function to create a new column
data_frame['Customer Satisfaction'] = data_frame['Spending Score (1-100)'].apply(fun)

# Display first 10 rows with the new column
print(data_frame.head(10))
const = data_frame['Age'].max()
data_frame['Age'] = data_frame['Age'].apply(lambda x: x/const)
data_frame.head()
data_frame.plot(x ='CustomerID', y='Spending Score (1-100)',kind = 'scatter')
```

Practical 8A: Perform data visualization Perform data visualization using Python on any sales data.

```
import pandas as pd

from matplotlib import pyplot as plt

plt.rcParams["figure.figsize"] = [8.00, 4.50]

plt.rcParams["figure.autolayout"] = True

columns = ["Item_Name", "Qty_sold"]

df = pd.read_csv("Sales.csv",usecols=columns)

print("Contents in csv file:", df)

plt.plot(df.Item_Name, df.Qty_sold)

plt.show()
```

8b Power BI link:

https://services.odata.org/V3/Northwind/Northwind.svc/

Sales table :

Item_Name	Qty_sold
Mouse	98
Keyboard	75
Pendrive	50
Speaker	92
Bluetooth	90

Customers Table:

CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
1	Male	25	45	72
2	Female	34	78	88
3	Male	42	55	65
4	Female	23	68	92
5	Male	36	47	75
6	Female	29	89	54
7	Male	51	102	60
8	Female	30	45	81

9	Male	22	56	78
10	Female	41	99	50
11	Male	37	120	40
12	Female	26	80	95
13	Male	50	105	67
14	Female	33	70	86
15	Male	40	60	58
16	Female	27	88	77
17	Male	55	130	45
18	Female	31	52	83
19	Male	48	75	66
20	Female	29	85	90
21	Male	60	110	55
22	Female	28	95	98
23	Male	35	40	70
24	Female	32	78	88
25	Male	45	65	79