**Slip 1:**

**Q1: R program to add, multiply, and divide two vectors.**

r

Copy code

# Vectors

vector1 <- c(1, 2, 3, 4)

vector2 <- c(5, 6, 7, 8)

# Operations

addition <- vector1 + vector2

multiplication <- vector1 \* vector2

division <- vector1 / vector2

# Display results

cat("Addition:", addition, "\n")

cat("Multiplication:", multiplication, "\n")

cat("Division:", division, "\n")

**Q2: Python program for Simple Linear Regression.**

python

Copy code

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Data (x and y can be downloaded)

x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)

y = np.array([7, 14, 15, 18, 19, 21, 26, 23])

# Model

model = LinearRegression()

model.fit(x, y)

y\_pred = model.predict(x)

# Errors

mae = mean\_absolute\_error(y, y\_pred)

mse = mean\_squared\_error(y, y\_pred)

rmse = np.sqrt(mse)

print(f"MAE: {mae}, MSE: {mse}, RMSE: {rmse}")

**Slip 2:**

**Q1: R program to calculate multiplication table using a function.**

r

Copy code

multiplication\_table <- function(num) {

for (i in 1:10) {

cat(num, "\*", i, "=", num \* i, "\n")

}

}

num <- as.integer(readline(prompt="Enter a number: "))

multiplication\_table(num)

**Q2: Python program for K-Means Algorithm.**

python

Copy code

from sklearn.cluster import KMeans

import numpy as np

# Synthetic dataset

data = np.random.rand(100, 2)

# K-means clustering

kmeans = KMeans(n\_clusters=3)

kmeans.fit(data)

print("Cluster centers:", kmeans.cluster\_centers\_)

print("Labels:", kmeans.labels\_)

**Slip 3:**

**Q1: R program to reverse a number and calculate sum of digits.**

r

Copy code

reverse\_number <- function(num) {

rev\_num <- as.numeric(paste(rev(strsplit(as.character(num), "")[[1]]), collapse=""))

cat("Reversed Number:", rev\_num, "\n")

cat("Sum of Digits:", sum(as.numeric(strsplit(as.character(num), "")[[1]])), "\n")

}

num <- as.integer(readline(prompt="Enter a number: "))

reverse\_number(num)

**Q2: Python program for Simple Linear Regression with numpy.**

python

Copy code

import numpy as np

# Data

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12, 16, 18])

# Simple Linear Regression

n = len(x)

mean\_x, mean\_y = np.mean(x), np.mean(y)

b1 = sum((x - mean\_x) \* (y - mean\_y)) / sum((x - mean\_x)\*\*2)

b0 = mean\_y - b1 \* mean\_x

print(f"Coefficients: b0 = {b0}, b1 = {b1}")

**Slip 4:**

**Q1: R program to calculate sum of two matrices.**

r

Copy code

# Matrices

mat1 <- matrix(c(1, 2, 3, 4), nrow=2, ncol=2)

mat2 <- matrix(c(5, 6, 7, 8), nrow=2, ncol=2)

# Sum of Matrices

sum\_matrix <- mat1 + mat2

print("Sum of matrices:")

print(sum\_matrix)

**Q2: Python program for Naïve Bayes algorithm.**

python

Copy code

import numpy as np

from sklearn.naive\_bayes import CategoricalNB

# Dataset

weather = ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast',

'Sunny','Sunny','Rainy','Sunny','Overcast','Overcast','Rainy']

temp = ['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool',

'Mild','Mild','Mild','Hot','Mild']

play = ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

# Encoding categorical data

from sklearn.preprocessing import LabelEncoder

weather\_encoder, temp\_encoder, play\_encoder = LabelEncoder(), LabelEncoder(), LabelEncoder()

weather\_encoded, temp\_encoded, play\_encoded = weather\_encoder.fit\_transform(weather), temp\_encoder.fit\_transform(temp), play\_encoder.fit\_transform(play)

# Combining Features

features = np.array(list(zip(weather\_encoded, temp\_encoded)))

# Naive Bayes

model = CategoricalNB()

model.fit(features, play\_encoded)

# Prediction

pred = model.predict([[0, 2]]) # Overcast, Mild

print("Prediction:", play\_encoder.inverse\_transform(pred))

**Slip 5:**

**Q1: R program to concatenate two factors.**

r

Copy code

factor1 <- factor(c("apple", "banana", "cherry"))

factor2 <- factor(c("mango", "banana", "grape"))

# Concatenate

concatenated <- c(factor1, factor2)

print(concatenated)

**Q2: Python program for Decision Tree Classifier (Diabetes Dataset).**

python

Copy code

from sklearn.datasets import load\_diabetes

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Dataset (Load diabetes dataset)

data = load\_diabetes()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.data, data.target > data.target.mean(), test\_size=0.3)

# Decision Tree

tree = DecisionTreeClassifier()

tree.fit(X\_train, y\_train)

y\_pred = tree.predict(X\_test)

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred)}")

**Slip 6:**

**Q1: R program to create data frame and display duplicate elements.**

r

Copy code

# Vectors

v1 <- c(1, 2, 3, 4, 5, 5)

v2 <- c(5, 6, 7, 8, 9, 10)

# Data Frame

df <- data.frame(v1, v2)

# Display duplicates

duplicates <- df[duplicated(df$v1), ]

print(duplicates)

**Q2: Python program for Agglomerative Clustering.**

python

Copy code

from sklearn.cluster import AgglomerativeClustering

import pandas as pd

# Dataset

data = pd.read\_csv('Customer.csv')

# Agglomerative Clustering

agglo = AgglomerativeClustering(n\_clusters=3)

agglo.fit(data[['Annual Income (k$)', 'Spending Score (1-100)']])

print("Cluster Labels:", agglo.labels\_)

**Slip 7**

**Q1. R Program: Sequence of numbers from 20 to 50, mean of numbers from 20 to 60, and sum of numbers from 51 to 91.**

r

Copy code

# Creating a sequence from 20 to 50

sequence <- 20:50

print(sequence)

# Mean of numbers from 20 to 60

mean\_value <- mean(20:60)

print(mean\_value)

# Sum of numbers from 51 to 91

sum\_value <- sum(51:91)

print(sum\_value)

**Q2. Python Program: Simple Linear Regression using sklearn.**

python

Copy code

import numpy as np

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

# Data

x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape((-1, 1))

y = np.array([7, 14, 15, 18, 19, 21, 26, 23])

# Linear Regression model

model = LinearRegression()

model.fit(x, y)

# Coefficients

b0 = model.intercept\_

b1 = model.coef\_[0]

print(f"Intercept (b0): {b0}, Coefficient (b1): {b1}")

# Plot

plt.scatter(x, y, color="blue")

plt.plot(x, model.predict(x), color="red")

plt.show()

**Slip 8**

**Q1. R Program: First 10 Fibonacci Numbers.**

r

Copy code

fibonacci <- numeric(10)

fibonacci[1] <- 0

fibonacci[2] <- 1

for (i in 3:10) {

fibonacci[i] <- fibonacci[i - 1] + fibonacci[i - 2]

}

print(fibonacci)

**Q2. Python Program: K-means Algorithm for Credit Card Dataset.**

python

Copy code

import pandas as pd

from sklearn.cluster import KMeans

# Load dataset (Assuming dataset is loaded as 'data')

data = pd.read\_csv('CC GENERAL.csv')

# Select features and apply KMeans

kmeans = KMeans(n\_clusters=3)

kmeans.fit(data)

# Predicted clusters

data['Cluster'] = kmeans.labels\_

print(data.head())

**Slip 9**

**Q1. R Program: Create Dataframe with 5 Employees and display summary.**

r

Copy code

empno <- c(101, 102, 103, 104, 105)

empname <- c("John", "Alex", "Emily", "David", "Sophia")

gender <- c("Male", "Male", "Female", "Male", "Female")

age <- c(25, 30, 28, 35, 22)

designation <- c("Manager", "Developer", "Analyst", "HR", "Intern")

employee\_data <- data.frame(empno, empname, gender, age, designation)

print(employee\_data)

summary(employee\_data)

**Q2. Python Program: SVM for Cancer Dataset with precision and recall.**

python

Copy code

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import classification\_report

# Load Cancer dataset

cancer = datasets.load\_breast\_cancer()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cancer.data, cancer.target, test\_size=0.3, random\_state=42)

# SVM model

model = SVC(kernel='linear')

model.fit(X\_train, y\_train)

# Predictions

y\_pred = model.predict(X\_test)

# Evaluate model

print(classification\_report(y\_test, y\_pred))

**Slip 10**

**Q1. R Program: Find Maximum and Minimum of a Vector.**

r

Copy code

vector <- c(1, 2, 8, 3, 5, 9)

max\_value <- max(vector)

min\_value <- min(vector)

print(paste("Maximum:", max\_value))

print(paste("Minimum:", min\_value))

**Q2. Python Program: Apply Apriori Algorithm on Iris Dataset.**

python

Copy code

from mlxtend.frequent\_patterns import apriori, association\_rules

import pandas as pd

# Load Iris dataset

data = pd.read\_csv('Iris.csv')

# Apply Apriori algorithm (Dummy transformation assumed for Iris data)

# Perform apriori and generate association rules.

frequent\_itemsets = apriori(data, min\_support=0.5, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.7)

print(rules)

**Slip 11**

**Q1. R Program: List elements not in another list.**

r

Copy code

list1 <- list("x", "y", "z")

list2 <- list("X", "Y", "Z", "x", "y", "z")

# Finding elements not in list2

not\_in\_list2 <- setdiff(list1, list2)

print(not\_in\_list2)

**Q2. Python Program: Hierarchical Clustering on Wholesale Customers Data.**

python

Copy code

import pandas as pd

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

# Load dataset

data = pd.read\_csv('Wholesale\_customers\_data.csv')

# Apply hierarchical clustering

Z = linkage(data, method='ward')

dendrogram(Z)

plt.show()

**Slip 12**

**Q1. R Program: Create a Dataframe with 5 employees and display the details.**

r

Copy code

# Create a dataframe

empno <- c(101, 102, 103, 104, 105)

empname <- c("John", "Alex", "Emily", "Sophia", "David")

gender <- c("Male", "Male", "Female", "Female", "Male")

age <- c(25, 30, 28, 22, 35)

designation <- c("Manager", "Developer", "Analyst", "Intern", "HR")

# Create dataframe

employee\_df <- data.frame(empno, empname, gender, age, designation)

# Display the details

print(employee\_df)

**Q2. Python Program: Implement Multiple Linear Regression Model for a car dataset.**

python

Copy code

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

# Load dataset (car dataset is hypothetical here)

data = pd.read\_csv('car\_data.csv') # assuming the dataset is available

X = data[['horsepower', 'engine\_size', 'weight']] # Features

y = data['price'] # Target variable

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Coefficients

print('Intercept:', model.intercept\_)

print('Coefficients:', model.coef\_)

**Slip 13**

**Q1. R Program: Draw a pie chart for the data distribution of a dice.**

r

Copy code

# Data distribution

slices <- c(7, 2, 6, 3, 4, 8)

labels <- c("1", "2", "3", "4", "5", "6")

# Draw the pie chart

pie(slices, labels = labels, main = "Dice Frequency Distribution", col=rainbow(length(slices)))

**Q2. Python Program: Display the shape and top rows of the StudentsPerformance.csv dataset.**

python

Copy code

import pandas as pd

# Load dataset

data = pd.read\_csv('StudentsPerformance.csv')

# Display the shape

print("Shape of the dataset:", data.shape)

# Display the top rows

print("Top rows of the dataset:")

print(data.head())

**Slip 14**

**Q1. R Script: Create a list of employees and perform operations.**

r

Copy code

# Create list

employee\_list <- list("John", "Alex", "Emily")

# a. Display the names

print("Names of employees:")

print(employee\_list)

# b. Add an employee

employee\_list <- append(employee\_list, "Sophia")

print("After adding a new employee:")

print(employee\_list)

# c. Remove the third element

employee\_list <- employee\_list[-3]

print("After removing the third employee:")

print(employee\_list)

**Q2. Python Program: Apply Apriori algorithm on Groceries dataset.**

python

Copy code

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

# Load groceries dataset

data = pd.read\_csv('Groceries\_dataset.csv')

# Apply Apriori

frequent\_itemsets = apriori(data, min\_support=0.1, use\_colnames=True)

# Generate rules

rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.5)

# Display support and confidence

print(rules[['antecedents', 'consequents', 'support', 'confidence']])

**Slip 15**

**Q1. R Program: Add, multiply and divide two vectors.**

r

Copy code

# Create two vectors

v1 <- c(4, 8, 12, 16)

v2 <- c(2, 4, 6, 8)

# Addition, multiplication, division

add\_result <- v1 + v2

multiply\_result <- v1 \* v2

divide\_result <- v1 / v2

# Display results

print("Addition:", add\_result)

print("Multiplication:", multiply\_result)

print("Division:", divide\_result)

**Q2. Python Program: Decision Tree Classifier for shows.csv dataset.**

python

Copy code

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

# Create a DataFrame (for illustration)

data = pd.DataFrame({

'Age': [30, 40, 50, 60],

'Experience': [5, 10, 15, 20],

'Rank': [7, 8, 6, 9],

'Class': ['No', 'Yes', 'No', 'Yes']

})

# Features and target

X = data[['Age', 'Experience', 'Rank']]

y = data['Class']

# Build the Decision Tree model

clf = DecisionTreeClassifier()

clf.fit(X, y)

# Predict class for a 40 years old American comedian with 10 years of experience and a ranking of 7

prediction = clf.predict([[40, 10, 7]])

print("Predicted class:", prediction)

**Slip 16**

**Q1. R Program: Create a bar plot for given data.**

r

Copy code

# Data

year <- c(2001, 2002, 2003)

export <- c(26, 32, 35)

import <- c(35, 40, 50)

# Create bar plot

barplot(rbind(export, import), beside=TRUE, names.arg=year, col=c("blue", "red"),

legend = c("Export", "Import"), main="Export-Import Data")

**Q2. Python Program: Decision Tree Classifier for Diabetes dataset.**

python

Copy code

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load dataset (from kaggle)

data = pd.read\_csv('diabetes.csv')

# Features and target

X = data.drop('Outcome', axis=1)

y = data['Outcome']

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Train the model

clf = DecisionTreeClassifier()

clf.fit(X\_train, y\_train)

# Predictions

y\_pred = clf.predict(X\_test)

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

**Slip 17**

**Q1. R Program: Get the first 20 Fibonacci numbers.**

r

Copy code

# Function to generate Fibonacci sequence

fibonacci <- numeric(20)

fibonacci[1] <- 0

fibonacci[2] <- 1

for (i in 3:20) {

fibonacci[i] <- fibonacci[i - 1] + fibonacci[i - 2]

}

print(fibonacci)

**Q2. Python Program: Multiple Linear Regression for Stock Market data and plot.**

python

Copy code

import pandas as pd

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

# Create a DataFrame for Stock Market data

Stock\_Market = {

'Year': [2017]\*12 + [2016]\*12,

'Month': list(range(12, 0, -1)) \* 2,

'Interest\_Rate': [2.75, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.25, 2.25, 2.25, 2, 2, 2, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75],

'Unemployment\_Rate': [5.3, 5.3, 5.3, 5.3, 5.4, 5.6, 5.5, 5.5, 5.5, 5.6, 5.7, 5.9, 6, 5.9, 5.8, 6.1, 6.2, 6.1, 6.1, 6.1, 5.9, 6.2, 6.2, 6.1],

'Stock\_Index\_Price': [1464, 1394, 1357, 1293, 1256, 1254, 1234, 1195, 1159, 1167, 1130, 1075, 1047, 965, 943, 958, 971, 949, 884, 866, 876, 822, 704, 719]

}

# Convert to DataFrame

df = pd.DataFrame(Stock\_Market)

# Multiple Linear Regression

X = df[['Interest\_Rate', 'Unemployment\_Rate']]

y = df['Stock\_Index\_Price']

model = LinearRegression()

model.fit(X, y)

# Plot the Stock Market Price vs Interest Rate

plt.scatter(df['Interest\_Rate'], y, color='blue')

plt.plot(df['Interest\_Rate'], model.predict(X), color='red')

plt.xlabel('Interest Rate')

plt.ylabel('Stock Index Price')

plt.show()

**Slip 18**

**Q1. R Program: Find the maximum and minimum value of a given vector.**

r

Copy code

# Given vector

vector <- c(12, 45, 67, 89, 23, 15, 90)

# Find max and min

max\_value <- max(vector)

min\_value <- min(vector)

print(paste("Maximum value:", max\_value))

print(paste("Minimum value:", min\_value))

**Q2. Python Program: Simple Linear Regression and Model Performance (sklearn).**

python

Copy code

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Data

x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape((-1, 1))

y = np.array([7, 14, 15, 18, 19, 21, 26, 23])

# Linear Regression

model = LinearRegression()

model.fit(x, y)

# Predict

y\_pred = model.predict(x)

# Calculate performance metrics

mse = mean\_squared\_error(y, y\_pred)

rmse = np.sqrt(mse)

print(f"Mean Squared Error: {mse}")

print(f"Root Mean Squared Error: {rmse}")

**Slip 19**

**Q1. R Program: Create Dataframe of 5 Students and display their details.**

r

Copy code

# Data for 5 students

Rollno <- c(1, 2, 3, 4, 5)

Studname <- c("John", "Alex", "Emily", "Sophia", "David")

Address <- c("City A", "City B", "City C", "City D", "City E")

Marks <- c(78, 85, 92, 88, 80)

# Create a dataframe

students <- data.frame(Rollno, Studname, Address, Marks)

# Display the details

print(students)

**Q2. Python Program: Multiple Linear Regression for Car Dataset.**

python

Copy code

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

# Load the car dataset (Assuming it is available as 'car\_data.csv')

car\_data = pd.read\_csv('car\_data.csv')

# Features and Target variable

X = car\_data[['engine\_size', 'horsepower', 'weight']]

y = car\_data['price']

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Display coefficients

print(f"Intercept: {model.intercept\_}")

print(f"Coefficients: {model.coef\_}")

**Slip 20**

**Q1. R Program: Create a Dataframe from four given vectors.**

r

Copy code

# Create vectors

v1 <- c(101, 102, 103, 104)

v2 <- c("A", "B", "C", "D")

v3 <- c(25, 30, 28, 22)

v4 <- c("Engineer", "Doctor", "Teacher", "Lawyer")

# Create dataframe

data <- data.frame(v1, v2, v3, v4)

colnames(data) <- c("ID", "Name", "Age", "Profession")

# Display the dataframe

print(data)

**Q2. Python Program: Hierarchical Agglomerative Clustering for Customer dataset.**

python

Copy code

import pandas as pd

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

# Load dataset

customer\_data = pd.read\_csv('Customer.csv')

# Perform hierarchical clustering

Z = linkage(customer\_data, method='ward')

# Plot dendrogram

dendrogram(Z)

plt.title('Dendrogram for Customer Data')

plt.xlabel('Customers')

plt.ylabel('Distance')

plt.show()