Slip 01 - Data Mining

Q1.Write a R program to add, multiply and divide two vectors of integer type. (Vector length should be minimum 4) [10 Marks]

```
v1 &lt;- c(10L, 20L, 30L, 40L)
v2 &lt;- c(2L, 4L, 6L, 8L)

cat("Addition: ", v1 + v2, "\n")
cat("Multiplication: ", v1 * v2, "\n")
cat("Division: ", v1 / v2, "\n")
```

```
Addition:  12 24 36 48
Multiplication:  20 80 180 320
Division:  5 5 5 5
```

Q2.Consider the student data set. It can be downloaded from: https://drive.google.com/open?id=1oakZCv7g3mlmCSdv9J8kdSaqO 5_6dIOw . Write a programme in python to apply simple linear regression and find out mean absolute error, mean squared error and root mean squared error. [20 Marks]

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error,
mean_squared_error

df = pd.read_csv("Slip1.csv")

X = df[["hours"]]
y = df["score"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=1)

model = LinearRegression().fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```python
print("Intercept:", model.intercept_)
print("Slope:", model.coef_[0])
print("MAE :", mean_absolute_error(y_test, y_pred))
print("MSE :", mean_squared_error(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Intercept: 60.70645475186507
Slope: 0.10768731754784296
MAE : 9.703211157963024
MSE : 117.73090338035513
RMSE: 10.850387245640366
```

Slip 02 - Data Mining

Q1. Write an R program to calculate the multiplication table using a function.[10 Marks]

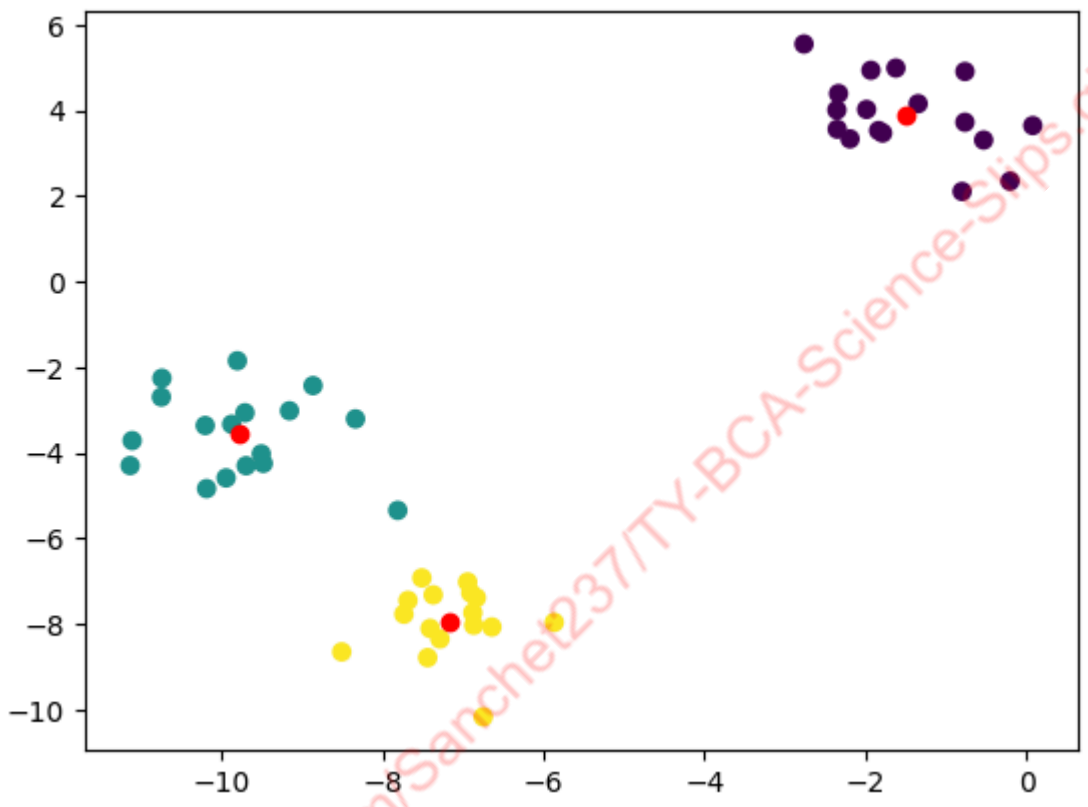
```r
multi_tab &lt;- function(n) {
  cat("Multiplication Table for", n, "\n")
  for(i in 1:10) {
    cat(n, "x", i, "=", n * i, "\n")
  }
}

num &lt;- as.integer(readline("Enter a number for multiplication table: "))
multi_tab(num)
```

```
Enter a number for multiplication table: 7
Multiplication Table for 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```


Q2. Write a python program to implement k-means algorithms on a synthetic dataset. [20 Marks]

```python
import pandas as pd
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

X, _ = make_blobs(n_samples=50, centers=3, n_features=2, random_state=1)
df = pd.DataFrame(X, columns=["x","y"])
```

```
kmeans = KMeans(n_clusters=3, random_state=1)
df['cluster'] = kmeans.fit_predict(df)

plt.scatter(df['x'], df['y'], c=df['cluster'])
plt.scatter(kmeans.cluster_centers_[:,0],
kmeans.cluster_centers_[:,1], color='red')
plt.show()
```



Slip 03 - Data Mining

Q1. Write a R program to reverse a number and also calculate the sum of digits of that number. [10 Marks]

```
num &lt;- as.integer(readline("Enter a number: "))

org_num &lt;- num
rev_num &lt;- 0
total &lt;- 0

while(num &gt; 0) {
  digit &lt;- num %% 10
  rev_num &lt;- rev_num * 10 + digit
  total &lt;- total + digit
  num &lt;- num %/% 10
}

cat("Reversed Number:", rev_num, "\n")
cat("Sum of Digits:", total, "\n")
```

```
Enter a number: 237
Reversed Number: 732
Sum of Digits: 12
```

Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b0 and b1.(use numpy package)

x=[0,1,2,3,4,5,6,7,8,9,11,13]

y = [1, 3, 2, 5, 7, 8, 8, 9, 10, 12,16, 18] [20 Marks]

```
import numpy as np

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])

# Mean of x and y
x_mean = np.mean(x)
```

```
y_mean = np.mean(y)

# Calculate coefficients
b1 = np.sum((x - x_mean) * (y - y_mean)) / np.sum((x -
x_mean)**2)
b0 = y_mean - b1 * x_mean

print("Estimated Coefficients:")
print("b0 (Intercept):", b0)
print("b1 (Slope):", b1)
```

```
Estimated Coefficients:
b0 (Intercept): 0.838709677419355
b1 (Slope): 1.2889200561009817
```

Slip 04 - Data Mining

Q1. Write a R program to calculate the sum of two matrices of given size. [10 Marks]

```
m1 &lt;- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow = TRUE)
m2 &lt;- matrix(c(6, 5, 4, 3, 2, 1), nrow = 2, ncol = 3, byrow = TRUE)

cat("\nFirst Matrix:\n")
print(m1)
cat("\nSecond Matrix:\n")
print(m2)
cat("\nSum of Matrices:\n")
print(m1 + m2)
```

```
First Matrix:
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6

Second Matrix:
     [,1] [,2] [,3]
[1,]    6    5    4
[2,]    3    2    1

Sum of Matrices:
     [,1] [,2] [,3]
[1,]    7    7    7
[2,]    7    7    7
```

Q2. Consider the following dataset:

```
weather = ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcas
           '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']
```

Use Naïve Bayes algorithm to predict [0: Overcast, 2: Mild] tuple belo
whether to play the sports or not.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

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

LE = LabelEncoder()
w = LE.fit_transform(weather)
t = LE.fit_transform(temp)
p = LE.fit_transform(play)

X = list(zip(w, t))
Y = p



classifier = GaussianNB()
classifier.fit(X, Y)

y_pred = classifier.predict([[0, 2]])

print("Prediction Value: ", y_pred)
```

```
Prediction Value:  [1]
```

Slip 05 - Data Mining

Q1. Write a R program to concatenate two given factors. [10 Marks]

```
f1 &lt;- factor(c("Pune", "Mumbai", "Delhi"))
f2 &lt;- factor(c("Chennai", "Kolkata"))

cat("Concatenated Factor:", c(as.character(f1),
as.character(f2)), "\n")
```

```
Concatenated Factor: Pune Mumbai Delhi Chennai Kolkata
```

Q2. Write a Python program build Decision Tree Classifier using Scikit- learn package for diabetes data set (download database from https://www.kaggle.com/uciml/pima indians-diabetes-database) [20 Marks]

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix

df = pd.read_csv("Slip5.csv")

print("First 5 rows of the dataset:")
print(df.head())

X = df.drop('Outcome', axis=1)
y = df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

clf = DecisionTreeClassifier(criterion='entropy',
random_state=42)
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)

print("\nModel Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
```

```
First 5 rows of the dataset:
   Glucose  BMI  Age  Outcome
0       85   22   21        0
1       89   24   25        0
2       95   26   29        0
3      105   28   33        0
4      120   30   37        1

Model Accuracy: 1.0

Confusion Matrix:
 [[1 0]
 [0 1]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

Slip 06 - Data Mining

Q1. Write a R program to create a data frame using two given vectors and display the duplicate elements. [10 Marks]

```
names &lt;- c("Sanchet", "Gaurav", "Ajinkya", "Gaurav",
"Sanchet")
ages  &lt;- c(21, 23, 22, 23, 21)

df &lt;- data.frame(Name = names, Age = ages)

cat("Data Frame:\n")
print(df)

cat("\nDuplicate Rows:\n")
print(df[duplicated(df), ])
```

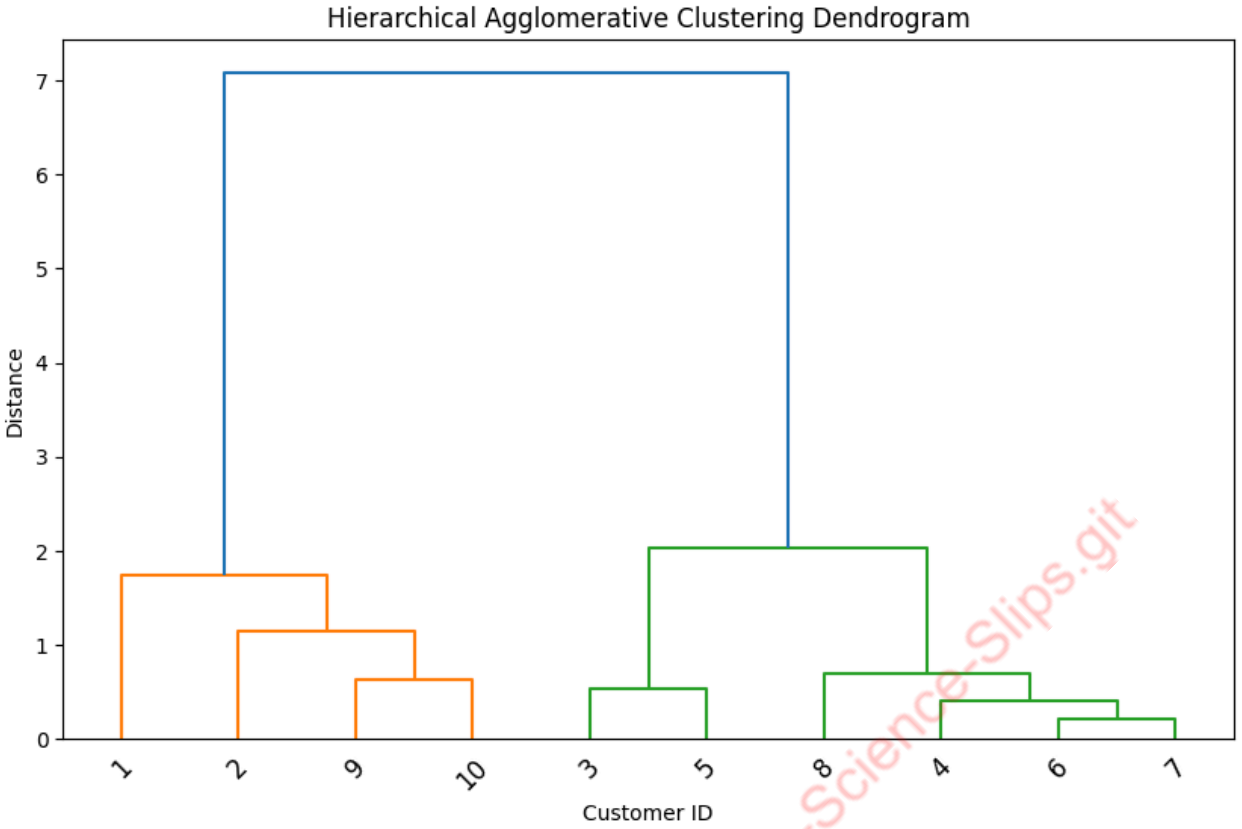
```
Data Frame:
     Name Age
1 Sanchet  21
2  Gaurav  23
3 Ajinkya  22
4  Gaurav  23
5 Sanchet  21

Duplicate Rows:
     Name Age
4  Gaurav  23
5 Sanchet  21
```


Q2. Write a python program to implement hierarchical Agglomerative clustering algorithm. (Download Customer.csv dataset from github.com). [20 Marks]

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage

df = pd.read_csv("Slip6.csv")
```

```
print("First 5 rows of the dataset:")
print(df.head())

X = df[["Age", "Income", "Spending"]]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

linkage_matrix = linkage(X_scaled, method='ward')

plt.figure(figsize=(10, 6))
dendrogram(linkage_matrix, labels=df["CustomerID"].values,
leaf_rotation=45)
plt.title("Hierarchical Agglomerative Clustering Dendrogram")
plt.xlabel("Customer ID")
plt.ylabel("Distance")
plt.show()
```

```
First 5 rows of the dataset:
   CustomerID  Age  Income  Spending
0           1   22      15        39
1           2   25      18        81
2           3   47      36         6
3           4   52      52         4
4           5   46      44         8
```



Slip 07 - Data Mining

Q1. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91. [10 Marks]

```r
cat("Sequence 20-50:", 20:50, "\n")
cat("Mean of 20-60:", mean(20:60), "\n")
cat("Sum of 51-91:", sum(51:91), "\n")
```

```
Sequence 20-50: 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
Mean of 20-60: 40
Sum of 51-91: 2911
```

Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b1 and b1 Also analyse the performance of the model (Use sklearn package) x = np.array([1,2,3,4,5,6,7,8]) y = np.array([7,14,15,18,19,21,26,23]) [20 Marks]

```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv("Slip7.csv")
X = df[['x']]
y = df['y']

model = LinearRegression()
model.fit(X, y)

print("b0 (Intercept):", model.intercept_)
print("b1 (Slope):", model.coef_[0])

y_pred = model.predict(X)
print("Mean Squared Error:", mean_squared_error(y, y_pred))
print("R^2 Score:", r2_score(y, y_pred))
```

```
b0 (Intercept): 7.642857142857139
b1 (Slope): 2.2738095238095246
```

```
Mean Squared Error: 3.4657738095238106
R^2 Score: 0.886774107294781
```

Slip 08 - Data Mining

Q1. Write a R program to get the first 10 Fibonacci numbers. [10 Marks

```
n &lt;- as.integer(readline("Enter the range : "))

fib &lt;- numeric(n)
fib[1] &lt;- 0
fib[2] &lt;- 1
for(i in 3:n) {
  fib[i] &lt;- fib[i-1] + fib[i-2]
}

cat("First", n, "Fibonacci Numbers:", fib, "\n")
```

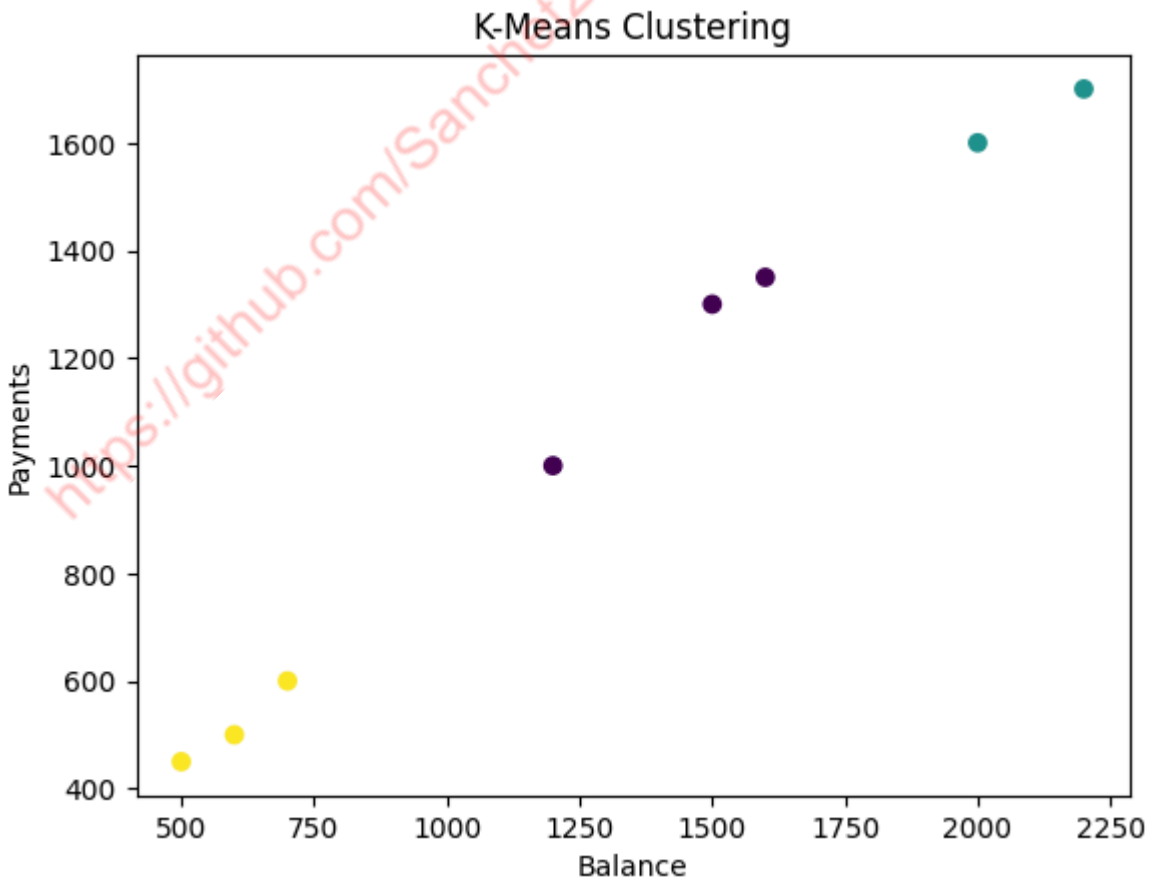
```
Enter the range : 8
First 8 Fibonacci Numbers: 0 1 1 2 3 5 8 13
```

Q2. Write a python program to implement k-means algorithm to build prediction model (Use Credit Card Dataset CC GENERAL.csv Download from kaggle.com) [20 Marks]

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

df = pd.read_csv("Slip8.csv")
X = df[['balance', 'purchases', 'payments']]


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, n_init=10, random_state=1)
labels = kmeans.fit_predict(X_scaled)

df['Cluster'] = labels

print("Cluster Centers:\n", kmeans.cluster_centers_)
```

```
print("\nFirst 5 rows with cluster labels:\n", df.head())

plt.scatter(X['balance'], X['payments'], c=labels,
cmap='viridis')
plt.xlabel('Balance')
plt.ylabel('Payments')
plt.title('K-Means Clustering')
plt.show()
```

```
Cluster Centers:
 [[ 0.2410242   0.37597382  0.33021239]
 [ 1.3428491   1.20945796  1.25837695]
 [-1.13625693 -1.18227913 -1.16913036]]

First 5 rows with cluster labels:
    balance  purchases  payments  Cluster
0      500        200       450        2
1      600        250       500        2
2      700        300       600        2
3     1200        800      1000        0
4     1500       1000      1300        0
```



Slip 09 - Data Mining

Q1. Write an R program to create a Data frames which contain details of 5 employees and display summary of the data. [10 Marks]

```r
emp &lt;- data.frame(
  EmpNo = 1:5,
  Name = c("Sanchet", "Gaurav", "Ajinkya", "Rahil", "Pranav"),
  Age = c(21, 23, 22, 24, 25),
  Salary = c(50000, 55000, 52000, 58000, 60000)
)
cat("\nEmployee Data:\n")
print(emp)
cat("\n\nSummary of Employee Data:\n")
print(summary(emp))
```

```
Employee Data:
  EmpNo    Name Age Salary
1     1 Sanchet  21  50000
2     2  Gaurav  23  55000
3     3 Ajinkya  22  52000
4     4   Rahil  24  58000
5     5  Pranav  25  60000


Summary of Employee Data:
     EmpNo         Name                Age          Salary     
 Min.   :1   Length:5           Min.   :21   Min.   :50000  
 1st Qu.:2   Class :character   1st Qu.:22   1st Qu.:52000  
 Median :3   Mode  :character   Median :23   Median :55000  
 Mean   :3                      Mean   :23   Mean   :55000  
 3rd Qu.:4                      3rd Qu.:24   3rd Qu.:58000  
 Max.   :5                      Max.   :25   Max.   :60000  
```

Q2. Write a Python program to build an SVM model to Cancer dataset. The dataset is available in the scikit-learn library. Check the accuracyof model with precision and recall. [20 Marks]

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
```

```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score,
recall_score

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = SVC(kernel='linear', random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
```

```
Accuracy: 0.956140350877193
Precision: 0.9459459459459459
Recall: 0.9859154929577465
```

Slip 10 - Data Mining

Q1. Write a R program to find the maximum and the minimum value of a givenvector [10 Marks]

```
v &lt;- c(45, 12, 67, 89, 23, 56)
cat("Max Value:", max(v), "\n")
cat("Min Value:", min(v), "\n")
```

```
Max Value: 89
Min Value: 12
```

Q2. Write a Python Programme to read the dataset ("Iris.csv"). dataset download from (https://archive.ics.uci.edu/ml/datasets/iris) and apply Apriori algorithm. [20 Marks]

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

data = pd.read_csv("Slip10.csv")

# Convert numeric values into categories (Low/Med/High)
for col in data.columns[:-1]:
    data[col] = pd.qcut(data[col], q=3, labels=["Low", "Med",
"High"])

data_encoded = pd.get_dummies(data)

frequent_itemsets = apriori(data_encoded, min_support=0.3,
use_colnames=True)

rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.6)

rules["antecedents"] = rules["antecedents"].apply(lambda x: ',
'.join(list(x)))
rules["consequents"] = rules["consequents"].apply(lambda x: ',
'.join(list(x)))

print("Frequent Itemsets:")
print(frequent_itemsets)
```

```
print("\nAssociation Rules:")
print(rules[['antecedents','consequents','support','confidence','li
```

```
Frequent Itemsets:
     support                                                itemsets
0   0.346667                                  (sepallength_Low)
1   0.373333                                  (sepallength_Med)
2   0.380000                                   (sepalwidth_Low)
3   0.340000                                   (sepalwidth_Med)
4   0.333333                                  (petallength_Low)
5   0.360000                                  (petallength_Med)
6   0.306667                                 (petallength_High)
7   0.333333                                   (petalwidth_Low)
8   0.346667                                   (petalwidth_Med)
9   0.320000                                  (petalwidth_High)
10  0.333333                               (class_Iris-setosa)
11  0.333333                           (class_Iris-versicolor)
12  0.333333                            (class_Iris-virginica)
13  0.300000                (petallength_Low, sepallength_Low)
14  0.300000                 (petalwidth_Low, sepallength_Low)
15  0.300000              (sepallength_Low, class_Iris-setosa)
16  0.333333                 (petallength_Low, petalwidth_Low)
17  0.333333              (petallength_Low, class_Iris-setosa)
18  0.313333                 (petallength_Med, petalwidth_Med)
19  0.320000          (class_Iris-versicolor, petallength_Med)
20  0.333333                (petalwidth_Low, class_Iris-setosa)
21  0.320000           (class_Iris-versicolor, petalwidth_Med)
22  0.306667           (petalwidth_High, class_Iris-virginica)
23  0.300000  (petallength_Low, petalwidth_Low, sepallength_...
24  0.300000  (petallength_Low, sepallength_Low, class_Iris-...
25  0.300000  (petalwidth_Low, sepallength_Low, class_Iris-s...
26  0.333333  (petallength_Low, petalwidth_Low, class_Iris-s...
27  0.313333  (class_Iris-versicolor, petallength_Med, petal...
28  0.300000  (petallength_Low, petalwidth_Low, sepallength_...

Association Rules:
                          antecedents  \
0                    petallength_Low
1                    sepallength_Low
2                     petalwidth_Low
3                    sepallength_Low
4                    sepallength_Low
..                               ...
59  sepallength_Low, class_Iris-setosa
```

```
60                      petallength_Low
61                       petalwidth_Low
62                      sepallength_Low
63                    class_Iris-setosa

                                            consequents  support
confidence  \
0                                     sepallength_Low      0.3
0.900000
1                                     petallength_Low      0.3
0.865385
2                                     sepallength_Low      0.3
0.900000
3                                      petalwidth_Low      0.3
0.865385
4                                   class_Iris-setosa      0.3
0.865385
..                                                ...      ...
...
59                    petallength_Low, petalwidth_Low      0.3
1.000000
60  petalwidth_Low, sepallength_Low, class_Iris-se...      0.3
0.900000
61  petallength_Low, sepallength_Low, class_Iris-s...      0.3
0.900000
62  petallength_Low, petalwidth_Low, class_Iris-se...      0.3
0.865385
63   petallength_Low, petalwidth_Low, sepallength_Low      0.3
0.900000

        lift
0   2.596154
1   2.596154
2   2.596154
3   2.596154
4   2.596154
..       ...
59  3.000000
60  3.000000
61  3.000000
62  2.596154
63  3.000000

[64 rows x 5 columns]
```


</>

Slip 11 - Data Mining

Q1. Write a R program to find all elements of a given list that are not inanother given list.

AB st("x", "y", "z") st("X", "Y", "Z", " x", "y", "z")

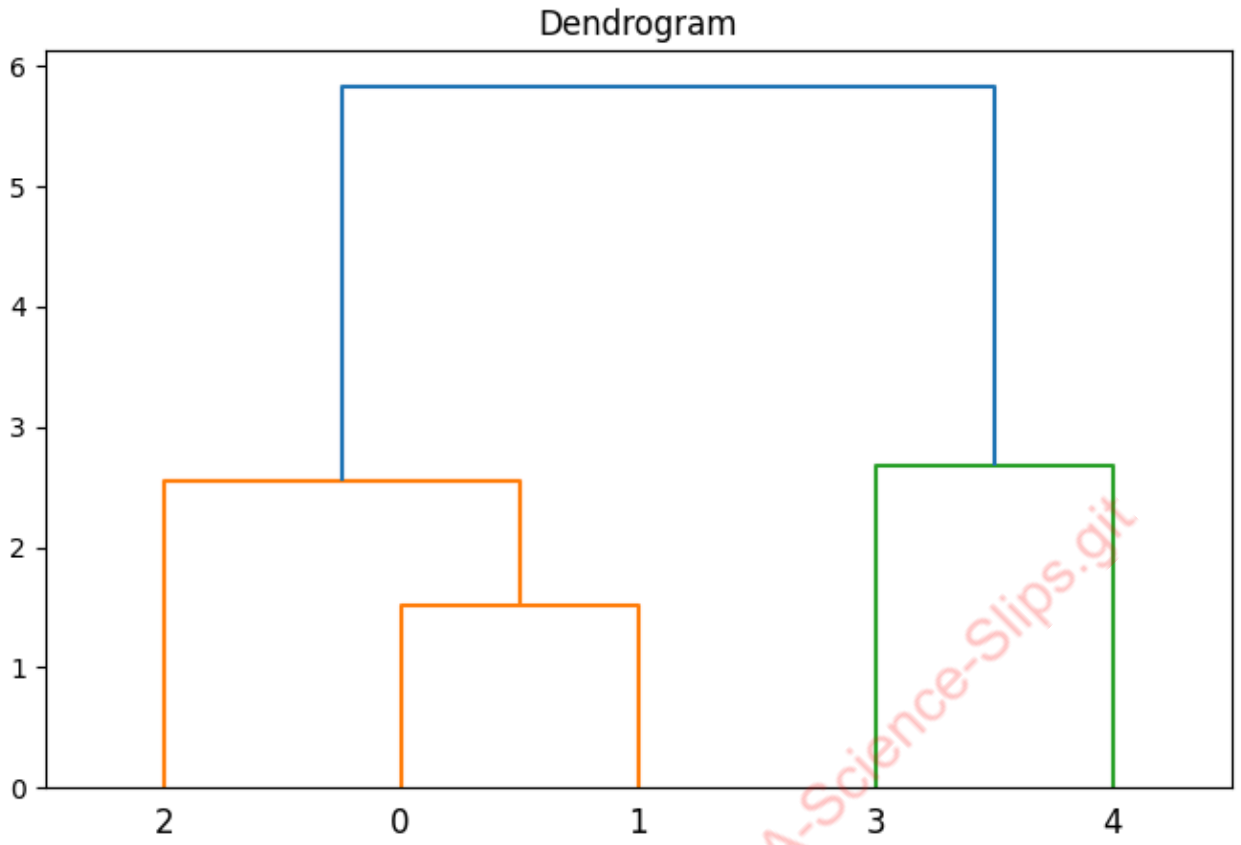
[10 Marks]

```
l1 &lt;- c("x", "y", "z", "w")
l2 &lt;- c("X", "Y", "Z", "x", "y", "z")
cat("Elements in l1 but not in l2:", setdiff(l1, l2), "\n")
```

```
Elements in l1 but not in l2: w
```

Q2. Write a python program to implement hierarchical clustering algorithm.(Download Wholesale customers data dataset from github.com). [20 Marks]

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering

df = pd.read_csv("Slip11.csv")
X = df.select_dtypes(include="number")
X_scaled = StandardScaler().fit_transform(X)


Z = linkage(X_scaled, method='ward')
plt.figure(figsize=(8,5))
dendrogram(Z)
plt.title("Dendrogram")
plt.show()

model = AgglomerativeClustering(n_clusters=3, linkage='ward')
labels = model.fit_predict(X_scaled)

print("Cluster labels:", labels)
```




```
Cluster labels: [0 0 0 2 1]
```

Slip 12 - Data Mining

Q1. Write a R program to create a Dataframes which contain details of 5employees and display the details. Employee contain (empno,empname,gender,age,designation) [10 Marks]

```
emp &lt;- data.frame(
  EmpNo = 1:5,
  EmpName = c("Sanchet", "Gaurav", "Ajinkya", "Rahil", "Pranav"),
  Gender = c("M", "M", "M", "M", "M"),
  Age = c(22, 22, 20, 21, 21),
  Designation = c("Developer", "Manager", "Tester", "Analyst",
"Designer")
)
cat("Employee Details:\n")
print(emp)
```

```
Employee Details:
  EmpNo EmpName Gender Age Designation
1     1 Sanchet      M  22   Developer
2     2  Gaurav      M  22     Manager
3     3 Ajinkya      M  20      Tester
4     4   Rahil      M  21     Analyst
5     5  Pranav      M  21    Designer
```

Q2. Write a python program to implement multiple Linear Regression modelfor a car dataset. Dataset can be downloaded from: https://www.w3schools.com/python/python_ml_multiple_regression.asp [20 Marks]

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv("Slip12.csv")
X = df[['Weight', 'Volume']]
y = df['CO2']

lr = LinearRegression()
```

```python
lr.fit(X, y)

print("Intercept:", lr.intercept_)
print("Coefficients:", lr.coef_)

y_pred = lr.predict(X)
print("MSE:", mean_squared_error(y, y_pred))
print("R2:", r2_score(y, y_pred))
```

```
Intercept: 13.297619047619122
Coefficients: [0.03595238 0.03595238]
MSE: 1.24702380952381
R2: 0.9954269124564793
```

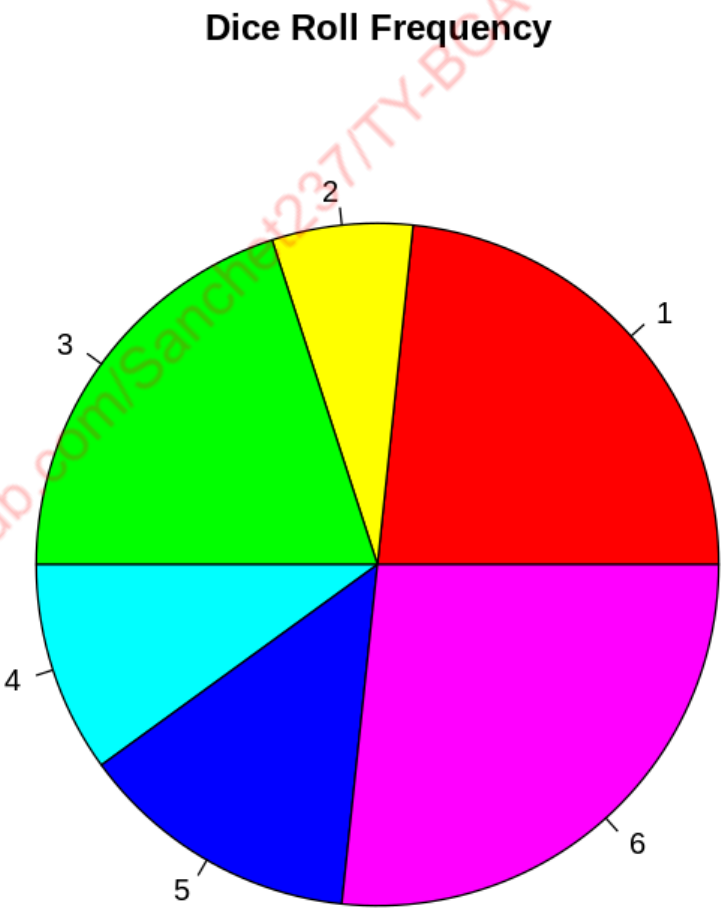
Slip 13 - Data Mining

Q1. Draw a pie chart using R programming for the following datadistribution:[10 Marks]

Digits on Dice 1 2 3 4 5 6

Frequency of getting each number 7 2 6 3 4 8

```
digits &lt;- c(1, 2, 3, 4, 5, 6)
freq &lt;- c(7, 2, 6, 3, 4, 8)
pie(freq, labels = digits, main = "Dice Roll Frequency", col =
rainbow(length(freq)))
```



Q2. Write a Python program to read "StudentsPerformance.csv" file. Solvefollowing: - To display the shape of dataset. - To display the top rows of the dataset with their columns.Note: Download dataset from following link : (https://www.kaggle.com/spscientist/students-performance-inexams? select=StudentsPerformance.csv) [20 Marks]

```python
import pandas as pd

df = pd.read_csv("Slip13.csv")

print("Shape of dataset:", df.shape)
print("\nTop rows of dataset:\n", df.head())
```

```
Shape of dataset: (5, 5)

Top rows of dataset:
    gender     race  parental_edu  math_score  reading_score
0  female  group A      bachelor          72             70
1    male  group B  some-college          65             63
2  female  group C        master          80             85
3    male  group D    highschool          58             55
4  female  group E      bachelor          90             88
```

Slip 14 - Data Mining

Q1. Write a script in R to create a list of employees (name) and perform thefollowing:

a. Display names of employees in the list.

b. Add an employee at the end of the list

c. Remove the third element of the list. [10 Marks]

```
employees &lt;- list("Sanchet", "Gaurav", "Ajinkya", "Rahil",
"Pranav")
cat("\nEmployees:", unlist(employees), "\n")

employees &lt;- append(employees, "Harsh")
cat("\nAfter Adding Harsh:", unlist(employees), "\n")

employees &lt;- employees[-3]
cat("\nAfter Removing 3rd Employee:", unlist(employees), "\n")
```

```
Employees: Sanchet Gaurav Ajinkya Rahil Pranav

After Adding Harsh: Sanchet Gaurav Ajinkya Rahil Pranav Harsh

After Removing 3rd Employee: Sanchet Gaurav Rahil Pranav Harsh
```

Q2. Write a Python Programme to apply Apriori algorithm on Groceries dataset. Dataset can be downloaded from (https://github.com/amankharwal/Websitedata/blob/master/Groceries_dataset.csv). Also display support and confidence for each rule. [20 Marks]

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

df = pd.read_csv("Slip14.csv")
trans = df['Items'].apply(lambda x: x.split(','))

te = TransactionEncoder()
data = te.fit(trans).transform(trans)
df_encoded = pd.DataFrame(data, columns=te.columns_)
```

```python
freq_items = apriori(df_encoded, min_support=0.3,
use_colnames=True)
rules = association_rules(freq_items, metric="confidence",
min_threshold=0.7)

print("Frequent Itemsets:")
print(freq_items)
print("\nRules with support and confidence:")
print(rules[['antecedents','consequents','support','confidence']])
```

```
Frequent Itemsets:
   support         itemsets
0    0.750          (bread)
1    0.500         (butter)
2    0.500           (eggs)
3    0.500           (milk)
4    0.375  (bread, butter)
5    0.375    (bread, eggs)
6    0.500    (bread, milk)

Rules with support and confidence:
  antecedents consequents  support  confidence
0    (butter)     (bread)    0.375        0.75
1      (eggs)     (bread)    0.375        0.75
2      (milk)     (bread)    0.500        1.00
```

Slip 15 - Data Mining

Q1.Write a R program to add, multiply and divide two vectors of integer type.(vector length should be minimum 4) [10 Marks]

```
v1 &lt;- c(10, 20, 30, 40)
v2 &lt;- c(2, 4, 6, 8)

cat("Addition:", v1 + v2, "\n")
cat("Multiplication:", v1 * v2, "\n")
cat("Division:", v1 / v2, "\n")
```

```
Addition: 12 24 36 48
Multiplication: 20 80 180 320
Division: 5 5 5 5
```

Q2. Write a Python program build Decision Tree Classifier for shows.csvfrom pandas and predict class label for show starring a 40 years old American comedian, with 10 years of experience, and a comedy ranking of 7? Create a csv file as shown in https://www.w3schools.com/python/python_ml_decision_tree.asp [20 Marks]

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier

df = pd.read_csv("Slip15.csv")

le = LabelEncoder()
df['Nationality'] = le.fit_transform(df['Nationality'])

X = df[['Age', 'Experience', 'Rank', 'Nationality']]
y = df['Go']


clf = DecisionTreeClassifier(criterion='entropy',
random_state=42)
clf.fit(X, y)
```

```python
# Predict for a new show: 40 years old, 10 years experience, rank 7, American
new_show = [[40, 10, 7, le.transform(['USA'])[0]]]
prediction = clf.predict(new_show)

print("Predicted class for the new show:", prediction[0])
```

```
Predicted class for the new show: YES
```

```
C:\Users\admin\AppData\Roaming\Python\Python313\site-packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Slip 16 - Data Mining

Q1. Write a R program to create a simple bar plot of given data [10 Marks]

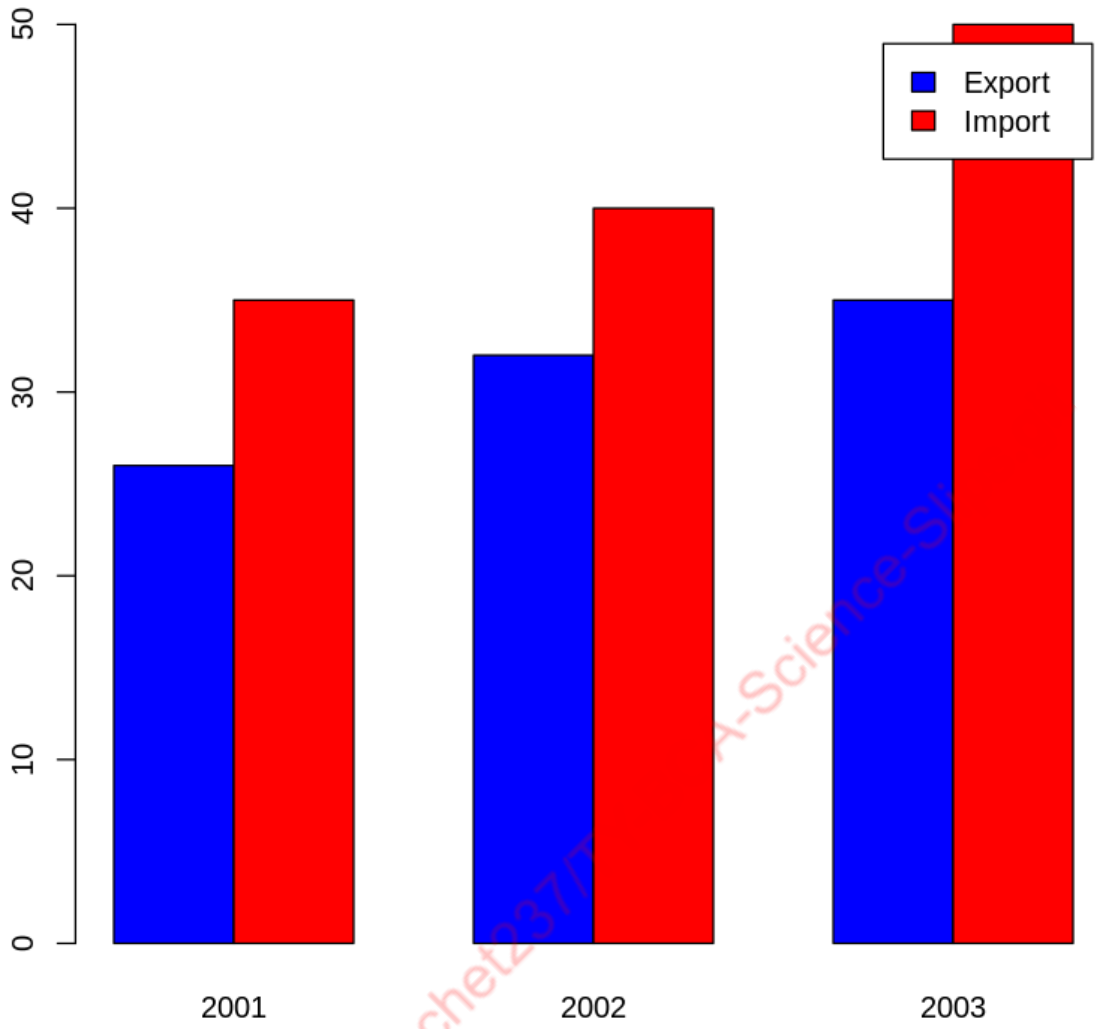
Year Export Import

2001 26 35

2002 32 40

2003 35 50

```
year &lt;- c(2001, 2002, 2003)
export &lt;- c(26, 32, 35)
import &lt;- c(35, 40, 50)

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



Q2. Write a Python program build Decision Tree Classifier using Scikit-learnpackage for diabetes data set (download database from https://www.kaggle.com/uciml/pima-indians- diabetes-database) [20 Marks]

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

df = pd.read_csv("Slip16.csv")

X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

clf = DecisionTreeClassifier(criterion='entropy',
random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test,
y_pred))
```

```
Accuracy: 1.0
Confusion Matrix:
 [[2]]
Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

```
C:\Users\admin\AppData\Roaming\Python\Python313\site-
packages\sklearn\metrics\_classification.py:534: UserWarning: A
single label was found in 'y_true' and 'y_pred'. For the
confusion matrix to have the correct shape, use the 'labels'
parameter to pass all known labels.
  warnings.warn(
```

Slip 17 - Data Mining

Q1. Write a R program to get the first 20 Fibonacci numbers. [10 Marks]

```
n &lt;- as.integer(readline("Enter the range : "))

fib &lt;- numeric(n)
fib[1] &lt;- 0
fib[2] &lt;- 1
for(i in 3:n) {
  fib[i] &lt;- fib[i-1] + fib[i-2]
}

cat("First", n, "Fibonacci Numbers:", fib, "\n")
```

```
Enter the range : 20
First 20 Fibonacci Numbers: 0 1 1 2 3 5 8 13 21 34 55 89 144 233
377 610 987 1597 2584 4181
```

Q2. Multiple Linear Regression on Stock Market Data

Write a Python program to implement a Multiple Linear Regression model for the given stock market dataset:

Stock_Market = { 'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,
2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'Month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],

'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],

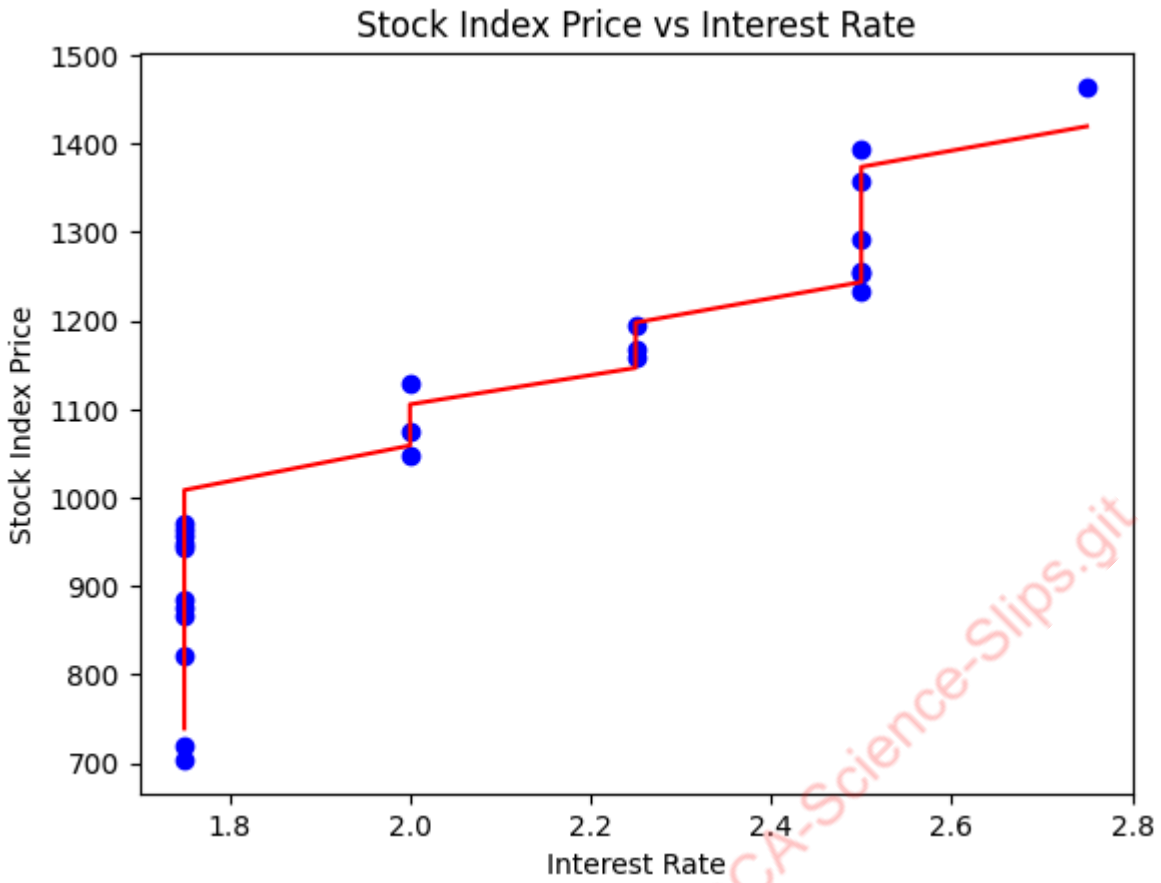
'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,
6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'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] }

Also, draw a graph of *Stock Market Price vs Interest Rate*.

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

df = pd.DataFrame({
    'Year': [2017]*12 + [2016]*12,
    'Month': [12,11,10,9,8,7,6,5,4,3,2,1]*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]
})

X = df[['Year','Month','Interest_Rate','Unemployment_Rate']]
y = df['Stock_Index_Price']

model = LinearRegression()
model.fit(X, y)

print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)

# Plot Stock Index Price vs Interest Rate
plt.scatter(df['Interest_Rate'], df['Stock_Index_Price'],
color='blue')
plt.plot(df['Interest_Rate'], model.predict(X), color='red')
plt.xlabel("Interest Rate")
plt.ylabel("Stock Index Price")
plt.title("Stock Index Price vs Interest Rate")
plt.show()
```

```
Intercept: -681638.0017816073
Coefficients: [338.26814287  27.82395958  71.92225193
45.09618248]
```



Slip 18 - Data Mining

Q1. Write a R program to find the maximum and the minimum value of a givenvector [10 Marks]

```
v &lt;- c(15, 78, 34, 92, 56)
cat("Max:", max(v), "\n")
cat("Min:", min(v), "\n")
```

```
Max: 92
Min: 15
```

Q2. Simple Linear Regression

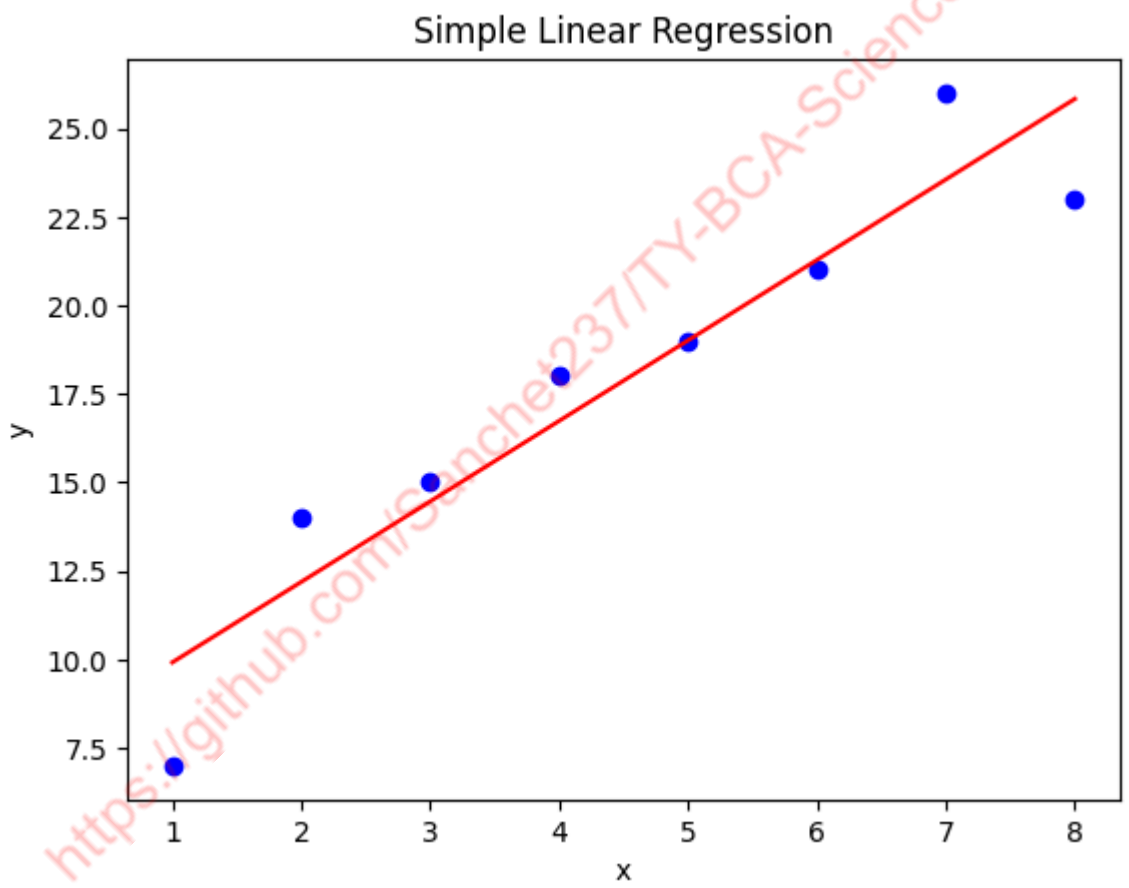
Consider the following observations/data and apply *Simple Linear Regression*. Find the estimated coefficients b_0 and b_1.
Also, analyse the performance of the model.(Use `sklearn` package) *[20 Marks]*

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([7, 14, 15, 18, 19, 21, 26, 23])

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

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 = LinearRegression()
model.fit(x, y)

b0 = model.intercept_
b1 = model.coef_[0]

print("Intercept (b0):", b0)
print("Slope (b1):", b1)

y_pred = model.predict(x)
print("Mean Squared Error:", mean_squared_error(y, y_pred))
```

```
print("R^2 Score:", r2_score(y, y_pred))

plt.scatter(x, y, color='blue')
plt.plot(x, y_pred, color='red')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Simple Linear Regression")
plt.show()
```

```
Intercept (b0): 7.642857142857139
Slope (b1): 2.2738095238095246
Mean Squared Error: 3.4657738095238106
R^2 Score: 0.886774107294781
```



Slip 19 - Data Mining

Q1. Write a R program to create a Dataframes which contain details of 5 Studentsand display the details. Students contain (Rollno,Studname,Address,Marks) [10 Marks]

```
students &lt;- data.frame(
  Rollno = 101:105,
  Studname = c("Sanchet", "Gaurav", "Ajinkya", "Rahil",
"Pranav"),
  Address = c("Pune", "West Bengal", "Ranjangaon", "Chakan",
"Jalgaon"),
  Marks = c(85, 90, 78, 88, 92)
)
cat("Students Data:\n")
print(students)
```

```
Students Data:
  Rollno Studname     Address Marks
1    101  Sanchet        Pune    85
2    102   Gaurav West Bengal    90
3    103  Ajinkya  Ranjangaon    78
4    104    Rahil      Chakan    88
5    105   Pranav     Jalgaon    92
```

Q2. Write a python program to implement multiple Linear Regression modelfor a car dataset. Dataset can be downloaded from: https://www.w3schools.com/python/python_ml_multiple_regression.asp [20 Marks]

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv("Slip19.csv")
X = df[['Weight', 'Volume']]
y = df['CO2']

lr = LinearRegression()
lr.fit(X, y)
```

```python
print("Intercept:", lr.intercept_)
print("Coefficients:", lr.coef_)

y_pred = lr.predict(X)
print("MSE:", mean_squared_error(y, y_pred))
print("R2:", r2_score(y, y_pred))
```

```
Intercept: 13.297619047619122
Coefficients: [0.03595238 0.03595238]
MSE: 1.24702380952381
R2: 0.9954269124564793
```

Slip 20 - Data Mining

Q1. Write a R program to create a data frame from four given vectors. [10 Marks]

```
v1 &lt;- c(11, 22, 33, 44)
v2 &lt;- c("A", "B", "C", "D")
v3 &lt;- c(TRUE, FALSE, TRUE, FALSE)
v4 &lt;- c(2.5, 3.6, 4.7, 5.8)

df &lt;- data.frame(v1, v2, v3, v4)
cat("Data Frame from Four Vectors:\n\n")
print(df)
```

```
Data Frame from Four Vectors:

  v1 v2    v3  v4
1 11  A  TRUE 2.5
2 22  B FALSE 3.6
3 33  C  TRUE 4.7
4 44  D FALSE 5.8
```

Q2. Write a python program to implement hierarchical Agglomerativeclustering algorithm. (Download Customer.csv dataset from github.com). [20 Marks]

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage

df = pd.read_csv("Slip20.csv")

print("First 5 rows of the dataset:")
print(df.head())

X = df[["Age", "Income", "Spending"]]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
linkage_matrix = linkage(X_scaled, method='ward')

plt.figure(figsize=(10, 6))
dendrogram(linkage_matrix, labels=df["CustomerID"].values,
leaf_rotation=45)
plt.title("Hierarchical Agglomerative Clustering Dendrogram")
plt.xlabel("Customer ID")
plt.ylabel("Distance")
plt.show()
```

```
First 5 rows of the dataset:
   CustomerID  Age  Income  Spending
0           1   22      15        39
1           2   25      18        81
2           3   47      36         6
3           4   52      52         4
4           5   46      44         8
```