COMPUTER LABORATORY MANUAL

Data Warehousing and Mining
(CS – 423)

Name:
Muhammad Razi-ur-Rehman
Ahmed Mujtaba Butt
Muhammad Jawad Liaqat
Muneeb Aslam

Course:
BESE-26C

DEPARTMENT OF COMPUTER SOFTWARE ENGINEERING
Military College of Signals
National University of Sciences and Technology
www.mcs.nust.edu.pk

PREFACE

This lab manual has been prepared to facilitate the students of Computer Software Engineering in studying and analysing various tools and concepts of Data Warehousing and Data Mining. Generally, data mining (sometimes called data or knowledge discovery) is the process of analysing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analysing data. It allows users to analyse data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

PREPARED BY

Lab manual is prepared by Assoc Prof Dr. Hammad Afzal, Asst Prof Malik Muhammad Zaki Murtaza Khan and Lab Engr Marium Hida.

GENERAL INSTRUCTIONS

a. Students are required to maintain the lab manual with them till the end of the semester.
b. All readings, answers to questions and illustrations must be solved on the place provided. If more space is required, then additional sheets may be attached.
c. It is the responsibility of the student to have the manual graded before deadlines as given by the instructor
d. Loss of manual will result in re submission of the complete manual.
e. Students are required to go through the experiment before coming to the lab session. Lab session details will be given in training schedule.
f. Students must bring the manual in each lab.
g. Keep the manual neat clean and presentable.
h. Plagiarism is strictly forbidden. No credit will be given if a lab session is plagiarised and no re submission will be entertained.
i. Marks will be deducted for late submission.
j. Error handling in a program is the responsibility of the student.

VERSION HISTORY

Date	Update By	Details
August 2014	Dr. Hammad Afzal	v1
September 2014	Dr. Hammad Afzal & TA Subhan Khan	v2
September 2016	Dr. Hammad Afzal & Lab Engr Marium Hida	v3
September 2019	Dr. Hammad Afzal, Dr. Malik Muhammad Zaki Murtaza Khan & Lab Engr Marium Hida	v4
April 2021	Dr. Hammad Afzal, Dr. Naima Iltaf, Lab Engr Sehrish Ferdous & Lab Engr Saba Siddique	V5
February 2022	Dr. Naima Iltaf, Dr. Hammad Afzal, Lab Engr Saba Siddique & Lab Engr Laraib Zainab	V6

Lab Rubrics (Group 1)

Criteria	Unacceptable (Marks=0)	Substandard Marks=1	Adequate Marks=2	Proficient Marks=3
R1 Completeness And Accuracy	The program failed to produce the right accurate result	The program execution let to inaccurate or incomplete results. It was not correctly functional or not all the features were implemented	The program was correctly functional and most of the features were implemented	The program was correctly functional, and all the features were implemented
R2 Syntax and Semantics	The student fails to figure out the syntax and semantic errors of the incorrect program	Student successfully figures out few of syntax and semantic errors of the program with extensive guidance	Student successfully figures out most of syntax and semantic errors of the program with minimum guidance	Student successfully figures out all syntax and semantic errors of the program without any guidance
R3 Demonstration	Student failed to demonstrate a clear understanding of the assigned task	Student has basic understanding, but asked questions were not answered.	Student has basic knowledge of understanding. Provides fundamental answers to asked questions	Student has demonstrated on accurate understanding of the lab objective and concepts. All the questions are answered completely and correctly
R4 Complexity and Readability	The code is poorly organized and very difficult to read	The code is readable only by someone who knows what it is supposed be doing	The code is fairly easy to read	The code is exceptionally well organized and very easy to follow
R5 Perseverance and plagiarism	Complete working program is copied indicating no effort on student's part resulting in a total score of zero for all rubrics	Most of working program is copied. Minor contribution by the student	Most of working program is contributed by the student. Minor copied components	Complete working program is contributed by the student

Lab Rubrics (Group 3)

Criteria	Unacceptable (Marks=0)	Substandard Marks=1	Adequate Marks=2	Proficient Marks=3
R1 Completeness and Accuracy	The system failed to produce the right accurate result	The system execution let to inaccurate or incomplete results. It was not correctly functional or not all the features were implemented	The system was correctly functional and most of the features were implemented	The system was correctly functional, and all the features were implemented
R2 Demonstration	The student failed to demonstrate a clear understanding of the assigned task	The student has basic knowledge of understanding, but asked questions were not answered.	The student has moderate knowledge of understanding. Answer to the question is basic	The student has demonstrated on accurate understanding of the lab objective and concepts. All the questions are answered completely and correctly
R3 Plagiarism	Complete working program is copied indicating no effort on student's part resulting in a total score of zero for all rubrics	Most of working program is copied. Minor contribution by the student	Most of working program is contributed by the student. Minor copied components	Complete working program is contributed by the student
R4 Contribution/ Group participation	Shows little commitment to group goals and fails to perform assigned roles	Demonstrates commitment to group goals, but has difficulty performing assigned roles	Demonstrates commitment to group goals and carries out assigned roles effectively	Actively helps to identify group goals and works effectively to meet them in all roles assumed
R5 Presentation skills	Poor presentation; cannot explain topic; scientific terminology lacking or confused; lacks understanding of topic	Presentation lacks clarity and organization; little use of scientific terms and vocabulary; poor understanding of topic	Presentation acceptable; adequate use of scientific terms; acceptable understanding of topic	Well-organized, clear presentation; good use of scientific vocabulary and terminology; good understanding of topic

Lab Rubrics (Group 6)

Criteria	Unacceptable (Marks=0)	Substandard Marks=1	Adequate Marks=2	Proficient Marks=3
R1 Completeness and Accuracy	The system failed to produce the right accurate result	The system execution let to inaccurate or incomplete results. It was not correctly functional or not all the features were implemented	The system was correctly functional and most of the features were implemented	The system was correctly functional, and all the features were implemented
R2 Complex Problems	Fails to comprehend the problem and its implications	The student is unable to decompose/transfer problem to conceptual model with adequate understanding of the complexity of his design	The student is able to convert conceptual model to simulation/program	The student is able to analyze and infer the results after execution and is able to relate the results to conceptual model's design choices/complexities
R3 Demonstration	The student failed to demonstrate a clear understanding of the assigned task	The student has basic knowledge of understanding, but asked questions were not answered.	The student has moderate knowledge of understanding. Answer to the question is basic	The student has demonstrated on accurate understanding of the lab objective and concepts. All the questions are answered completely and correctly
R4 Followed Directions	The student clearly failed to follow the verbal and written instructions to successfully complete the lab	The student failed to follow the some of the verbal and written instructions to successfully complete all requirements of the lab	The student followed most of the verbal and written instructions to complete all the requirements of the lab	The student followed the verbal and written instructions to successfully complete requirements of the lab
R5 Modern tool Usage	The student clearly failed to use simulation tools to design, configure, test and troubleshoot the given scenario.	The student knows the basic knowledge of simulation tools to design, configure, test and troubleshoot the given scenario.	The student has moderate knowledge of simulation tools to design, configure, test and troubleshoot the given scenario	The student effectively uses simulation tools to design, configure, test and troubleshoot given scenario.

COURSE LEVEL OUTCOMES

CS-423 Data Warehousing and Data Mining			
Course Learning Outcomes (CLOs)			
At the end of the course the students will be able to:		PLOs	BT Level*
1	Develop the understanding of the concepts of Data Warehousing and Data Mining fundamentals including various Data Cubes, Data Pre-Processing, and Frequent Patterns Analysis	1	C-2
2	Apply the concepts of Supervised and unsupervised learning on different types of data.	3	C-3
3	Practice modern tools and programming environments to learn various data mining tasks	5	P-3

S No	List of Experiments	CLO	R-G
1.	**Introduction to Python - I**	**3**	**1**
2.	**Introduction to Python - II**	**3**	**1**
3.	**Data Cleaning**	**3**	**1**
4.	**Feature Selection**	**3**	**1**
5.	**Dimensionality Reduction using PCA**	**3**	**1**
6.	**Understanding Clustering – I**	**3**	**1**
7.	**Understanding Clustering – II**	**3**	**1**
8.	**Experiment 7: Association Rule Analysis**	3	1
9.	**Understanding Classification using KNN**	3	1
10.	**Linear Regression**	3	1
11.	**Open-Ended Lab**	3	1
12.	Data Analytics using Rapid Miner	3	1
13.	Project	3	3

Table of Contents

MARKS

S No	Experiment	Max. Marks	Marks Obtained					Instructor Sign
			R1	R2	R3	R4	R5	
Grand Total								

Experiment 1: Introduction to Python-I

Task 1:

Write a program to:

Part A:

a) Define a string with your name and assign it to a variable. Print the contents of this variable in two ways, first by simply typing the variable name and pressing enter, then by using the print statement.

In [1]:
```
name="Muhammad Razi-ur-Rehman"
name
```

Out[1]:
```
'Muhammad Razi-ur-Rehman'
```

In [2]:
```
print(name)
```

```
Muhammad Razi-ur-Rehman
```

Part B:

b) Try adding the string to itself using my_string + my_string, or multiplying it by a number, e.g., my_string * 3. Notice that the strings are joined together without any spaces. How could you fix this?

In [3]:
```
print(name*3)
```

```
Muhammad Razi-ur-RehmanMuhammad Razi-ur-RehmanMuhammad Razi-ur-Rehman
```

To improve this task we can add space in variable or try this:

In [4]:
```
print((name+" ")*3)
```

```
Muhammad Razi-ur-Rehman Muhammad Razi-ur-Rehman Muhammad Razi-ur-Rehman
```

Task 2:

Write a python program to create a function arithematic_operation to perform arithmetic operations(+, -, *, /) using two numbers.

In [5]:
```
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
def multiple(a,b):
    return a*b
def divide(a,b):
    return a/b
def module(a,b):
    return a%b

num1=int(input("Enter 1st number="))
num2=int(input("Enter 2nd number="))
oper=input("Enter any arithmetic operator=")
if(oper=="+"):
    sol=add(num1,num2)
elif(oper=="-"):
    sol=sub(num1,num2)
elif(oper=="*"):
    sol=multiple(num1,num2)
elif(oper=="/"):
    sol=divide(num1,num2)
elif(oper=="%"):
    sol=module(num1,num2)
print(f"{num1} {oper} {num2} = {sol}")
```

```
Enter 1st number=23
Enter 2nd number=2
Enter any arithmetic operator=%
23 % 2 = 1
```

TASK 3:

Given two strings, s1, and s2 return a new string made of the first, middle, and last characters each input string.

In [6]:
```
import math
s1=input("Enter the first string=")
s2=input("Enter the second string=")
if(len(s1)%2==0):
    mid1=s1[math.ceil(len(s1)/2)]+s1[math.ceil(len(s1+1)/2)]
else:
    mid1=s1[int(len(s1)/2)]
if(len(s1)%2==0):
    mid2=s2[math.ceil(len(s2)/2)]+s2[math.ceil(len(s2+1)/2)]
else:
    mid2=s2[int(len(s2)/2)]

output=s1[0]+s2[0]+mid1+mid2+s1[-1]+s2[-1]
print(output)
```

```
Enter the first string=America
Enter the second string=Japan
AJrpan
```

TASK 4:

Write a program to create a function that takes wo strings s1 and s2 and create a new string by appending s2 in the middle of s1.

In [7]:
```
import math
s1=input("Enter the first string=")
s2=input("Enter the second string=")
output=s1[:math.ceil(len(s1)/2)]+s2+s1[math.ceil(len(s1)/2):]
print(output)
```

```
Enter the first string=Software
Enter the second string=Design
SoftDesignware
```

TASK 5:

Write a program to create five lists, one for each row in our dataset:

In [8]:
```
l1=["Facebook",0.0,"USD",297476,3.5]
l2=["Instagram",0.0,"USD",2161558,4.5]
l3=["Clash of Clans",0.0,"USD",2130805,4.5]
l4=["Temple Run",0.0,"USD",1724546,4.5]
l5=["Pandora_Music & Radio",0.0,"USD",1126879,4.0]

avg=(l1[3]+l2[3]+l3[3]+l4[3]+l5[3])/5
print(avg)
```

```
1488252.8
```

TASK 6:

Write a program to create the course variable then set the course variable to be an empty list.

1. Now, Add 'Machine Learning', 'Software Construction', and 'Formal Methods' to the users list in that order without reassigning the variable.
2. Delete software construction and display the updated list content.
3. Add the course 'Artificial Intelligence' to course where ' Software Construction' used to be.
4. Slice course to Return 1st and 3rd Elements

In [9]:
```
course=[]
```

In [10]:
```
course.append("Machine Learning")
course.append("Software Construction")
course.append("Formal Methods")
course
```

Out[10]:
```
['Machine Learning', 'Software Construction', 'Formal Methods']
```

In [11]:
```
del course[1]
course
```

Out[11]:
```
['Machine Learning', 'Formal Methods']
```

In [12]:
```
course.insert(1,"Artificail Intellegent")
course
```

Out[12]:
```
['Machine Learning', 'Artificail Intellegent', 'Formal Methods']
```

In [13]:
```
course[0],course[2]
```

Out[13]:
```
('Machine Learning', 'Formal Methods')
```

Experiment 2: Python Introduction - II

TASK 1

Write a program that takes two integers as input (lower limit and upper limit) and displays all the prime numbers including and between these two numbers.

In [1]:
```
import math
num1=int(input("Enter lower limit="))
num2=int(input("Enter higher limit="))
for i in range(num1,num2+1):
    prime=True
    for a in range(2,i):
        if(i%a==0):
            prime=False
            break
    if (prime):
        print(i,end=" ")
```

```
Enter lower limit=5
Enter higher limit=19
5 7 11 13 17 19
```

TASK 2

Given a list iterate it and display numbers which are divisible by 5 and if you find number greater than 150 stop the loop iteration.

list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]

In [2]:
```
# num=int(input("Enter the length of the list="))
# numList=list(map(int,input().splbit()))
numList=[12,15,32,42,55,75,122,132,150,180,200]
for i in numList:
    if(i>150):
        break
    if(i%5==0):
        print(i,end=" ")
```

```
15 55 75 150
```

Task 3

Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically. Suppose the following input is supplied to the program: without, hello, bag, world. Then, the output should be: bag, hello, without, world.

In [3]:
```
words=input("Enter the words=").split(",")
words=[i.strip() for i in words]
words.sort()
words
```

```
Enter the words=without, hello, bag, world
```

Out[3]:
```
['bag', 'hello', 'without', 'world']
```

Task 4

a. Write a simple calculator program. Follow the steps below:

1. Declare and define a function named Menu which displays a list of choices for the user such as addition, subtraction, multiplication, and classic division. It should take the choice from user as an input and return.
2. Define and declare a separate function for each choice (each mathematical operation).
3. In the main body of the program call the respective function depending on the user's choice.

b. Implement the following functions for the calculator you created in the above task.

1. Factorial
2. x_power_y (x raised to the power y)

In [4]:

```
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
def multiple(a,b):
    return a*b
def divide(a,b):
    return a/b
def module(a,b):
    return a%b
def factorial(a):
    if(a<=1):
        return 1
    else:
        return a*factorial(a-1)
def power(a,b):
    return a**b

print("__Choices__")
print("+ --> Addition\n- --> Subtraction\n* --> Multiplication\n/ --> Division
print("= --> To Display the result")
sol=int(input("Enter number="))
oper=input("Enter any operator:")
while(oper!="="):
    if(oper!="!"):
        num2=int(input("Enter number="))

    if(oper=="+"):
        sol=add(sol,num2)
    elif(oper=="-"):
        sol=sub(sol,num2)
    elif(oper=="*"):
        sol=multiple(sol,num2)
    elif(oper=="/"):
        sol=divide(sol,num2)
    elif(oper=="%"):
        sol=module(sol,num2)
    elif(oper=="!"):
        sol=factorial(sol)
    elif(oper=="^"):
        sol=power(sol,num2)
    oper=input("Enter any operator:")
print(f"Solution={sol}")
```

```
__Choices__
+ --> Addition
- --> Subtraction
* --> Multiplication
/ --> Division
% --> Module
! --> Factorial
^ --> Power
= --> To Display the result
Enter number=2
Enter any operator:+
Enter number=3
Enter any operator:-
Enter number=2
Enter any operator:!
Enter any operator:=
Solution=6
```

In [5]:
```
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
def multiple(a,b):
    return a*b
def divide(a,b):
    return a/b
def module(a,b):
    return a%b
def factorial(a):
    if(a<=1):
        return 1
    else:
        return a*factorial(a-1)
def power(a,b):
    return a**b
def menu():
    print("__Choices__")
    print("+ --> Addition\n- --> Subtraction\n* --> Multiplication\n/ --> Divi
    print("= --> To Display the result")
    return input("Enter any operator:")
if __name__=="__main__":
    sol=int(input("Enter number="))
    oper=menu()
    while(oper!="="):
        if(oper!="!"):
            num2=int(input("Enter number="))

        if(oper=="+"):
            sol=add(sol,num2)
        elif(oper=="-"):
            sol=sub(sol,num2)
        elif(oper=="*"):
            sol=multiple(sol,num2)
        elif(oper=="/"):
            sol=divide(sol,num2)
        elif(oper=="%"):
            sol=module(sol,num2)
        elif(oper=="!"):
            sol=factorial(sol)
        elif(oper=="^"):
            sol=power(sol,num2)
        oper=menu()
    print(f"Solution={sol}")
```

```
Enter number=2
__Choices__
+ --> Addition
- --> Subtraction
* --> Multiplication
/ --> Division
% --> Module
! --> Factorial
^ --> Power
= --> To Display the result
Enter any operator:!
__Choices__
+ --> Addition
- --> Subtraction
* --> Multiplication
/ --> Division
% --> Module
! --> Factorial
^ --> Power
= --> To Display the result
Enter any operator:=
Solution=2
```

Experiment 3: Data Cleaning

Import Libraries

In [1]:
```
import pandas as pd
```

Task 1: Load and view dataset

Load and view the dataset provided after importing important libraries.

In [18]:
```
rawdata=pd.read_csv("rawdata.csv")
rawdata.head()
```

Out[18]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0

In [3]:
```
print(rawdata.size)
print(rawdata.shape)
```

```
160
(32, 5)
```

In [4]:
```
rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  32 non-null     int64  
 1   Date      31 non-null     object 
 2   Pulse     32 non-null     int64  
 3   Maxpulse  32 non-null     int64  
 4   Calories  30 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

Task2: Dealing with Empty cells

In [5]:
```
rawdata.isna().sum()
```

Out[5]:
```
Duration    0
Date        1
Pulse       0
Maxpulse    0
Calories    2
dtype: int64
```

In [6]:
```
rawdata.dropna(inplace=True)
```

In [7]:
```
print(rawdata.shape)
rawdata.isna().sum()
```

```
(29, 5)
```

Out[7]:
```
Duration    0
Date        0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

Replace Empty Values with Specific Values

Getting new data

In [9]:
```
rawdata.Calories.mean()
```

Out[9]:
```
304.68
```

In [10]:
```
rawdata.Calories.fillna(rawdata.Calories.mean(),inplace=True)
```

In [11]:
```
rawdata.isna().sum()
```

Out[11]:
```
Duration    0
Date        1
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

In [12]:
```
rawdata.Date.fillna("2020/12/21",inplace=True)
```

In [13]:
```
rawdata.isna().sum()
```

Out[13]:
```
Duration    0
Date        0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

Dealing with wrong Data

Getting new Data

In [14]:
```
rawdata.Duration[rawdata["Duration"]>120].size
```

Out[14]:
```
1
```

In [15]:
```
rawdata.Duration[rawdata["Duration"]>120]=120
```

```
C:\Users\muham\AppData\Local\Temp\ipykernel_14484\2762304739.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  rawdata.Duration[rawdata["Duration"]>120]=120
```

In [16]:
```
rawdata.Duration[rawdata["Duration"]>120].size
```

Out[16]:
```
0
```

In [17]:
```
rawdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  32 non-null     int64
 1   Date      32 non-null     object
 2   Pulse     32 non-null     int64
 3   Maxpulse  32 non-null     int64
 4   Calories  32 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

Removing Duplicated Values

Getting new Data

In [19]:
```
rawdata.duplicated().sum()
```

Out[19]:
```
1
```

In [20]:
```
rawdata.drop_duplicates(inplace=True)
```

In [21]:
```
rawdata.duplicated().sum()
```

Out[21]:
```
0
```

Exporting Dataframe

In [ ]:
```
rawdata.to_csv("updated_rawdata.csv")
```

Post Lab Task

For given dataset ‘diabetes.csv’, perform data cleaning techniques. After applying the data cleaning methods, carry out fitting of data with a Regression Model and compute its accuracy.

Importing Libraries

In [57]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
# Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
```

In [76]:
```
diabetes=pd.read_csv("diabetes.csv")
diabetes.shape
```

Out[76]:
```
(768, 9)
```

In [77]:
```
diabetes.isna().size
```

Out[77]:
```
6912
```

In [78]:
```
diabetes.head()
```

Out[78]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

In [79]:
```
diabetes.isna().sum()
```

Out[79]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [80]:
```
diabetes.duplicated().sum()
```

Out[80]:
```
0
```

Normalization

We are applying z-score standarderization

In [81]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
diabetes[["Pregnancies","Glucose","BloodPressure","SkinThickness"]]=scaler.fit
diabetes[["Insulin","BMI","DiabetesPedigreeFunction","Age"]]=scaler.fit_transf
diabetes.head()
```

Out[81]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.

Applying ML Algorithm

Spliting Data

In [82]:
```
y=diabetes.Outcome
diabetes.drop(columns=["Outcome"],axis=1,inplace=True)
# Split Data into a training set and test set
X_train,X_test,y_train,y_test = train_test_split(diabetes,y,test_size=0.3 ,rar
#Note: Here x includes the features or independent variables where y includes
```

In [85]:
```
# Fitting with Logistic Regression
lr = LogisticRegression(max_iter=100)
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
```

In [84]:
```
# Compute Model Accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 79.22077922077922
```

Experiment 4: Feature Selection

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest,chi2
import seaborn as sns
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
```

Import Dataset

In [2]:
```
df= pd.read_csv("Data/lab4.csv")
df.head()
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_c
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

5 rows × 21 columns

In [3]:
```
df.isna().sum()
```

Out[3]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

Task 1: Univariate Selection

Use the chi-squared (chi²) statistical test for non-negative features to select 10 of the best features from the above dataset which is used for Mobile Price Range Prediction.

In [4]:
```
df1=df.copy()
```

In [5]:
```
y = df1['price_range']
df1.drop(['price_range'], axis=1, inplace=True)
# Feature selection
selector = SelectKBest(chi2, k=10)
X_new =selector.fit_transform(df1, y)
# Convert the selected features to a DataFrame
selected_features = selector.get_support(indices=True)
feature_names = df1.columns[selected_features]
X_new = pd.DataFrame(X_new, columns=feature_names)
X_new
```

Out[5]:

	battery_power	fc	int_memory	mobile_wt	px_height	px_width	ram	sc_h	sc_w	ta
0	842.0	1.0	7.0	188.0	20.0	756.0	2549.0	9.0	7.0	
1	1021.0	0.0	53.0	136.0	905.0	1988.0	2631.0	17.0	3.0	
2	563.0	2.0	41.0	145.0	1263.0	1716.0	2603.0	11.0	2.0	
3	615.0	0.0	10.0	131.0	1216.0	1786.0	2769.0	16.0	8.0	
4	1821.0	13.0	44.0	141.0	1208.0	1212.0	1411.0	8.0	2.0	
...	...	...	...	...	...	...	...	...	...	
1995	794.0	0.0	2.0	106.0	1222.0	1890.0	668.0	13.0	4.0	
1996	1965.0	0.0	39.0	187.0	915.0	1965.0	2032.0	11.0	10.0	
1997	1911.0	1.0	36.0	108.0	868.0	1632.0	3057.0	9.0	1.0	
1998	1512.0	4.0	46.0	145.0	336.0	670.0	869.0	18.0	10.0	
1999	510.0	5.0	45.0	168.0	483.0	754.0	3919.0	19.0	4.0	

2000 rows × 10 columns

Task 2: Feature Importance

Load the dataset again and use Extra Tree Classifier for extracting the top 10 features for the dataset and plot your results.

In [6]:
```
df2=df.copy()
```

In [7]:
```
y = df2['price_range']
df2.drop(['price_range'], axis=1, inplace=True)
```

In [8]:
```
model = ExtraTreesClassifier()
model.fit(df2, y)
```

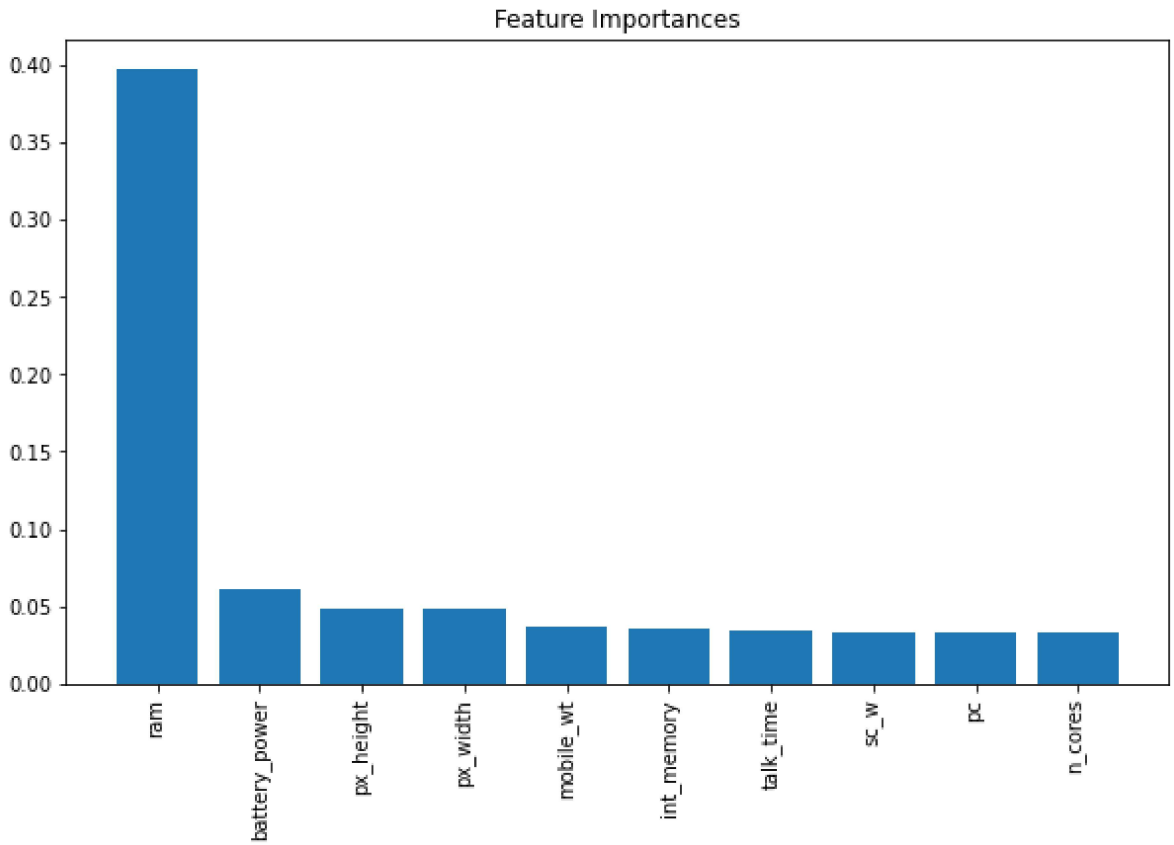
Out[8]:
```
ExtraTreesClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:
```
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
top_features = df2.columns[indices][:10]
top_features
```

Out[9]:
```
Index(['ram', 'battery_power', 'px_height', 'px_width', 'mobile_wt',
       'int_memory', 'talk_time', 'sc_w', 'pc', 'n_cores'],
      dtype='object')
```

In [10]:
```
plt.figure(figsize=(10, 6))
plt.title("Feature Importances")
plt.bar(range(10), importances[indices][:10])
plt.xticks(range(10), top_features, rotation=90)
plt.show()
```



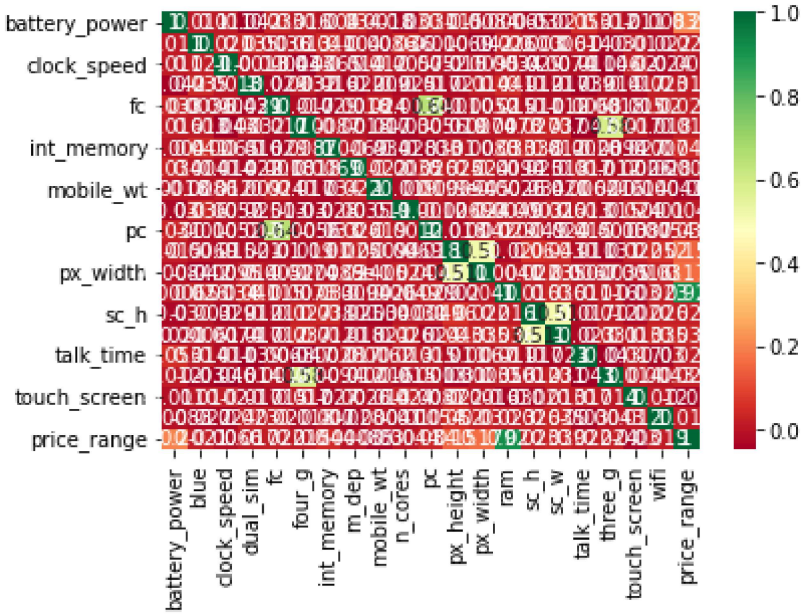
Task 3: Correlation Matrix with Heatmap

Load the dataset and plot heatmap of correlated features using the seaborn library.

In [11]:
```
df3=df.copy()
```

Plot Heatplot

In [12]:
```
corrmat = df3.corr()
g=sns.heatmap(corrmat , annot=True, cmap="RdYlGn")
```



Experiment 5: Dimensionality Reduction through PCA

Apply PCA on the Fisher's Iris data set. The data contains 3 classes of 50 instances each, where each class refers to a type of iris plant. There are 4 different attributes describing the data. You will use principal component analysis to transform the data to a lower dimensional space.

Import Libraries

Load all relevant packages and dataset.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import seaborn as sns
import matplotlib.pyplot as plt
```

Import Dataset

In [2]:
```
df=pd.read_csv("Data\\iris.csv")
df.head()
```

Out[2]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [3]:
```
df.isna().sum()
```

Out[3]:
```
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
variety         0
dtype: int64
```

In [4]:
```
df.dtypes
```

Out[4]:
```
sepal.length    float64
sepal.width     float64
petal.length    float64
petal.width     float64
variety          object
dtype: object
```

Splitting Features Vectors and Labels

Split feature vectors and labels.

In [5]:
```
x=df.iloc[:,0:4]
y=df.iloc[:,-1]
```

Normalization of Dataset

Normalize the dataset which is done by subtracting the mean of each feature vector from the dataset so that the dataset should be centered on the origin.

In [6]:
```
(x["sepal.length"].mean(),x["sepal.width"].mean(),x["petal.length"].mean(),x["
```

Out[6]:
```
(5.843333333333334, 3.0573333333333337, 3.7580000000000005, 3.758000000000000
5)
```

In [7]:
```
x1=x.copy()
x1=x1-x1.mean()
x1.head()
```

Out[7]:

	sepal.length	sepal.width	petal.length	petal.width
0	-0.743333	0.442667	-2.358	-0.999333
1	-0.943333	-0.057333	-2.358	-0.999333
2	-1.143333	0.142667	-2.458	-0.999333
3	-1.243333	0.042667	-2.258	-0.999333
4	-0.843333	0.542667	-2.358	-0.999333

Another method is by using standard scaler

In [8]:
```
scaler=StandardScaler()
x[["sepal.length","sepal.width","petal.length","petal.width"]]=scaler.fit_trar
x.head()
```

Out[8]:

	sepal.length	sepal.width	petal.length	petal.width
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

Covariance Matrix

In [9]:
```
x_cov=x.cov()
x_cov
```

Out[9]:

	sepal.length	sepal.width	petal.length	petal.width
sepal.length	1.006711	-0.118359	0.877604	0.823431
sepal.width	-0.118359	1.006711	-0.431316	-0.368583
petal.length	0.877604	-0.431316	1.006711	0.969328
petal.width	0.823431	-0.368583	0.969328	1.006711

Eigen Values and Eigen Vectors

Calculate the eigenvalues and eigenvectors.

Remember: The Eigenvectors of the Covariance matrix we get are Orthogonal to each other and each vector represents a principal axis. A higher Eigenvalue corresponds to a higher variability. Hence the principal axis with the higher Eigenvalue will be an axis capturing higher variability in the data. Orthogonal means the vectors are mutually perpendicular to each other.

In [10]:
```
values,vectors=np.linalg.eigh(x_cov)
values,vectors
```

Out[10]:
```
(array([0.02085386, 0.14774182, 0.9201649 , 2.93808505]),
 array([[ 0.26128628,  0.71956635,  0.37741762, -0.52106591],
        [-0.12350962, -0.24438178,  0.92329566,  0.26934744],
        [-0.80144925, -0.14212637,  0.02449161, -0.5804131 ],
        [ 0.52359713, -0.63427274,  0.06694199, -0.56485654]]))
```

Sorting the Eigen values

Sort the eigen values in descending order.

Remember: We order the eigenvalues from largest to smallest so that it gives us the components in order of significance. Each column in the Eigen vector-matrix corresponds to a principal component, so arranging them in descending order of their Eigenvalue will automatically arrange the principal component in descending order of their variability. Hence, the first column in our rearranged Eigen vector-matrix here will be a principal component that captures the highest variability.

In [11]:
```
index=values.argsort()[::-1]
index
```

Out[11]:
```
array([3, 2, 1, 0], dtype=int64)
```

sorting according to the index order

In [12]:
```
values=values[index]
vectors=vectors[:,index]
values,vectors
```

Out[12]:
```
(array([2.93808505, 0.9201649 , 0.14774182, 0.02085386]),
 array([[-0.52106591,  0.37741762,  0.71956635,  0.26128628],
        [ 0.26934744,  0.92329566, -0.24438178, -0.12350962],
        [-0.5804131 ,  0.02449161, -0.14212637, -0.80144925],
        [-0.56485654,  0.06694199, -0.63427274,  0.52359713]]))
```

Using PCA

In [13]:
```
pca=PCA(n_components=2)
x_pca=pca.fit(x)
eigenvector_subset = pca.components_.transpose()[:, :2]
eigenvector_subset
```

Out[13]:
```
array([[ 0.52106591,  0.37741762],
       [-0.26934744,  0.92329566],
       [ 0.5804131 ,  0.02449161],
       [ 0.56485654,  0.06694199]])
```

Transform the data by having a dot product between the Transpose of the Feature Vector and the Transpose of the mean-centered data. By transposing the outcome of the dot product, the result we get is the data reduced to lower dimensions (2-D) from higher dimensions (4-D).

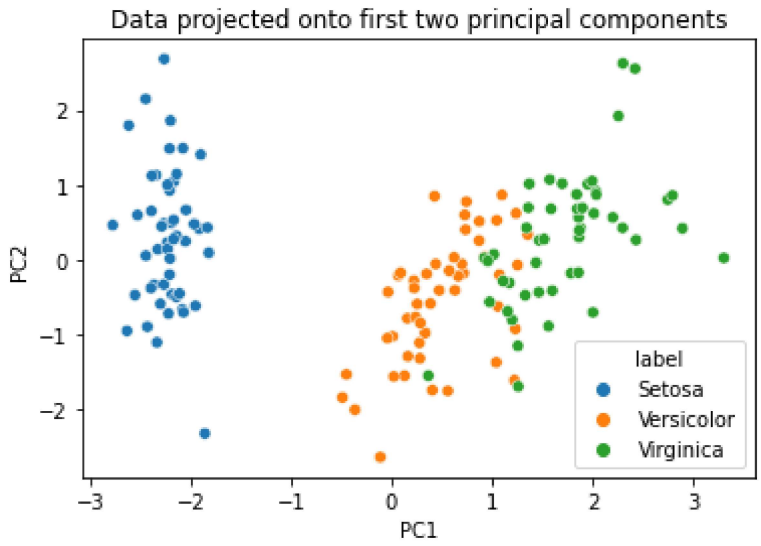
In [14]:
```
x_reduced=np.dot(eigenvector_subset.transpose(),  x.transpose()).transpose()
x_reduced
```

Out[14]:
```
array([[-2.26470281,  0.4800266 ],
       [-2.08096115, -0.67413356],
       [-2.36422905, -0.34190802],
       [-2.29938422, -0.59739451],
       [-2.38984217,  0.64683538],
       [-2.07563095,  1.48917752],
       [-2.44402884,  0.0476442 ],
       [-2.23284716,  0.22314807],
       [-2.33464048, -1.11532768],
       [-2.18432817, -0.46901356],
       [-2.1663101 ,  1.04369065],
       [-2.32613087,  0.13307834],
       [-2.2184509 , -0.72867617],
       [-2.6331007 , -0.96150673],
       [-2.1987406 ,  1.86005711],
       [-2.26221453,  2.68628449],
       [-2.2075877 ,  1.48360936],
       [-2.19034951,  0.48883832],
       [-1.898572  ,  1.40501879],
```

Ploting Principle Components

Project the data onto its first two principal components and plot the results using the seaborn and matplotlib libraries. (Hint: Create Data Frame of reduced dataset and concatenate it with Labels (target variable) to create a complete Dataset).

In [15]:
```
# Create a dataframe of the reduced dataset and concatenate it with the labels
df_reduced = pd.DataFrame(x_reduced, columns=['PC1', 'PC2'])
df_reduced['label'] = y

# Plot the results using seaborn and matplotlib
sns.scatterplot(x='PC1', y='PC2', hue='label', data=df_reduced)
plt.title('Data projected onto first two principal components')
plt.show()
```



Experiment 6: Understanding Clustering - I

Task 1

You have to solve the customer segmentation problem by using KMeans clustering and the dataset “Mall_Customers.csv”.

Import Libraries

Import the important libraries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
import sklearn.cluster as cluster
from sklearn.decomposition import PCA
import seaborn as sns
import matplotlib.pyplot as plt
```

Import dataset

Load and view the dataset

In [2]:
```
df=pd.read_csv("Mall_Customers.csv")
df.head()
```

Out[2]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Feature Scaling using MinMaxScaler

MinMaxScaler() is a data normalization technique in machine learning that scales and transforms the features of a dataset to have values between 0 and 1. This normalization method is used to ensure that all features are on a similar scale.

In [3]:
```
scaler = MinMaxScaler()
scale = scaler.fit_transform(df[['Annual Income (k$)','Spending Score (1-100)'
df_scale = pd.DataFrame(scale, columns = ['Annual Income (k$)','Spending Score
df_scale.head(5)
```

Out[3]:

	Annual Income (k$)	Spending Score (1-100)
0	0.000000	0.387755
1	0.000000	0.816327
2	0.008197	0.051020
3	0.008197	0.775510
4	0.016393	0.397959

Applying K-Means

Applying K-Means with 2 Clusters

In [4]:
```
km=KMeans(n_clusters=2,n_init=20)
y_predicted = km.fit_predict(df[['Annual Income (k$)','Spending Score (1-100)'
y_predicted
```

Out[4]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

Finding the centroid of the two clusters by using the attribute ‘cluster_centers_’

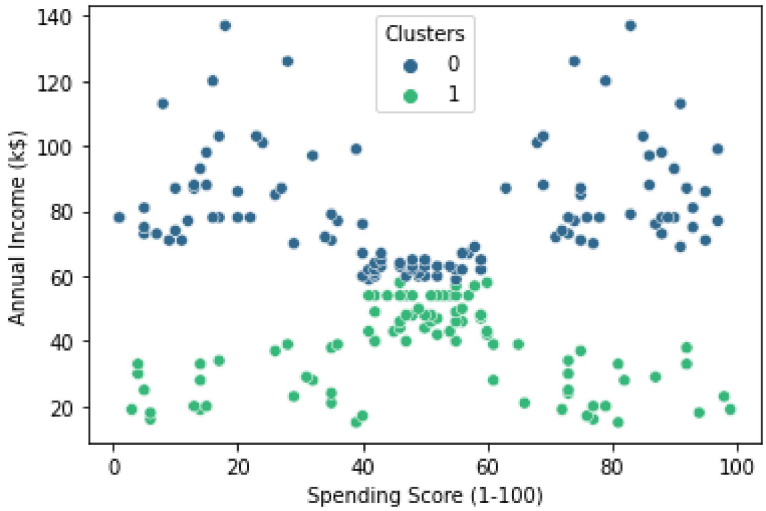
In [5]:
```
km.cluster_centers_
```

Out[5]:
```
array([[79.6       , 50.12727273],
       [37.28888889, 50.28888889]])
```

Visualize the results by using the scatterplot from seaborn library

In [6]:
```
df['Clusters'] = km.labels_
sns.scatterplot(x="Spending Score (1-100)", y="Annual Income (k$)",hue = 'Clus
```

Out[6]:
```
<AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Annual Income (k$)'>
```



Elbow Method with Within-Cluster-Sum of Squared Error (WCSS)

Task 2

finding optimun number of clusters in K means

Finding the optimum value of K

Calculating the WCSS for K=2 to k=12 and calculating the WCSS in each iteration by using the following code

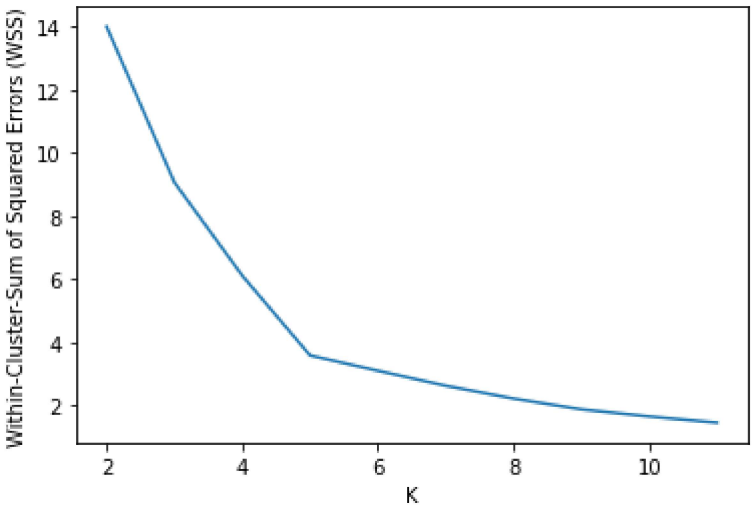
In [7]:
```
K=range(2,12)
wss = []
for k in K:
    kmeans=cluster.KMeans(n_clusters=k,n_init=20)
    kmeans=kmeans.fit(df_scale)
    wss_iter = kmeans.inertia_
    wss.append(wss_iter)
```

Plot the graph

Ploting the WCSS vs K cluster graph

In [8]:
```
plt.xlabel('K')
plt.ylabel('Within-Cluster-Sum of Squared Errors (WSS)')
plt.plot(K,wss)
```

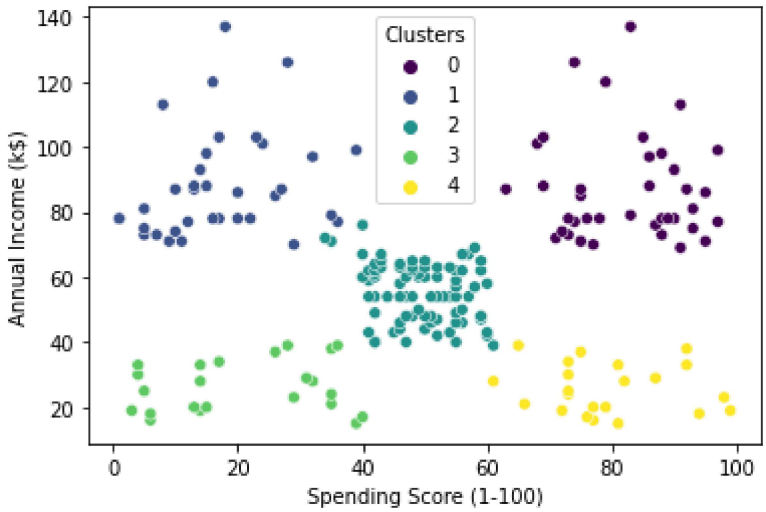
Out[8]:
```
[<matplotlib.lines.Line2D at 0x1d4fe871510>]
```



Using Optimal Value of K

In [9]:
```
kmeans = cluster.KMeans(n_clusters=5,n_init=20 )
kmeans = kmeans.fit(df[['Annual Income (k$)','Spending Score (1-100)']])
df['Clusters'] = kmeans.labels_
sns.scatterplot(x="Spending Score (1-100)", y="Annual Income (k$)",hue = 'Clus
```

Out[9]:
```
<AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Annual Income (k$)'>
```



Task 3: Applying PCA

Apply KMeans clustering after reducing the dimensionality of dataset into two components.

In [10]:
```
df_scale
```

Out[10]:

	Annual Income (k$)	Spending Score (1-100)
0	0.000000	0.387755
1	0.000000	0.816327
2	0.008197	0.051020
3	0.008197	0.775510
4	0.016393	0.397959
...	...	...
195	0.860656	0.795918
196	0.909836	0.275510
197	0.909836	0.744898
198	1.000000	0.173469
199	1.000000	0.836735

200 rows × 2 columns

In [11]:
```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df_scale)
pca_df = pd.DataFrame(data = principalComponents,columns = ['principal compone
pca_df.head()
```

Out[11]:

	principal component 1	principal component 2
0	0.123331	-0.370554
1	-0.305114	-0.380973
2	0.459767	-0.354173
3	-0.264509	-0.371786
4	0.112731	-0.354413

Experiment 07: Understanding Clutering - II

You have to solve the wholesale customer segmentation problem using hierarchical clustering. You can download the dataset using this link. The data is hosted on the UCI Machine Learning repository. The aim of this problem is to segment the clients of a wholesale distributor based on their annual spending on diverse product categories, like milk, grocery, region, etc.

Import Libraries

Import the important libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import normalize
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
```

Import Dataset

Load and view the dataset

In [2]:
```
df=pd.read_csv("Data\\Wholesale customers data.csv")
df.head()
```

Out[2]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

Normalize

Normalize the data so that the scale of each variable is the same. If the scale of the variables is not the same, the model might become biased towards the variables with a higher magnitude like Fresh or Milk.

In [3]:
```
scale_data = normalize(df)
scale_data = pd.DataFrame(scale_data, columns=df.columns)
scale_data.head()
```

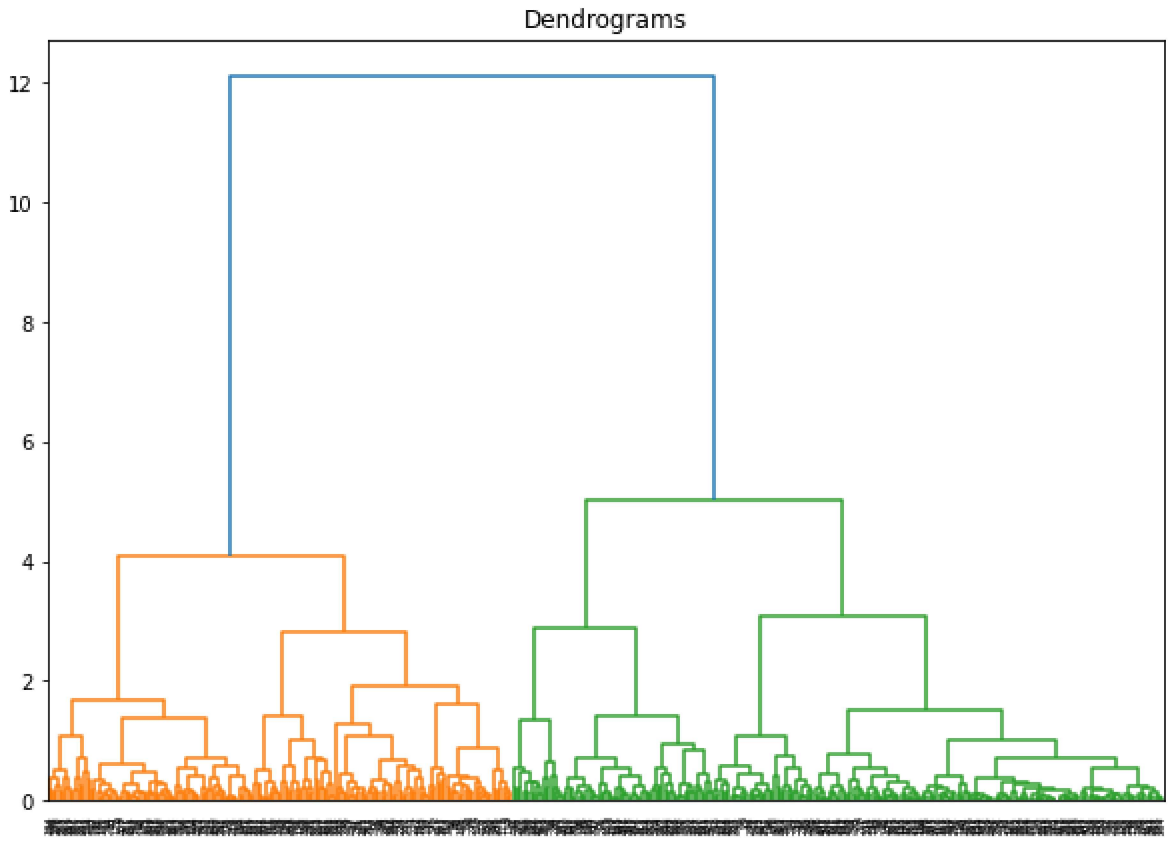
Out[3]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0.000112	0.000168	0.708333	0.539874	0.422741	0.011965	0.149505	0.074809
1	0.000125	0.000188	0.442198	0.614704	0.599540	0.110409	0.206342	0.111286
2	0.000125	0.000187	0.396552	0.549792	0.479632	0.150119	0.219467	0.489619
3	0.000065	0.000194	0.856837	0.077254	0.272650	0.413659	0.032749	0.115494
4	0.000079	0.000119	0.895416	0.214203	0.284997	0.155010	0.070358	0.205294

Draw Dendogram

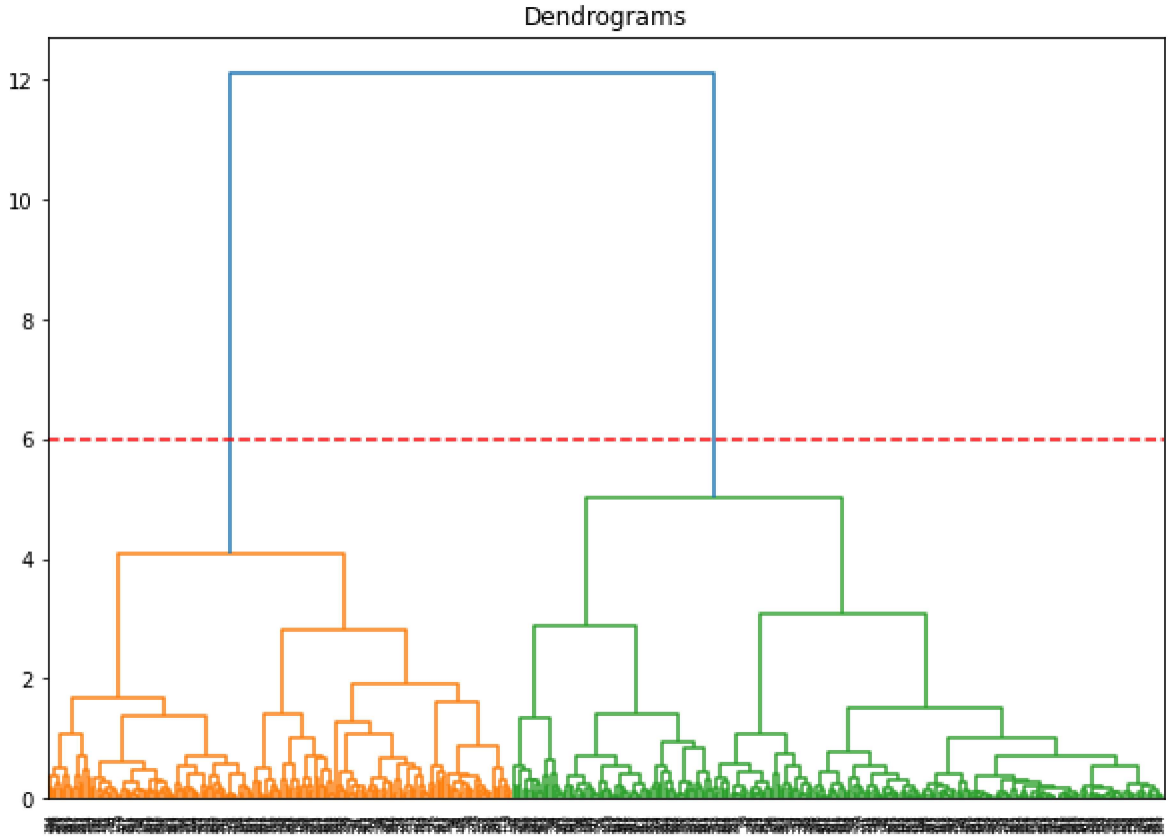
Draw the dendrogram to help you decide the number of clusters for this particular problem.After drawing the dendogram, you will see that the x-axis contains the samples and y-axis represents the distance between these samples. The vertical line with maximum distance is the blue line and hence you can decide a threshold of 6 and cut the dendrogram

In [4]:
```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = sch.dendrogram(sch.linkage(scale_data, method='ward'))
```



In [5]:
```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = sch.dendrogram(sch.linkage(scale_data, method='ward'))
plt.axhline(y=6, color='r', linestyle='--')
```

Out[5]:
```
<matplotlib.lines.Line2D at 0x257e42e6770>
```



Apply hierarchical Clustering

Apply hierarchical clustering for 2 clusters.

In [6]:
```
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage=
cluster.fit_predict(scale_data)
```

```
C:\Users\muham\AppData\Local\Programs\Python\Python310\lib\site-packages\skle
arn\cluster\_agglomerative.py:983: FutureWarning: Attribute `affinity` was de
precated in version 1.2 and will be removed in 1.4. Use `metric` instead
  warnings.warn(
```

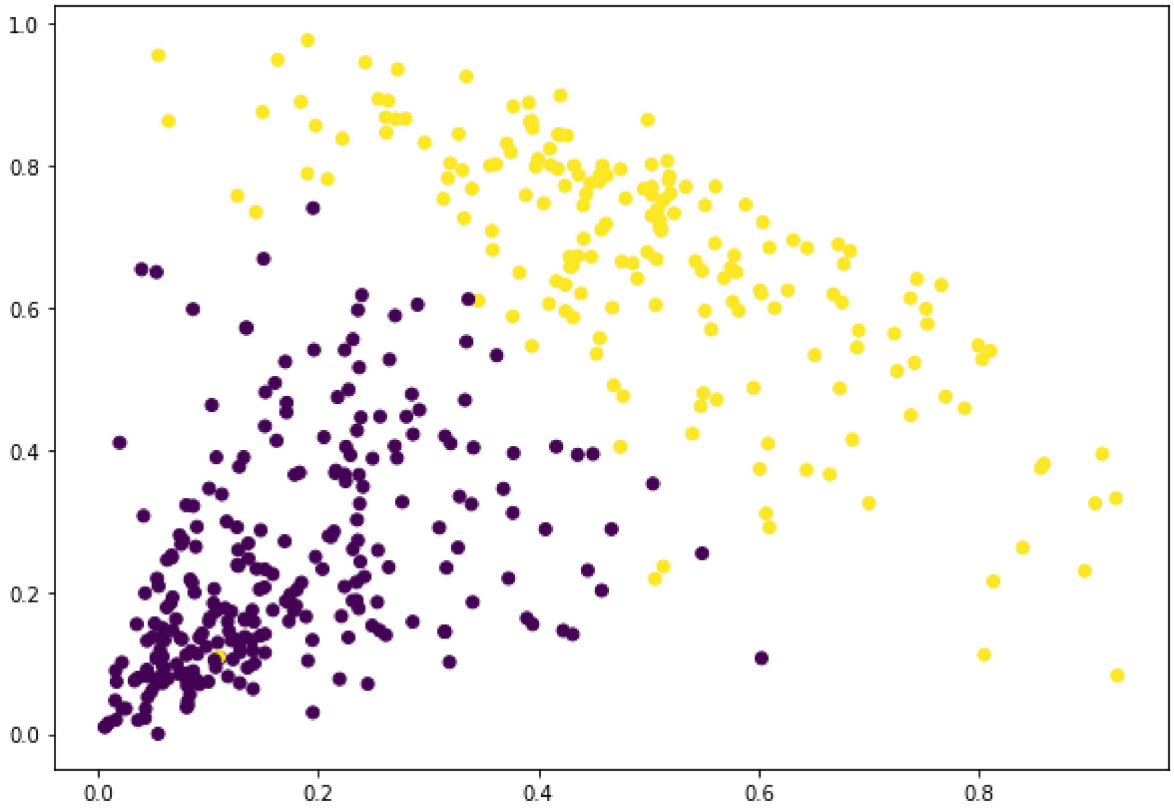
Out[6]:
```
array([1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1],
      dtype=int64)
```

Data visualization

Plot the clusters to visualize them

In [7]:
```
plt.figure(figsize=(10, 7))
plt.scatter(scale_data['Milk'], scale_data['Grocery'], c=cluster.labels_)
```

Out[7]: <matplotlib.collections.PathCollection at 0x257e1e925f0>



Experiment 8: Apriori Algorithm ¶

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori, association_rules
from sklearn.preprocessing import StandardScaler
```

Import Dataset

Load the transaction data from the 'Online Retail.xlsx' file into a panda DataFrame.

In [2]:
```
df=pd.read_excel('Data\\Online Retail.xlsx')
df.head()
```

Out[2]:

	InvoiceNo	StockCode	lower	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	C
0	536365	85123A	white hanging heart t-light holder	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Ki
1	536365	71053	white metal lantern	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Ki
2	536365	84406B	cream cupid hearts coat hanger	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Ki
3	536365	84029G	knitted union flag hot water bottle	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Ki
4	536365	84029E	red woolly hottie white heart.	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Ki

Data Cleaning

Preprocess the data by removing extra spaces in the 'Description' column, dropping rows without invoice numbers, and filtering out credit transactions. Create separate transaction baskets for each country of interest (France, United Kingdom, Portugal, and Sweden) by grouping the data based on 'Country', 'InvoiceNo', and 'Description' columns. Calculate the sum of 'Quantity' for each unique combination of 'InvoiceNo' and 'Description'. Reshape the resulting DataFrame to have 'InvoiceNo' as the index and each unique 'Description' as a column, representing the quantity of the corresponding item in the transaction.

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype         
---  ------       --------------   -----         
 0   InvoiceNo    541909 non-null  object        
 1   StockCode    541909 non-null  object        
 2   lower        1816 non-null    object        
 3   Description  540455 non-null  object        
 4   Quantity     541909 non-null  int64         
 5   InvoiceDate  541909 non-null  datetime64[ns]
 6   UnitPrice    541909 non-null  float64       
 7   CustomerID   406829 non-null  float64       
 8   Country      541909 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(5)
memory usage: 37.2+ MB
```

In [4]:
```
df.dropna(subset=["InvoiceNo"], axis=0,inplace=True)
df["Description"].str.replace("  "," ")
```

Out[4]:
```
0          WHITE HANGING HEART T-LIGHT HOLDER
1                         WHITE METAL LANTERN
2              CREAM CUPID HEARTS COAT HANGER
3         KNITTED UNION FLAG HOT WATER BOTTLE
4              RED WOOLLY HOTTIE WHITE HEART.
                       ...
541904            PACK OF 20 SPACEBOY NAPKINS
541905          CHILDREN'S APRON DOLLY GIRL
541906          CHILDRENS CUTLERY DOLLY GIRL
541907        CHILDRENS CUTLERY CIRCUS PARADE
541908          BAKING SET 9 PIECE RETROSPOT
Name: Description, Length: 541909, dtype: object
```

In [5]:
```
df["InvoiceNo"]=df["InvoiceNo"].astype(str)
df=df[~ df["InvoiceNo"].str.contains('C')]
len(df)
```

Out[5]:
```
532621
```

In [6]:
```
def makeBasket(country):
          return (df[df['Country'] =="France"]
         .groupby(['InvoiceNo', 'Description'])['Quantity']
         .sum().unstack().reset_index().fillna(0)
         .set_index('InvoiceNo'))
```

In [7]:
```
FranceBasket=makeBasket("France")
UkBasket=makeBasket("United Kingdom")
PortugalBasket=makeBasket("Portugal")
SwedenBasket=makeBasket("Sweden")
```

Applying Apriori Alogrithm

In [8]:
```
def hot_encode(x):
    if(x<= 0):
        return 0
    if(x>= 1):
        return 1

FranceBasket=FranceBasket.applymap(hot_encode)
UkBasket=UkBasket.applymap(hot_encode)
PortugalBasket=PortugalBasket.applymap(hot_encode)
SwedenBasket=SwedenBasket.applymap(hot_encode)
```

In [9]:
```
def getSortedRules(basket):
    frequency_items=apriori(FranceBasket,min_support=0.05, use_colnames=True)
    rules= association_rules(frequency_items, metric="lift",min_threshold=1)
    return rules.sort_values(['confidence','lift'],ascending=[False,False])
```

In [10]:
```
france_rules=getSortedRules(FranceBasket)
uk_rules=getSortedRules(UkBasket)
portugal_rules=getSortedRules(PortugalBasket)
sweden_rules=getSortedRules(SwedenBasket)
```

```
C:\Users\muham\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxt
end\frequent_patterns\fpcommon.py:110: DeprecationWarning: DataFrames with no
n-bool types result in worse computationalperformance and their support might
be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
C:\Users\muham\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxt
end\frequent_patterns\fpcommon.py:110: DeprecationWarning: DataFrames with no
n-bool types result in worse computationalperformance and their support might
be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
C:\Users\muham\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxt
end\frequent_patterns\fpcommon.py:110: DeprecationWarning: DataFrames with no
n-bool types result in worse computationalperformance and their support might
be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
C:\Users\muham\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxt
end\frequent_patterns\fpcommon.py:110: DeprecationWarning: DataFrames with no
n-bool types result in worse computationalperformance and their support might
be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

Displaying Rules

In [11]:
```
france_rules.head()
```

Out[11]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
44	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.076531	0.765306	0.076531	1.000	1.306667	0.01
258	(PLASTERS IN TIN CIRCUS PARADE , RED TOADSTOOL...	(POSTAGE)	0.051020	0.765306	0.051020	1.000	1.306667	0.01
270	(PLASTERS IN TIN WOODLAND ANIMALS, RED TOADSTO...	(POSTAGE)	0.053571	0.765306	0.053571	1.000	1.306667	0.01
301	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975	7.644000	0.08
302	(SET/20 RED RETROSPOT PAPER NAPKINS , SET/6 RE...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975	7.077778	0.08

In [12]:
```
uk_rules.head()
```

Out[12]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
44	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.076531	0.765306	0.076531	1.000	1.306667	0.01
258	(PLASTERS IN TIN CIRCUS PARADE , RED TOADSTOOL...	(POSTAGE)	0.051020	0.765306	0.051020	1.000	1.306667	0.01
270	(PLASTERS IN TIN WOODLAND ANIMALS, RED TOADSTO...	(POSTAGE)	0.053571	0.765306	0.053571	1.000	1.306667	0.01
301	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975	7.644000	0.08
302	(SET/20 RED RETROSPOT PAPER NAPKINS , SET/6 RE...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975	7.077778	0.08

In [13]:
```
portugal_rules.head()
```

Out[13]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
44	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.076531	0.765306	0.076531	1.000	1.306667	0.01
258	(PLASTERS IN TIN CIRCUS PARADE , RED TOADSTOOL...	(POSTAGE)	0.051020	0.765306	0.051020	1.000	1.306667	0.01
270	(PLASTERS IN TIN WOODLAND ANIMALS, RED TOADSTO...	(POSTAGE)	0.053571	0.765306	0.053571	1.000	1.306667	0.01
301	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975	7.644000	0.08
302	(SET/20 RED RETROSPOT PAPER NAPKINS , SET/6 RE...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975	7.077778	0.08

In [14]:
```
sweden_rules.head()
```

Out[14]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
44	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.076531	0.765306	0.076531	1.000	1.306667	0.01
258	(PLASTERS IN TIN CIRCUS PARADE , RED TOADSTOOL...	(POSTAGE)	0.051020	0.765306	0.051020	1.000	1.306667	0.01
270	(PLASTERS IN TIN WOODLAND ANIMALS, RED TOADSTO...	(POSTAGE)	0.053571	0.765306	0.053571	1.000	1.306667	0.01
301	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975	7.644000	0.08
302	(SET/20 RED RETROSPOT PAPER NAPKINS , SET/6 RE...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975	7.077778	0.08

Interpretation

On analyzing the above rules, it is found that boys' and girls' cutlery are paired together. This makes practical sense because when a parent goes shopping for cutlery for his/her children, he/she would want the product to be a little customized according to the kid's wishes.

Experiment 9: Understanding Classification using KNN

Task 1

You must predict whether a person will have diabetes or not using KNN classifier

Import Libraries

Import all the important libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score, confusion_matrix,accuracy_score,precisic
from sklearn.model_selection import train_test_split
from sklearn.metrics.pairwise import cosine_similarity
```

Import Dataset

Load and view the provided dataset 'diabetes.csv'

In [2]:
```
df=pd.read_csv("data\\diabetes.csv")
df.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

Data Cleaning

Perform data cleaning by replacing empty values with the mean of respective column so that it won't affect the outcome. Also split the dependent variables (features) and independent variables (label) of the dataset.

In [3]:
```
df.isna().sum()
```

Out[3]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [4]:
```
y=df.Outcome
df.drop(columns=["Outcome"],axis=1,inplace=True)
```

In [5]:
```
colName=df.columns
for col in colName:
    df[col]=df[col].replace(0,df[col].mean())
df.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6.000000	148.0	72.0	35.000000	79.799479	33.6	0
1	1.000000	85.0	66.0	29.000000	79.799479	26.6	0
2	8.000000	183.0	64.0	20.536458	79.799479	23.3	0
3	1.000000	89.0	66.0	23.000000	94.000000	28.1	0
4	3.845052	137.0	40.0	35.000000	168.000000	43.1	2

Normalization

Split data into training set and test set. Perform feature scaling to the training and test set of independent variables for reducing the size to smaller values

In [6]:
```
scaler=StandardScaler()
df[["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","D
df.head()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
0	0.536251	0.865276	-0.021044	0.872057	-0.417768	0.167255	
1	-1.140353	-1.205989	-0.516583	0.248678	-0.417768	-0.851535	
2	1.206893	2.015979	-0.681762	-0.630654	-0.417768	-1.331821	
3	-1.140353	-1.074480	-0.516583	-0.374700	-0.265107	-0.633222	
4	-0.186348	0.503626	-2.663916	0.872057	0.530423	1.549899	

In [7]:
```
X_train,X_test,y_train,y_test=train_test_split(df,y,test_size=0.2,random_state
```

Define The Model

Define the K Nearest Neighbor model with the training set. Fit your defined model and predict the test results

In [8]:
```
cls=KNeighborsClassifier(n_neighbors=11,p=2,metric="euclidean")
cls.fit(X_train,y_train)
```

Out[8]:
```
KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

Prediction

Evaluate the model using the confusion matrix, f1_score and accuracy score by comparing the predicted and actual test values.

In [9]:
```
y_pred=cls.predict(X_test)
accuracy=accuracy_score(y_pred,y_test)
precision=precision_score(y_pred,y_test)
accuracy,precision
```

Out[9]:
```
(0.7792207792207793, 0.6274509803921569)
```

Confusion Matrix

In [10]:
```
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[10]:
```
array([[88, 15],
       [19, 32]], dtype=int64)
```

F1 Score

In [11]:
```
f1=f1_score(y_test,y_pred)
f1
```

Out[11]:
```
0.6530612244897959
```

Task 2: Cosine Similarity

Using the above implemented code, vary the model by using cosine similarity measure instead of Euclidean and determine which one is producing better values in terms of accuracy and f1_score.

In [12]:
```
cls=KNeighborsClassifier(n_neighbors=11,p=2,metric="cosine")
cls.fit(X_train,y_train)
```

Out[12]:
```
KNeighborsClassifier(metric='cosine', n_neighbors=11)
```

In [13]:
```
y_pred2=cls.predict(X_test)
accuracy2=accuracy_score(y_pred2,y_test)
precision2=precision_score(y_pred2,y_test)
accuracy2,precision2
```

Out[13]:
```
(0.7857142857142857, 0.6078431372549019)
```

Confusion Matrix

In [14]:
```
cm=confusion_matrix(y_test,y_pred2)
cm
```

Out[14]:
```
array([[90, 13],
       [20, 31]], dtype=int64)
```

f1 Score

In [15]:
```
f1=f1_score(y_test,y_pred)
f1
```

Out[15]:
```
0.6530612244897959
```

Experiment 10: Linear Regression

Apply Linear Regression on Advertising data set that involves advertising expenditures on TV, radio, and newspaper, and the corresponding sales figures.

Import Libraries

Load the relevent Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

Import Dataset

Load the relavent dataset and display it

In [2]:
```
df=pd.read_csv("Data\\Advertising.csv")
df.head()
```

Out[2]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

Spliting Feature and Label

Split feature vectors and labels

In [3]:
```
y=df["Sales"]
df.drop(["Sales"],axis=1,inplace=True)
df.head()
```

Out[3]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

Cleaning and Preprocessing Data

Clean and preprocess the data to prepare it for the linear regression model. This step may involve handling missing values, encoding categorical variables, and splitting the data into features (X) and target variable (y).

In [4]:
```
df.isna().sum()
```

Out[4]:
```
TV           0
Radio        0
Newspaper    0
dtype: int64
```

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
0
```

In [6]:
```
scaler=StandardScaler()
df[['TV','Radio','Newspaper']]=scaler.fit_transform(df[['TV','Radio','Newspape
df.head()
```

Out[6]:

	TV	Radio	Newspaper
0	0.969852	0.981522	1.778945
1	-1.197376	1.082808	0.669579
2	-1.516155	1.528463	1.783549
3	0.052050	1.217855	1.286405
4	0.394182	-0.841614	1.281802

Spliting into Training and Testing data

Split the data into training and testing set

In [7]:
```
X_train,X_test,y_train,y_test=train_test_split(df,y,test_size=0.3,random_state
len(X_train),len(X_test)
```

Out[7]:
```
(140, 60)
```

Training of model

Instantiate the linear regression model and fit it to the training data. The model will learn the relationship between the features and the target variable

In [8]:
```
model=LinearRegression()
model.fit(X_train,y_train)
```

Out[8]:
```
LinearRegression
LinearRegression()
```

Model Evaluation

Evaluate the trained model using various metrics such as mean squared error (MSE) and coefficient of determination (R^2).

In [9]:
```
y_pred=model.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
mse,r2
```

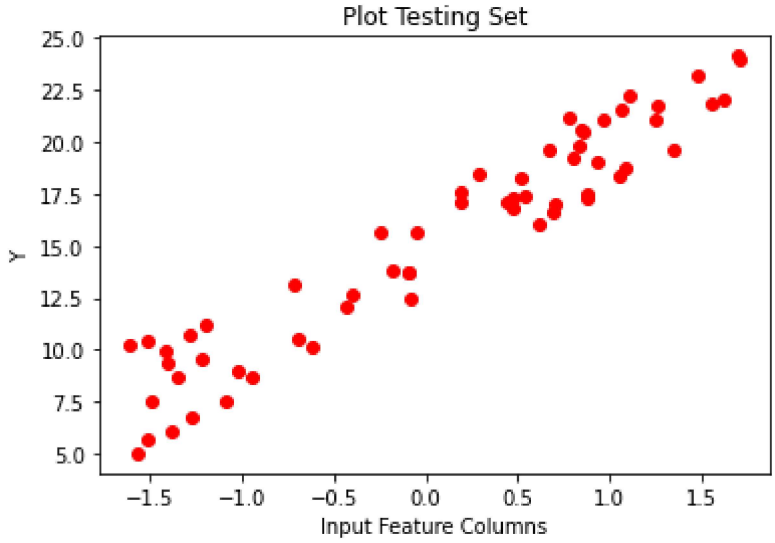
Out[9]:
```
(3.4134090612952006, 0.84425057938104)
```

Model Ploting

Plot a scatter plot of the training set and testing set using the first feature column of X as the x-axis and Y as the y-axis.

In [10]:
```
plt.scatter(X_test['TV'],y_pred,color='red')
plt.title("Plot Testing Set")
plt.ylabel('Y')
plt.xlabel('Input Feature Columns')
```

Out[10]:
```
Text(0.5, 0, 'Input Feature Columns')
```



Experiment 11 Open Ended Lab

Automobile Manufacture

Problem Statement

The automobile manufacturer is seeking to identify the closest competitors to their newly developed vehicle prototypes before launching the new model. To achieve this, they need to group existing vehicles on the market based on similarities, determine which group is the most similar to the prototypes, and use this information to identify the primary competitors for their new model. The objective is to utilize clustering techniques to identify clusters of vehicles that possess unique characteristics. This analysis will provide an overview of the current market of vehicles and aid manufacturers in deciding on the development of new models based on the identified distinct clusters.

Import Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import scipy
import pylab
import scipy.cluster.hierarchy as sch
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics.pairwise import euclidean_distances
```

Import Dataset

In [2]:
```
car=pd.read_csv("Data/cars_clus.csv")
car.head()
```

Out[2]:

	manufact	model	sales	resale	type	price	engine_s	horsepow	wheelbas	width	len
0	Acura	Integra	16.919	16.360	0.000	21.500	1.800	140.000	101.200	67.300	172.
1	Acura	TL	39.384	19.875	0.000	28.400	3.200	225.000	108.100	70.300	192.
2	Acura	CL	14.114	18.225	0.000	*null*	3.200	225.000	106.900	70.600	192.
3	Acura	RL	8.588	29.725	0.000	42.000	3.500	210.000	114.600	71.400	196.
4	Audi	A4	20.397	22.255	0.000	23.990	1.800	150.000	102.600	68.200	178.

Data Preparation

In data Preparation, following steps are performed. Not specifically in this particular order

1. Dealing With Null Values
2. Dealing With wrong values
3. Dealing with Duplicates
4. Data Normalization
5. Label Encorder
6. Converting into required Class

In [3]:
```
car.model=car.model.apply(lambda x:pd.NA if "9" in str(x) else x)
```

In [4]:
```
car.isna().sum()
```

Out[4]:
```
manufact     2
model        2
sales        0
resale       0
type         0
price        0
engine_s     0
horsepow     0
wheelbas     0
width        0
length       0
curb_wgt     0
fuel_cap     0
mpg          0
lnsales      0
partition    0
dtype: int64
```

In [5]:
```
car_test=car.dropna(axis=0,inplace=True)
```

In [6]:
```
for column in car.columns:
    car[column]=car[column].apply(lambda x:pd.NA if str(x)=="$null$" else x)
car.isna().sum()
```

Out[6]:
```
manufact      0
model         0
sales         0
resale       34
type          0
price         2
engine_s      1
horsepow      1
wheelbas      1
width         1
length        1
curb_wgt      2
fuel_cap      1
mpg           3
lnsales       0
partition     0
dtype: int64
```

In [7]:
```
car[car.isna().sum(axis=1)>2]
```

Out[7]:

	manufact	model	sales	resale	type	price	engine_s	horsepow	wheelbas	width	leng
33	Chrysler	Town & Country	53.480	19.540	1.000	<NA>	<NA>	<NA>	<NA>	<NA>	<NA

In [8]:
```
car.drop(car.index[33],axis=0,inplace=True)
```

Removing Single Unique Column

As Partitions data has only 1 entity and it is not changing so it would not be contributing anything in the prediction model. Release this feature would be a perference as less computation may be require

In [9]:
```
car.drop(["partition"],axis=1,inplace=True)
```

Filling the null values

In [10]:
```
car["price"] = pd.to_numeric(car["price"], errors="coerce")
car["price"].fillna(car["price"].mean(), inplace=True)

car["curb_wgt"] = pd.to_numeric(car["curb_wgt"], errors="coerce")
car["curb_wgt"].fillna(car["curb_wgt"].mean(), inplace=True)

car["mpg"] = pd.to_numeric(car["mpg"], errors="coerce")
car["mpg"].fillna(car["mpg"].mean(), inplace=True)
```

Converting into Required Class Type

In [11]:
```
car["sales"]=car["sales"].astype("float16")
car["type"]=car["type"].astype("float16").astype("int16")
car["engine_s"]=car["engine_s"].astype("float16")
car["horsepow"]=car["horsepow"].astype("float16").astype("int16")
car["wheelbas"]=car["wheelbas"].astype("float16")
car["width"]=car["width"].astype("float16")
car["length"]=car["length"].astype("float16")
car["fuel_cap"]=car["fuel_cap"].astype("float16")
car["lnsales"]=car["lnsales"].astype("float16")
car["price"]=car["price"].astype("float16")
car["curb_wgt"]=car["curb_wgt"].astype("float16")
car["mpg"]=car["mpg"].astype("int16")
car["resale"] = pd.to_numeric(car["resale"], errors="coerce")
```

In [12]:
```
car.dtypes
```

Out[12]:
```
manufact     object
model        object
sales       float16
resale      float64
type          int16
price       float16
engine_s    float16
horsepow      int16
wheelbas    float16
width       float16
length      float16
curb_wgt    float16
fuel_cap    float16
mpg           int16
lnsales     float16
dtype: object
```

Predicting Values using Linear Regression

In [13]:
```
df=car.copy()
```

Label Encoding

In [14]:
```
encoder=LabelEncoder()
df.model=encoder.fit_transform(df.model)
df.manufact=encoder.fit_transform(df.manufact)
```

Normalization

In [15]:
```
scaler=StandardScaler()
df[["manufact","model","sales","price","engine_s","horsepow","wheelbas","width
df.head()
```

Out[15]:

	manufact	model	sales	resale	type	price	engine_s	horsepow	wheelbas	
0	-1.721175	0.018859	-0.536252	16.360	0	-0.408665	-1.220855	-0.809898	-0.827477	-1.1
1	-1.721175	1.471021	-0.207467	19.875	0	0.072769	0.121176	0.684754	0.078434	-0.2
2	-1.721175	-1.206402	-0.577321	18.225	0	-0.000209	0.121176	0.684754	-0.084793	-0.1
3	-1.721175	0.835700	-0.658316	29.725	0	1.020388	0.409624	0.420992	0.927215	0.0
4	-1.600529	-1.592133	-0.485458	22.255	0	-0.235480	-1.220855	-0.634057	-0.639766	-0.8

Spliting data

In [16]:
```
y = df["resale"].dropna(axis=0)
X_test = df[df.isna().any(axis=1)].copy()
X_test.drop(["resale"],axis=1,inplace=True)
df.dropna(axis=0,inplace=True)
df.drop(["resale"],axis=1, inplace=True)
```

Model Selection

In [17]:
```
model=LinearRegression()
model.fit(df,y)
```

Out[17]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Model Prediction

In [18]:
```
resale_pred=model.predict(X_test)
```

In [19]:
```
nan_indices = car['resale'].isnull()
car.loc[nan_indices, 'resale'] = resale_pred
```

Checking Duplicate Value

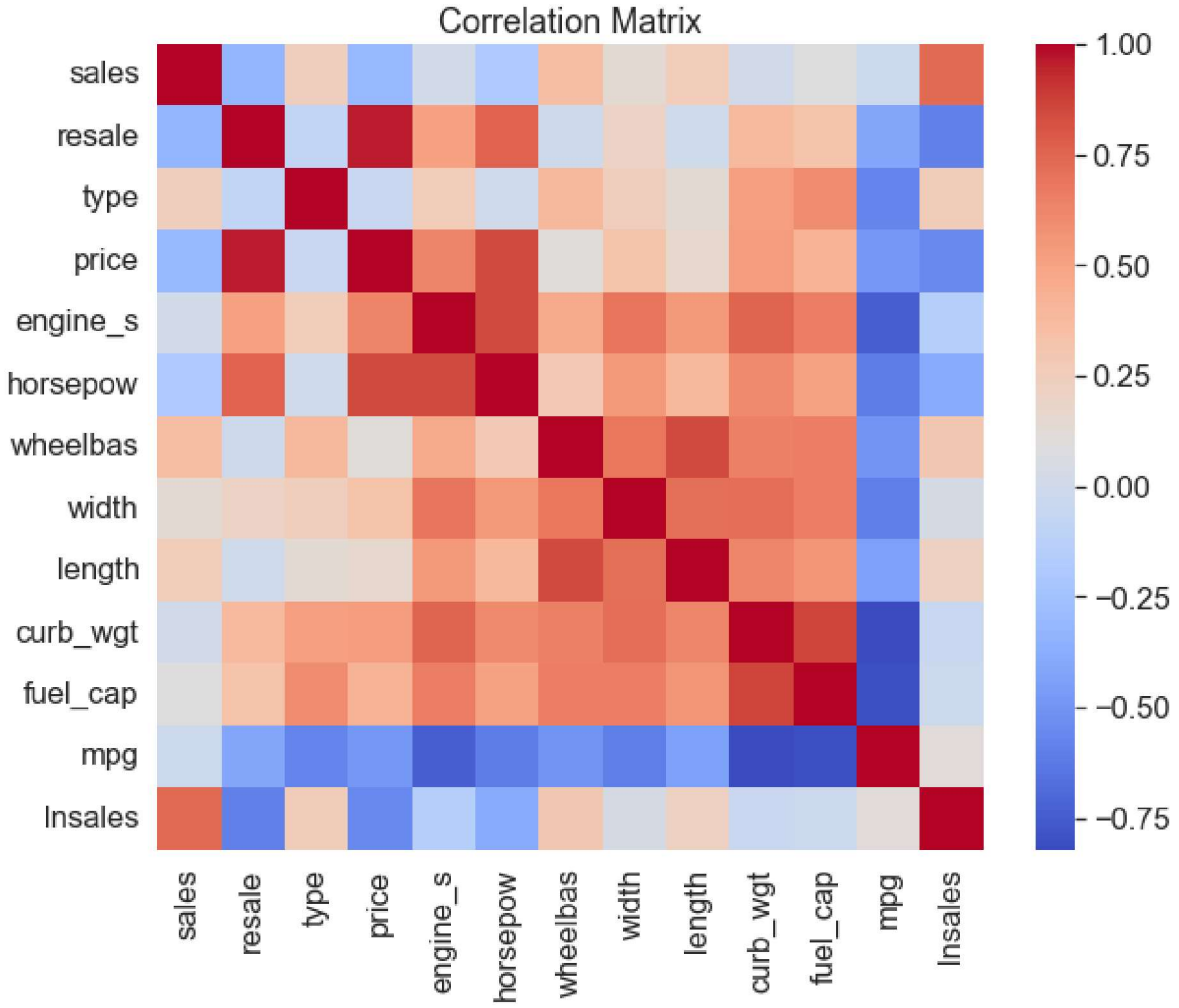
In [20]:
```
car.duplicated().sum()
```

Out[20]:
```
0
```

Feature Selection

In [21]:
```
correlation_matrix = car.corr()
plt.figure(figsize=(10, 8))
sns.set(font_scale=1.5)
plt.title('Correlation Matrix')
sns.heatmap(correlation_matrix,cmap='coolwarm');
```

```
C:\Users\muham\AppData\Local\Temp\ipykernel_23440\899194147.py:1: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a f
uture version, it will default to False. Select only valid columns or specify
the value of numeric_only to silence this warning.
  correlation_matrix = car.corr()
```



Using the above table, we can see that price and rescale are highly corelated.

In [22]:
```
car.drop(['resale'],axis=1,inplace=True)
```

In [23]:
```
car=car.reset_index(drop=True)
```

Data Modeling

In [24]:
```
featureset = car[['sales','price','engine_s',  'horsepow', 'wheelbas', 'width'
```

Normalization

In [25]:
```
featureset[['sales','price','engine_s',  'horsepow', 'wheelbas', 'width', 'ler
featureset.head()
```

```
C:\Users\muham\AppData\Local\Temp\ipykernel_23440\1567509421.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  featureset[['sales','price','engine_s',  'horsepow', 'wheelbas', 'width',
'length', 'curb_wgt', 'fuel_cap', 'mpg']]=scaler.fit_transform(featureset[['s
ales','price','engine_s',  'horsepow', 'wheelbas', 'width', 'length', 'curb_w
gt', 'fuel_cap', 'mpg']])
```

Out[25]:

	sales	price	engine_s	horsepow	wheelbas	width	length	curb_wgt	fuel_ca
0	-0.536252	-0.408665	-1.220855	-0.809898	-0.827477	-1.119705	-1.112550	-1.179290	-1.21878
1	-0.207467	0.072769	0.121176	0.684754	0.078434	-0.250241	0.408923	0.216618	-0.19293
2	-0.577321	-0.000209	0.121176	0.684754	-0.084793	-0.159672	0.343982	0.142169	-0.19293
3	-0.658316	1.020388	0.409624	0.420992	0.927216	0.057694	0.687241	0.743960	0.01143
4	-0.485458	-0.235480	-1.220855	-0.634057	-0.639766	-0.866111	-0.695073	-0.608519	-0.39730

Here, we are calcuting distance using Euclidean formula

In [26]:
```
distance=euclidean_distances(featureset.values,featureset.values)
len(distance)
```

Out[26]:
```
154
```

Finding the linkage

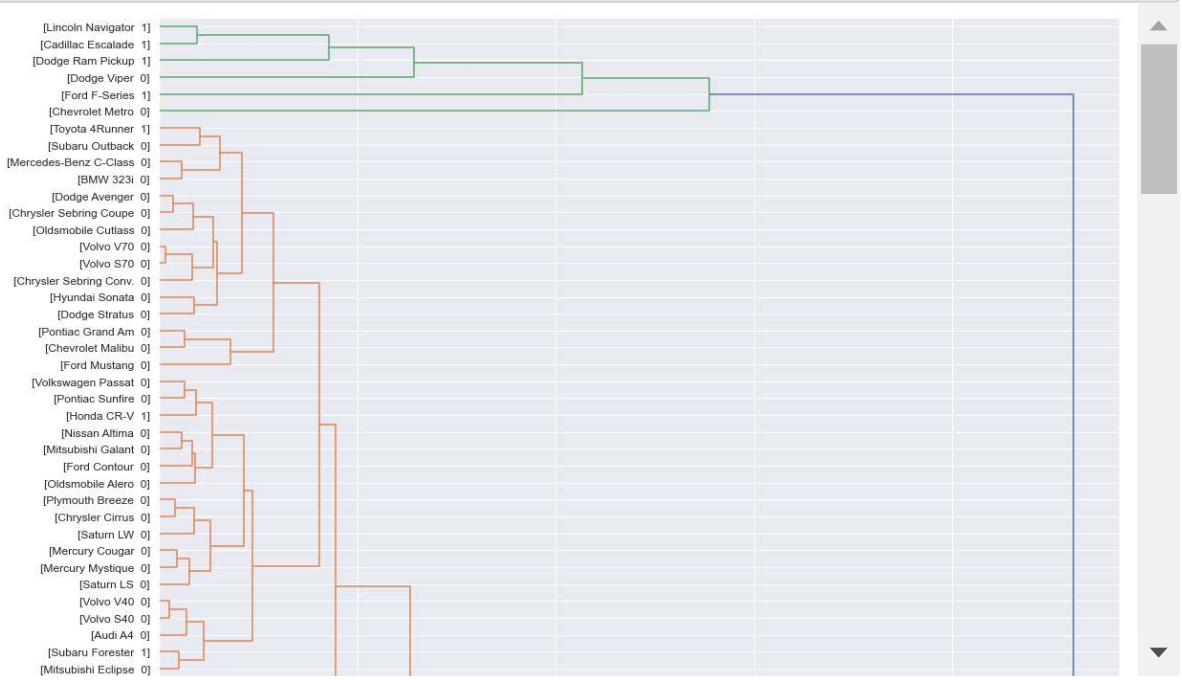
In [27]:
```
z=sch.linkage(distance,"complete")
len(z)
```

```
C:\Users\muham\AppData\Local\Temp\ipykernel_23440\3041572958.py:1: ClusterWar
ning: scipy.cluster: The symmetric non-negative hollow observation matrix loo
ks suspiciously like an uncondensed distance matrix
  z=sch.linkage(distance,"complete")
```

Out[27]: 153

Dendrogram

In [28]:
```
fig = pylab.figure(figsize=(18,50))
def llf(id):
    return f"[{car['manufact'][id]} {car['model'][id]}  {car['type'][id]}]"
dendro = sch.dendrogram(z,  leaf_label_func=llf, leaf_rotation=0, leaf_font_si
```



Modeling the Data

In [29]:
```
agglom = AgglomerativeClustering(n_clusters = 6, linkage = 'complete')
agglom.fit(distance)
agglom.labels_
```

```
C:\Users\muham\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\cluster\_agglomerative.py:544: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix looks suspiciously like an uncondensed distance matrix
  out = hierarchy.linkage(X, method=linkage, metric=affinity)
```

Out[29]:
```
array([4, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 3, 4, 1, 1,
       1, 1, 0, 4, 5, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 3, 3, 0, 0, 0, 0,
       1, 4, 1, 1, 1, 4, 0, 0, 0, 0, 1, 2, 4, 1, 1, 1, 0, 4, 4, 1, 1, 1,
       4, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 3, 4, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 4, 4, 1, 0, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 4, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 4, 4, 4, 1, 1, 1,
       1, 4, 1, 1, 4, 4, 1, 4, 1, 0, 4, 4, 1, 4, 4, 4, 1, 1, 1, 1, 1, 1],
      dtype=int64)
```

In [30]:
```
car['cluster_'] = agglom.labels_
car.head()
```

Out[30]:

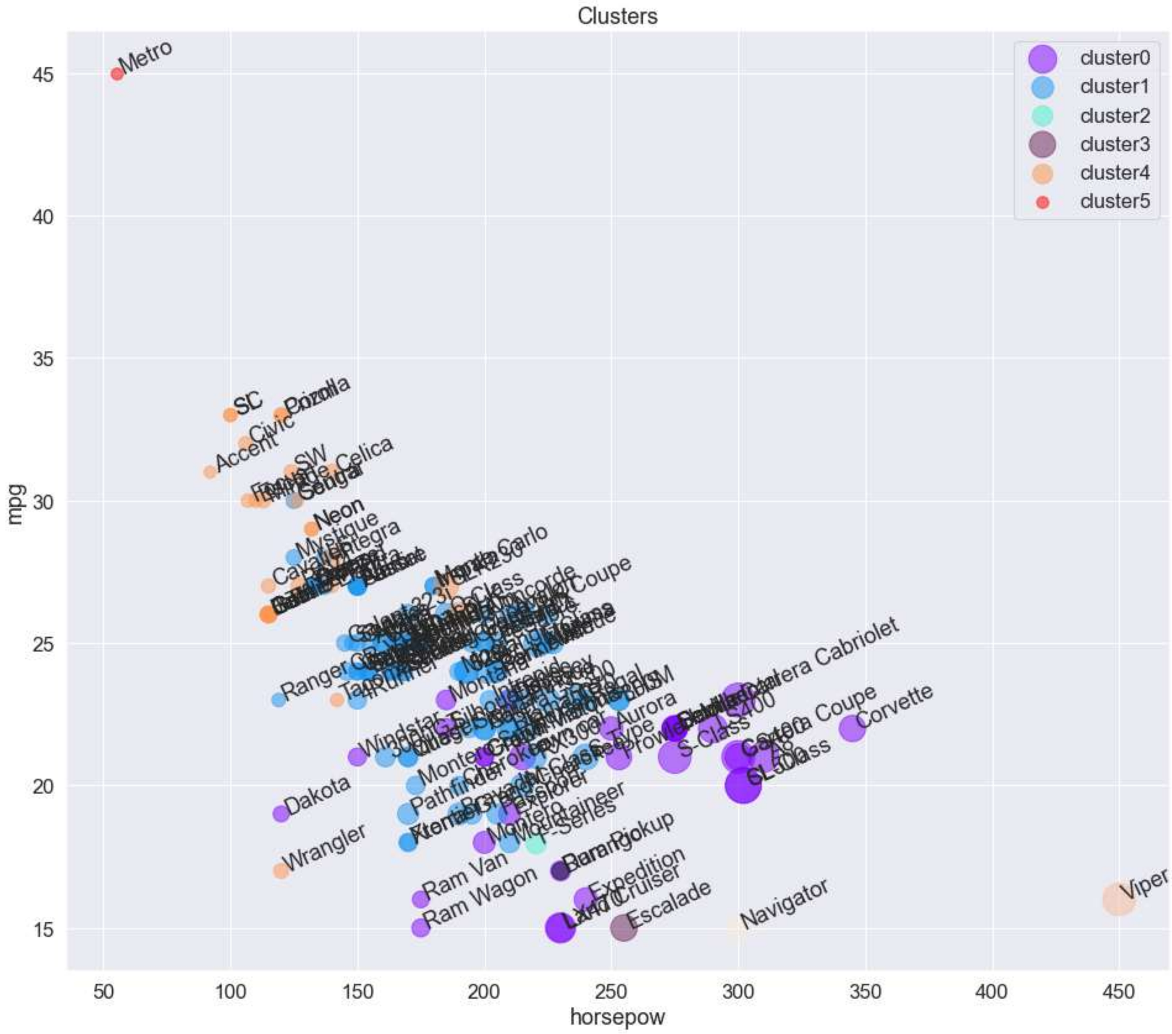
	manufact	model	sales	type	price	engine_s	horsepow	wheelbas	width	leng
0	Acura	Integra	16.921875	0	21.500000	1.799805	140	101.1875	67.3125	172.3
1	Acura	TL	39.375000	0	28.406250	3.199219	225	108.1250	70.3125	192.8
2	Acura	CL	14.117188	0	27.359375	3.199219	225	106.8750	70.6250	192.0
3	Acura	RL	8.585938	0	42.000000	3.500000	210	114.6250	71.3750	196.6
4	Audi	A4	20.390625	0	23.984375	1.799805	150	102.6250	68.1875	178.0

Data Visualization

In [31]:
```
n_clusters = max(agglom.labels_)+1
colors = plt.cm.rainbow(np.linspace(0, 1, n_clusters))
cluster_labels = list(range(0, n_clusters))

plt.figure(figsize=(16,14))

for color, label in zip(colors, cluster_labels):
    subset = car[car.cluster_ == label]
    for i in subset.index:
            plt.text(subset.horsepow[i], subset.mpg[i],str(subset['model'][i])
    plt.scatter(subset.horsepow, subset.mpg, s= subset.price*10, c=color, labe

plt.legend()
plt.title('Clusters')
plt.xlabel('horsepow')
plt.ylabel('mpg')
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should b
e avoided as value-mapping will have precedence in case its length matches wi
th *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array
with a single row if you intend to specify the same RGB or RGBA value for all
points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should b
e avoided as value-mapping will have precedence in case its length matches wi
th *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array
with a single row if you intend to specify the same RGB or RGBA value for all
points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should b
e avoided as value-mapping will have precedence in case its length matches wi
th *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array
with a single row if you intend to specify the same RGB or RGBA value for all
points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should b
e avoided as value-mapping will have precedence in case its length matches wi
th *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array
with a single row if you intend to specify the same RGB or RGBA value for all
points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should b
e avoided as value-mapping will have precedence in case its length matches wi
th *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array
with a single row if you intend to specify the same RGB or RGBA value for all
points.
```

Out[31]:
```
Text(0, 0.5, 'mpg')
```



Clusters
cluster0
cluster1
cluster2
cluster3
cluster4
cluster5
mpg
horsepow
45
40
35
30
25
20
15
50
100
150
200
250
300
350
400
450
Metro
SC
Civic
Accent
SW
Celica
Cougar
Neon
Mystique
Integra
Cavalier
Monte Carlo
Concorde
Ranger
Windstar
Montero
Pathfinder
Dakota
Wrangler
Ram Van
Ram Wagon
Explorer
Mountaineer
F-Series
Durango
Expedition
Land Cruiser
Escalade
Navigator
Aurora
S-Class
Prowler
Cabriolet
Corvette
Viper

Experiment 12: Support Vector Machine

Apply SVM on the Breast Cancer data set. You will use principal component analysis to transform the data to a lower dimensional space

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn import svm
```

Import Dataset

Load the dataset.

In [2]:
```
df=pd.read_csv("Data\\breast-cancer.csv")
df.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_m
0	842302	M	17.99	10.38	122.80	1001.0	0.1
1	842517	M	20.57	17.77	132.90	1326.0	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.1(
3	84348301	M	11.42	20.38	77.58	386.1	0.14
4	84358402	M	20.29	14.34	135.10	1297.0	0.1(

5 rows × 32 columns

Preprocessing

Pre-process and visualize the data.

1. Replace the '?' mark in the 'bare' column by np.nan and change the type to 'float'.
2. Fill any missing data with the median of the column.
3. Drop the ID column.

4. Using Pandas, Matplotlib, seaborn (you can use any or a mix) generate 3-5 plots and add them to your written response explaining what the key insights and findings from the plots are.
5. Separate the features from the class.
6. Split your data into train 80% train and 20% test, use the last two digits of your student number for the seed

In [3]:
```
df.isna().sum()
```

Out[3]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

In [4]:
```
df.drop(["id"],axis=1,inplace=True)
```

In [5]:
```
y=df["diagnosis"]
df.drop("diagnosis",axis=1,inplace=True)
```

In [6]:
```
X_train,X_test,y_train,y_test=train_test_split(df,y,test_size=0.2,random_state
```

Build Classification Models

1. Support vector machine classifier with linear kernel
2. Train an SVM classifier using the training data, set the kernel to linear and set the regularization parameter to C= 0.1. Name the classifier clf_linear_firstname.
3. Print out two accuracy score one for the model on the training set i.e. X_train, y_train and the other on the testing set i.e. X_test, y_test. Record both results in your written response.

Model Creation and Fitting

In [7]:
```
clf_linear_razi=svm.SVC(kernel='linear',C=0.1)
clf_linear_razi.fit(X_train,y_train)
```

Out[7]:
```
SVC(C=0.1, kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [8]:
```
y_train_pred = clf_linear_razi.predict(X_train)
y_test_pred = clf_linear_razi.predict(X_test)
```

Model Evaluation

In [9]:
```
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)
train_accuracy,test_accuracy
```

Out[9]:
```
(0.9692307692307692, 0.9122807017543859)
```