Program 1: Program to multiply two given array of same size element by element

CODE

```
import numpy as np
n1=np.array([[2,3,4],[5,2,3]])
n2=np.array([[1,2,3],[1,2,1]])
print(f"Array 1\n{n1}")
print(f"Array 2\n{n2}")
print(f"Multiplied array{np.multiply(n1,n2)}")
```

OUTPUT

```
Array 1
[[2 3 4]
 [5 2 3]]
Array 2
[[1 2 3]
 [1 2 1]]
Multiplied array[[ 2  6 12]
 [ 5  4  3]]
```

RESULT

The program was successfully implemented using NumPy arrays, and the element-wise multiplication operation produced the expected output.

Program 2: Write a program to compare each element in a array.

CODE

```
import numpy as np
x=np.array([1,2,8,4,5])
y=np.array([6,7,3,9,10])
print(x,"\n",y)
print("x>y\n",np.greater(x,y))
print("x>=y\n",np.greater_equal(x,y))
print("x<y\n",np.less(x,y))
print("x<=y\n",np.less_equal(x,y))
```

OUTPUT

```
[1 2 8 4 5]
 [ 6  7  3  9 10]
x>y
 [False False  True False False]
x>=y
 [False False  True False False]
x<y
 [ True  True False  True  True]
x<=y
 [ True  True False  True  True]
```

RESULT

The program was successfully developed using NumPy comparison functions, and the results for all relational operators were displayed correctly.

Program 3: Program to compute sum of all elements, sum of each column and sum of each row

CODE

```
import numpy as np
x=np.array([[1,0],[0,1]])
print("array is \n",x)
print("Sum of elements\n",np.sum(x))
print("Sum of columns\n",np.sum(x,axis=0))
print("Sum of rows\n",np.sum(x,axis=1))
```

OUTPUT

```
array is
 [[1 0]
 [0 1]]
Sum of elements
 2
Sum of columns
 [1 1]
Sum of rows
 [1 1]
```

RESULT

The program was executed successfully, computing the total sum, row sums, and column sums accurately using NumPy functions.

Program 4: Write a python program to implement list to series conversion**.**

CODE

```
import pandas as pd
name=['a','b','c']
x=pd.Series(name)
print(x)
```

OUTPUT

```
0  a
1  b
2  c
dtype: object
```

RESULT

The program was implemented successfully using the pandas library, and the list was accurately converted into a Series object.

Program 5: Write a program to generate the series of dates from 1st may to 12th may 2021 (both inclusive)

CODE

```
import pandas as pd

s=pd.Series(pd.date_range('2021-05-01','2021-05-12',freq='D'))

print(s.to_string(index=False))
```

OUTPUT

```
2021-05-01
2021-05-02
2021-05-03
2021-05-04
2021-05-05
2021-05-06
2021-05-07
2021-05-08
2021-05-09
2021-05-10
2021-05-11
2021-05-12
```

RESULT

The program was successfully developed using pandas, and the generated date range from May 1 to May 12, 2021, was displayed correctly.

Program 6: Given a 2D list, convert it into corresponding dataframe and display it.

CODE

```
import pandas as pd
details=[[1,2],[3,4]]
df=pd.DataFrame(details)
print(df)
```

OUTPUT

```
   0  1
0  1  2
1  3  4
```

RESULT

The program was successfully executed, converting the 2D list into a structured pandas DataFrame and displaying the expected output.

Program 7: Given a dataframe sort it by multiple columns

CODE

```
import pandas as pd
df=pd.DataFrame({'Name':['John','Alic','Bob','Alic','John'],
                 'Age':[25,30,22,25,20],
                 'Score':[85,90,88,70,95]})
print("Original data frame\n",df)
df_sort=df.sort_values(by=['Name','Age'],ascending=[True,True])
print("Sorted data frame\n",df_sort)
```

OUTPUT

```
Original data frame
   Name  Age  Score
0  John   25     85
1  Alic   30     90
2   Bob   22     88
3  Alic   25     70
4  John   20     95
Sorted data frame
   Name  Age  Score
3  Alic   25     70
1  Alic   30     90
2   Bob   22     88
4  John   20     95
0  John   25     85
```

RESULT

The program was successfully designed and executed, sorting the given DataFrame by multiple columns and displaying the sorted output.

Program 8: Given a dataframe select first two rows and output them.

CODE

```
import pandas as pd
df=pd.DataFrame({'Name':['John','Alic','Bob','Alic','John'],
                 'Age':[25,30,22,25,20],
                 'Score':[85,90,88,70,95]})
print(df.head(2))
```

OUTPUT

```
   Name  Age  Score
0  John   25     85
1  Alic   30     90
```

RESULT

The program was executed successfully, and the first two rows of the DataFrame were retrieved and displayed correctly.

Program 9: Different Types of Plots using Matplotlib

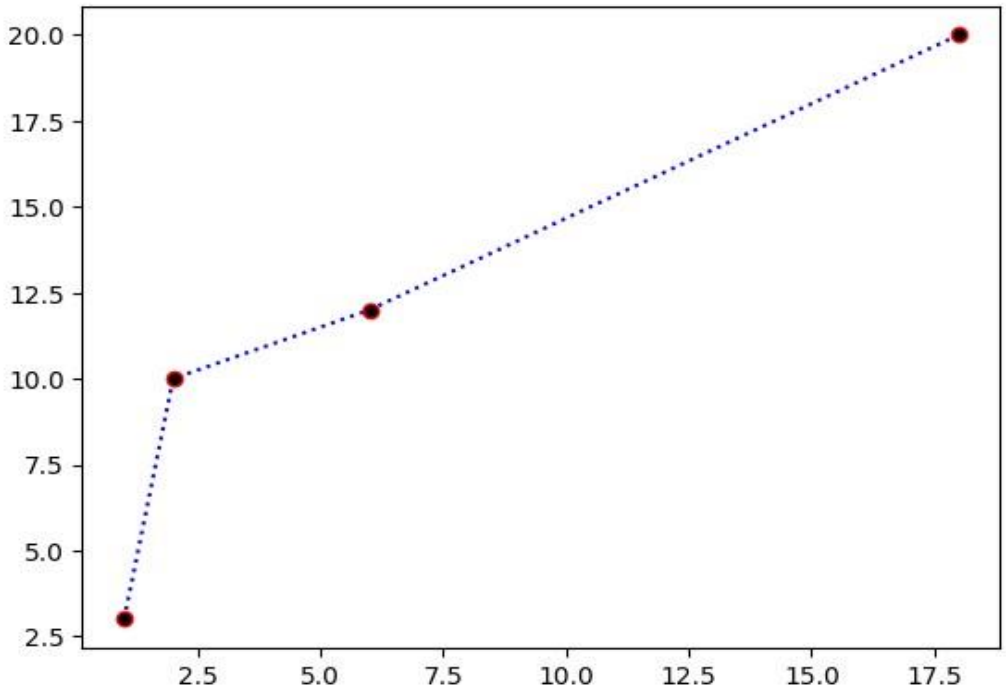
1)Line Plot
2) Bar Plot
3) Histogram
4) Scatter Plot
5) Pie Chart
6) Box Plot
7) Scatter Multiple
8) Bubble Chart
9) Subplots

1. Line Plot

CODE

```
import matplotlib.pyplot as plt
import numpy as np
x=np.array([1,2,6,18])
y=np.array([3,10,12,20])
plt.plot(x,y,marker='o',color='blue',mec='red',mfc='black',linestyle='dotted')
plt.show()
```

OUTPUT

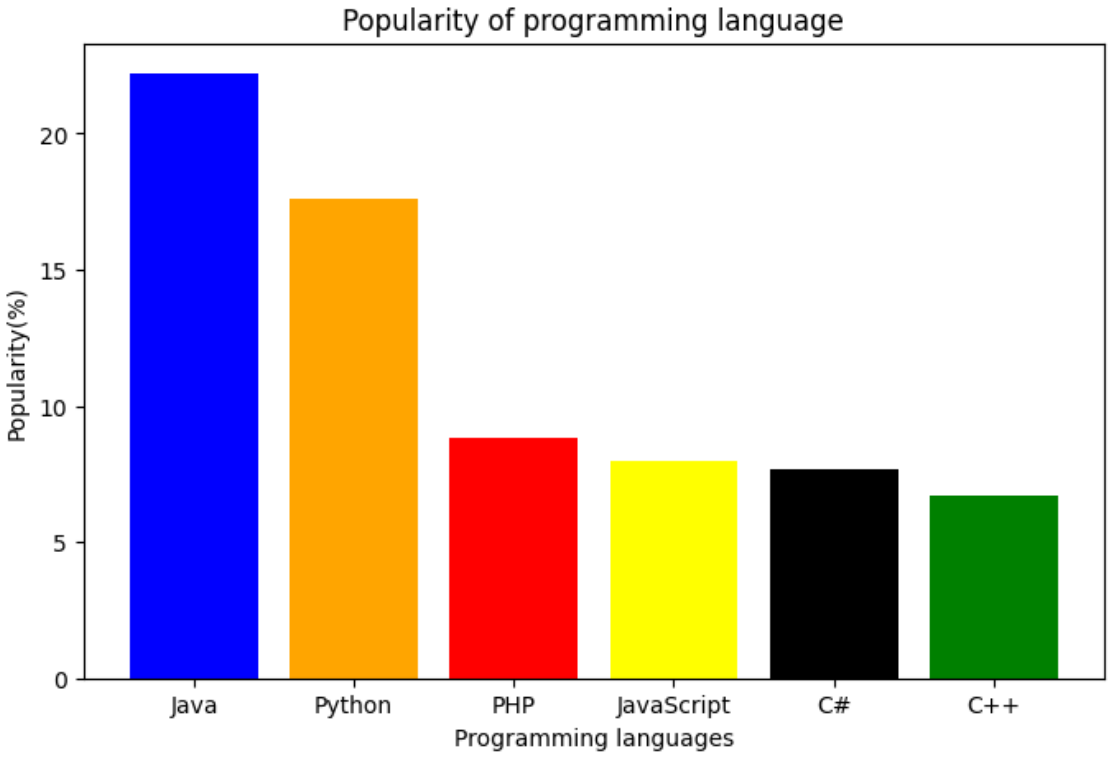


2. Bar Plot

CODE

```
import matplotlib.pyplot as plt
import numpy as np
languages=['Java','Python','PHP','JavaScript','C#','C++']
colors=['blue','orange','red','yellow','black','green']
popularity=[22.2,17.6,8.8,8,7.7,6.7]
plt.figure(figsize=(8,5))
plt.bar(languages,popularity,color=colors)
plt.title('Popularity of programming language')
plt.xlabel('Programming languages')
plt.ylabel('Popularity(%)')
plt.show()
```

OUTPUT

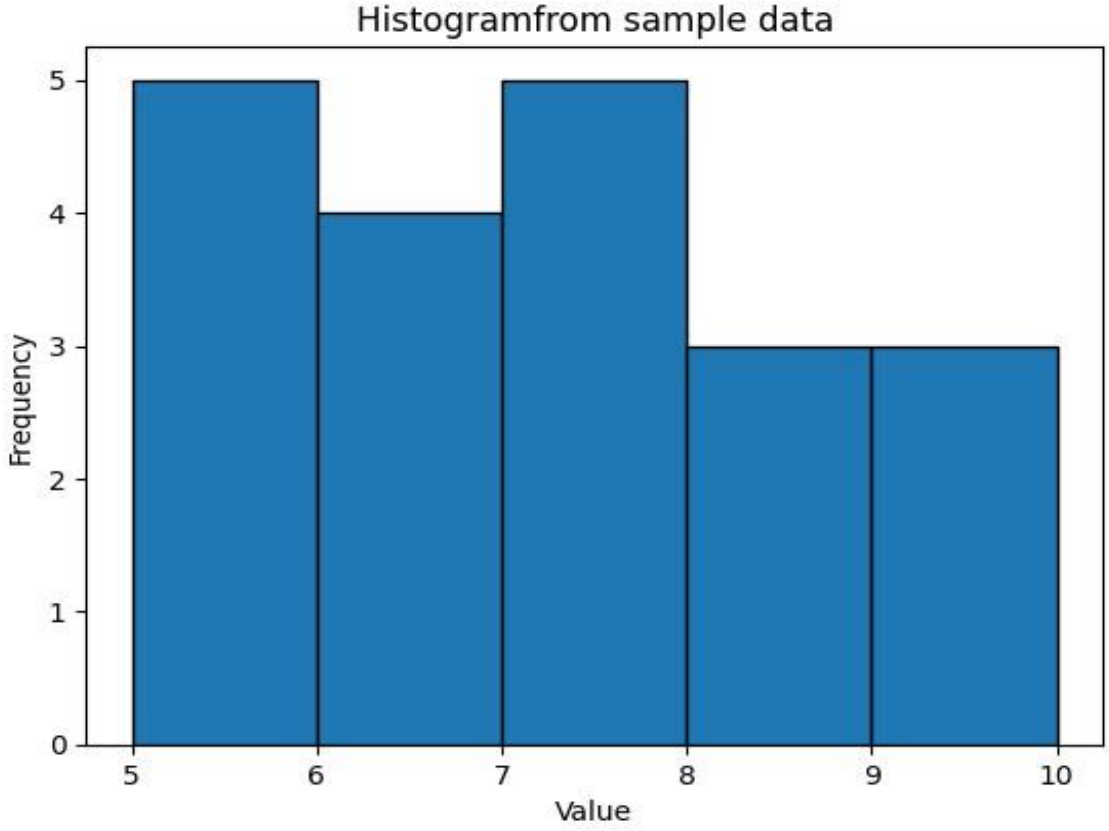


3. Histogram

CODE

```
import matplotlib.pyplot as plt
data=[5,7,8,5,6,7,9,5,6,7,8,7,6,5,7,8,9,10,5,6]
plt.hist(data,bins=5,edgecolor='black')
plt.title("Histogramfrom sample data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

OUTPUT

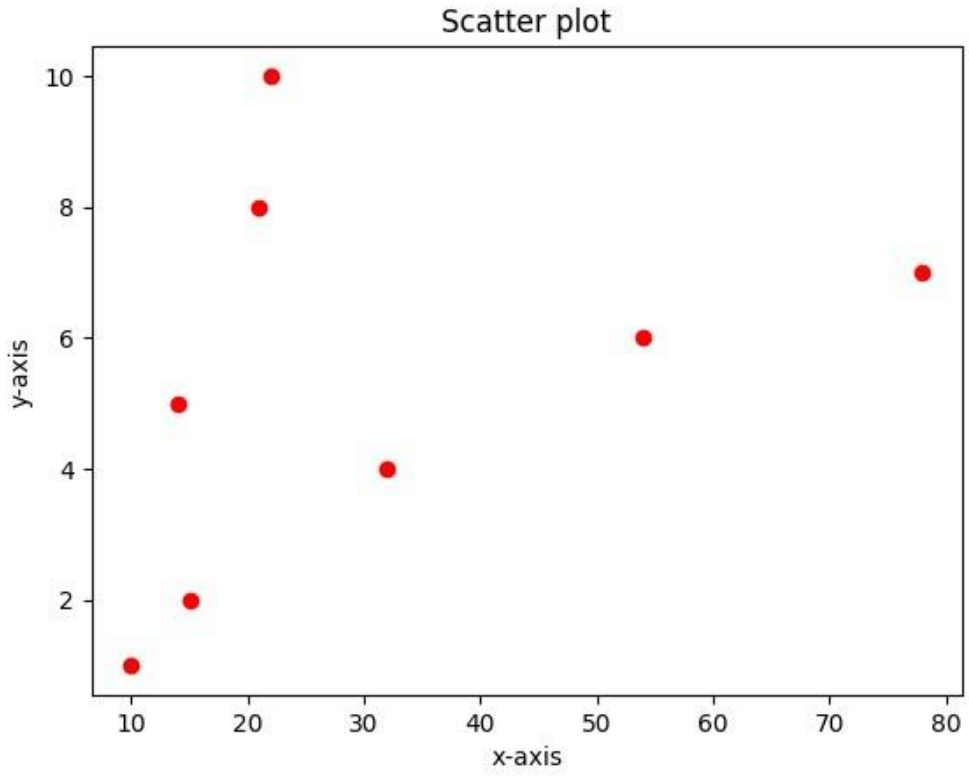


4. Scatter Plot

CODE

```
import matplotlib.pyplot as plt
x=[10,15,14,32,54,78,21,22]
y=[1,2,5,4,6,7,8,10]
plt.scatter(x,y, color='red')
plt.title('Scatter plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```

OUTPUT

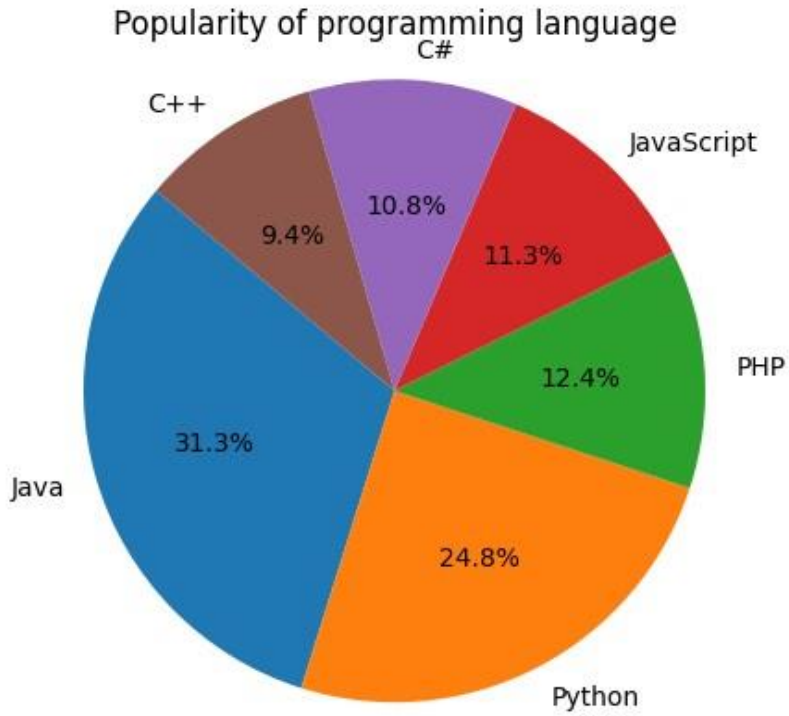


5. Pie Chart

CODE

```
import matplotlib.pyplot as plt
import numpy as np
languages=['Java','Python','PHP','JavaScript','C#','C++']
colors=['blue','orange','red','yellow','black','green']
popularity=[22.2,17.6,8.8,8,7.7,6.7]
plt.pie(popularity,labels=languages,autopct='%1.1f%%',startangle=140)
plt.title('Popularity of programming language')
plt.axis('equal')
plt.show()
```

OUTPUT

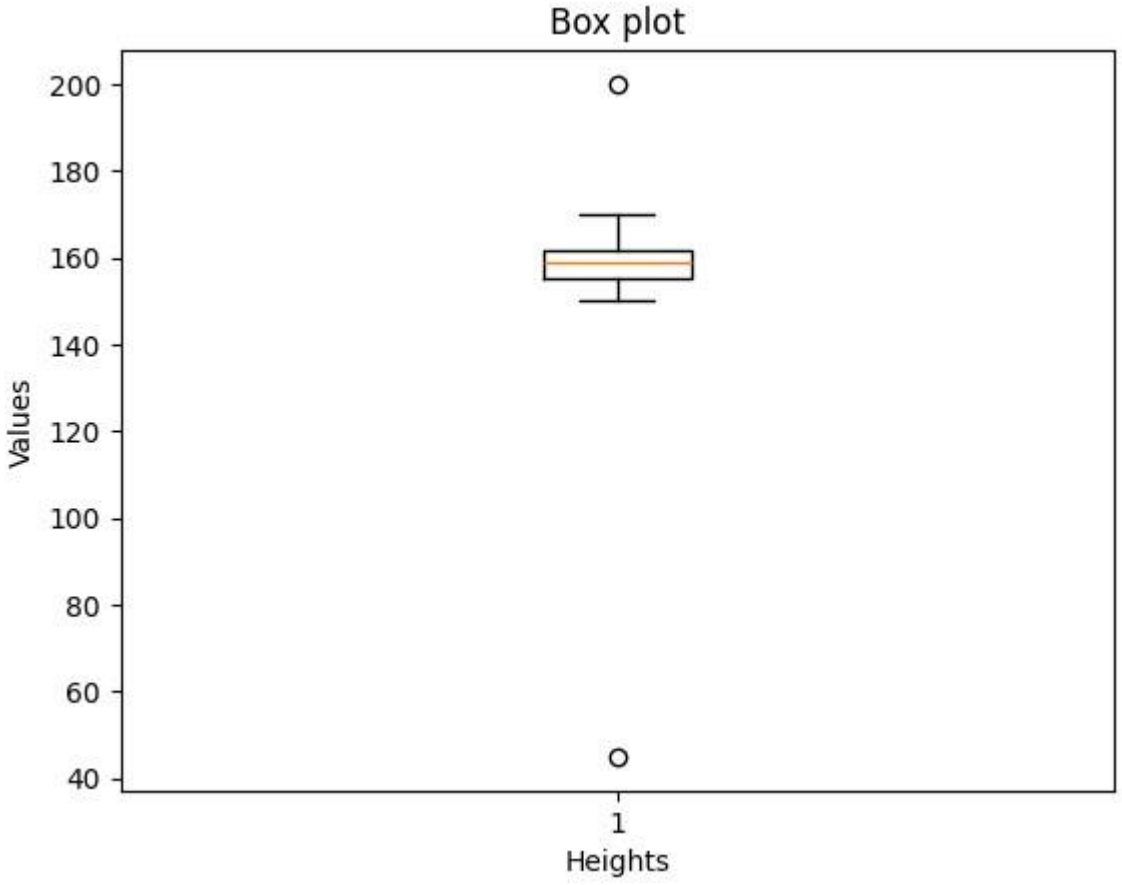


6. Box Plot:

CODE

```
import matplotlib.pyplot as plt
heights=[150,155,160,165,170,155,160,158,160,157,163,151,159,200,45]
plt.boxplot(heights)
plt.title('Box plot')
plt.xlabel('Heights')
plt.ylabel('Values')
plt.show()
```

OUTPUT

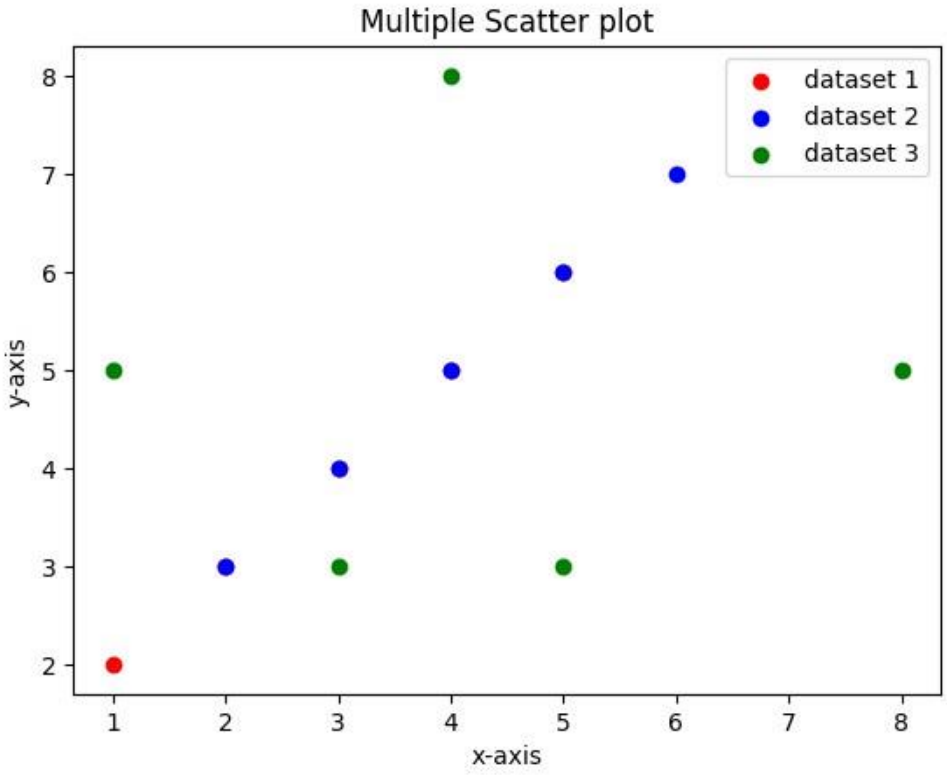


7. Scatter Multiple:

CODE

```
import matplotlib.pyplot as plt
x1=[1,2,3,4,5]
y1=[2,3,4,5,6]
x2=[2,3,4,5,6]
y2=[3,4,5,6,7]
x3=[1,3,4,5,8]
y3=[5,3,8,3,5]
plt.scatter(x1,y1, color='red' ,label='dataset 1')
plt.scatter(x2,y2, color='blue' ,label='dataset 2')
plt.scatter(x3,y3, color='green' ,label='dataset 3')
plt.title('Multiple Scatter plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()
plt.show()
```

OUTPUT

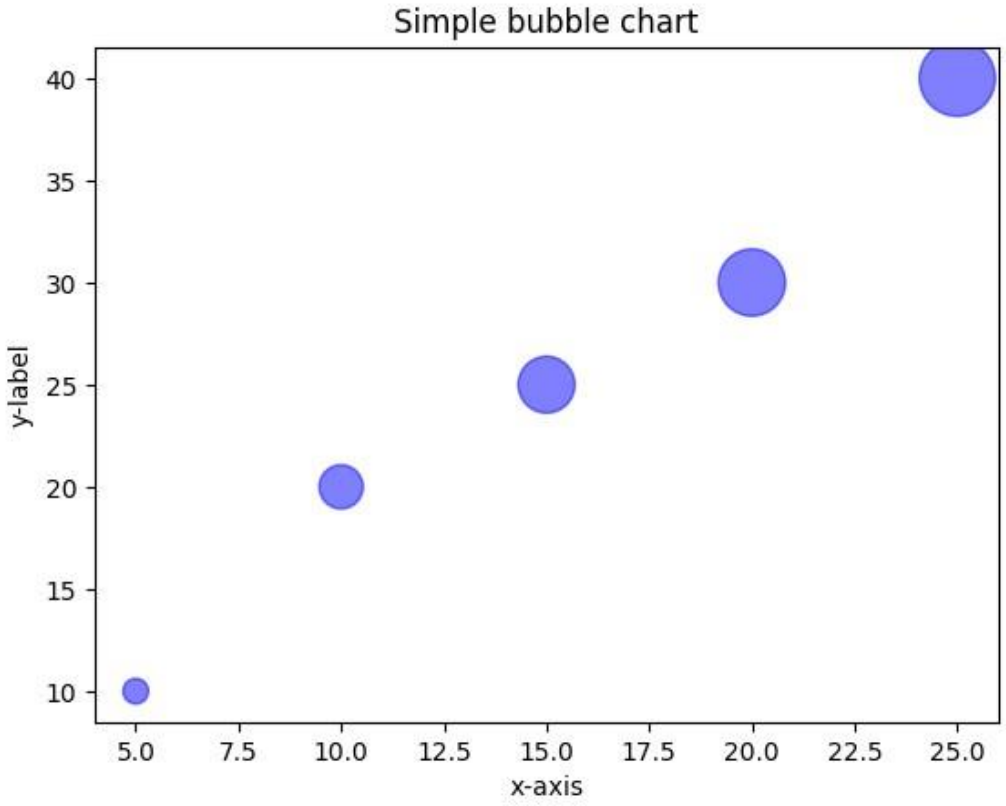


8. Bubble Chart:

CODE

```
import matplotlib.pyplot as plt
x=[5,10,15,20,25]
y=[10,20,25,30,40]
sizes=[100,300,500,700,900]
plt.scatter(x,y, s=sizes, alpha=0.5, color='blue')
plt.title('Simple bubble chart')
plt.xlabel('x-axis')
plt.ylabel('y-label')
plt.show()
```

OUTPUT

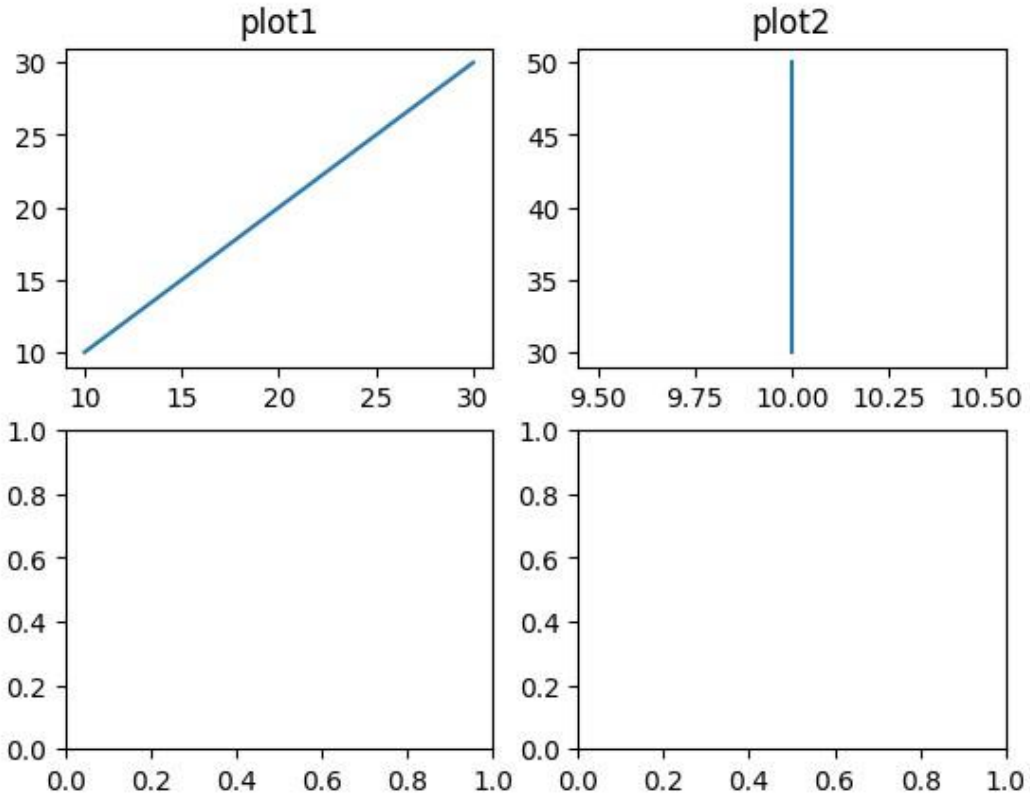


9. Subplots:

CODE

```
import matplotlib.pyplot as plt
figure,axis=plt.subplots(2,2)
x1=[10,20,30]
y1=[10,20,30]
axis[0,0].plot(x1,y1)
axis[0,0].set_title("plot1")
x2=[10,10,10]
y2=[30,40,50]
axis[0,1].plot(x2,y2)
axis[0,1].set_title("plot2")
plt.show()
```

OUTPUT



RESULT

The program was successfully developed using Matplotlib, and various plots such as line, bar, histogram, scatter, and pie charts were displayed as expected.

Program 10: Sarah bought a new car in 2001 for $24000. The dollar value of her car changed each year as shown in the table below.

Value of Sarah's Car

Year	Value
2001	**$24,000**
2002	**$22,500**
2003	**$19,700**
2004	**$17,500**
2005	**$14,500**
2006	**$10,000**
2007	**$5,800**

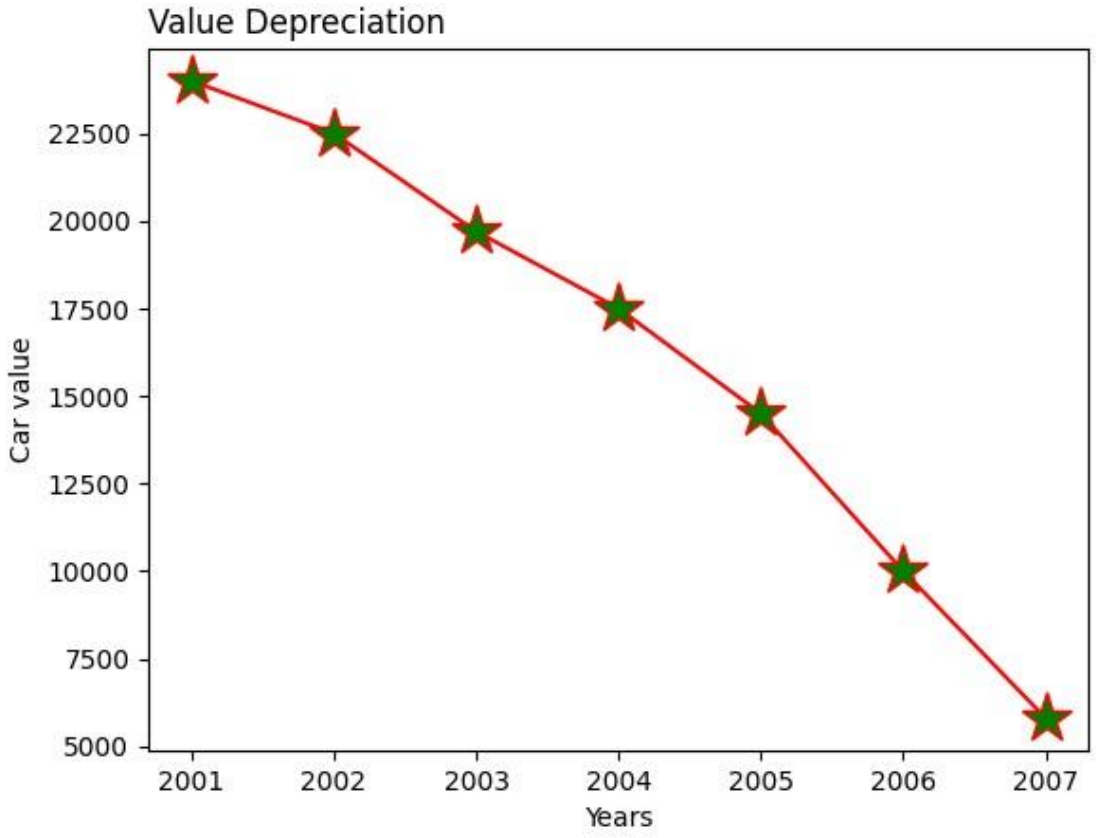
represent the following information using a line graph with following style properties

- X-axis – year
- Y-axis – car value
- Title – value depreciation (left aligned)
- Line style dash dot & line color should be red
- Point using * symbol with green color & size 20

CODE

```
import matplotlib.pyplot as plt
years=[2001,2002,2003,2004,2005,2006,2007]
value=[24000,22500,19700,17500,14500,10000,5800]
plt.plot(years,value,marker='*', linestyle='-',color='red',markersize=20, mfc='green')
plt.title("Value Depreciation",loc="left")
plt.xlabel('Years')
plt.ylabel('Car value')
plt.show()
```

OUTPUT



RESULT

The program was successfully implemented to plot car value depreciation using Matplotlib, and the styled line chart was generated correctly.

Program 11: Create scatter plot for the below data (use scatter function)

Product	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Affordable Segment	173	153	195	147	120	144	148	109	174	130	172	131
Luxury Segment	189	189	105	112	173	109	151	197	174	145	177	161
Super Luxury Segment	185	185	126	134	196	153	112	133	200	145	167	110

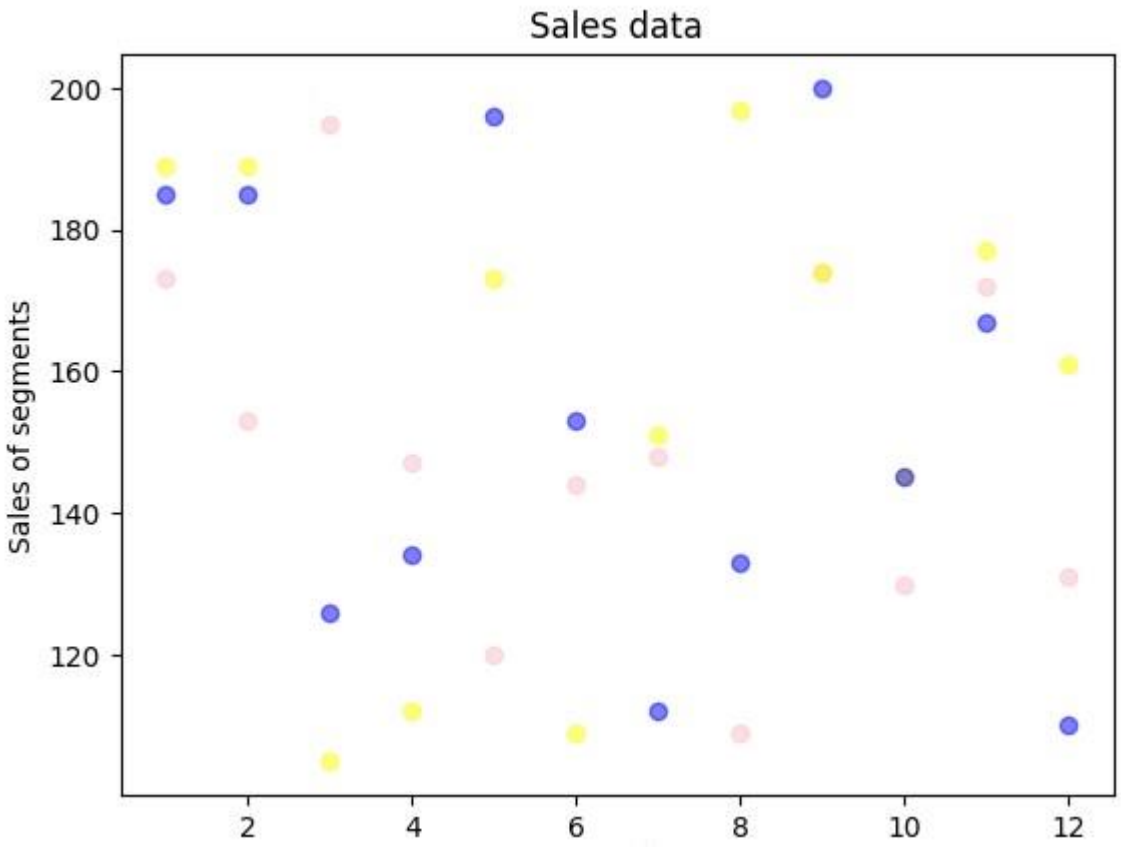
Create scatter plot for each segment with following properties within one graph

- X-axis – months of year with font size 18
- Y-axis – sales of segments
- Title – sales data
- Color for affordable segment- pink
- Color for luxury segment – yellow
- Color for super luxury segment – blue

CODE

```
import matplotlib.pyplot as plt
y1=[173,153,195,147,120,144,148,109,174,130,172,131]
y2=[189,189,105,112,173,109,151,197,174,145,177,161]
y3=[185,185,126,134,196,153,112,133,200,145,167,110]
x=[1,2,3,4,5,6,7,8,9,10,11,12]

plt.scatter(x,y1, label='Affordable segment',color='pink',alpha=0.5)
plt.scatter(x,y2,label='Luxury segment',color='yellow',alpha=0.5)
plt.scatter(x,y3, label='Super luxury segmrnt',color='blue',alpha=0.5)
plt.title("Sales data")
plt.xlabel('Mont of Year', fontsize=18)
plt.ylabel('Sales of segments')
plt.show()
```

OUTPUT



RESULT

The program was executed successfully, and the scatter plot displaying sales data for different segments was plotted accurately.

Program 12: Following table gives the daily sales of the following items in a shop.

Day	Mon	Tues	Wed	Thurs	Fri
Drinks	300	450	150	400	650
Food	400	500	350	300	500

Use subplot function to draw the line graphs with grids (color as blue & line style dotted) for the above information as 2 separate graphs in 2 rows

a) Properties for the graph1:

X label – days of week

Y label – Sale of drinks

Title – sales data1 (right aligned)

Line – dotted with cyan color

Points – hexagon shape with color magenta & outline black

b) Properties for the graph2:

X label – days of week

Y label – Sale of food

Title – sales data2 (center aligned)

Line – dashed with yellow color

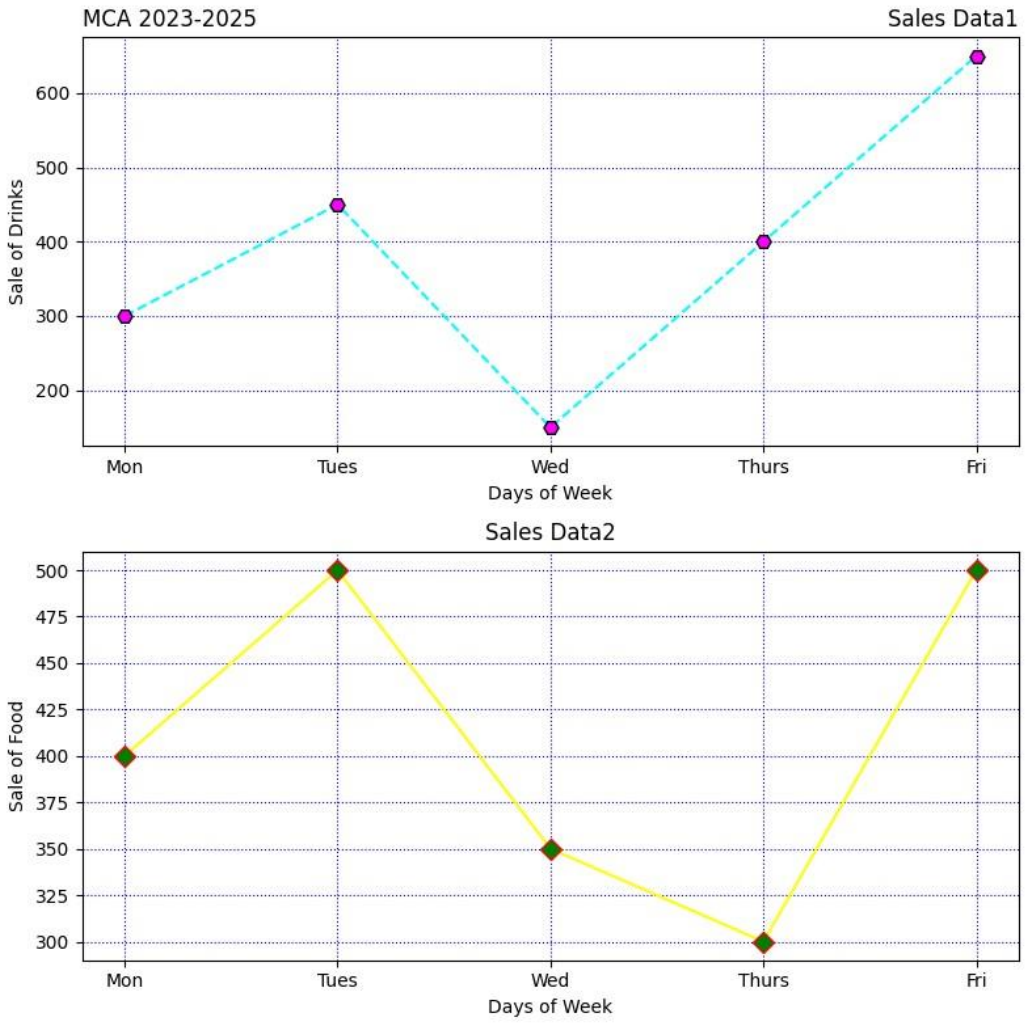
Points – diamond shape with color green & outline red

CODE

```
import matplotlib.pyplot as plt

days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri']

drinks_sales = [300, 450, 150, 400, 650]

food_sales = [400, 500, 350, 300, 500]

fig, axs = plt.subplots(2, 1, figsize=(8, 8))

axs[0].plot(days, drinks_sales, linestyle='--', color='cyan', marker='H',
markersize=8,markerfacecolor='magenta', markeredgecolor='black')

axs[0].set_xlabel('Days of Week')

axs[0].set_ylabel('Sale of Drinks')

axs[0].set_title('Sales Data1', loc='right')

axs[0].set_title('MCA 2023-2025', loc='left')
```

```
axs[0].grid(True, color='blue', linestyle='dotted')
axs[1].plot(days, food_sales, linestyle='-', color='yellow', marker='D', markersize=8,
markerfacecolor='green', markeredgecolor='red')
axs[1].set_xlabel('Days of Week')
axs[1].set_ylabel('Sale of Food')
axs[1].set_title('Sales Data2', loc='center')
axs[1].grid(True, color='blue', linestyle='dotted')
plt.tight_layout()
plt.show()
```

OUTPUT



RESULT

The program was successfully developed using Matplotlib subplots, and both line graphs were displayed with proper grid and styling properties..

Program 13: Implement the k-NN classification algorithm using the Dataset: <Breast_Cancer.csv>

a) Conduct exploratory data analysis on the given dataset and report the details.

b) Visualize the analysis results using

(i) scatter plot (ii) histogram & (iii) box plot

c) Try with different k values and show the accuracy

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
print(df.head(5))
df.info()
print(df.describe)
print(df.isnull().sum())
print(df.describe())
print(df['target'].value_counts())
plt.figure(figsize=(8, 6))
sns.scatterplot(x='mean radius', y='mean texture', hue='target', data=df)
plt.title("Scatter plot of Mean Radius vs Mean Texture")
plt.show()
```

```
plt.figure(figsize=(8, 6))
df['mean radius'].hist(bins=30)
plt.title("Histogram of Mean Radius")
plt.xlabel("Mean Radius")
plt.ylabel("Frequency")
plt.show()
plt.figure(figsize=(8, 6))
sns.boxplot(x='target', y='mean radius', data=df)
plt.title("Box plot of Mean Radius grouped by Target")
plt.show()



X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


k_values = [3, 5, 7, 9]
for k in k_values:
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
  y_pred = knn.predict(X_test)
  accuracy = metrics.accuracy_score(y_test, y_pred)
  print(f"Accuracy with k={k}: {accuracy:.4f}")
```

OUTPUT

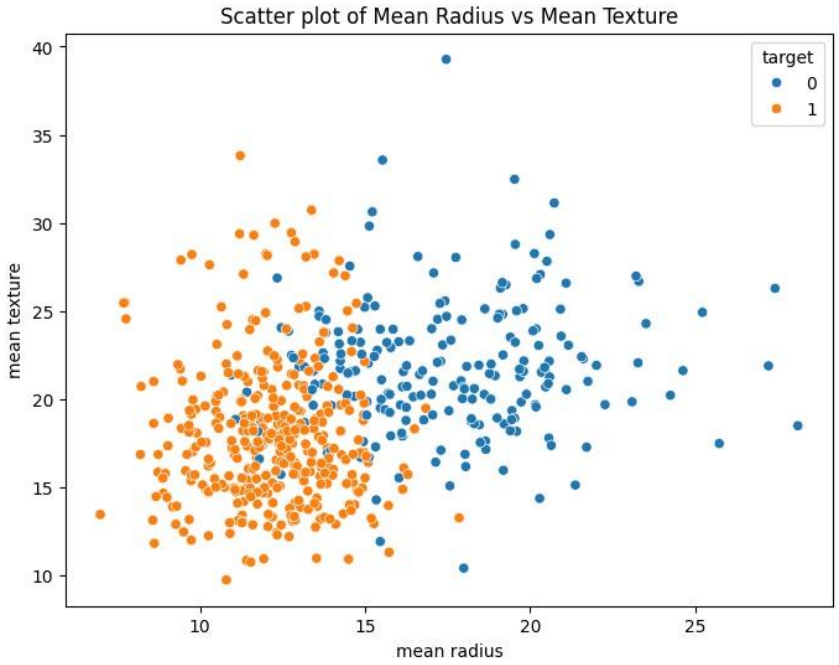
```
   mean radius  mean texture  mean perimeter  mean area  mean smoothnes
s  \
0        17.99         10.38          122.80     1001.0          0.1184
0
1        20.57         17.77          132.90     1326.0          0.0847
4
2        19.69         21.25          130.00     1203.0          0.1096
0
3        11.42         20.38           77.58      386.1          0.1425
0
4        20.29         14.34          135.10     1297.0          0.1003
0
```

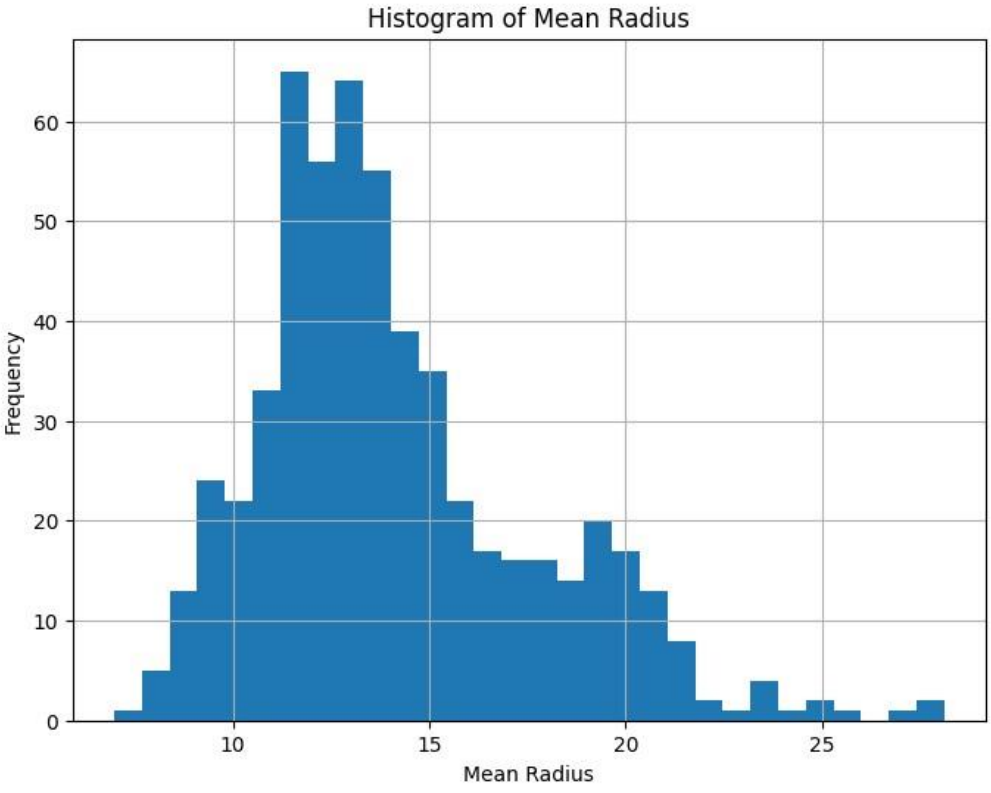
```
   mean compactness  mean concavity  mean concave points  mean symmetry
\
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst a
rea  \
0                 0.07871  ...          17.33           184.60      201
9.0
1                 0.05667  ...          23.41           158.80      195
6.0
2                 0.05999  ...          25.53           152.50      170
9.0
3                 0.09744  ...          26.50            98.87       56
7.7
4                 0.05883  ...          16.67           152.20      157
5.0

   worst smoothness  worst compactness  worst concavity  worst concave
points  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
```

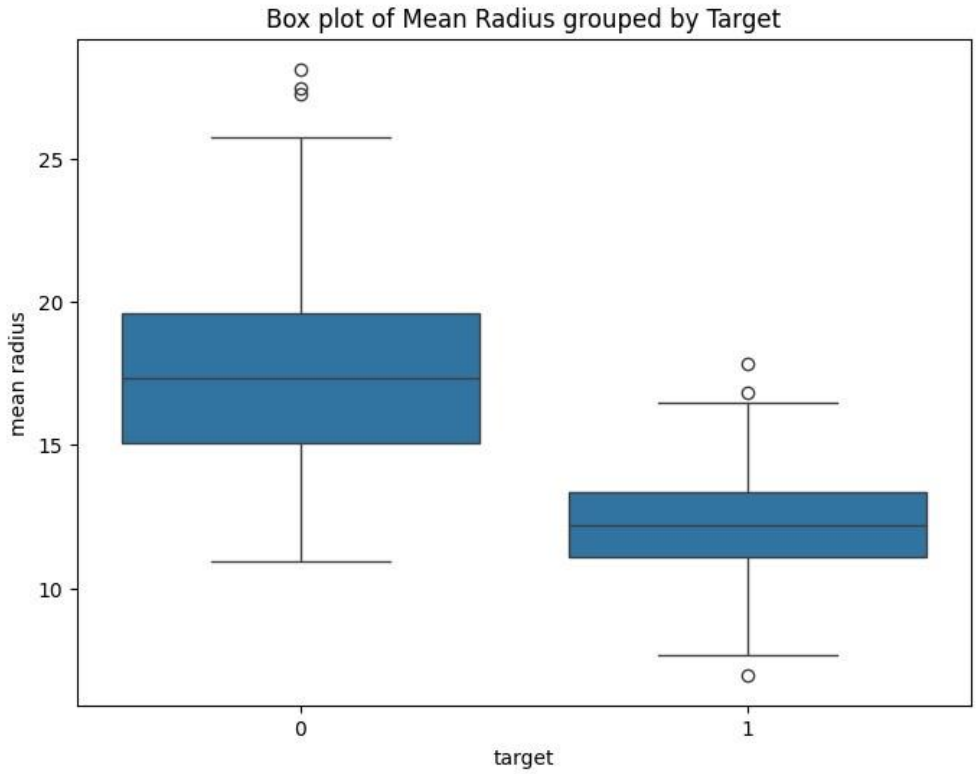
```
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB

[569 rows x 31 columns]>
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
       mean radius  mean texture  mean perimeter    mean area  \
```

```
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      28.110000     39.280000      188.500000  2501.000000

       mean smoothness  mean compactness  mean concavity  mean concave
points  \
count       569.000000        569.000000      569.000000           569.
000000
mean          0.096360          0.104341        0.088799             0.
048919
std           0.014064          0.052813        0.079720             0.
038803
min           0.052630          0.019380        0.000000             0.
000000
25%           0.086370          0.064920        0.029560             0.
020310
50%           0.095870          0.092630        0.061540             0.
033500
75%           0.105300          0.130400        0.130700             0.
074000
max           0.163400          0.345400        0.426800             0.
201200

       worst fractal dimension      target
count                569.000000  569.000000
mean                   0.083946    0.627417
std                    0.018061    0.483918
min                    0.055040    0.000000
25%                    0.071460    0.000000
50%                    0.080040    1.000000
75%                    0.092080    1.000000
max                    0.207500    1.000000

[8 rows x 31 columns]
target
1    357
0    212
```

```
Name: count, dtype: int64
```








```
Accuracy with k=3: 0.9415
Accuracy with k=5: 0.9591
Accuracy with k=7: 0.9649
Accuracy with k=9: 0.9708
```

RESULT

The program was implemented and tested successfully using the breast cancer dataset, and the EDA, visualizations, and accuracy results were generated as expected.

Program 14: Implement the k-NN classification algorithm using the Dataset: <Wine_Quality.csv>

a) Conduct exploratory data analysis on the given dataset and report the details.

b) Visualize the analysis results using (i) scatter plot (ii) histogram & (iii) box plot.

c) Try with different K values and show the accuracy.

CODE

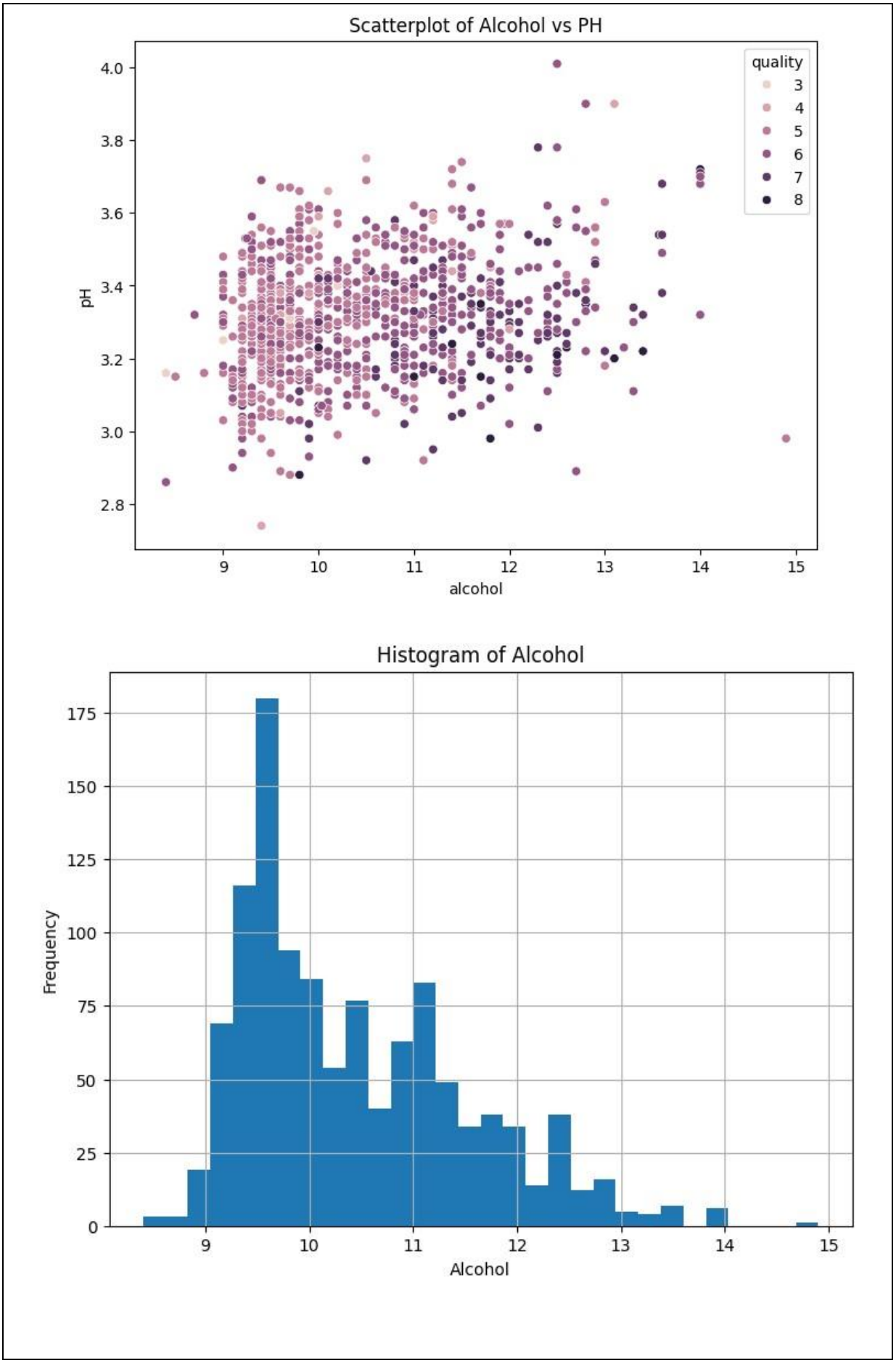
```
import pandas as pd
df=pd.read_csv('WineQT.csv')
print(df.head())
print(df.isnull())
print(df.describe())
print(df['quality'].value_counts())

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8,6))
sns.scatterplot(x='alcohol',y='pH',hue='quality',data=df)
plt.title('Scatterplot of Alcohol vs PH')
plt.show()

plt.figure(figsize=(8,6))
df['alcohol'].hist(bins=30)
plt.title('Histogram of Alcohol')
plt.xlabel('Alcohol')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(8,6))
sns.boxplot(x='quality',y='alcohol',data=df)
plt.title('Boxplot of alcohol grouped by Quality')
```

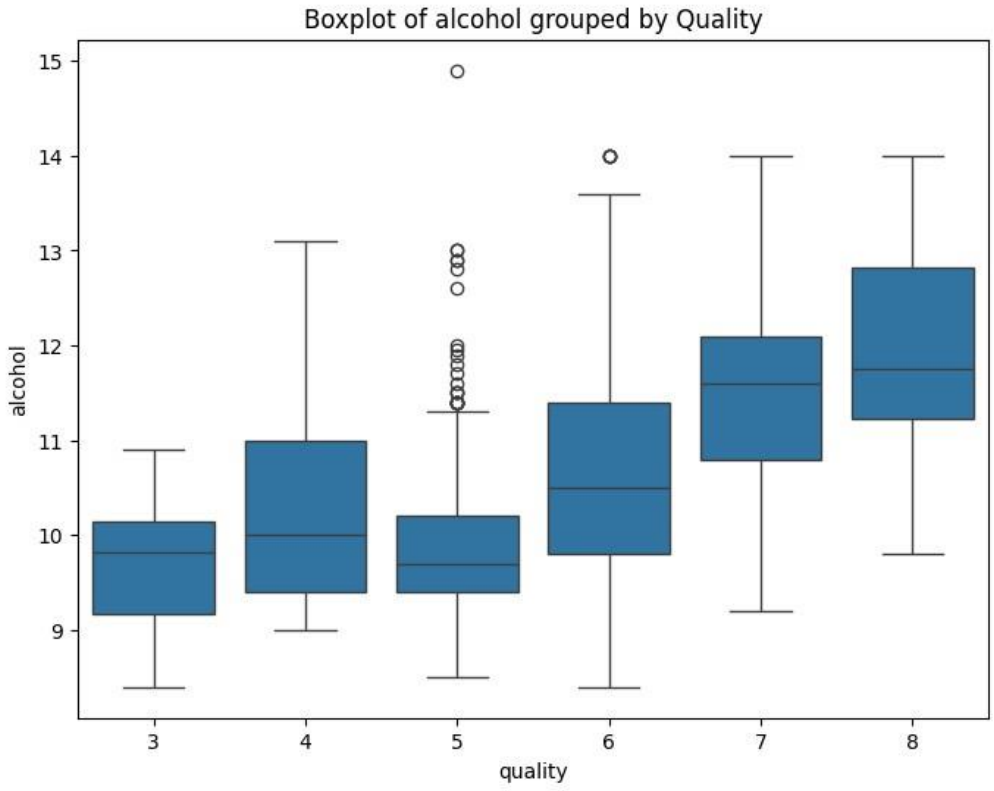
```
plt.show()


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score


df_cleaned=df.dropna()
x=df_cleaned.drop(['quality'],axis=1)
y=df_cleaned['quality']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
scaler=StandardScaler()
x_train_scale=scaler.fit_transform(x_train)
x_test_scaled=scaler.transform(x_test)
k_value=[1,3,5,7,9,11,13,15]
for k in k_value:
  knn=KNeighborsClassifier(n_neighbors=k)
  knn.fit(x_train_scale,y_train)
  y_pred=knn.predict(x_test_scaled)
  accuracy=accuracy_score(y_test,y_pred)
  print(f"Accuracy with k={k}:{accuracy:4f}")
```

OUTPUT

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlori
des  \
0            7.4              0.70         0.00             1.9      0.
076
1            7.8              0.88         0.00             2.6      0.
098
2            7.8              0.76         0.04             2.3      0.
092
3           11.2              0.28         0.56             1.9      0.
075
4            7.4              0.70         0.00             1.9      0.
076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
```

```
0                11.0                 34.0   0.9978  3.51       0.56
1                25.0                 67.0   0.9968  3.20       0.68
2                15.0                 54.0   0.9970  3.26       0.65
3                17.0                 60.0   0.9980  3.16       0.58
4                11.0                 34.0   0.9978  3.51       0.56

   alcohol  quality  Id
0      9.4        5   0
1      9.8        5   1
2      9.8        5   2
3      9.8        6   3
4      9.4        5   4
      fixed acidity  volatile acidity  citric acid  residual sugar  chl
orides  \
0             False             False        False           False
False
1             False             False        False           False
False
2             False             False        False           False
False
3             False             False        False           False
False
4             False             False        False           False
False
...             ...               ...          ...             ...
...
1138          False             False        False           False
False
1139          False             False        False           False
False
1140          False             False        False           False
False
1141          False             False        False           False
False
1142          False             False        False           False
False

                pH    sulphates      alcohol      quality           Id
count  1143.000000  1143.000000  1143.000000  1143.000000  1143.000000
mean      3.311015     0.657708    10.442111     5.657043   804.969379
std       0.156664     0.170399     1.082196     0.805824   463.997116
min       2.740000     0.330000     8.400000     3.000000     0.000000
25%       3.205000     0.550000     9.500000     5.000000   411.000000
50%       3.310000     0.620000    10.200000     6.000000   794.000000
75%       3.400000     0.730000    11.100000     6.000000  1209.500000
max       4.010000     2.000000    14.900000     8.000000  1597.000000
quality
5    483
6    462
7    143
4     33
8     16
3      6
Name: count, dtype: int64
```


Scatterplot of Alcohol vs PH
quality
3
4
5
6
7
8
pH
alcohol
4.0
3.8
3.6
3.4
3.2
3.0
2.8
9
10
11
12
13
14
15
Histogram of Alcohol
Frequency
Alcohol
175
150
125
100
75
50
25
0




```
Accuracy with k=1:0.572052
Accuracy with k=3:0.528384
Accuracy with k=5:0.558952
Accuracy with k=7:0.558952
Accuracy with k=9:0.624454
Accuracy with k=11:0.606987
Accuracy with k=13:0.620087
Accuracy with k=15:0.615721
```

RESULT

The program was successfully executed on the wine quality dataset, performing data analysis, visualization, and k-value accuracy testing accurately.

Program 15: Implement the Naïve Bayes classification algorithm using the Dataset:<Breast_Cancer.csv>

a) Conduct exploratory data analysis on the given dataset and report the details.

b) Visualize the analysis results using (i) scatter plot (ii) histogram & (iii) box plot.

c) Display the classification report with the accuracy.

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score
from sklearn import metrics

data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

print(df.head())
print(df.isnull().sum())
print(df.describe())
if 'target' in df.columns:
  print(df['target'].value_counts())

plt.figure(figsize=(8, 6))
df['mean radius'].hist(bins=30)
plt.title("Histogram of Mean Radius")
plt.xlabel("Mean Radius")
```

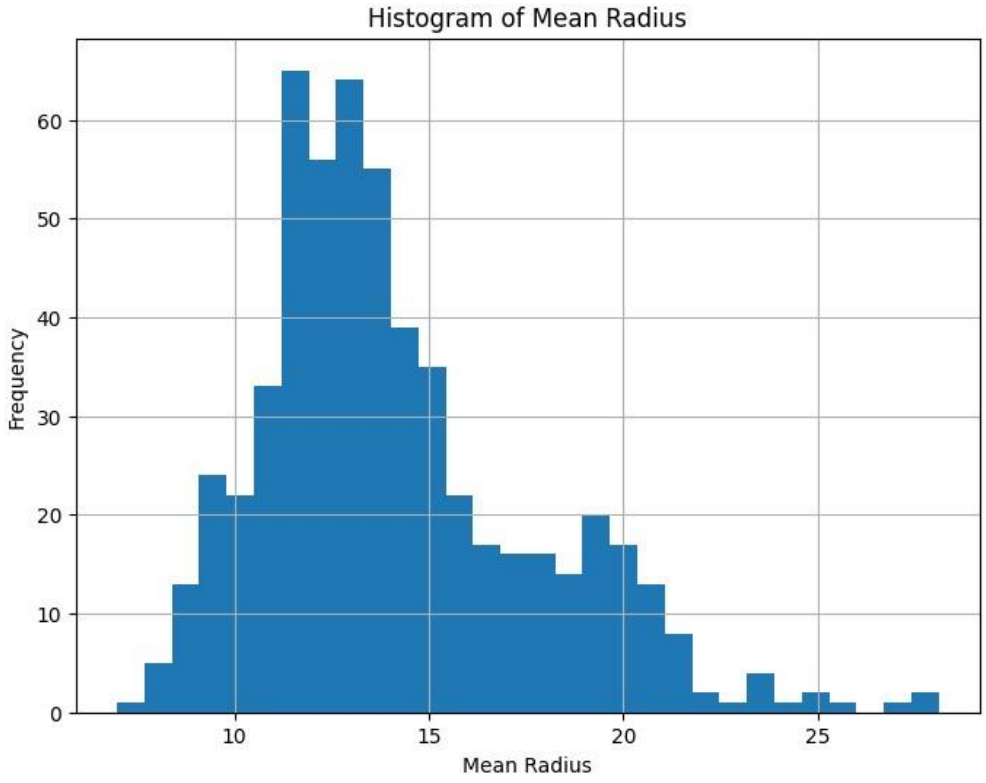
```
plt.ylabel("Frequency")
plt.show()


plt.figure(figsize=(8, 6))
sns.boxplot(x='target', y='mean radius', data=df)
plt.title("Box plot of Radius Mean grouped by Target")
plt.show()



df_cleaned = df.dropna()
X = df_cleaned.drop(['target'], axis=1)
y = df_cleaned['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred))
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy {accuracy:4f}")
```

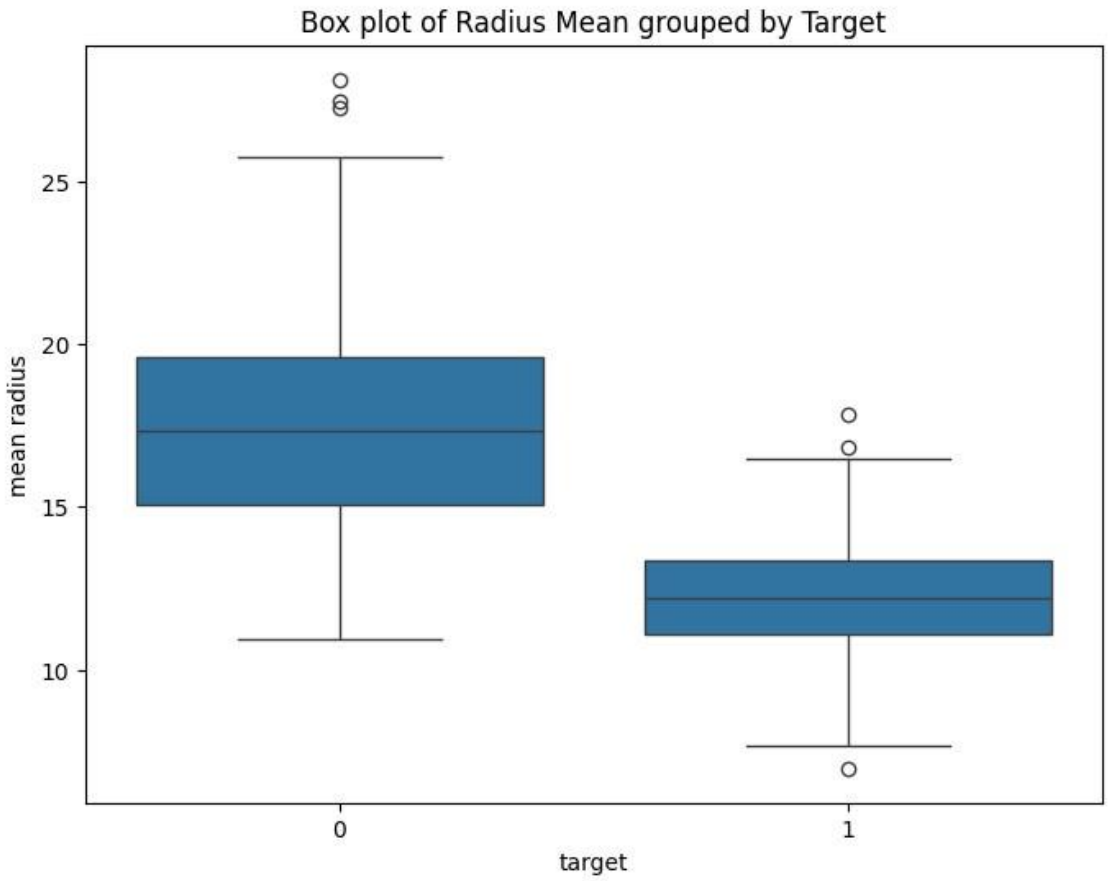
OUTPUT

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothnes
s  \
0        17.99         10.38          122.80     1001.0          0.1184
0
1        20.57         17.77          132.90     1326.0          0.0847
4
2        19.69         21.25          130.00     1203.0          0.1096
0
3        11.42         20.38           77.58      386.1          0.1425
0
4        20.29         14.34          135.10     1297.0          0.1003
0

   mean compactness  mean concavity  mean concave points  mean symmetry
\
```

```
0           0.27760          0.3001            0.14710         0.2419
1           0.07864          0.0869            0.07017         0.1812
2           0.15990          0.1974            0.12790         0.2069
3           0.28390          0.2414            0.10520         0.2597
4           0.13280          0.1980            0.10430         0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst a
rea  \
0                 0.07871  ...          17.33           184.60      201
9.0
1                 0.05667  ...          23.41           158.80      195
6.0
2                 0.05999  ...          25.53           152.50      170
9.0
3                 0.09744  ...          26.50            98.87       56
7.7
4                 0.05883  ...          16.67           152.20      157
5.0

   worst smoothness  worst compactness  worst concavity  worst concave
points  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
```

```
symmetry error              0
fractal dimension error     0
worst radius                0
worst texture               0
worst perimeter             0
worst area                  0
worst smoothness            0
worst compactness           0
worst concavity             0
worst concave points        0
worst symmetry              0
worst fractal dimension     0
target                      0
dtype: int64

[8 rows x 31 columns]
target
1    357
0    212
Name: count, dtype: int64
```






```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        43
           1       0.96      1.00      0.98        71

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Accuracy 0.973684
```

RESULT

The program was successfully implemented using the Naïve Bayes algorithm, and the classification report and accuracy results were displayed correctly.

Program 16: Implement the Naïve Bayes classification algorithm using the Dataset: <Wine_Quality.csv>

a) Conduct exploratory data analysis on the given dataset and report the details.

b) Visualize the analysis results using (i) histogram and (ii) box plot.

c) Display the classification report with the accuracy.

CODE

```
import pandas as pd
df = pd.read_csv('WineQT.csv')
print(df.head())
print(df.isnull().sum())
print(df.describe())
print(df['quality'].value_counts())

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 6))
df['alcohol'].hist(bins=30)
plt.title("Histogram of Alcohol")
plt.xlabel("Alcohol")
plt.ylabel("Frequency")
plt.show()
plt.figure(figsize=(8, 6))
sns.boxplot(x='quality', y='alcohol', data=df)
plt.title("Box plot of Alcohol grouped by Quality")
plt.show()

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score
from sklearn.impute import SimpleImputer
df['quality_label'] = df['quality'].apply(lambda x: 1 if x >= 7 else 0)
```

```
X = df.drop(columns=[ 'quality', 'quality_label'])
y = df['quality_label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred = nb_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print("Accuracy: {:2f}%".format(accuracy * 100))
print("\nClassification Report:\n", classification_rep)
```

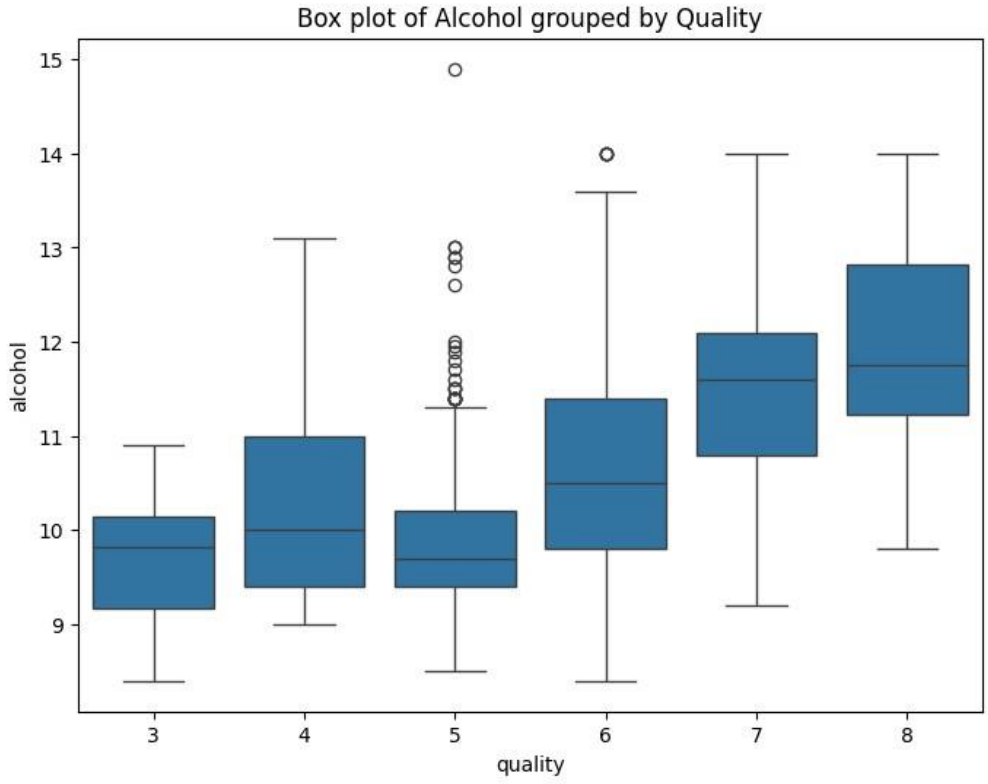
OUTPUT

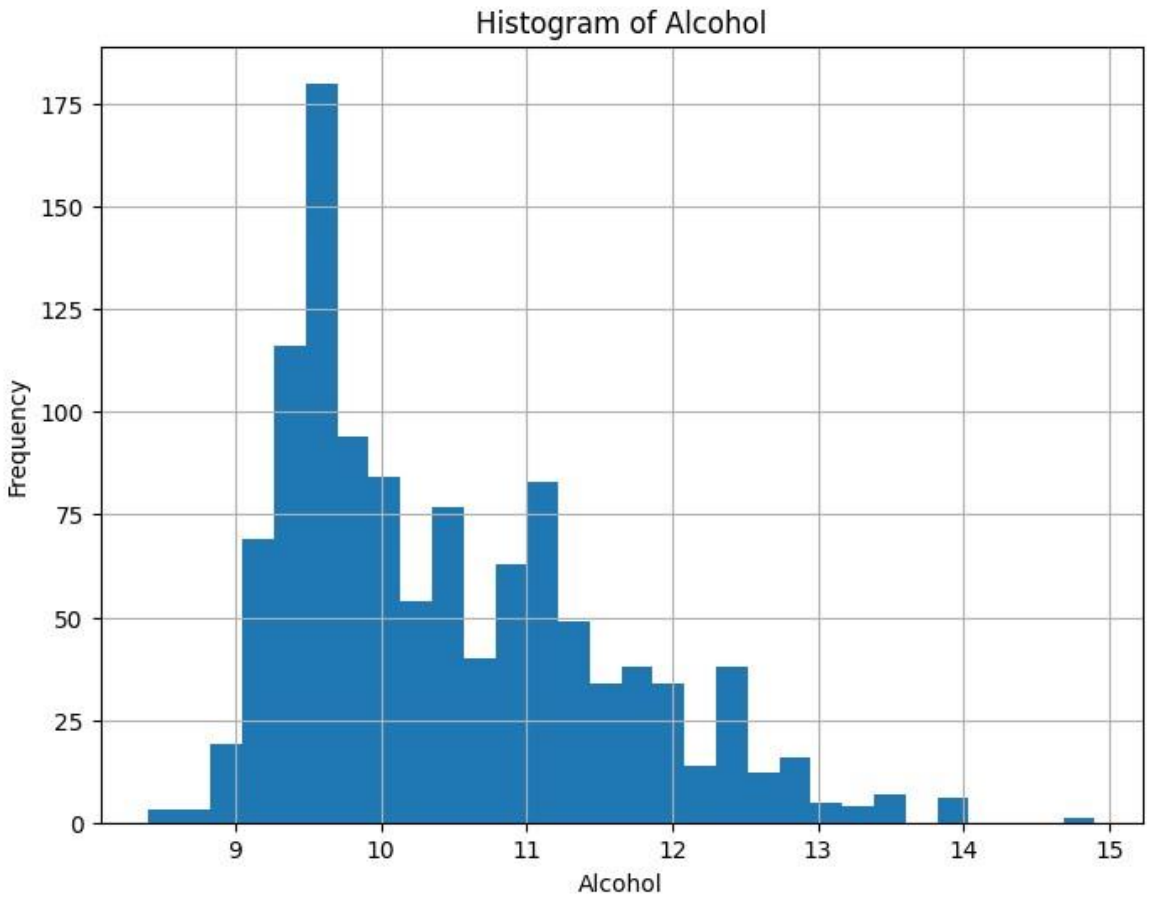
```
 fixed acidity  volatile acidity  citric acid  residual sugar  chloride
s  \
0            7.4             0.70         0.00             1.9      0.
076
1            7.8             0.88         0.00             2.6      0.
098
2            7.8             0.76         0.04             2.3      0.
092
3           11.2             0.28         0.56             1.9      0.
075
4            7.4             0.70         0.00             1.9      0.
076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality  Id
0      9.4        5   0
1      9.8        5   1
2      9.8        5   2
3      9.8        6   3
4      9.4        5   4
fixed acidity           0
```

```
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
Id                      0
dtype: int64
```

	pH	sulphates	alcohol	quality	Id
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	3.311015	0.657708	10.442111	5.657043	804.969379
std	0.156664	0.170399	1.082196	0.805824	463.997116
min	2.740000	0.330000	8.400000	3.000000	0.000000
25%	3.205000	0.550000	9.500000	5.000000	411.000000
50%	3.310000	0.620000	10.200000	6.000000	794.000000
75%	3.400000	0.730000	11.100000	6.000000	1209.500000
max	4.010000	2.000000	14.900000	8.000000	1597.000000

```
quality
5    483
6    462
7    143
4     33
8     16
3      6
Name: count, dtype: int64
```






```
Accuracy: 82.969432%

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.84      0.90       201
           1       0.40      0.79      0.53        28

    accuracy                           0.83       229
   macro avg       0.68      0.81      0.71       229
weighted avg       0.90      0.83      0.85       229
```

RESULT

The program was successfully developed using the Naïve Bayes classifier, and the model accuracy and classification report were generated as expected.

Program 17: Apply the Decision Tree Classifier on the Wine quality (winequality-red.csv) dataset to generate the following results.

(i) Classification Report

(ii) Confusion Matrix

(iii)Plot Decision Tree

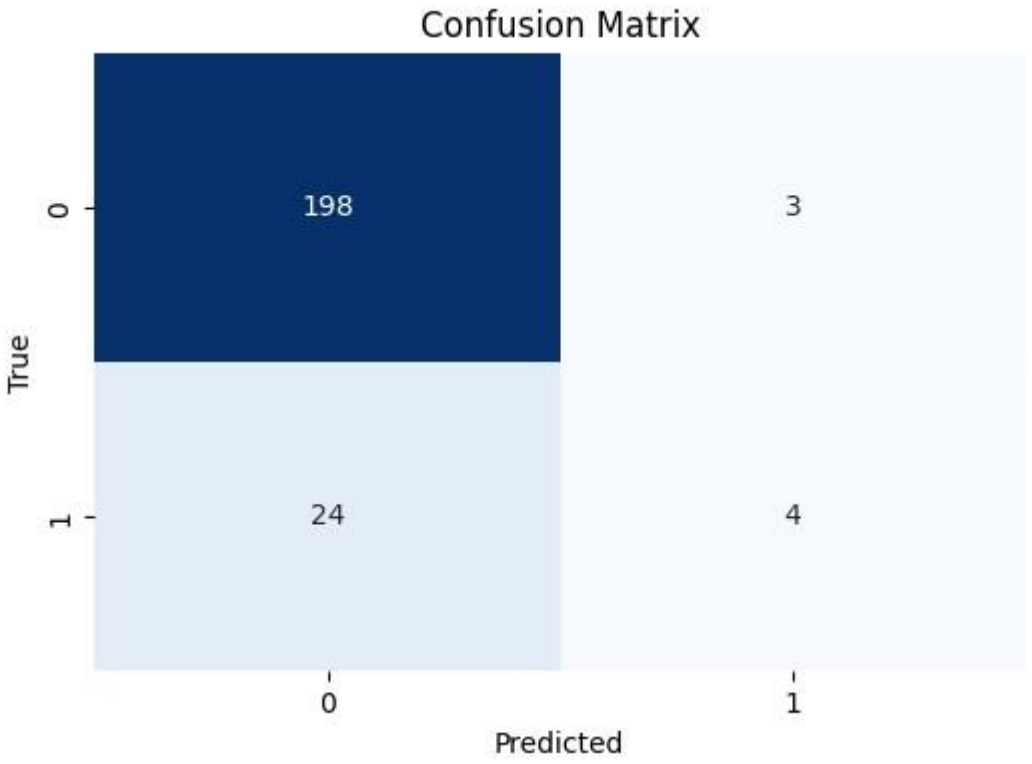
CODE

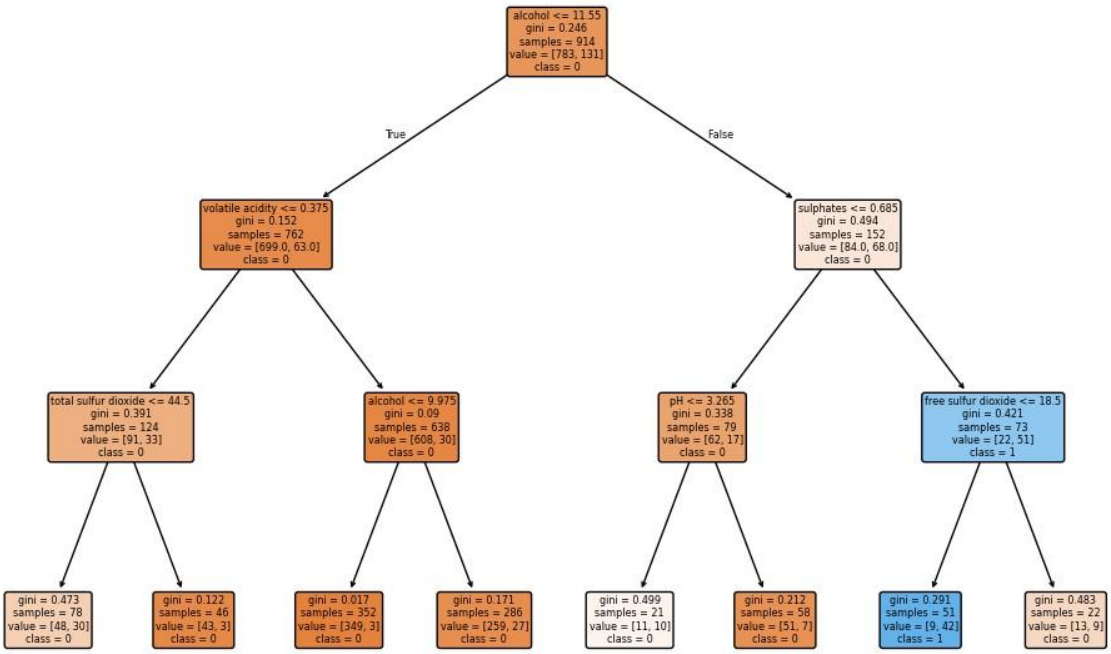
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import tree
import matplotlib.pyplot as plt
wine_data = pd.read_csv("WineQT.csv", sep=';')
print(wine_data.head())
if 'color' in wine_data.columns:
    wine_data = pd.get_dummies(wine_data, columns=['color'], drop_first=True)
    X = wine_data.drop('quality', axis=1)
    y = wine_data['quality']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
dt_classifier = DecisionTreeClassifier(random_state=42, max_depth=3)
dt_classifier.fit(X_train, y_train)
y_pred = dt_classifier.predict(X_test)
classification_rep = classification_report(y_test, y_pred, zero_division=1)
print("Classification Report:")
print(classification_rep)
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
```

```
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
plt.figure(figsize=(12, 8))
tree.plot_tree(
   dt_classifier,
   feature_names=X.columns,
   class_names=[str(label) for label in dt_classifier.classes_],
   filled=True,
   rounded=True
)
plt.show()
```

OUTPUT

```
fixed acidity,volatile acidity,citric acid,residual sugar,chlorides,fre
e sulfur dioxide,total sulfur dioxide,density,pH,sulphates,alcohol,qual
ity,Id
0  7.4,0.7,0.0,1.9,0.076,11.0,34.0,0.9978,3.51,0....
1  7.8,0.88,0.0,2.6,0.098,25.0,67.0,0.9968,3.2,0....
2  7.8,0.76,0.04,2.3,0.092,15.0,54.0,0.997,3.26,0...
3  11.2,0.28,0.56,1.9,0.075,17.0,60.0,0.998,3.16,...
4  7.4,0.7,0.0,1.9,0.076,11.0,34.0,0.9978,3.51,0....
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94       201
           1       0.57      0.14      0.23        28

    accuracy                           0.88       229
   macro avg       0.73      0.56      0.58       229
weighted avg       0.85      0.88      0.85       229


Confusion Matrix:
[[198   3]
 [ 24   4]]
```





RESULT

The program was implemented successfully using the Decision Tree Classifier, and the classification report, confusion matrix, and tree visualization were obtained.

Program 18: Apply the Decision Tree Classifier on iris dataset to generate the following results.

(i) Classification Report

(ii) Confusion Matrix

(iii) Plot Decision Tree

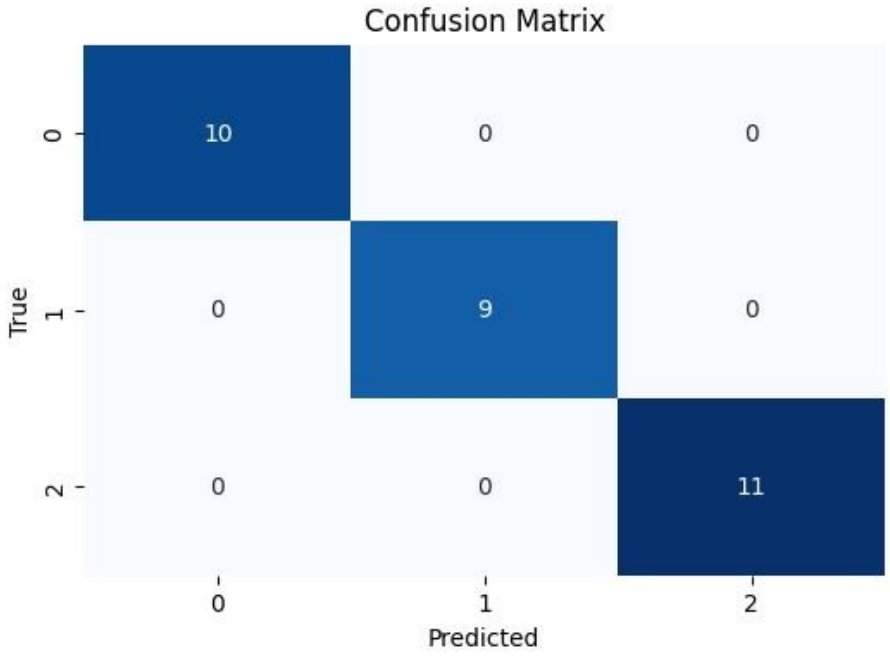
CODE

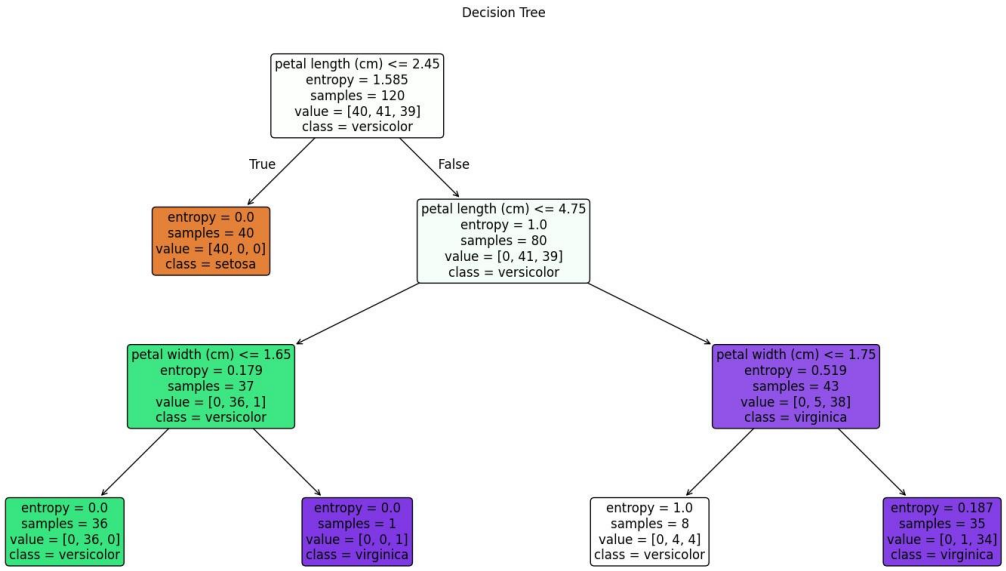
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.metrics import classification_report,confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
iris=load_iris()
x=pd.DataFrame(iris.data,columns=iris.feature_names)
y=pd.Series(iris.target)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
clf=DecisionTreeClassifier(criterion='entropy',max_depth=3,random_state=42)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
print("classification Report\n",classification_report(y_test,y_pred))
cm=confusion_matrix(y_test,y_pred)
print(cm)
plt.figure(figsize=(6,4))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues',cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title("Confusion Matrix")
plt.show()
plt.figure(figsize=(20,10))
```

```
plot_tree(clf,filled=True,feature_names=iris.feature_names,class_names=iris.target_names,rounded=
True,fontsize=12)

plt.title("Decision Tree")

plt.show()
```

OUTPUT

```
classification Report
               precision    recall  f1-score   support

            0       1.00      1.00      1.00        10
            1       1.00      1.00      1.00         9
            2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```





RESULT

The program was successfully executed using the iris dataset, and the decision tree visualization, confusion matrix, and report were generated correctly.

Program 19: Given dataset contains 200 records and five columns, two of which describe the customer's annual income and spending score. The latter s a value from 0 to 100. The higher the number, the more this customer has spent with the company in the past: Using kmeans clustering creates 6 clusters of customers based on their spending pattern.

• Visualize the same in a scatter plot with each cluster in a different color scheme.

• Display the cluster labels of each point.(print cluster indexes)

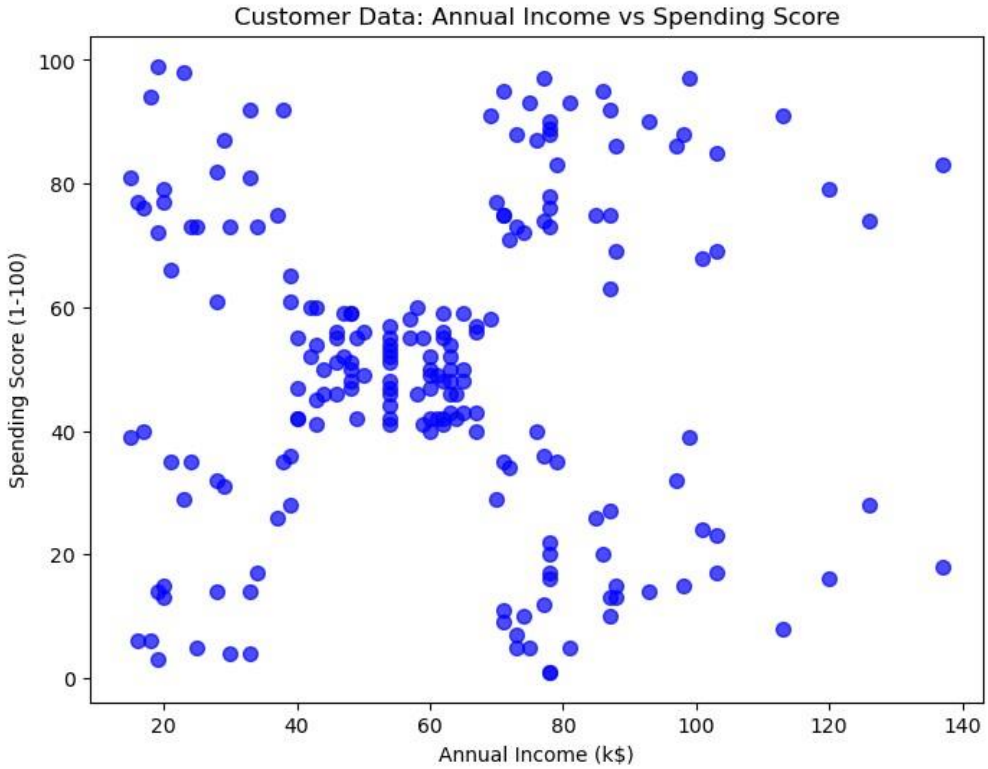
• Display the cluster centers.

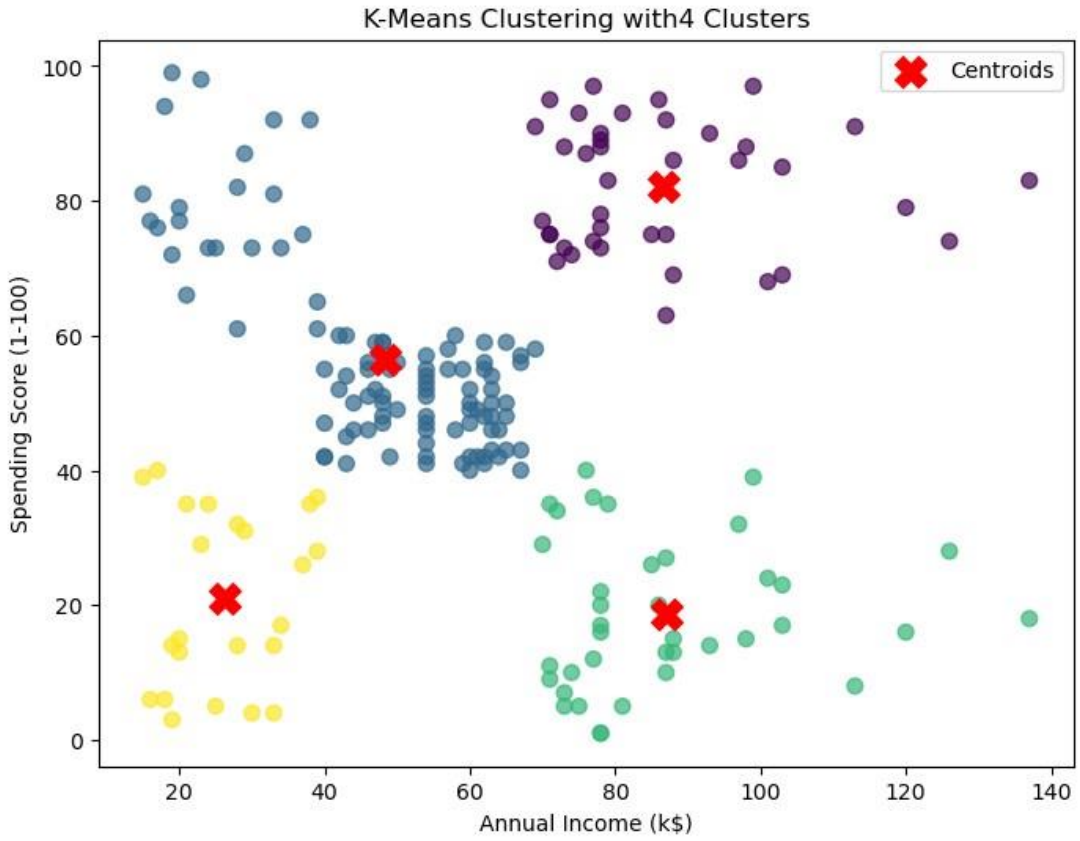
CODE

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np
customer = pd.read_csv('Mall_Customers.csv')
points = customer.iloc[:, 3:5].values
x = points[:, 0]
y = points[:, 1]
plt.figure(figsize=(8, 6))
plt.scatter(x, y, s=50, alpha=0.7, c='blue')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Customer Data: Annual Income vs Spending Score')
plt.show()
def kmeans_clustering_and_plot(points,n_clusters):
    kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    kmeans.fit(points)
    cluster_labels = kmeans.predict(points)
    cluster_centers = kmeans.cluster_centers_
    plt.figure(figsize=(8, 6))
```

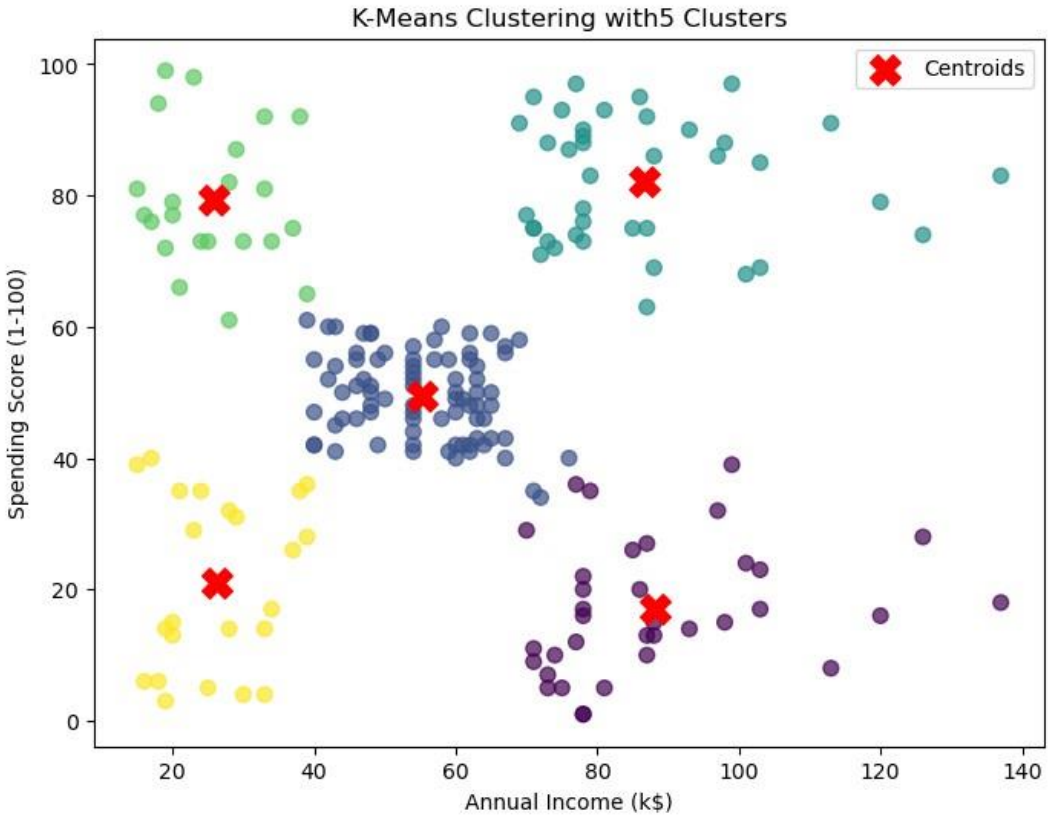
```
    scatter = plt.scatter(x, y, c=cluster_labels, s=50, alpha=0.7,cmap='viridis')
    plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1],s=200, c='red',marker='X',
label='Centroids')
    plt.xlabel('Annual Income (k$)')
    plt.ylabel('Spending Score (1-100)')
    plt.title(f'K-Means Clustering with{n_clusters} Clusters')
    plt.legend()
    plt.show()
    print(f"Cluster labels for K={n_clusters}:\n",cluster_labels)
    print(f"Cluster centers for K={n_clusters}:\n",cluster_centers)


for k in range(4, 9): # Testing for K=4 to K=8
    kmeans_clustering_and_plot(points, n_clusters=k)
```

OUTPUT

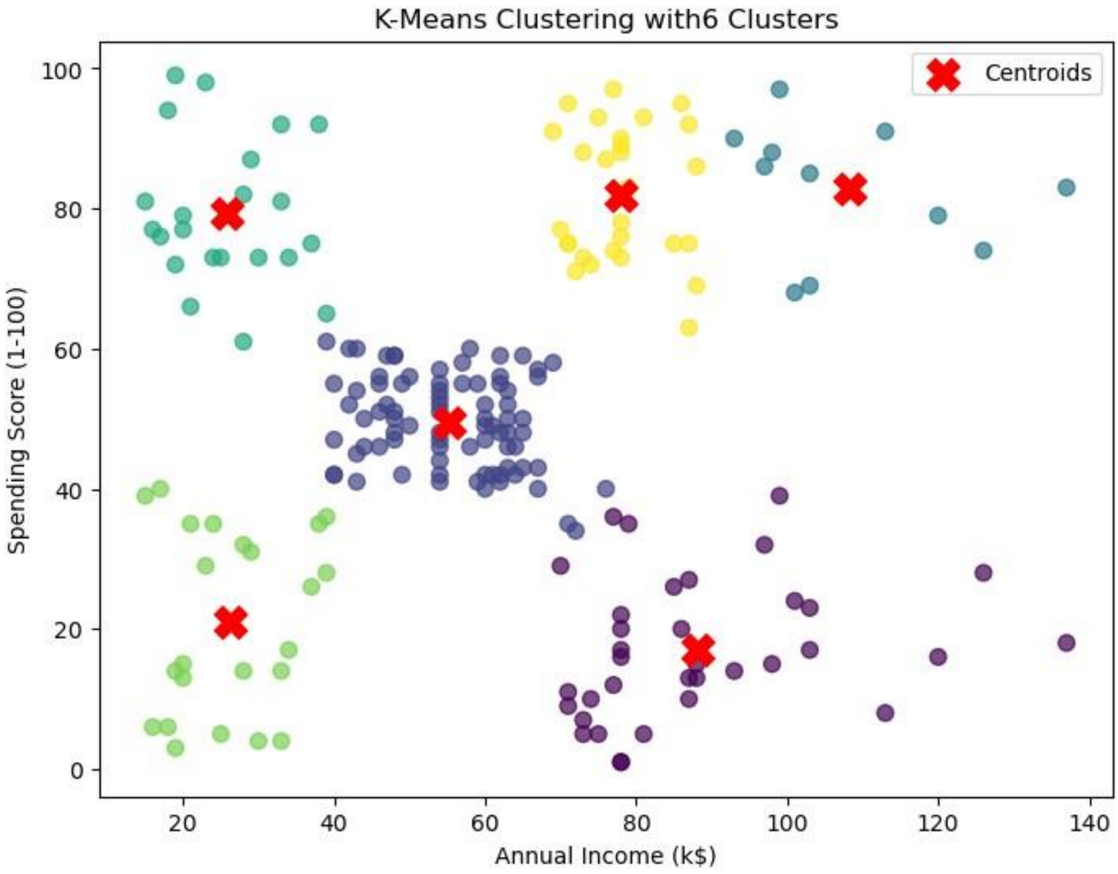




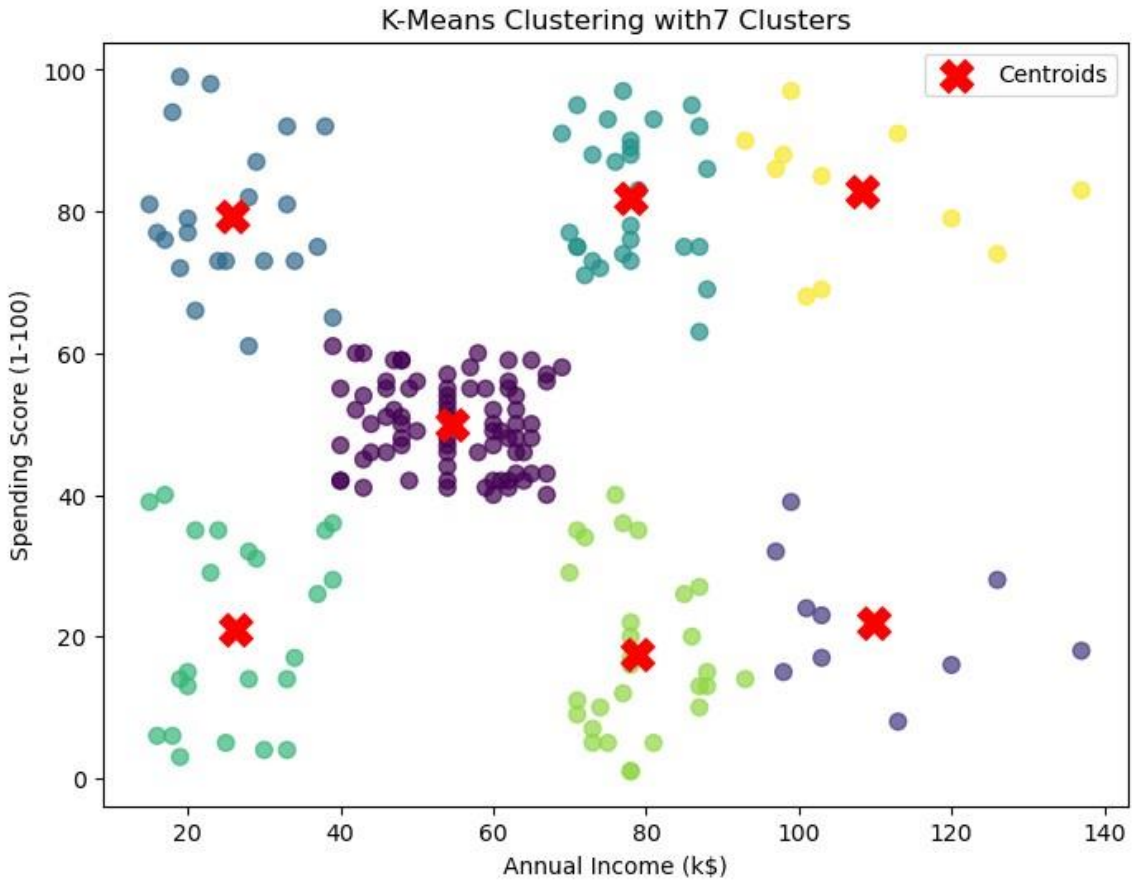

```
Cluster labels for K=4:
 [3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3 1 3
 1 3 1 3 1 3 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2
 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0]
Cluster centers for K=4:
 [[86.53846154 82.12820513]
 [48.26       56.48      ]
 [87.         18.63157895]
 [26.30434783 20.91304348]]
```



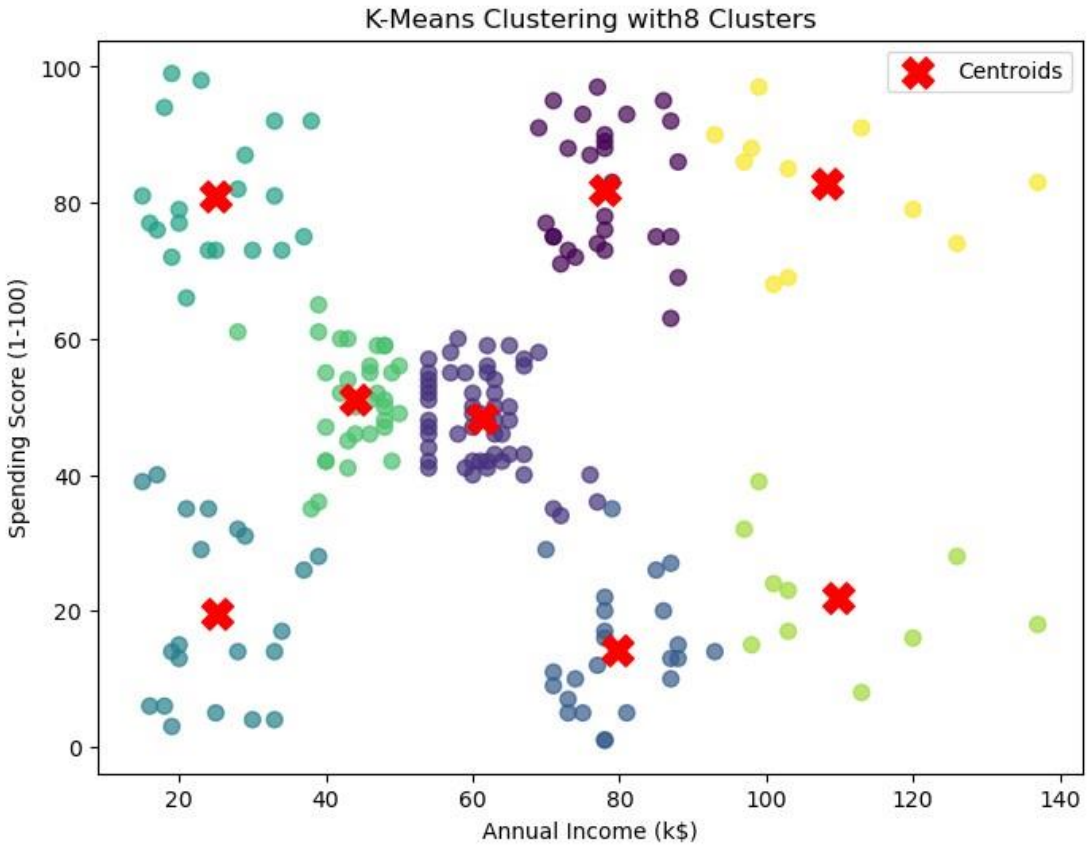

```
Cluster labels for K=5:
 [4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
 3 4 3 4 3 4 1 4 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 0 2 1 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 1 2 0 2
 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]
Cluster centers for K=5:
 [[88.2        17.11428571]
 [55.2962963  49.51851852]
 [86.53846154 82.12820513]
 [25.72727273 79.36363636]
 [26.30434783 20.91304348]]
```




```
Cluster labels for K=6:
 [4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
3 4
 3 4 3 4 3 4 1 4 3 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 5 0 5 1 5 0 5 0 5 1 5 0 5 0 5 0 5 0 5 1 5 0 5
0 5
 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 2 0 2 0
2 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]
Cluster centers for K=6:
 [[ 88.2         17.11428571]
 [ 55.2962963   49.51851852]
 [108.18181818  82.72727273]
 [ 25.72727273  79.36363636]
 [ 26.30434783  20.91304348]
 [ 78.03571429  81.89285714]]
```




```
Cluster labels for K=7:
 [4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4
2 4
 2 4 2 4 2 4 0 4 2 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 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5
5 3
 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 3 5 6 1 6 1 6
6 1
 6 1 6 1 6 1 6 1 6 1 6 1 6 1 6]
Cluster centers for K=7:
 [[ 54.61538462  50.02564103]
 [109.7         22.        ]
 [ 25.72727273  79.36363636]
 [ 78.03571429  81.89285714]
 [ 26.30434783  20.91304348]
 [ 78.89285714  17.42857143]
 [108.18181818  82.72727273]]
```




```
Cluster labels for K=8:
 [3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 5 3 4 3 4 3 4 3
4 3
 4 3 4 5 4 5 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5
 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 2 0 1 0 2 0 2 0 1 0 2 0 2 0 2 0 2 0 1 0 2 0
1 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 7 6 7 6
7 6
 7 6 7 6 7 6 7 6 7 6 7 6 7 6 7]
Cluster centers for K=8:
 [[ 78.03571429  81.89285714]
 [ 61.30188679  48.24528302]
 [ 79.70833333  14.29166667]
 [ 25.14285714  19.52380952]
 [ 24.95        81.        ]
 [ 43.96969697  51.12121212]
 [109.7         22.        ]
 [108.18181818  82.72727273]]
```

RESULT

The program was successfully developed and executed using the K-Means algorithm, and the cluster visualization, labels, and centroids were displayed as expected.