x8ufct61u

December 8, 2023

DAV PRACTICALS BY ABHIGYAN 21HCS4103

[2]:
```
#importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Q1

[3]:
```
height = {'Boys':[72, 68, 70, 69, 74], 'Girls':[63, 65, 69, 62, 61]}
result = [{'Boy':bh, 'Girl':gh} for bh, gh in zip(height['Boys'],␣
 ↪height['Girls'])]
result
```

[3]:
```
[{'Boy': 72, 'Girl': 63},
 {'Boy': 68, 'Girl': 65},
 {'Boy': 70, 'Girl': 69},
 {'Boy': 69, 'Girl': 62},
 {'Boy': 74, 'Girl': 61}]
```

Q2

[4]:
```
#Part(a)
arr = np.random.randint(1,100,size=(3,5))
mean = arr.mean(axis=1)
var = arr.var(axis=1)
sd = arr.std(axis=1)
print("Part (a)")
print("Original array : \n", arr)
print("Mean = ", mean)
print("Standard Deviation = ", sd)
print("Variation = ", var)
print()

#Part(b)
print("Part (b)")
B=np.array([56, 48, 22, 41, 78, 91, 24, 46, 8, 33])
Sorted_indices = B.argsort()
```

```python
print("Original Array = ", B)
print("Sorted_Indices = ", Sorted_indices)
print()

#Part(c)
print("Part (c)")
m = int(input("Enter the value of m : "))
n = int(input("Enter the value of n : "))
arr2 = np.random.randint(1, 50, size=(m, n))
print("m x n Array : \n", arr2)
print("Shape = ", arr2.shape)
print("Type = ", type(arr2))
print("Data Type = ", arr2.dtype)
arr3 = arr2.reshape(n, m)
print()
print("n x m Array : \n", arr3)

#Part(d)
```

```
Part (a)
Original array :
 [[24 19 93 71 72]
 [35 74 76 68 62]
 [22 38 18 81 17]]
Mean =  [55.8 63.  35.2]
Standard Deviation =  [29.13005321 14.83239697 24.11140809]
Variation =  [848.56 220.   581.36]

Part (b)
Original Array =  [56 48 22 41 78 91 24 46  8 33]
Sorted_Indices =  [8 2 6 9 3 7 1 0 4 5]

Part (c)

Enter the value of m :  5
Enter the value of n :  4

m x n Array :
 [[44 14 17 10]
 [40 33 23 14]
 [31 40 10 12]
 [19 42 21  5]
 [46 25 45 49]]
Shape =  (5, 4)
Type =  <class 'numpy.ndarray'>
Data Type =  int32

n x m Array :
```

```
[[44 14 17 10 40]
 [33 23 14 31 40]
 [10 12 19 42 21]
 [ 5 46 25 45 49]]
```

Q3

[5]:

```
p = 0.10 #setting the probability
df = pd.DataFrame(np.random.randint(1, 100, size=(50, 3)), columns=['C1', 'C2',␣
 ↪'C3'])
mask = np.random.choice([True, False], size=df.shape, p=[p,1-p])
new_df = df.mask(mask)
print(new_df)

#part(a)
print("Part (a)")
null_count = new_df.isnull().sum()
print("Null_Count = ",null_count)
print()

#part(b)
print("Part (b)")
new_df2 = new_df.dropna(axis=1, thresh = df.shape[0] - 5)
print(new_df2)
print()

#part(c)
print("Part (c)")
max_sum_row = new_df.sum(axis=1).idxmax()
new_df = new_df.drop(max_sum_row)
print()

#part(d)
print("Part (d)")
new_df3 = new_df.sort_values(by = 'C1')
print(new_df)
print()

#part(e)
print("Part (e)")
new_df4 = new_df.drop_duplicates(subset='C1')
print(new_df4)
print()

#part(f)
print("Part (f)")
correlation = new_df['C1'].corr(new_df['C2'])
covariance = new_df['C2'].cov(new_df['C3'])
```

```
print("Correlation between C1 and C2 = ", correlation)
print()
print("Covarinace between C2 and C3 = ", covariance)
print()

#part (g)
print("Part (g)")
z_scores = (new_df - new_df.mean()) / new_df.std()
outliers = (z_scores > 3) | (z_scores < -3)
new_df = new_df[~outliers.any(axis=1)]
print("\ng. DataFrame after removing rows with outliers:")
print(new_df)

#part(h)
new_df['Column2_bins'] = pd.cut(new_df['C2'], bins=5)
print("\nh. DataFrame with second column discretized into 5 bins:")
print(df)
```

```
      C1    C2    C3
0   50.0  34.0  64.0
1   58.0  58.0  29.0
2   17.0  54.0   9.0
3   29.0  20.0  54.0
4   30.0  51.0   NaN
5   38.0  76.0   NaN
6   13.0  85.0  48.0
7   94.0  14.0  79.0
8   27.0  32.0  40.0
9   65.0  68.0  29.0
10  19.0  58.0   5.0
11  94.0  19.0  16.0
12  47.0  17.0  72.0
13  44.0  58.0  25.0
14  37.0  54.0  90.0
15  73.0  95.0  25.0
16  81.0  91.0   NaN
17  54.0  43.0  41.0
18  56.0  24.0  43.0
19  47.0   NaN  71.0
20  76.0  93.0   4.0
21   3.0  54.0  14.0
22   1.0  60.0  53.0
23  80.0  43.0  94.0
24  78.0  93.0  30.0
25  81.0  59.0  18.0
26  33.0  30.0  56.0
27  99.0  66.0  43.0
```

```
28  94.0  49.0  57.0
29  75.0  17.0  96.0
30  43.0  79.0  74.0
31  82.0  76.0  87.0
32  70.0  40.0  58.0
33  70.0  24.0  15.0
34  87.0  32.0  93.0
35  84.0  97.0  84.0
36   NaN  95.0   6.0
37  39.0   NaN  21.0
38  99.0  72.0  12.0
39   NaN  56.0  85.0
40  91.0  61.0  36.0
41  98.0  27.0  93.0
42  78.0   NaN  72.0
43  10.0  56.0  24.0
44  55.0  87.0  61.0
45  44.0   NaN  19.0
46  16.0  74.0  58.0
47  92.0   8.0  85.0
48   8.0   4.0  23.0
49  92.0  80.0   4.0
Part (a)
Null_Count =  C1    2
C2    4
C3    3
dtype: int64

Part (b)
      C1    C2    C3
0   50.0  34.0  64.0
1   58.0  58.0  29.0
2   17.0  54.0   9.0
3   29.0  20.0  54.0
4   30.0  51.0   NaN
5   38.0  76.0   NaN
6   13.0  85.0  48.0
7   94.0  14.0  79.0
8   27.0  32.0  40.0
9   65.0  68.0  29.0
10  19.0  58.0   5.0
11  94.0  19.0  16.0
12  47.0  17.0  72.0
13  44.0  58.0  25.0
14  37.0  54.0  90.0
15  73.0  95.0  25.0
16  81.0  91.0   NaN
17  54.0  43.0  41.0
```

```
18  56.0  24.0  43.0
19  47.0   NaN  71.0
20  76.0  93.0   4.0
21   3.0  54.0  14.0
22   1.0  60.0  53.0
23  80.0  43.0  94.0
24  78.0  93.0  30.0
25  81.0  59.0  18.0
26  33.0  30.0  56.0
27  99.0  66.0  43.0
28  94.0  49.0  57.0
29  75.0  17.0  96.0
30  43.0  79.0  74.0
31  82.0  76.0  87.0
32  70.0  40.0  58.0
33  70.0  24.0  15.0
34  87.0  32.0  93.0
35  84.0  97.0  84.0
36   NaN  95.0   6.0
37  39.0   NaN  21.0
38  99.0  72.0  12.0
39   NaN  56.0  85.0
40  91.0  61.0  36.0
41  98.0  27.0  93.0
42  78.0   NaN  72.0
43  10.0  56.0  24.0
44  55.0  87.0  61.0
45  44.0   NaN  19.0
46  16.0  74.0  58.0
47  92.0   8.0  85.0
48   8.0   4.0  23.0
49  92.0  80.0   4.0

Part (c)

Part (d)
      C1    C2    C3
0   50.0  34.0  64.0
1   58.0  58.0  29.0
2   17.0  54.0   9.0
3   29.0  20.0  54.0
4   30.0  51.0   NaN
5   38.0  76.0   NaN
6   13.0  85.0  48.0
7   94.0  14.0  79.0
8   27.0  32.0  40.0
9   65.0  68.0  29.0
10  19.0  58.0   5.0
```

```
11  94.0  19.0  16.0
12  47.0  17.0  72.0
13  44.0  58.0  25.0
14  37.0  54.0  90.0
15  73.0  95.0  25.0
16  81.0  91.0   NaN
17  54.0  43.0  41.0
18  56.0  24.0  43.0
19  47.0   NaN  71.0
20  76.0  93.0   4.0
21   3.0  54.0  14.0
22   1.0  60.0  53.0
23  80.0  43.0  94.0
24  78.0  93.0  30.0
25  81.0  59.0  18.0
26  33.0  30.0  56.0
27  99.0  66.0  43.0
28  94.0  49.0  57.0
29  75.0  17.0  96.0
30  43.0  79.0  74.0
31  82.0  76.0  87.0
32  70.0  40.0  58.0
33  70.0  24.0  15.0
34  87.0  32.0  93.0
36   NaN  95.0   6.0
37  39.0   NaN  21.0
38  99.0  72.0  12.0
39   NaN  56.0  85.0
40  91.0  61.0  36.0
41  98.0  27.0  93.0
42  78.0   NaN  72.0
43  10.0  56.0  24.0
44  55.0  87.0  61.0
45  44.0   NaN  19.0
46  16.0  74.0  58.0
47  92.0   8.0  85.0
48   8.0   4.0  23.0
49  92.0  80.0   4.0

Part (e)
      C1    C2    C3
0   50.0  34.0  64.0
1   58.0  58.0  29.0
2   17.0  54.0   9.0
3   29.0  20.0  54.0
4   30.0  51.0   NaN
5   38.0  76.0   NaN
6   13.0  85.0  48.0
```

```
7   94.0  14.0  79.0
8   27.0  32.0  40.0
9   65.0  68.0  29.0
10  19.0  58.0   5.0
12  47.0  17.0  72.0
13  44.0  58.0  25.0
14  37.0  54.0  90.0
15  73.0  95.0  25.0
16  81.0  91.0   NaN
17  54.0  43.0  41.0
18  56.0  24.0  43.0
20  76.0  93.0   4.0
21   3.0  54.0  14.0
22   1.0  60.0  53.0
23  80.0  43.0  94.0
24  78.0  93.0  30.0
26  33.0  30.0  56.0
27  99.0  66.0  43.0
29  75.0  17.0  96.0
30  43.0  79.0  74.0
31  82.0  76.0  87.0
32  70.0  40.0  58.0
34  87.0  32.0  93.0
36   NaN  95.0   6.0
37  39.0   NaN  21.0
40  91.0  61.0  36.0
41  98.0  27.0  93.0
43  10.0  56.0  24.0
44  55.0  87.0  61.0
46  16.0  74.0  58.0
47  92.0   8.0  85.0
48   8.0   4.0  23.0

Part (f)
Correlation between C1 and C2 =  0.003811387816605662

Covarinace between C2 and C3 =  -273.1440185830429

Part (g)

g. DataFrame after removing rows with outliers:
      C1    C2    C3
0   50.0  34.0  64.0
1   58.0  58.0  29.0
2   17.0  54.0   9.0
3   29.0  20.0  54.0
4   30.0  51.0   NaN
5   38.0  76.0   NaN
```

```
6   13.0  85.0  48.0
7   94.0  14.0  79.0
8   27.0  32.0  40.0
9   65.0  68.0  29.0
10  19.0  58.0   5.0
11  94.0  19.0  16.0
12  47.0  17.0  72.0
13  44.0  58.0  25.0
14  37.0  54.0  90.0
15  73.0  95.0  25.0
16  81.0  91.0   NaN
17  54.0  43.0  41.0
18  56.0  24.0  43.0
19  47.0   NaN  71.0
20  76.0  93.0   4.0
21   3.0  54.0  14.0
22   1.0  60.0  53.0
23  80.0  43.0  94.0
24  78.0  93.0  30.0
25  81.0  59.0  18.0
26  33.0  30.0  56.0
27  99.0  66.0  43.0
28  94.0  49.0  57.0
29  75.0  17.0  96.0
30  43.0  79.0  74.0
31  82.0  76.0  87.0
32  70.0  40.0  58.0
33  70.0  24.0  15.0
34  87.0  32.0  93.0
36   NaN  95.0   6.0
37  39.0   NaN  21.0
38  99.0  72.0  12.0
39   NaN  56.0  85.0
40  91.0  61.0  36.0
41  98.0  27.0  93.0
42  78.0   NaN  72.0
43  10.0  56.0  24.0
44  55.0  87.0  61.0
45  44.0   NaN  19.0
46  16.0  74.0  58.0
47  92.0   8.0  85.0
48   8.0   4.0  23.0
49  92.0  80.0   4.0

h. DataFrame with second column discretized into 5 bins:
    C1  C2  C3
0   50  34  64
1   58  58  29
```

```
2    17  54   9
3    29  20  54
4    30  51   8
5    38  76  64
6    13  85  48
7    94  14  79
8    27  32  40
9    65  68  29
10   19  58   5
11   94  19  16
12   47  17  72
13   44  58  25
14   37  54  90
15   73  95  25
16   81  91  46
17   54  43  41
18   56  24  43
19   47  77  71
20   76  93   4
21    3  54  14
22    1  60  53
23   80  43  94
24   78  93  30
25   81  59  18
26   33  30  56
27   99  66  43
28   94  49  57
29   75  17  96
30   43  79  74
31   82  76  87
32   70  40  58
33   70  24  15
34   87  32  93
35   84  97  84
36    4  95   6
37   39  64  21
38   99  72  12
39   52  56  85
40   91  61  36
41   98  27  93
42   78  20  72
43   10  56  24
44   55  87  61
45   44  38  19
46   16  74  58
47   92   8  85
48    8   4  23
49   92  80   4
```

Q4

[6]:

```
df1 = pd.read_excel('day1.xlsx')
df2 = pd.read_excel('day2.xlsx')
print(df1)
print(df2)
```

```
         Name Time of Joining  Duration
0   Abhimanyu        11:00:00        40
1    Abhishek        11:04:00        30
2       Aasif        11:08:00        30
3        Aman        11:01:00        40
4       Anand        11:12:00        50
5     Anubhav        11:10:00        30
6      Anurag        11:11:00        30
7       Arpit        11:07:00        40
8    Akanksha        11:08:00        50
9     Bhavana        11:15:00        30
10  Deepanshu        11:02:00        40
11     Ishant        11:03:00        30
12     Gourav        11:19:00        30
13    Harshit        11:13:00        40
14   Kartikey        11:05:00        50
         Name Time of Joining  Duration
0   Abhimanyu        11:00:00        40
1    Abhishek        11:06:00        30
2   Deepanshu        11:10:00        40
3        Aman        11:09:00        40
4     Anubhav        11:10:00        50
5      Bharat        11:12:00        30
6      Anurag        11:08:00        30
7       Arpit        11:08:00        40
8   Divyanshu        11:13:00        40
9     Bhavana        11:14:00        30
10     Deepak        11:02:00        50
11     Ishant        11:00:00        30
12     Jayesh        11:08:00        30
13    Harshit        11:09:00        40
14      Jeeva        11:06:00        30
```

[7]:

```
#PART A
common_attendees = pd.merge(df1, df2, on="Name", how='inner')
print(common_attendees['Name'])
```

```
0    Abhimanyu
1     Abhishek
2         Aman
3      Anubhav
```

```
4        Anurag
5         Arpit
6       Bhavana
7     Deepanshu
8        Ishant
9       Harshit
Name: Name, dtype: object
```

```
[8]: #PART B
     single_day_attendees = pd.merge(df1, df2, on='Name', how='outer')
     print(single_day_attendees['Name'])
```

```
0     Abhimanyu
1      Abhishek
2         Aasif
3          Aman
4         Anand
5       Anubhav
6        Anurag
7         Arpit
8      Akanksha
9       Bhavana
10    Deepanshu
11       Ishant
12       Gourav
13      Harshit
14     Kartikey
15       Bharat
16    Divyanshu
17       Deepak
18       Jayesh
19        Jeeva
Name: Name, dtype: object
```

```
[9]: #PART C
     merged_data_rw = pd.concat([df1, df2], ignore_index=True)
     merged_data_rw
```

```
[9]:          Name Time of Joining  Duration
     0   Abhimanyu        11:00:00        40
     1    Abhishek        11:04:00        30
     2       Aasif        11:08:00        30
     3        Aman        11:01:00        40
     4       Anand        11:12:00        50
     5     Anubhav        11:10:00        30
     6      Anurag        11:11:00        30
     7       Arpit        11:07:00        40
```

```
8    Akanksha        11:08:00        50
9     Bhavana        11:15:00        30
10  Deepanshu        11:02:00        40
11     Ishant        11:03:00        30
12     Gourav        11:19:00        30
13    Harshit        11:13:00        40
14   Kartikey        11:05:00        50
15  Abhimanyu        11:00:00        40
16   Abhishek        11:06:00        30
17  Deepanshu        11:10:00        40
18       Aman        11:09:00        40
19    Anubhav        11:10:00        50
20     Bharat        11:12:00        30
21     Anurag        11:08:00        30
22      Arpit        11:08:00        40
23  Divyanshu        11:13:00        40
24    Bhavana        11:14:00        30
25     Deepak        11:02:00        50
26     Ishant        11:00:00        30
27     Jayesh        11:08:00        30
28    Harshit        11:09:00        40
29      Jeeva        11:06:00        30
```

[10]:

```
#PART D
merged_multi_index = pd.merge(df1, df2, on=['Name', 'Duration'], how='inner')
multi_merge_stats = merged_multi_index.groupby(['Name', 'Duration']).describe()
multi_merge_stats
```

[10]:

```
                   Time of Joining_x                       Time of Joining_y  \
                               count unique       top freq             count
Name      Duration
Abhimanyu 40                       1      1  11:00:00    1                 1
Abhishek  30                       1      1  11:04:00    1                 1
Aman      40                       1      1  11:01:00    1                 1
Anurag    30                       1      1  11:11:00    1                 1
Arpit     40                       1      1  11:07:00    1                 1
Bhavana   30                       1      1  11:15:00    1                 1
Deepanshu 40                       1      1  11:02:00    1                 1
Harshit   40                       1      1  11:13:00    1                 1
Ishant    30                       1      1  11:03:00    1                 1


                   unique       top freq
Name      Duration
Abhimanyu 40            1  11:00:00    1
Abhishek  30            1  11:06:00    1
Aman      40            1  11:09:00    1
```

```
Anurag    30            1  11:08:00    1
Arpit     40            1  11:08:00    1
Bhavana   30            1  11:14:00    1
Deepanshu 40            1  11:10:00    1
Harshit   40            1  11:09:00    1
Ishant    30            1  11:00:00    1
```

Q5

[12]:

```python
from ucimlrepo import fetch_ucirepo
#FETCHING THE IRIS DATASET
iris = fetch_ucirepo(id=53)
iris_data = iris.data.original
# data (as pandas dataframes)
X = iris.data.features
y = iris.data.targets
print(X)
print(y)
```

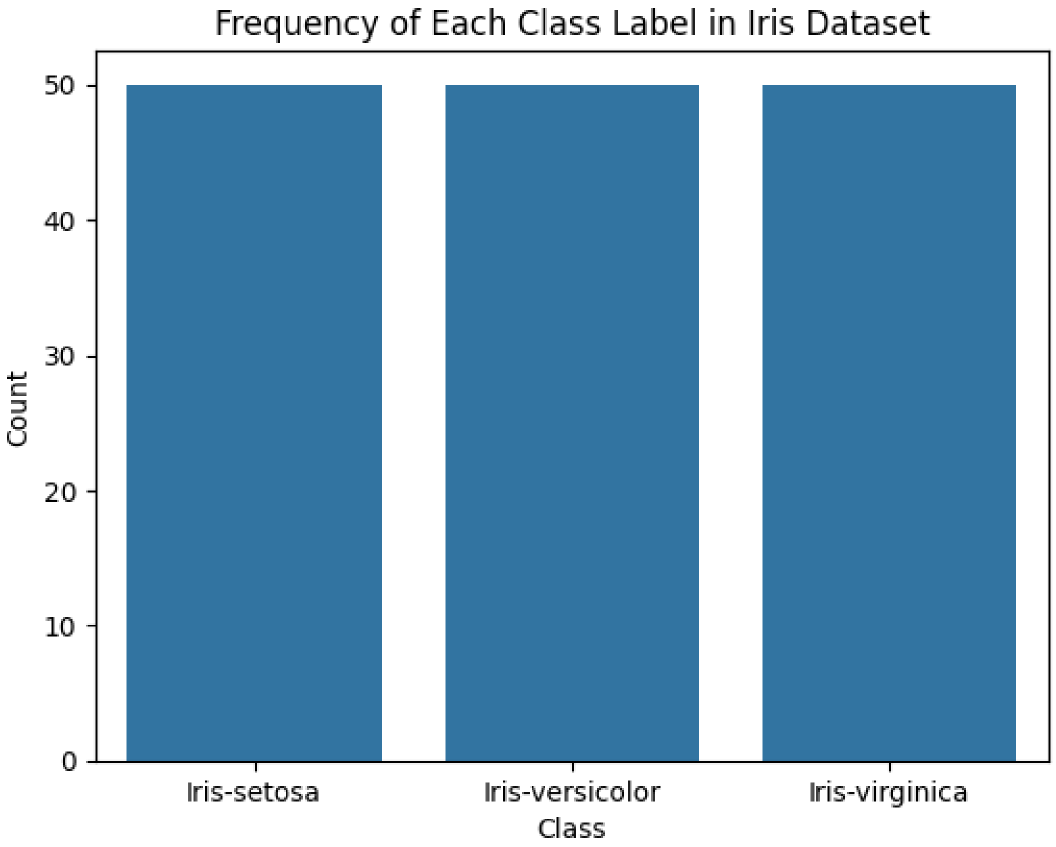
```
     sepal length  sepal width  petal length  petal width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            …          …           …          …
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
              class
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 1 columns]
```

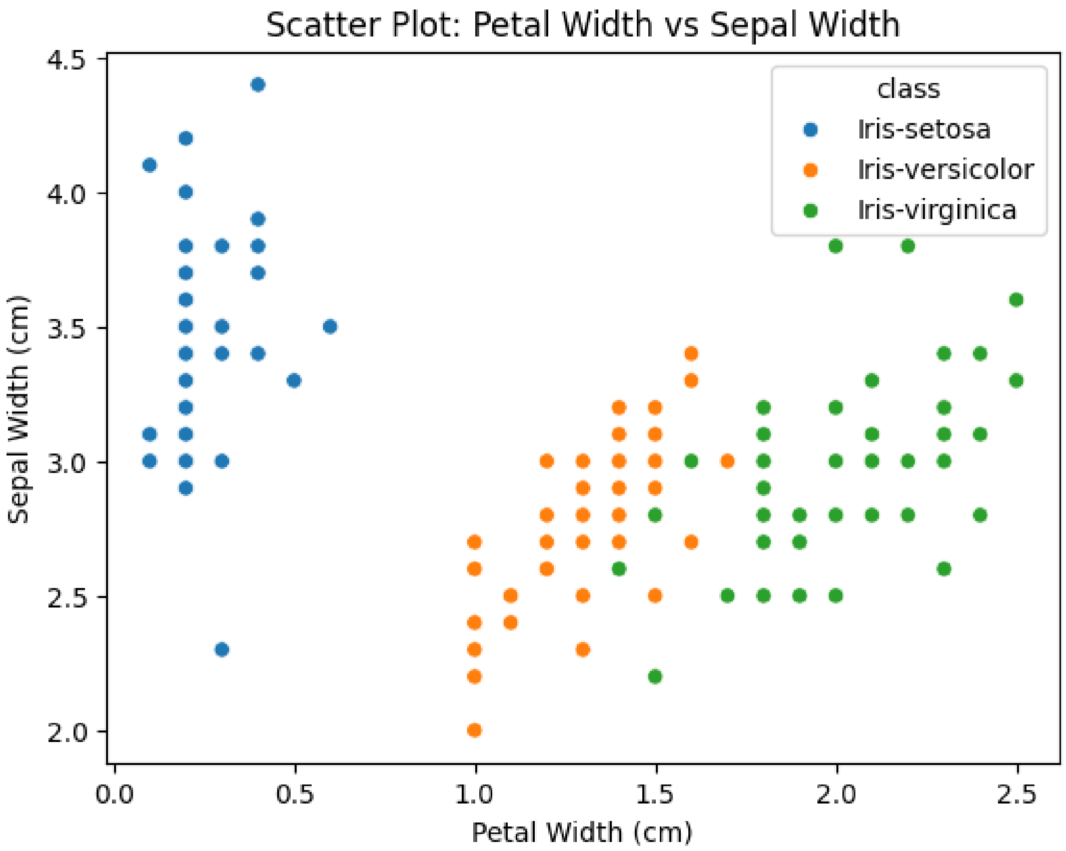
[13]:

```
#PART A
sns.countplot(x='class', data = y)
plt.title('Frequency of Each Class Label in Iris Dataset')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```



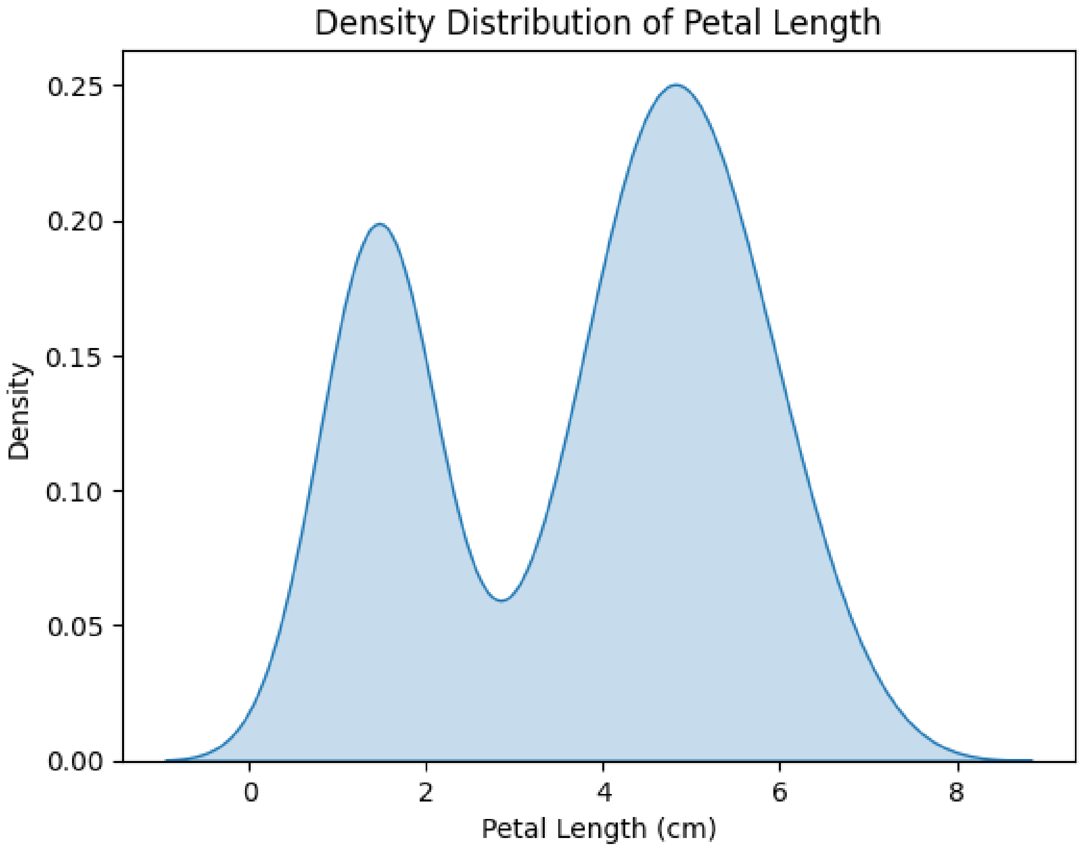
[14]:

```
#PART B
sns.scatterplot(x='petal width', y='sepal width', hue = y['class'],data=X)
plt.title('Scatter Plot: Petal Width vs Sepal Width')
plt.xlabel('Petal Width (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```



[15]:

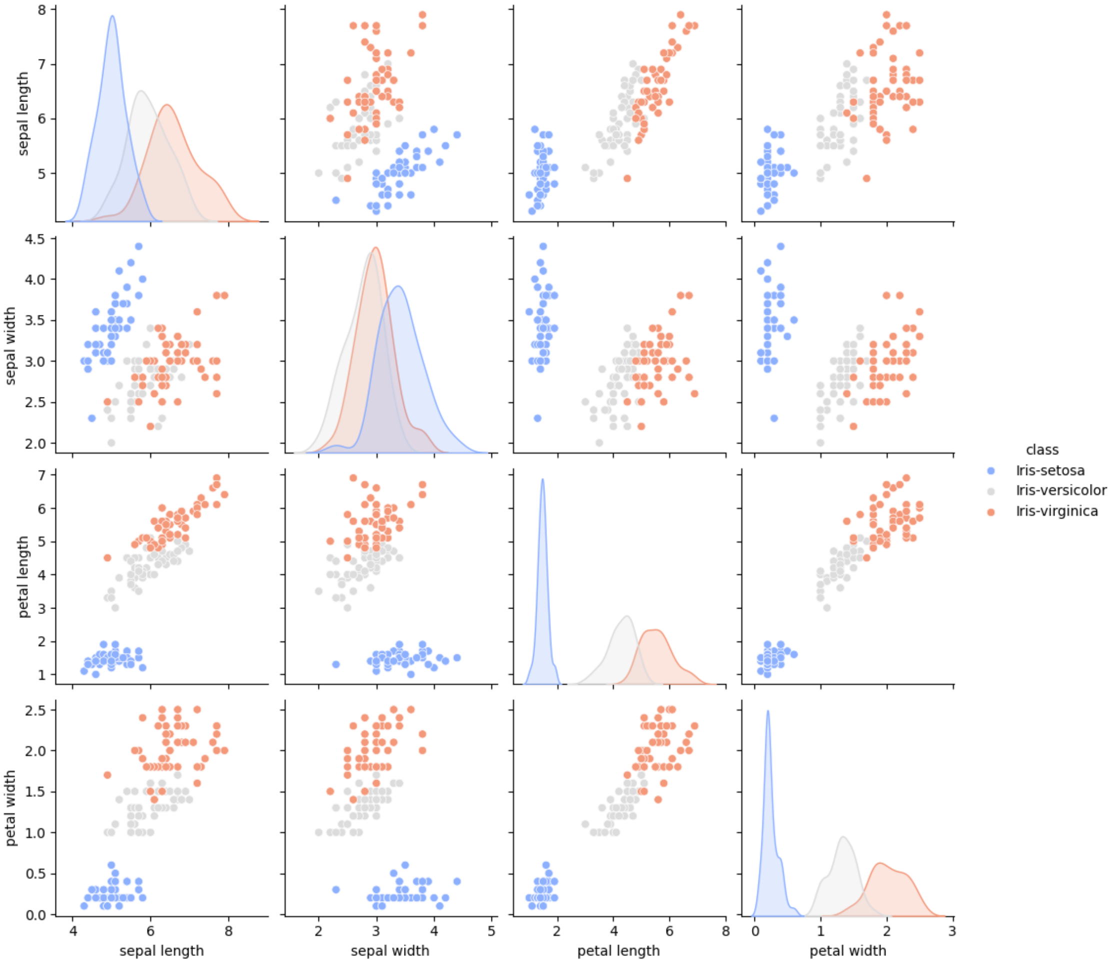
```
#PART C
sns.kdeplot(X['petal length'], fill=True)
plt.title('Density Distribution of Petal Length')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Density')
plt.show()
```



```
[16]: #PART D
      sns.pairplot(iris_data, hue ='class',palette='coolwarm')
```

```
C:\Users\acer\AppData\Local\Programs\Python\Python311\Lib\site-
packages\seaborn\axisgrid.py:123: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

```
[16]: <seaborn.axisgrid.PairGrid at 0x29f9c70df10>
```



Q6

[17]:
```
weather_rep = pd.read_csv('DailyDelhiClimateTest.csv')
weather_rep
```

[17]:
```
           date   meantemp   humidity  wind_speed  meanpressure
0    01-01-2017  15.913043  85.869565    2.743478     59.000000
1    02-01-2017  18.500000  77.222222    2.894444   1018.277778
2    03-01-2017  17.111111  81.888889    4.016667   1018.333333
3    04-01-2017  18.700000  70.050000    4.545000   1015.700000
4           NaN  18.388889  74.944444    3.300000   1014.333333
..          …          …          …          …            …
109  20-04-2017  34.500000  27.500000    5.562500    998.625000
110  21-04-2017  34.250000  39.375000    6.962500    999.875000
111         NaN  32.900000  40.900000    8.890000   1001.600000
112  23-04-2017  32.875000  27.500000    9.962500   1002.125000
113  24-04-2017  32.000000  27.142857   12.157143   1004.142857
```

```
[114 rows x 5 columns]
```

[18]:

```
#PART A
mean_humidity_by_temperature = weather_rep.groupby('meantemp')['humidity'].
 ↪mean()
mean_humidity_by_temperature
```

[18]:

```
meantemp
11.000000    72.111111
11.722222    84.444444
11.789474    74.578947
12.111111    71.944444
13.041667    78.333333
               …
32.900000    40.900000
33.500000    24.125000
34.000000    27.333333
34.250000    39.375000
34.500000    27.500000
Name: humidity, Length: 105, dtype: float64
```

[19]:

```
#PART B
#df_weather_filled = weather_rep.set_index('date').asfreq('D', method='pad')
#print("DataFrame with Missing Dates Filled:")
#print(df_weather_filled)
```

[20]:

```
#PART C
weather_rep['YearMonth'] = pd.to_datetime(weather_rep['date'],␣
 ↪format="%d-%m-%Y").dt.to_period('M')
print("Converted Year-Month:")
print(weather_rep[['date', 'YearMonth']])
```

```
Converted Year-Month:
           date YearMonth
0    01-01-2017   2017-01
1    02-01-2017   2017-01
2    03-01-2017   2017-01
3    04-01-2017   2017-01
4           NaN       NaT
..          …         …
109  20-04-2017   2017-04
110  21-04-2017   2017-04
111         NaN       NaT
112  23-04-2017   2017-04
113  24-04-2017   2017-04
```

```
[114 rows x 2 columns]
```

[21]:

```
#PART D
sorted_weather_by_pressure = weather_rep.groupby(['meanpressure', 'YearMonth']).
 ↪agg({
    'meantemp':'mean',
    'humidity':'mean'
}).reset_index()
sorted_weather_by_pressure
```

[21]:

```
     meanpressure YearMonth   meantemp   humidity
0       59.000000   2017-01  15.913043  85.869565
1      998.625000   2017-04  34.500000  27.500000
2      999.875000   2017-04  34.250000  39.375000
3     1000.875000   2017-04  33.500000  24.125000
4     1002.125000   2017-04  32.875000  27.500000
..            …       …        …          …
100   1021.375000   2017-02  16.875000  65.500000
101   1021.555556   2017-02  16.333333  67.000000
102   1021.789474   2017-01  15.263158  66.473684
103   1021.958333   2017-01  13.041667  78.333333
104   1022.809524   2017-01  14.619048  75.142857

[105 rows x 4 columns]
```

[22]:

```
#PART E
temp_bins = [0, 15, 25, 35]
weather_rep['TempBins'] = pd.cut(weather_rep['meantemp'], bins=temp_bins)
groupby_bins = weather_rep.groupby('TempBins')
print(groupby_bins.describe())
```

```
         meantemp                                                         \
            count       mean       std     min        25%        50%
TempBins
(0, 15]      13.0  13.398375  1.381566  11.000  12.111111  13.235294
(15, 25]     67.0  18.999372  2.790567  15.125  16.472222  18.631579
(25, 35]     34.0  30.239829  2.269097  25.625  29.132692  30.194444

                                humidity             … wind_speed            \
                75%        max     count       mean  …        75%        max
TempBins                                             …
(0, 15]   14.650000  14.863636      13.0  77.502871  …   9.772222  10.380000
(15, 25]  20.842857  25.000000      67.0  63.864985  …   9.473333  16.662500
(25, 35]  31.336806  34.500000      34.0  33.145938  …  12.939286  19.314286

         meanpressure                                                      \
                count          mean          std        min          25%
TempBins
```

```
(0, 15]          13.0  1017.641666    2.894354  1011.375  1016.368421
(15, 25]         67.0  1000.470917  116.827770    59.000  1011.830808
(25, 35]         34.0  1005.856092    3.299112   998.625  1003.473214


                50%          75%          max
TempBins
(0, 15]   1017.1500  1018.840000  1022.809524
(15, 25]  1015.2500  1017.676136  1021.789474
(25, 35]  1006.0625  1008.799107  1010.625000

[3 rows x 32 columns]
```

Q7

[23]:
```
data = {
    'Name': ['Mudit Chauhan', 'Seema Chopra', 'Rani Gupta', 'Aditya Narayan',␣
 ↪'Sanjeev Sahni',
             'Prakash Kumar', 'Ritu Agarwal', 'Akshay Goel', 'Meeta Kulkarni',␣
 ↪'Preeti Ahuja',
             'Sunil Das Gupta', 'Sonali Sapre', 'Rashmi Talwar', 'Ashish␣
 ↪Dubey', 'Kiran Sharma',
             'Sameer Bansal'],
    'Birth_Month': ['December', 'January', 'March', 'October', 'February',␣
 ↪'December', 'September',
                    'August', 'July', 'November', 'April', 'January', 'June',␣
 ↪'May', 'February', 'October'],
    'Gender': ['M', 'F', 'F', 'M', 'M', 'M', 'F', 'M', 'F', 'F', 'M', 'F', 'F',␣
 ↪'M', 'F', 'M'],
    'Pass_Division': ['III', 'II', 'I', 'I', 'II', 'III', 'I', 'I', 'II', 'II',␣
 ↪'III', 'I', 'III', 'II', 'II', 'I']
}

df = pd.DataFrame(data)
df
```

[23]:
```
              Name Birth_Month Gender Pass_Division
0    Mudit Chauhan    December      M           III
1     Seema Chopra     January      F            II
2       Rani Gupta       March      F             I
3   Aditya Narayan     October      M             I
4    Sanjeev Sahni    February      M            II
5    Prakash Kumar    December      M           III
6     Ritu Agarwal   September      F             I
7      Akshay Goel      August      M             I
8   Meeta Kulkarni        July      F            II
9     Preeti Ahuja    November      F            II
```

```
10  Sunil Das Gupta       April      M           III
11     Sonali Sapre     January      F             I
12    Rashmi Talwar        June      F           III
13     Ashish Dubey         May      M            II
14     Kiran Sharma    February      F            II
15    Sameer Bansal     October      M             I
```

[24]:

```
#PART A
df_encoded = pd.get_dummies(df, columns=['Gender','Pass_Division'])
df_encoded
```

[24]:

```
                Name Birth_Month  Gender_F  Gender_M  Pass_Division_I  \
0     Mudit Chauhan    December     False      True            False
1      Seema Chopra     January      True     False            False
2        Rani Gupta       March      True     False             True
3    Aditya Narayan     October     False      True             True
4     Sanjeev Sahni    February     False      True            False
5     Prakash Kumar    December     False      True            False
6      Ritu Agarwal   September      True     False             True
7       Akshay Goel      August     False      True             True
8    Meeta Kulkarni        July      True     False            False
9      Preeti Ahuja    November      True     False            False
10  Sunil Das Gupta       April     False      True            False
11     Sonali Sapre     January      True     False             True
12    Rashmi Talwar        June      True     False            False
13     Ashish Dubey         May     False      True            False
14     Kiran Sharma    February      True     False            False
15    Sameer Bansal     October     False      True             True

    Pass_Division_II  Pass_Division_III
0              False               True
1               True              False
2              False              False
3              False              False
4               True              False
5              False               True
6              False              False
7              False              False
8               True              False
9               True              False
10             False               True
11             False              False
12             False               True
13              True              False
14              True              False
15             False              False
```

[25]:
```
#PART B
month_order=['January', 'February', 'March',␣
 ↪'April','May','June','July','August','September','October','November','December']
df_encoded['Birth_Month'] = pd.Categorical(df_encoded['Birth_Month'],␣
 ↪categories=month_order, ordered=True)
df_sorted = df_encoded.sort_values('Birth_Month')
df_sorted
```

[25]:
```
               Name Birth_Month  Gender_F  Gender_M  Pass_Division_I  \
1      Seema Chopra     January      True     False            False
11     Sonali Sapre     January      True     False             True
4     Sanjeev Sahni    February     False      True            False
14     Kiran Sharma    February      True     False            False
2        Rani Gupta       March      True     False             True
10  Sunil Das Gupta       April     False      True            False
13     Ashish Dubey         May     False      True            False
12    Rashmi Talwar        June      True     False            False
8    Meeta Kulkarni        July      True     False            False
7       Akshay Goel      August     False      True             True
6      Ritu Agarwal   September      True     False             True
3    Aditya Narayan     October     False      True             True
15    Sameer Bansal     October     False      True             True
9      Preeti Ahuja    November      True     False            False
0     Mudit Chauhan    December     False      True            False
5     Prakash Kumar    December     False      True            False

    Pass_Division_II  Pass_Division_III
1               True              False
11             False              False
4               True              False
14              True              False
2              False              False
10             False               True
13              True              False
12             False               True
8               True              False
7              False              False
6              False              False
3              False              False
15             False              False
9               True              False
0              False               True
5              False               True
```

Q8

[26]:
```
data = {
    'Name': ['Shah', 'Vats', 'Vats', 'Kumar', 'Vats', 'Kumar', 'Shah', 'Shah',␣
 ↪'Kumar', 'Vats'],
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Female', 'Male', 'Male',␣
 ↪'Female', 'Female', 'Male'],
    'MonthlyIncome': [114000.00, 65000.00, 43150.00, 69500.00, 155000.00,␣
 ↪103000.00, 55000.00, 112400.00, 81030.00, 71900.00]
}
df = pd.DataFrame(data)
df
```

[26]:
```
    Name  Gender  MonthlyIncome
0   Shah    Male       114000.0
1   Vats    Male        65000.0
2   Vats  Female        43150.0
3  Kumar  Female        69500.0
4   Vats  Female       155000.0
5  Kumar    Male       103000.0
6   Shah    Male        55000.0
7   Shah  Female       112400.0
8  Kumar  Female        81030.0
9   Vats    Male        71900.0
```

[27]:
```
#PART A
familywise_gross_income = df.groupby('Name')['MonthlyIncome'].sum()
familywise_gross_income
```

[27]:
```
Name
Kumar    253530.0
Shah     281400.0
Vats     335050.0
Name: MonthlyIncome, dtype: float64
```

[28]:
```
#PART B
familywise_max_income = df.groupby("Name")['MonthlyIncome'].max()
familywise_max_income
```

[28]:
```
Name
Kumar    103000.0
Shah     114000.0
Vats     155000.0
Name: MonthlyIncome, dtype: float64
```

[29]:
```
#PART C
df[df['MonthlyIncome']>60000.0]
```

```
[29]:      Name  Gender  MonthlyIncome
      0    Shah    Male       114000.0
      1    Vats    Male        65000.0
      3   Kumar  Female        69500.0
      4    Vats  Female       155000.0
      5   Kumar    Male       103000.0
      7    Shah  Female       112400.0
      8   Kumar  Female        81030.0
      9    Vats    Male        71900.0
```

```
[30]: #PART D
      shah_female_avg_income = df[(df['Name'] == 'Shah') & (df['Gender'] ==␣
       ↪'Female')]['MonthlyIncome'].mean()
      shah_female_avg_income
```

```
[30]: 112400.0
```