**Assignment - 4**

**Customer Segmentation Analysis**

| **Assignment Date** | **26th Oct 2022** |
| --- | --- |
| **Student Name** | **Keerthanaa Anand** |
| **Team ID** | **PNT2022TMID04987** |
| **Maximum Marks** | **2 Marks** |

**Customer Segmentation Analysis**

**1.Downloading and Loading the Dataset**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

data = pd.read\_csv('Mall\_Customers.csv')

data.head()

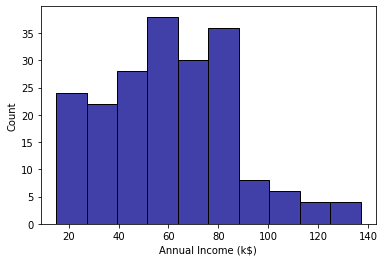
CustomerID Gender Age Annual Income (k$) Spending Score (1-100) 0 1 Male 19 15 39 1 2 Male 21 15 81 2 3 Female 20 16 6 3 4 Female 23 16 77 4 5 Female 31 17 40

**2.Performing Visualizations**

**(i) Univariate Analysis**

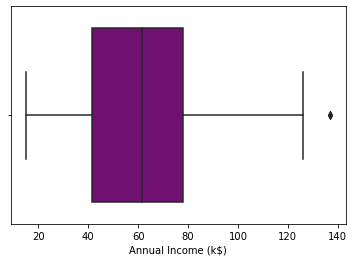
**Histplot**

sns.distplot(data['Annual Income (k$)'], color="darkblue") <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Count'>



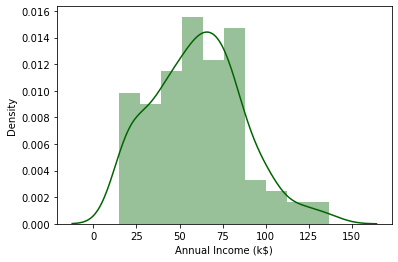
**Box Plot**

sns.boxplot(data['Annual Income (k$)'], color="purple") <AxesSubplot:xlabel='Annual Income (k$)'>



**Dist Plot**

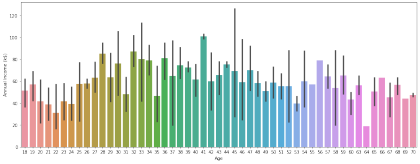
sns.distplot(data['Annual Income (k$)'], color="dark green")

<AxesSubplot:xlabel='Annual Income (k$)', ylabel='Density'> **(ii) Bi-variate Analysis**

**Barplot**

plt.figure(figsize=(16,6))

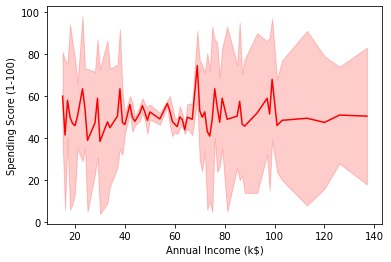
sns.barplot(data['Age'],data['Annual Income (k$)'])

<AxesSubplot:xlabel='Age', ylabel='Annual Income (k$)'> 

**Lineplot**

sns.lineplot(data['Annual Income (k$)'], data['Spending Score (1- 100)'], color="red")

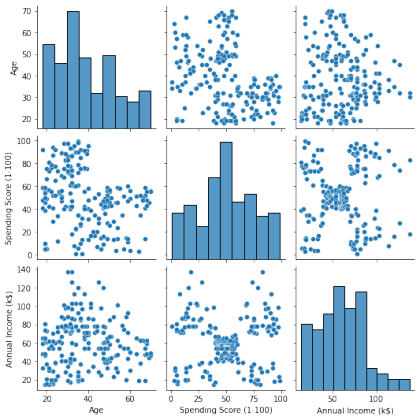
<AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1- 100)'>



**(iii) Multivariate Analysis**

sns.pairplot(data=data[["Age", "Gender","Spending Score (1- 100)","Annual Income (k$)"]])

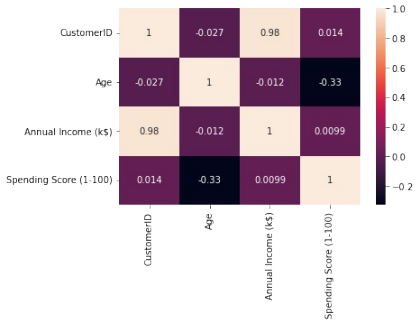
<seaborn.axisgrid.PairGrid at 0x298ddce51c0>



**Correlation between the different attributes**

sns.heatmap(data.corr(),annot=True)

<AxesSubplot:>



**3.Performing Descriptive Statistics on the Dataset**

data.describe()

CustomerID Age Annual Income (k$) Spending Score (1- 100)

count 200.000000 200.000000 200.000000

200.000000

mean 100.500000 38.850000 60.560000

50.200000

std 57.879185 13.969007 26.264721

25.823522

min 1.000000 18.000000 15.000000

1.000000

25% 50.750000 28.750000 41.500000

34.750000

50% 100.500000 36.000000 61.500000

50.000000

75% 150.250000 49.000000 78.000000

73.000000

max 200.000000 70.000000 137.000000

99.000000

data.info

data.shape

(200, 5)

**4.Check for Missing values**

data.isnull().any() *#Inference: The dataset has no null values*

CustomerID False

Gender False

Age False

Annual Income (k$) False

Spending Score (1-100) False

dtype: bool

data.drop('CustomerID',axis=1,inplace=True)

data.head()

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

**5.Finding the outliers and replacing them**

**for** i **in** data:

**if** data[i].dtype=='int64':

q1=data[i].quantile(0.25)

q3=data[i].quantile(0.75)

iqr=q3-q1

upper=q3+1.5\*iqr

lower=q1-1.5\*iqr

data[i]=np.where(data[i] >upper, upper, data[i]) data[i]=np.where(data[i] <lower, lower, data[i])

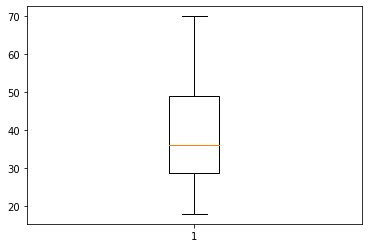
**After removing outliers, boxplot will be like**

plt.boxplot(data['Age'])

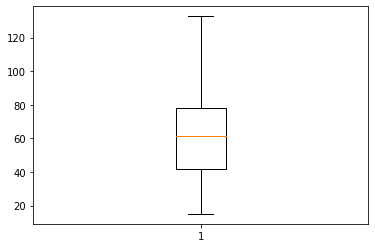
{'whiskers': [<matplotlib.lines.Line2D at 0x298de535e20>, <matplotlib.lines.Line2D at 0x298de545130>],

'caps': [<matplotlib.lines.Line2D at 0x298de545400>, <matplotlib.lines.Line2D at 0x298de5456d0>],

'boxes': [<matplotlib.lines.Line2D at 0x298de535b50>], 'medians': [<matplotlib.lines.Line2D at 0x298de545940>], 'fliers': [<matplotlib.lines.Line2D at 0x298de545c70>], 'means': []}

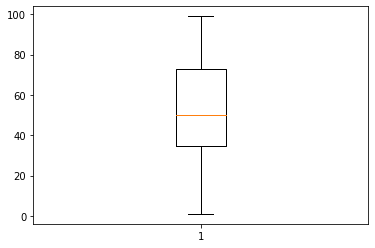
plt.boxplot(data['Annual Income (k$)'])

{'whiskers': [<matplotlib.lines.Line2D at 0x298de59d9a0>, <matplotlib.lines.Line2D at 0x298de59dc10>], 'caps': [<matplotlib.lines.Line2D at 0x298de59dee0>, <matplotlib.lines.Line2D at 0x298de5ac1f0>], 'boxes': [<matplotlib.lines.Line2D at 0x298de5904f0>], 'medians': [<matplotlib.lines.Line2D at 0x298de5ac4c0>], 'fliers': [<matplotlib.lines.Line2D at 0x298de5ac790>], 'means': []}



plt.boxplot(data['Spending Score (1-100)'])

{'whiskers': [<matplotlib.lines.Line2D at 0x298de6034c0>, <matplotlib.lines.Line2D at 0x298de603790>], 'caps': [<matplotlib.lines.Line2D at 0x298de603a60>, <matplotlib.lines.Line2D at 0x298de603d30>], 'boxes': [<matplotlib.lines.Line2D at 0x298de6031f0>], 'medians': [<matplotlib.lines.Line2D at 0x298de60f040>], 'fliers': [<matplotlib.lines.Line2D at 0x298de60f310>], 'means': []}



**6.Checking for categorical columns and performing encoding** from sklearn.preprocessing import LabelEncoder l\_en = LabelEncoder()

data['Gender'] = l\_en.fit\_transform(data['Gender']) data.head()

Gender Age Annual Income (k$) Spending Score (1-100) 0 1 19.0 15.0 39.0 1 1 21.0 15.0 81.0 2 0 20.0 16.0 6.0 3 0 23.0 16.0 77.0 4 0 31.0 17.0 40.0

**7.Scaling the data**

from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler()

data\_scaled = scaler.fit\_transform(data)

data\_scaled[0:5]

array([[1. , 0.01923077, 0. , 0.3877551 ], [1. , 0.05769231, 0. , 0.81632653], [0. , 0.03846154, 0.00849257, 0.05102041], [0. , 0.09615385, 0.00849257, 0.7755102 ], [0. , 0.25 , 0.01698514, 0.39795918]])

**8.Performing any of the clustering algorithms**

from sklearn.cluster import KMeans

km = KMeans()

res = km.fit\_predict(data\_scaled)

res

array([4, 4, 1, 1, 1, 1, 5, 1, 0, 1, 0, 1, 5, 1, 4, 4, 1, 4, 0, 1, 4, 4,

5, 4, 5, 4, 5, 4, 5, 1, 0, 1, 0, 4, 5, 1, 5, 1, 5, 1, 5, 4, 0, 1,

5, 1, 5, 1, 1, 1, 5, 4, 1, 0, 5, 0, 5, 0, 1, 0, 0, 4, 5, 5, 0, 4,

5, 5, 4, 1, 0, 5, 5, 5, 0, 4, 5, 4, 1, 5, 0, 4, 0, 5, 1, 0, 5, 1,

1, 5, 5, 4, 0, 5, 1, 4, 5, 1, 0, 4, 1, 5, 0, 4, 0, 1, 5, 0, 0, 0,

0, 1, 5, 4, 1, 1, 5, 5, 5, 5, 4, 5, 5, 2, 1, 6, 3, 2, 0, 2, 3, 2,

1, 6, 3, 6, 7, 2, 3, 6, 7, 2, 1, 6, 3, 2, 3, 6, 7, 2, 3, 2, 7, 6,

7, 6, 3, 6, 3, 6, 5, 6, 3, 6, 3, 6, 3, 6, 7, 2, 3, 2, 3, 2, 7, 6,

3, 2, 3, 2, 7, 6, 3, 6, 7, 2, 7, 2, 7, 6, 7, 6, 3, 6, 7, 6, 7, 2,

3, 2])

data1 = pd.DataFrame(data\_scaled, columns = data.columns) data1.head()

Gender Age Annual Income (k$) Spending Score (1-100) 0 1.0 0.019231 0.000000 0.387755 1 1.0 0.057692 0.000000 0.816327 2 0.0 0.038462 0.008493 0.051020 3 0.0 0.096154 0.008493 0.775510 4 0.0 0.250000 0.016985 0.397959

data1['kclus'] = pd.Series(res)

data1.head()

Gender Age Annual Income (k$) Spending Score (1-100) kclus 0 1.0 0.019231 0.000000 0.387755 4 1 1.0 0.057692 0.000000 0.816327 4 2 0.0 0.038462 0.008493 0.051020 1 3 0.0 0.096154 0.008493 0.775510 1 4 0.0 0.250000 0.016985 0.397959 1

data1['kclus'].unique()

array([4, 1, 5, 0, 2, 6, 3, 7])

data1['kclus'].value\_counts()

5 40

1 37

0 26

4 25

6 21

3 19

2 18

7 14

Name: kclus, dtype: int64

import matplotlib.pyplot as plt

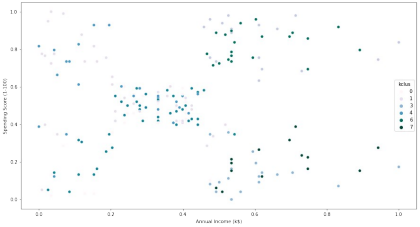
fig,ax = plt.subplots(figsize=(15,8))

sns.scatterplot(x=data1['Annual Income (k$)'],

y=data1['Spending Score (1-100)'], hue=data1['kclus'],

palette='PuBuGn')

plt.show()



ind = data1.iloc[:,0:4]

ind.head()

Gender Age Annual Income (k$) Spending Score (1-100) 0 1.0 0.019231 0.000000 0.387755 1 1.0 0.057692 0.000000 0.816327 2 0.0 0.038462 0.008493 0.051020

3 0.0 0.096154 0.008493 0.775510 4 0.0 0.250000 0.016985 0.397959

dep = data1.iloc[:,4:]

dep.head()

kclus

0 4

1 4

2 1

3 1

4 1

**9.ADD THE CLUSTER DATA WITH THE PRIMARY DATASET**

df['clust']=mb

df.head()

|  | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) | Gender\_Female | Gender\_Male | clust |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 19 | 15.0 | 39 | 0 | 1 | 2 |
| 1 | 2 | 21 | 15.0 | 81 | 0 | 1 | 2 |
| 2 | 3 | 20 | 16.0 | 6 | 1 | 0 | 3 |
| 3 | 4 | 23 | 16.0 | 77 | 1 | 0 | 1 |
| 4 | 5 | 31 | 17.0 | 40 | 1 | 0 | 1 |

# 10.SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

*# dependent variable*

y**=**df['clust']**.**values

y

array([2, 2, 3, 1, 1, 1, 3, 1, 0, 1, 0, 1, 3, 1, 0, 2, 3, 2, 0, 1, 0, 2,

3, 2, 3, 2, 3, 2, 3, 1, 0, 1, 0, 2, 3, 1, 3, 1, 3, 1, 3, 2, 0, 1,

3, 1, 3, 1, 1, 1, 3, 2, 1, 0, 3, 0, 3, 0, 1, 0, 0, 2, 3, 3, 0, 2,

3, 3, 2, 1, 0, 3, 3, 3, 0, 2, 3, 0, 1, 3, 0, 2, 0, 3, 1, 0, 3, 1,

1, 3, 3, 2, 0, 3, 1, 2, 3, 1, 0, 2, 1, 3, 0, 2, 0, 1, 3, 0, 0, 0,

0, 1, 3, 2, 1, 1, 3, 3, 3, 3, 2, 3, 1, 2, 1, 1, 0, 2, 0, 2, 0, 2,

1, 1, 0, 1, 3, 2, 0, 1, 3, 2, 1, 1, 0, 2, 0, 1, 3, 2, 0, 2, 3, 1,

3, 1, 0, 1, 0, 1, 3, 1, 0, 1, 0, 1, 0, 1, 3, 2, 0, 2, 0, 2, 3, 1,

0, 2, 0, 2, 3, 1, 0, 1, 3, 2, 3, 2, 3, 1, 3, 1, 0, 1, 3, 1, 3, 2,

0, 2], dtype=int32)

*#independent variable*

x**=**df**.**drop(columns**=**['clust','CustomerID'],axis**=**1)**.**values

x

array([[ 19. , 15. , 39. , 0. , 1. ],

[ 21. , 15. , 81. , 0. , 1. ],

[ 20. , 16. , 6. , 1. , 0. ],

[ 23. , 16. , 77. , 1. , 0. ],

[ 31. , 17. , 40. , 1. , 0. ],

[ 22. , 17. , 76. , 1. , 0. ],

[ 35. , 18. , 6. , 1. , 0. ],

[ 23. , 18. , 94. , 1. , 0. ],

[ 64. , 19. , 3. , 0. , 1. ],

[ 30. , 19. , 72. , 1. , 0. ],

[ 67. , 19. , 14. , 0. , 1. ],

[ 35. , 19. , 99. , 1. , 0. ],

[ 58. , 20. , 15. , 1. , 0. ],

[ 24. , 20. , 77. , 1. , 0. ],

[ 37. , 20. , 13. , 0. , 1. ],

[ 22. , 20. , 79. , 0. , 1. ],

[ 35. , 21. , 35. , 1. , 0. ],

[ 20. , 21. , 66. , 0. , 1. ],

[ 52. , 23. , 29. , 0. , 1. ],

[ 35. , 23. , 98. , 1. , 0. ],

[ 35. , 24. , 35. , 0. , 1. ],

[ 25. , 24. , 73. , 0. , 1. ],

[ 46. , 25. , 5. , 1. , 0. ],

[ 31. , 25. , 73. , 0. , 1. ],

[ 54. , 28. , 14. , 1. , 0. ],

[ 29. , 28. , 82. , 0. , 1. ],

[ 45. , 28. , 32. , 1. , 0. ],

[ 35. , 28. , 61. , 0. , 1. ],

[ 40. , 29. , 31. , 1. , 0. ],

[ 23. , 29. , 87. , 1. , 0. ],

[ 60. , 30. , 4. , 0. , 1. ],

[ 21. , 30. , 73. , 1. , 0. ],

[ 53. , 33. , 4. , 0. , 1. ],

[ 18. , 33. , 92. , 0. , 1. ],

[ 49. , 33. , 14. , 1. , 0. ],

[ 21. , 33. , 81. , 1. , 0. ],

[ 42. , 34. , 17. , 1. , 0. ],

[ 30. , 34. , 73. , 1. , 0. ],

[ 36. , 37. , 26. , 1. , 0. ],

[ 20. , 37. , 75. , 1. , 0. ],

[ 65. , 38. , 35. , 1. , 0. ],

[ 24. , 38. , 92. , 0. , 1. ],

[ 48. , 39. , 36. , 0. , 1. ],

[ 31. , 39. , 61. , 1. , 0. ],

[ 49. , 39. , 28. , 1. , 0. ],

[ 24. , 39. , 65. , 1. , 0. ],

[ 50. , 40. , 55. , 1. , 0. ],

[ 27. , 40. , 47. , 1. , 0. ],

[ 29. , 40. , 42. , 1. , 0. ],

[ 31. , 40. , 42. , 1. , 0. ],

[ 49. , 42. , 52. , 1. , 0. ],

[ 33. , 42. , 60. , 0. , 1. ],

[ 31. , 43. , 54. , 1. , 0. ],

[ 59. , 43. , 60. , 0. , 1. ],

[ 50. , 43. , 45. , 1. , 0. ],

[ 47. , 43. , 41. , 0. , 1. ],

[ 51. , 44. , 50. , 1. , 0. ],

[ 69. , 44. , 46. , 0. , 1. ],

[ 27. , 46. , 51. , 1. , 0. ],

[ 53. , 46. , 46. , 0. , 1. ],

[ 70. , 46. , 56. , 0. , 1. ],

[ 19. , 46. , 55. , 0. , 1. ],

[ 67. , 47. , 52. , 1. , 0. ],

[ 54. , 47. , 59. , 1. , 0. ],

[ 63. , 48. , 51. , 0. , 1. ],

[ 18. , 48. , 59. , 0. , 1. ],

[ 43. , 48. , 50. , 1. , 0. ],

[ 68. , 48. , 48. , 1. , 0. ],

[ 19. , 48. , 59. , 0. , 1. ],

[ 32. , 48. , 47. , 1. , 0. ],

[ 70. , 49. , 55. , 0. , 1. ],

[ 47. , 49. , 42. , 1. , 0. ],

[ 60. , 50. , 49. , 1. , 0. ],

[ 60. , 50. , 56. , 1. , 0. ],

[ 59. , 54. , 47. , 0. , 1. ],

[ 26. , 54. , 54. , 0. , 1. ],

[ 45. , 54. , 53. , 1. , 0. ],

[ 40. , 54. , 48. , 0. , 1. ],

[ 23. , 54. , 52. , 1. , 0. ],

[ 49. , 54. , 42. , 1. , 0. ],

[ 57. , 54. , 51. , 0. , 1. ],

[ 38. , 54. , 55. , 0. , 1. ],

[ 67. , 54. , 41. , 0. , 1. ],

[ 46. , 54. , 44. , 1. , 0. ],

[ 21. , 54. , 57. , 1. , 0. ],

[ 48. , 54. , 46. , 0. , 1. ],

[ 55. , 57. , 58. , 1. , 0. ],

[ 22. , 57. , 55. , 1. , 0. ],

[ 34. , 58. , 60. , 1. , 0. ],

[ 50. , 58. , 46. , 1. , 0. ],

[ 68. , 59. , 55. , 1. , 0. ],

[ 18. , 59. , 41. , 0. , 1. ],

[ 48. , 60. , 49. , 0. , 1. ],

[ 40. , 60. , 40. , 1. , 0. ],

[ 32. , 60. , 42. , 1. , 0. ],

[ 24. , 60. , 52. , 0. , 1. ],

[ 47. , 60. , 47. , 1. , 0. ],

[ 27. , 60. , 50. , 1. , 0. ],

[ 48. , 61. , 42. , 0. , 1. ],

[ 20. , 61. , 49. , 0. , 1. ],

[ 23. , 62. , 41. , 1. , 0. ],

[ 49. , 62. , 48. , 1. , 0. ],

[ 67. , 62. , 59. , 0. , 1. ],

[ 26. , 62. , 55. , 0. , 1. ],

[ 49. , 62. , 56. , 0. , 1. ],

[ 21. , 62. , 42. , 1. , 0. ],

[ 66. , 63. , 50. , 1. , 0. ],

[ 54. , 63. , 46. , 0. , 1. ],

[ 68. , 63. , 43. , 0. , 1. ],

[ 66. , 63. , 48. , 0. , 1. ],

[ 65. , 63. , 52. , 0. , 1. ],

[ 19. , 63. , 54. , 1. , 0. ],

[ 38. , 64. , 42. , 1. , 0. ],

[ 19. , 64. , 46. , 0. , 1. ],

[ 18. , 65. , 48. , 1. , 0. ],

[ 19. , 65. , 50. , 1. , 0. ],

[ 63. , 65. , 43. , 1. , 0. ],

[ 49. , 65. , 59. , 1. , 0. ],

[ 51. , 67. , 43. , 1. , 0. ],

[ 50. , 67. , 57. , 1. , 0. ],

[ 27. , 67. , 56. , 0. , 1. ],

[ 38. , 67. , 40. , 1. , 0. ],

[ 40. , 69. , 58. , 1. , 0. ],

[ 39. , 69. , 91. , 0. , 1. ],

[ 23. , 70. , 29. , 1. , 0. ],

[ 31. , 70. , 77. , 1. , 0. ],

[ 43. , 71. , 35. , 0. , 1. ],

[ 40. , 71. , 95. , 0. , 1. ],

[ 59. , 71. , 11. , 0. , 1. ],

[ 38. , 71. , 75. , 0. , 1. ],

[ 47. , 71. , 9. , 0. , 1. ],

[ 39. , 71. , 75. , 0. , 1. ],

[ 25. , 72. , 34. , 1. , 0. ],

[ 31. , 72. , 71. , 1. , 0. ],

[ 20. , 73. , 5. , 0. , 1. ],

[ 29. , 73. , 88. , 1. , 0. ],

[ 44. , 73. , 7. , 1. , 0. ],

[ 32. , 73. , 73. , 0. , 1. ],

[ 19. , 74. , 10. , 0. , 1. ],

[ 35. , 74. , 72. , 1. , 0. ],

[ 57. , 75. , 5. , 1. , 0. ],

[ 32. , 75. , 93. , 0. , 1. ],

[ 28. , 76. , 40. , 1. , 0. ],

[ 32. , 76. , 87. , 1. , 0. ],

[ 25. , 77. , 12. , 0. , 1. ],

[ 28. , 77. , 97. , 0. , 1. ],

[ 48. , 77. , 36. , 0. , 1. ],

[ 32. , 77. , 74. , 1. , 0. ],

[ 34. , 78. , 22. , 1. , 0. ],

[ 34. , 78. , 90. , 0. , 1. ],

[ 43. , 78. , 17. , 0. , 1. ],

[ 39. , 78. , 88. , 0. , 1. ],

[ 44. , 78. , 20. , 1. , 0. ],

[ 38. , 78. , 76. , 1. , 0. ],

[ 47. , 78. , 16. , 1. , 0. ],

[ 27. , 78. , 89. , 1. , 0. ],

[ 37. , 78. , 1. , 0. , 1. ],

[ 30. , 78. , 78. , 1. , 0. ],

[ 34. , 78. , 1. , 0. , 1. ],

[ 30. , 78. , 73. , 1. , 0. ],

[ 56. , 79. , 35. , 1. , 0. ],

[ 29. , 79. , 83. , 1. , 0. ],

[ 19. , 81. , 5. , 0. , 1. ],

[ 31. , 81. , 93. , 1. , 0. ],

[ 50. , 85. , 26. , 0. , 1. ],

[ 36. , 85. , 75. , 1. , 0. ],

[ 42. , 86. , 20. , 0. , 1. ],

[ 33. , 86. , 95. , 1. , 0. ],

[ 36. , 87. , 27. , 1. , 0. ],

[ 32. , 87. , 63. , 0. , 1. ],

[ 40. , 87. , 13. , 0. , 1. ],

[ 28. , 87. , 75. , 0. , 1. ],

[ 36. , 87. , 10. , 0. , 1. ],

[ 36. , 87. , 92. , 0. , 1. ],

[ 52. , 88. , 13. , 1. , 0. ],

[ 30. , 88. , 86. , 1. , 0. ],

[ 58. , 88. , 15. , 0. , 1. ],

[ 27. , 88. , 69. , 0. , 1. ],

[ 59. , 93. , 14. , 0. , 1. ],

[ 35. , 93. , 90. , 0. , 1. ],

[ 37. , 97. , 32. , 1. , 0. ],

[ 32. , 97. , 86. , 1. , 0. ],

[ 46. , 98. , 15. , 0. , 1. ],

[ 29. , 98. , 88. , 1. , 0. ],

[ 41. , 99. , 39. , 1. , 0. ],

[ 30. , 99. , 97. , 0. , 1. ],

[ 54. , 101. , 24. , 1. , 0. ],

[ 28. , 101. , 68. , 0. , 1. ],

[ 41. , 103. , 17. , 1. , 0. ],

[ 36. , 103. , 85. , 1. , 0. ],

[ 34. , 103. , 23. , 1. , 0. ],

[ 32. , 103. , 69. , 1. , 0. ],

[ 33. , 113. , 8. , 0. , 1. ],

[ 38. , 113. , 91. , 1. , 0. ],

[ 47. , 120. , 16. , 1. , 0. ],

[ 35. , 120. , 79. , 1. , 0. ],

[ 45. , 126. , 28. , 1. , 0. ],

[ 32. , 126. , 74. , 0. , 1. ],

[ 32. , 60.56, 18. , 0. , 1. ],

[ 30. , 60.56, 83. , 0. , 1. ]])

**11.Splitting dataset into train and test data**

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test =

train\_test\_split(ind,dep,test\_size=0.3,random\_state=1) x\_train.head()

Gender Age Annual Income (k$) Spending Score (1-100) 116 0.0 0.865385 0.424628 0.428571 67 0.0 0.961538 0.280255 0.479592 78 0.0 0.096154 0.331210 0.520408 42 1.0 0.576923 0.203822 0.357143 17 1.0 0.038462 0.050955 0.663265

x\_test.head()

Gender Age Annual Income (k$) Spending Score (1-100) 58 0.0 0.173077 0.263270 0.510204 40 0.0 0.903846 0.195329 0.346939 34 0.0 0.596154 0.152866 0.132653 102 1.0 0.942308 0.399151 0.591837 184 0.0 0.442308 0.713376 0.387755

y\_train.head()

kclus

116 5

67 5

78 1

42 0

17 4

y\_test.head()

kclus

58 1

40 5

34 5

102 0

184 7

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

lr.fit(x\_train,y\_train)

LinearRegression()

pred\_test = lr.predict(x\_test)

pred\_test[0:5]

array([[3.95042116],

[3.52672483],

[3.51129248],

[2.16561932],

[5.40782042]])

**12.BUILD THE MODEL**

from sklearn.ensemble import RandomForestClassifier

rf**=**RandomForestClassifier()

# 13.TRAIN THE MODEL

rf**.**fit(x\_train,y\_train)

RandomForestClassifier()

**14.TEST THE MODEL**

pred**=**rf**.**predict(x\_test)

pred

array([0, 0, 0, 0, 2, 0, 1, 0, 3, 3, 2, 1, 3, 3, 0, 1, 2, 0, 1, 0, 1, 2,

1, 1, 0, 2, 2, 2, 3, 3, 3, 0, 0, 1, 3, 1, 2, 1, 3, 3], dtype=int32)

**15.Measuring the performance using Evaluation Metrics**

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

mse = mean\_squared\_error(pred\_test,y\_test)

print("The Mean squared error is: ", mse)

The Mean squared error is: 3.785711485954305

rmse = np.sqrt(mse)

print("The Root mean squared error is: ", rmse)

The Root mean squared error is: 1.945690490790944

mae = mean\_absolute\_error(pred\_test,y\_test)

print("The Mean absolute error is: ", mae)

The Mean absolute error is: 1.7183473427088407

acc = lr.score(x\_test,y\_test)

print("The accuracy is: ", acc)

The accuracy is: 0.2718229670102855