assignment-5

October 3, 2023

1 Kaggle Connection & DataFrame setup

[137]:
```
!pip install -q kaggle
```

[138]:
```
!mkdir ~/.kaggle
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

[139]:
```
!cp kaggle.json ~/.kaggle
```

[140]:
```
! kaggle datasets download -d vjchoudhary7/
↪customer-segmentation-tutorial-in-python
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix
this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
customer-segmentation-tutorial-in-python.zip: Skipping, found more recently
modified local copy (use --force to force download)
```

[141]:
```
!unzip /content/customer-segmentation-tutorial-in-python.zip
```

```
Archive:  /content/customer-segmentation-tutorial-in-python.zip
replace Mall_Customers.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
```

2 Pre-Processing

[142]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[143]:
```
df = pd.read_csv('./Mall_Customers.csv')
df.head()
```

[143]:

	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
```

[144]:
```
df.describe()
```

[144]:
```
       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
```

[145]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

[146]:
```
df.isnull().values.any()
```

[146]:
```
False
```

[147]:
```
df.shape
```

[147]:
```
(200, 5)
```

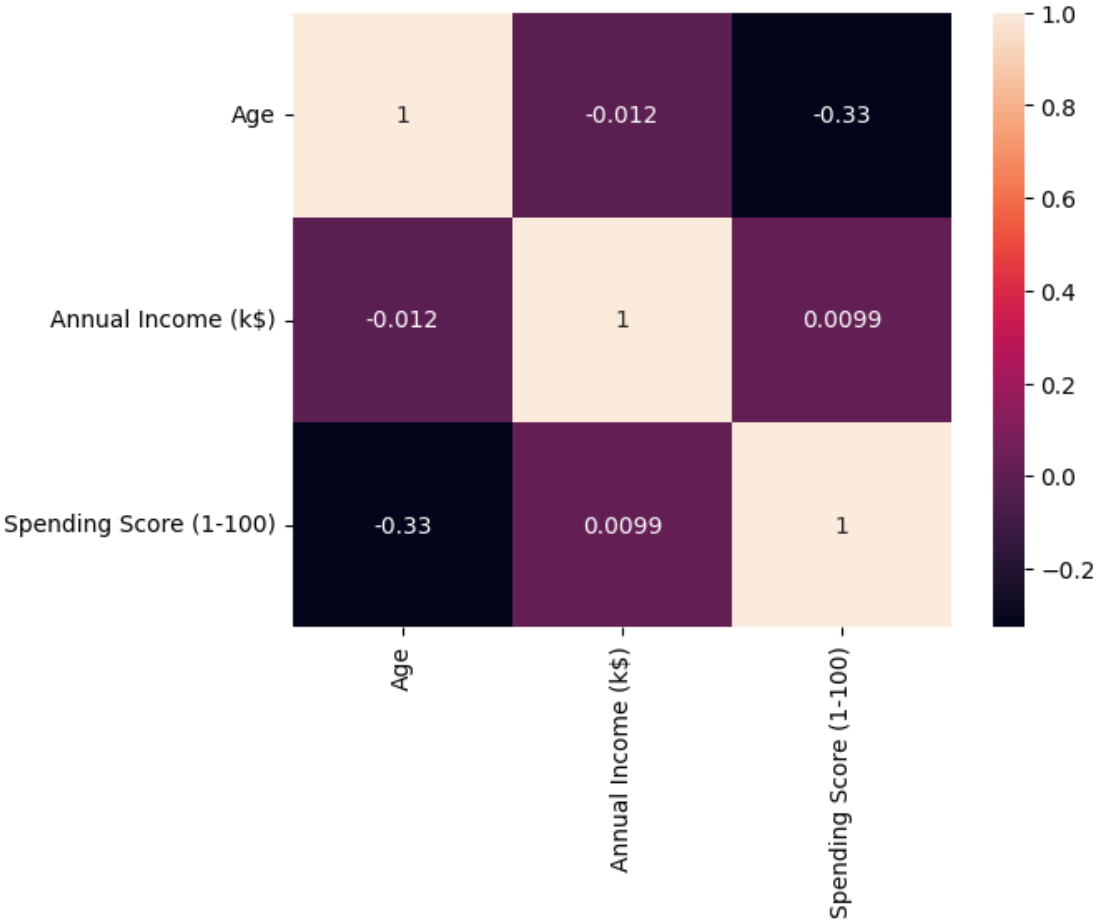
[148]:
```
# Dropping 'CustomerID' as it has no impact or connection to dataset or data␣
↪values
df.drop(['CustomerID'], axis=1, inplace=True)
```

[149]:
```
sns.heatmap(df.corr(), annot=True)
```

```
<ipython-input-149-6dc1c4c1753e>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
```

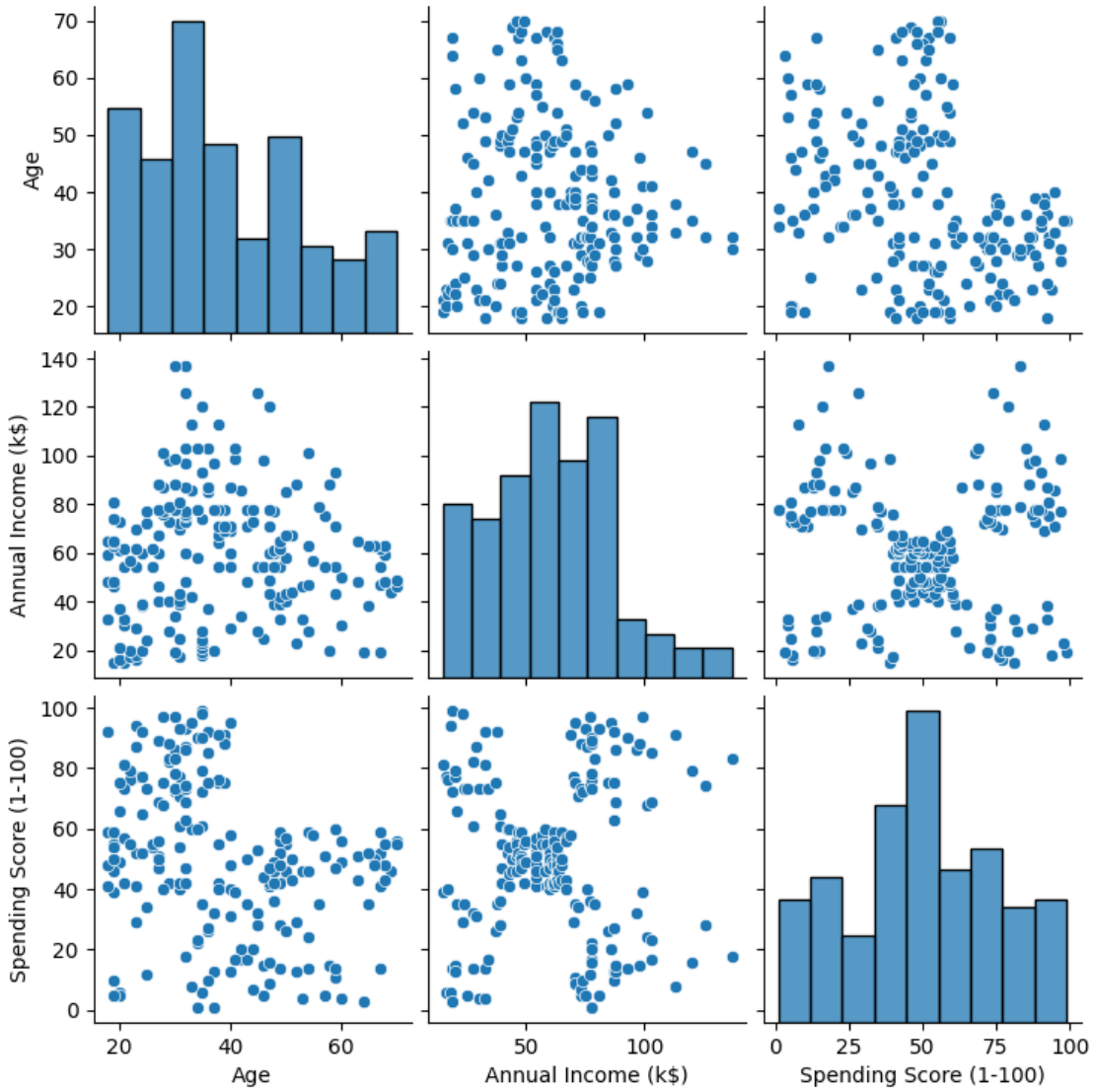
```
to silence this warning.
  sns.heatmap(df.corr(), annot=True)
```

[149]: <Axes: >



[150]:
```
sns.pairplot(df)
```

[150]: <seaborn.axisgrid.PairGrid at 0x7b174a045600>



3 Converting Categorical Data (Columns) to Numerical

[151]:
```
df['Gender'].value_counts()
```

[151]:
```
Female    112
Male       88
Name: Gender, dtype: int64
```

[152]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[153]:
```
# Label Encoding 'Gender' column
# '1' == 'Male' && 0 == 'Female'
df['Gender'] = le.fit_transform(df.Gender)
```

[154]:
```
df.head()
```

[154]:

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

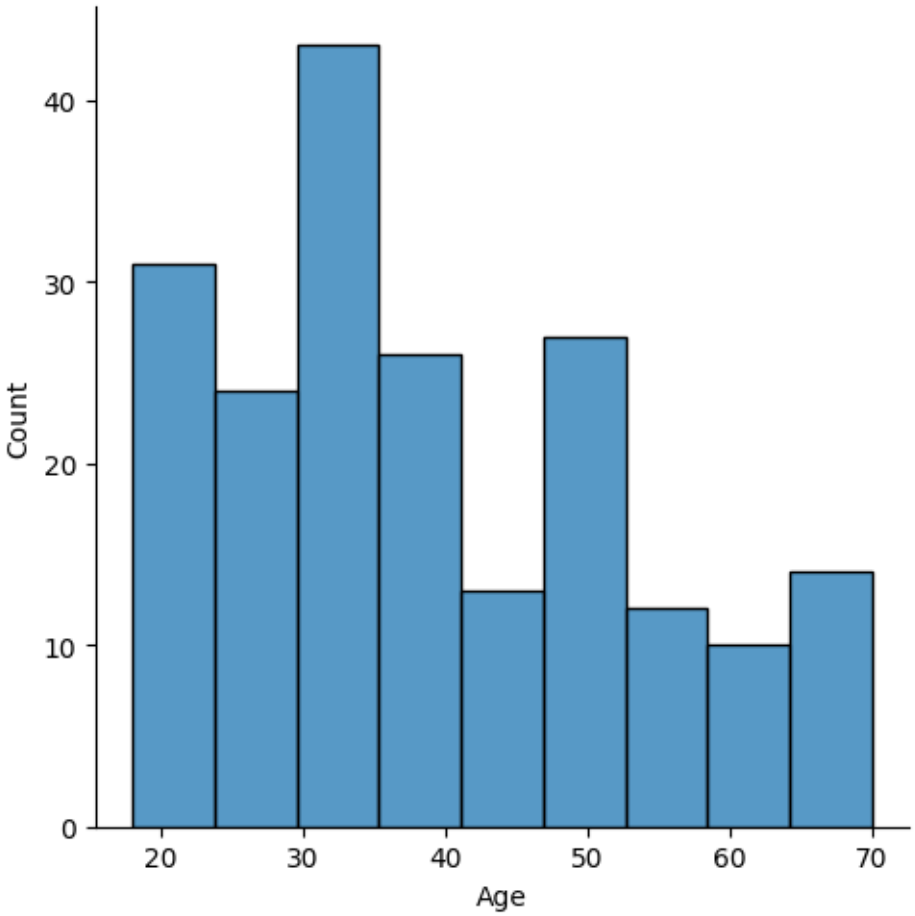
[155]:
```
df.info()
```

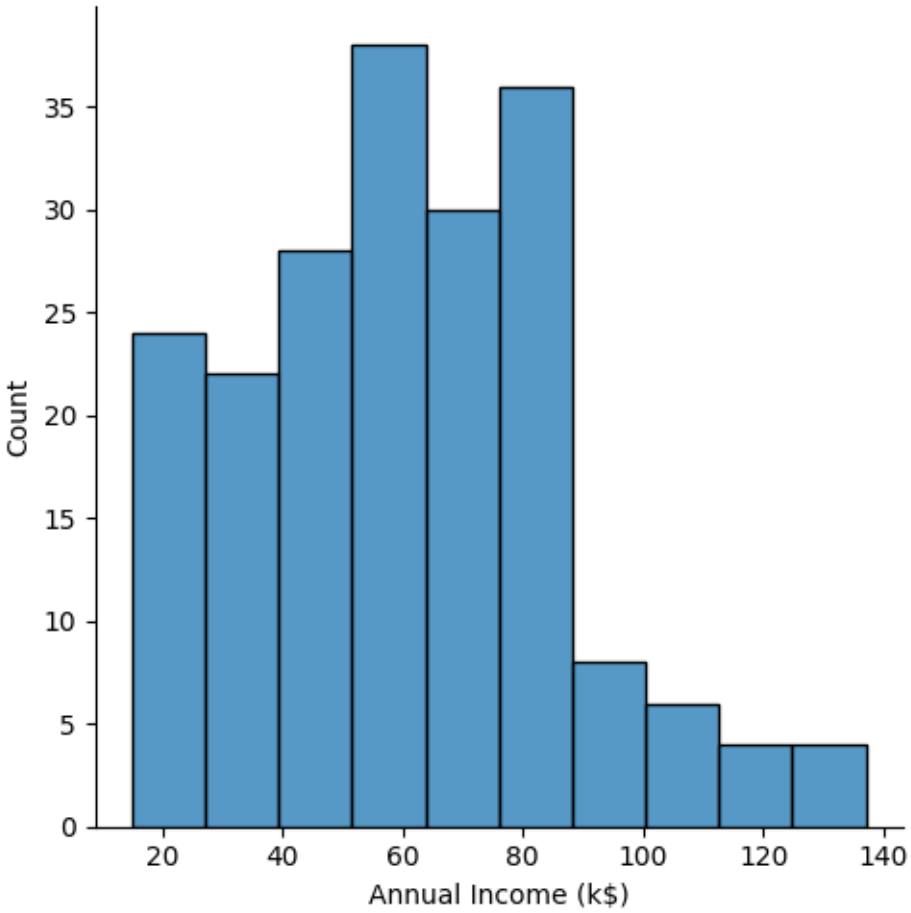
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    int64
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4)
memory usage: 6.4 KB
```

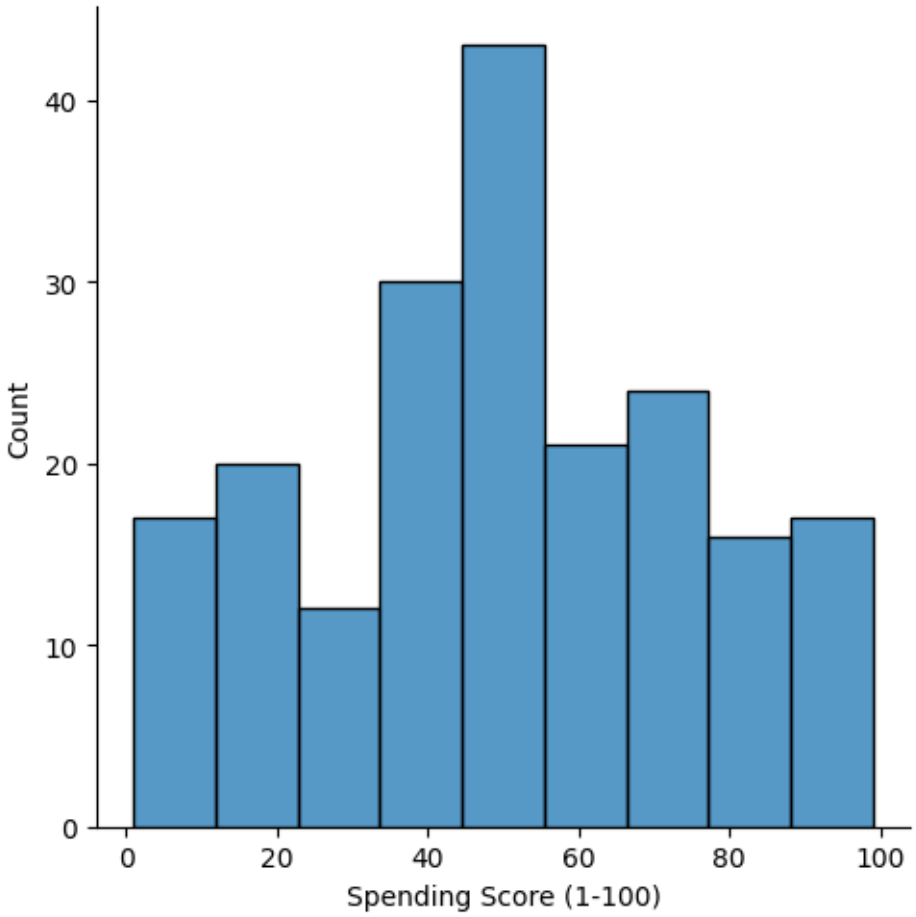
4 Data Analysis, Outlier Detection & Outlier Elimination

[159]:
```
sns.displot(df['Age'])
sns.displot(df['Annual Income (k$)'])
sns.displot(df['Spending Score (1-100)'])
```

[159]:
```
<seaborn.axisgrid.FacetGrid at 0x7b17496d1ff0>
```


Count
0
10
20
30
40
20
30
40
50
60
70
Age

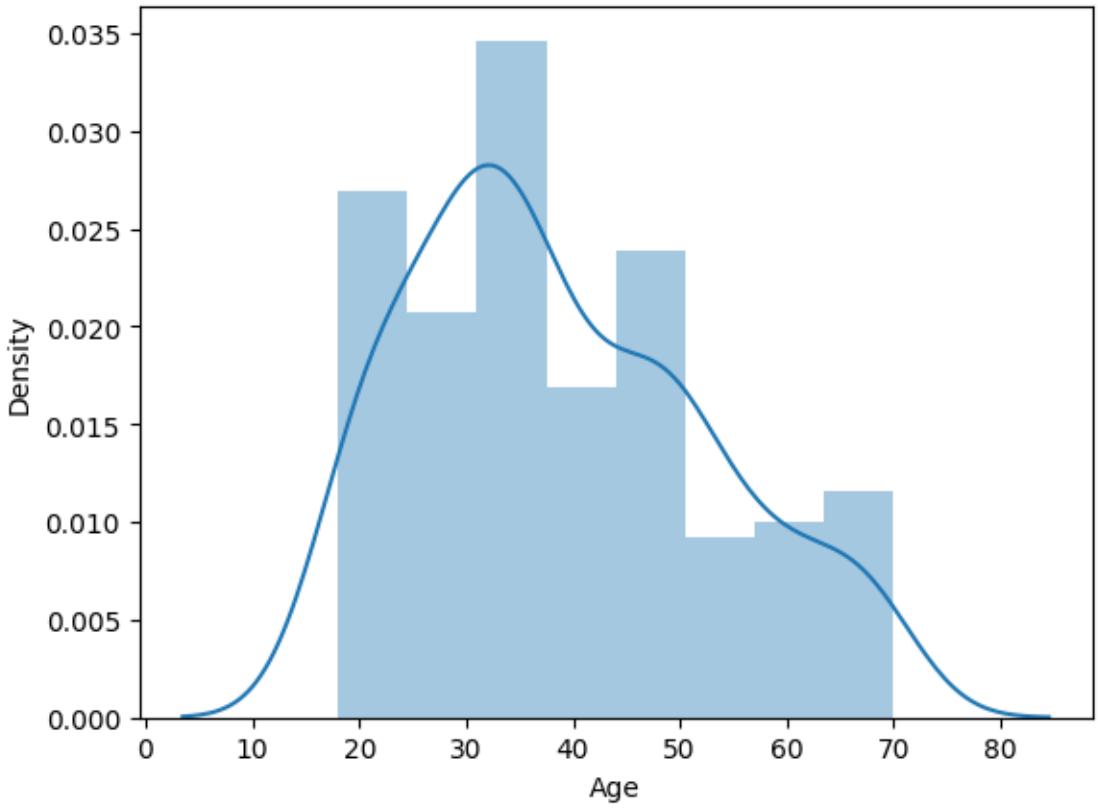

Count
0
5
10
15
20
25
30
35
20
40
60
80
100
120
140
Annual Income (k$)



[160]:
```
sns.distplot(df['Age'])
```

```
<ipython-input-160-0fafe04ea3f6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Age'])
```

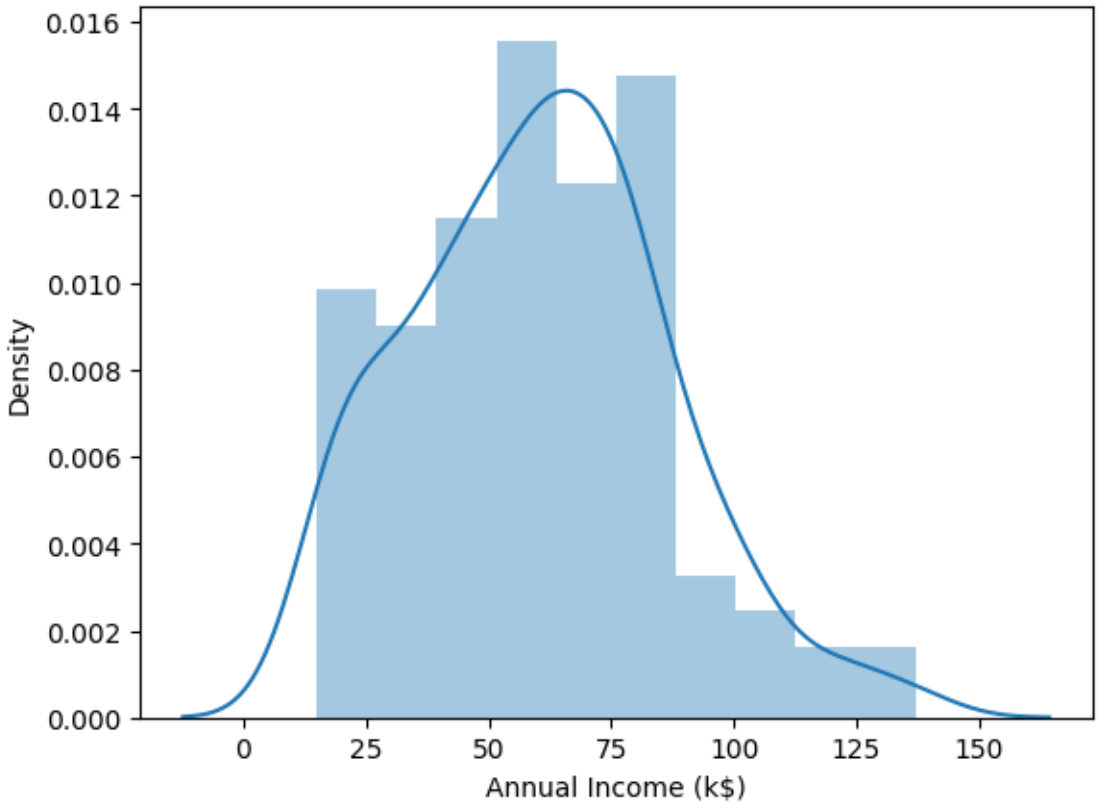
[160]: `<Axes: xlabel='Age', ylabel='Density'>`



[161]:
```
sns.distplot(df['Annual Income (k$)'])
```

```
<ipython-input-161-5c9bfeb4bab1>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Annual Income (k$)'])
```

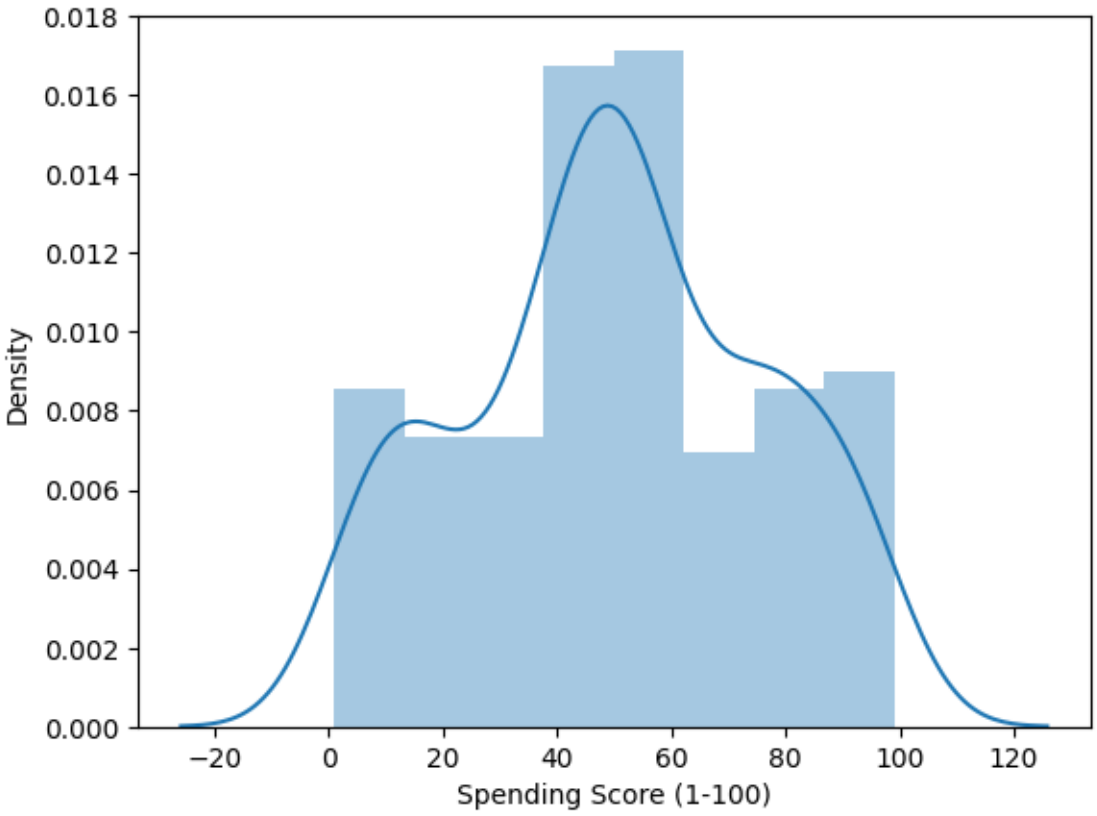
[161]: <Axes: xlabel='Annual Income (k$)', ylabel='Density'>



[162]:
```
sns.distplot(df['Spending Score (1-100)'])
```

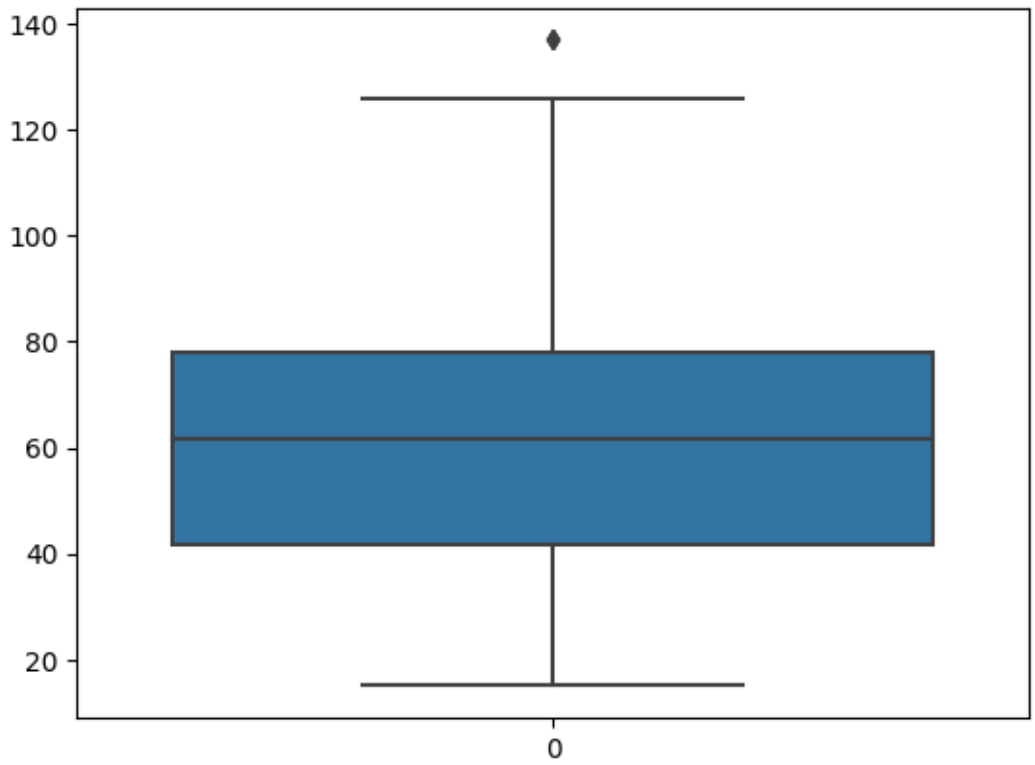
```
<ipython-input-162-beed7b40d5ab>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Spending Score (1-100)'])
```

[162]: <Axes: xlabel='Spending Score (1-100)', ylabel='Density'>



[165]:
```
sns.boxplot(df['Annual Income (k$)'])
```

[165]: <Axes: >



[166]:
```
Q1 = df['Annual Income (k$)'].quantile(0.25)
Q3 = df['Annual Income (k$)'].quantile(0.75)
```

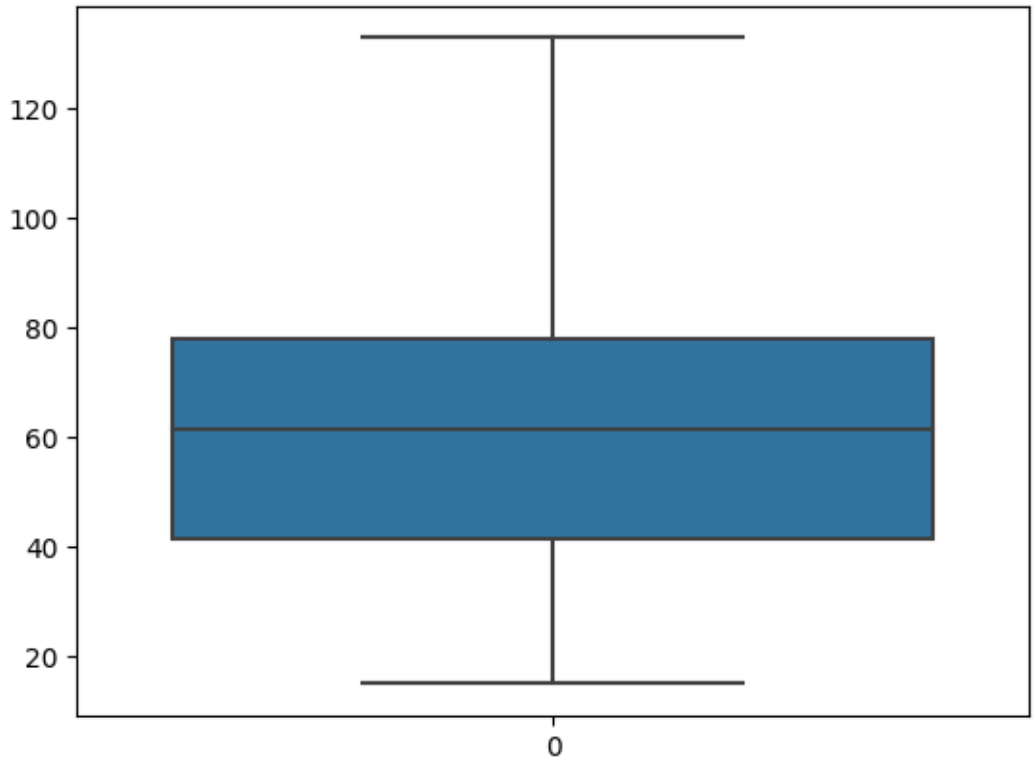
[167]:
```
IQR = Q3 - Q1
whisker_width = 1.5
```

[168]:
```
lower_whisker = Q1 - (whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
```

[169]:
```
df['Annual Income (k$)'] = np.where(df['Annual Income (k$)'] > upper_whisker,␣
 ↪upper_whisker, np.where(df['Annual Income (k$)'] < lower_whisker,␣
 ↪lower_whisker, df['Annual Income (k$)']))
```

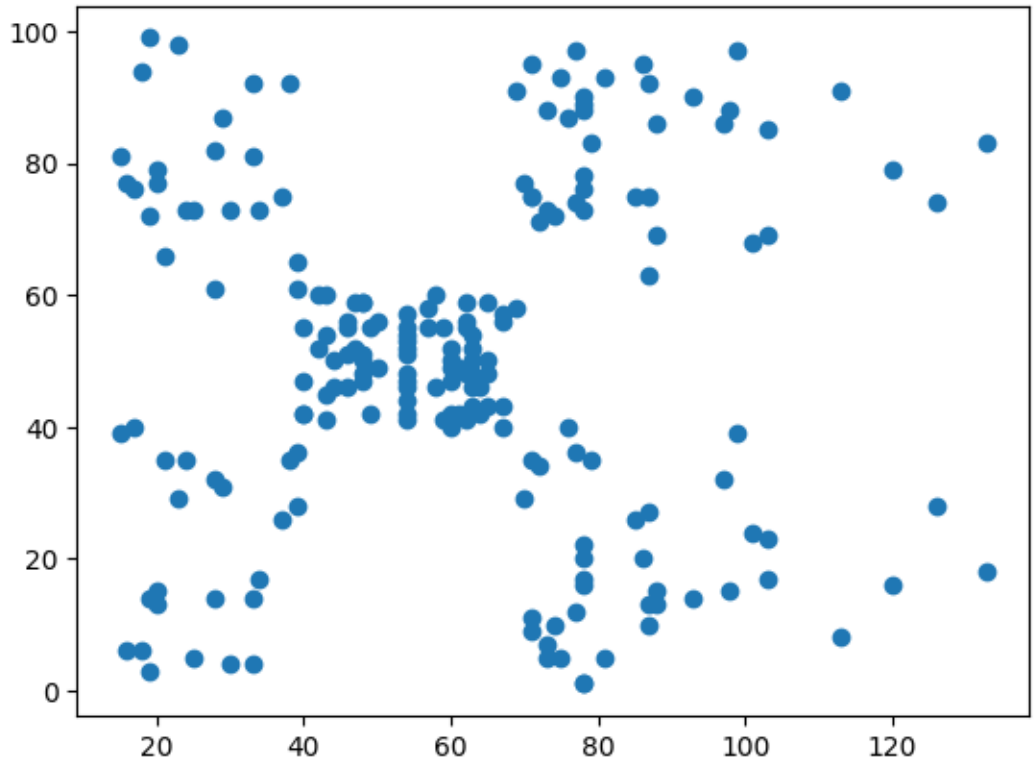
[170]:
```
sns.boxplot(df['Annual Income (k$)'])
```

[170]: <Axes: >



[171]:
```
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'])
```

[171]: <matplotlib.collections.PathCollection at 0x7b1749661db0>



[172]:
```
X_train = df.drop(['Spending Score (1-100)'], axis=1)
Y_train = df['Spending Score (1-100)']
```

[173]:
```
X_train.head(), Y_train.head()
```

[173]:
```
(   Gender  Age  Annual Income (k$)
 0       1   19                15.0
 1       1   21                15.0
 2       0   20                16.0
 3       0   23                16.0
 4       0   31                17.0,
 0    39
 1    81
 2     6
 3    77
 4    40
 Name: Spending Score (1-100), dtype: int64)
```

5 Finding Elbow Point (Possible 'K' value)

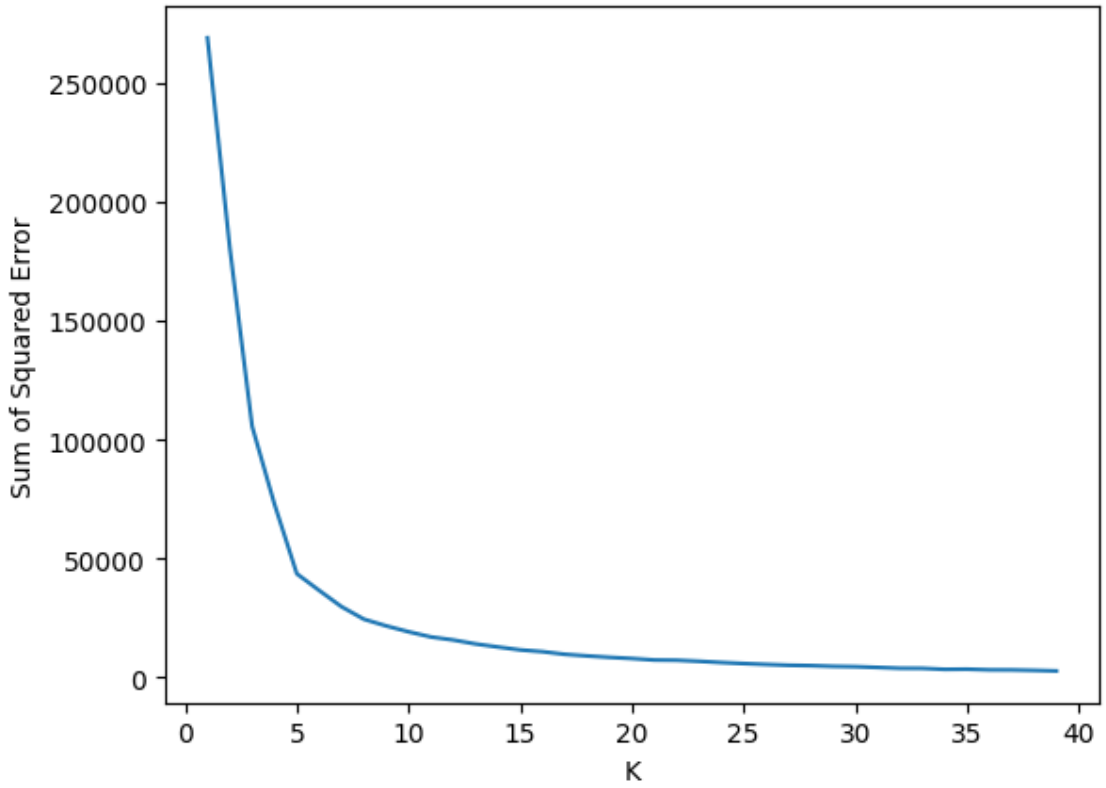
[174]:
```
from sklearn.cluster import KMeans
```

[ ]:
```
k_rng = range(1,40)
sse = []

for k in k_rng:
  km = KMeans(n_clusters=k)
  km.fit(df[['Annual Income (k$)', 'Spending Score (1-100)']])
  sse.append(km.inertia_)
```

[ ]:
```
sse
```

[177]:
```
plt.xlabel('K')
plt.ylabel('Sum of Squared Error')
plt.plot(k_rng, sse)
```

[177]: [<matplotlib.lines.Line2D at 0x7b174a86f9d0>]



[178]:
```
kmeans = KMeans(n_clusters=5)
kmeans
```

[178]:
```
KMeans(n_clusters=5)
```

[179]:
```
z = kmeans.fit_predict(df[['Annual Income (k$)','Spending Score (1-100)']])
z
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[179]:
```
array([3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2,
       3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 0,
       3, 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, 4, 1, 4, 0, 4, 1, 4, 1, 4,
       0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
       1, 4], dtype=int32)
```

[180]:
```
df_2 = df
```

[181]:
```
df_2['Cluster'] = z
df_2.head()
```

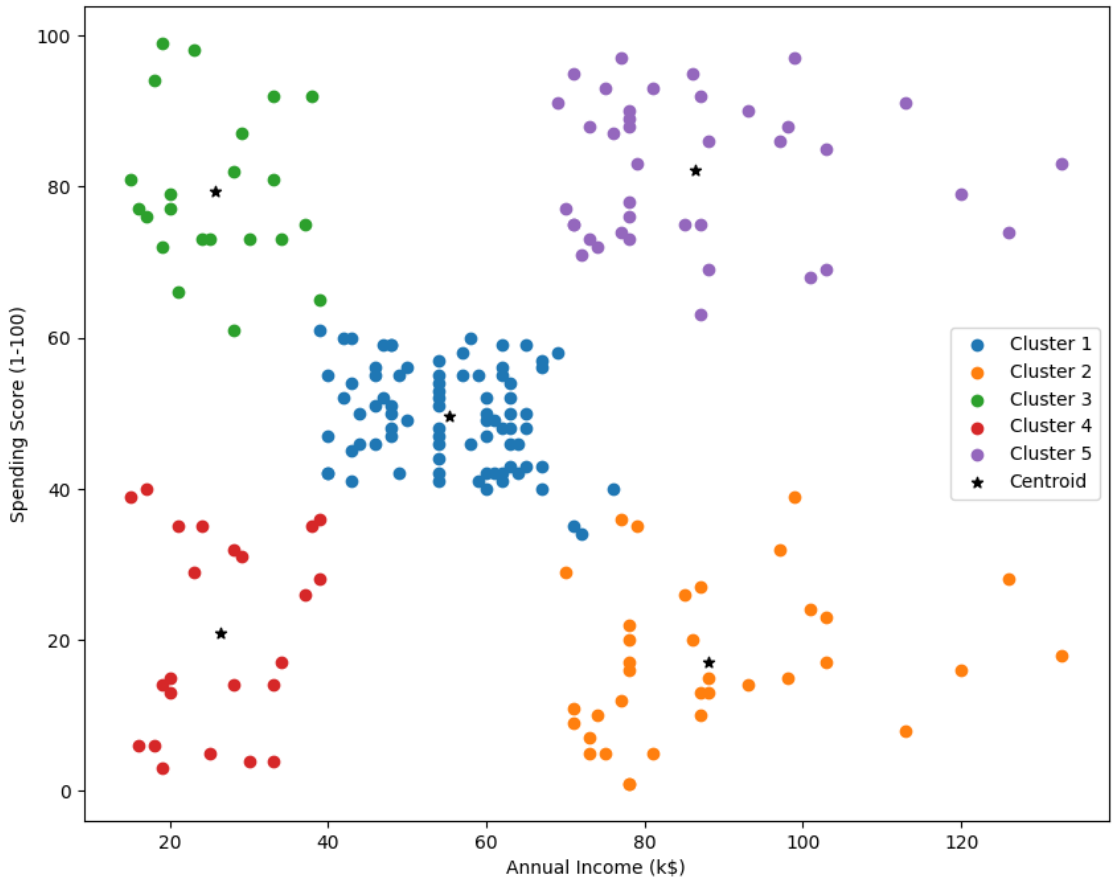
[181]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1	19	15.0	39	3
1	1	21	15.0	81	2
2	0	20	16.0	6	3
3	0	23	16.0	77	2
4	0	31	17.0	40	3

[182]:
```
plt.figure(figsize=(10,8))

df1 = df_2[df_2.Cluster==0]
df2 = df_2[df_2.Cluster==1]
df3 = df_2[df_2.Cluster==2]
df4 = df_2[df_2.Cluster==3]
df5 = df_2[df_2.Cluster==4]


plt.scatter(df1['Annual Income (k$)'], df1['Spending Score (1-100)'],␣
 ↪label='Cluster 1')
```

```
plt.scatter(df2['Annual Income (k$)'], df2['Spending Score (1-100)'],␣
 ↪label='Cluster 2')
plt.scatter(df3['Annual Income (k$)'], df3['Spending Score (1-100)'],␣
 ↪label='Cluster 3')
plt.scatter(df4['Annual Income (k$)'], df4['Spending Score (1-100)'],␣
 ↪label='Cluster 4')
plt.scatter(df5['Annual Income (k$)'], df5['Spending Score (1-100)'],␣
 ↪label='Cluster 5')

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],␣
 ↪color='black', marker='*', label='Centroid')

plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
```

[182]: <matplotlib.legend.Legend at 0x7b17490659f0>



6 Train - Test Split

[183]:
```
df.drop(['Cluster'], axis=1, inplace=True)
```

[184]:
```
df.head()
```

[184]:

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

[185]:
```
from sklearn.model_selection import train_test_split
```

[186]:
```
X = df.drop(['Spending Score (1-100)'], axis=1)
y = df['Spending Score (1-100)']
```

7 KNN and Logistic Regression Modeling

[383]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
```

7.1 KNN Model

[384]:
```
lr = LogisticRegression()
knn = KNeighborsClassifier(n_neighbors=3)
```

[350]:
```
xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3,␣
↪random_state=12)
```

[351]:
```
xtest.shape, xtrain.shape
```

[351]:
```
((60, 3), (140, 3))
```

[352]:
```
ytest.shape, ytrain.shape
```

[352]:
```
((60,), (140,))
```

[353]:
```
knn.fit(xtrain, ytrain)
```

[353]:
```
KNeighborsClassifier(n_neighbors=3)
```

[378]:
```
acc = knn.score(xtest, ytest)
print(f"Accuracy for the KNN model is {acc*100:.2f}%")
```

```
Accuracy for the KNN model is 3.33%
```

7.2 Logistic Regression Model

[446]:
```
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=0.3,␣
↪random_state=10)
```

[447]:
```
Xtest.shape, Xtrain.shape
```

[447]:
```
((60, 3), (140, 3))
```

[448]:
```
Ytest.shape, Ytrain.shape
```

[448]:
```
((60,), (140,))
```

[449]:
```
lr.fit(Xtrain, Ytrain)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[449]:
```
LogisticRegression()
```

[450]:
```
acc = lr.score(xtest, ytest)
print(f"Accuracy for the Logistic Regression model is {acc*100:.2f}%")
```

```
Accuracy for the Logistic Regression model is 13.33%
```

8 Prediction

[380]:
```
prediction = knn.predict([[1.0, 36.0 , 24.0]])[0]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature
names
  warnings.warn(
```

[381]:
```
print("Gender: Male, Age: 36, Salary(k$): 24.0, Spending Score(1-100): {}".
↪format(prediction))
```

```
Gender: Male, Age: 36, Salary(k$): 24.0, Spending Score(1-100): 35
```