Friday, 10 March 2023 08:40

steps
import libraries
pandas for data manipulation lib it allows to clean and wrangle data
seaborn is a visualization lib it allows us to look at statistical visualization
matplot another visualization lib
sk learn is our clustering lib using k means so from sk learn we got our clustering lib

```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        import warnings
        warnings.filterwarnings('ignore')
```

Screen clipping taken: 10/03/2023 09:42

then bring in the data set with variable mk

```
In [2]: mk = pd.read_csv("C:/Users/davyy/Downloads/mall customer segmenta
```

Screen clipping taken: 10/03/2023 09:43

mk.head() to see our data

```
In [3]: mk.head()
```

Out[3]:

	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

Screen clipping taken: 10/03/2023 09:44

mk.describe to see the count of our data

Univariate Analysis

```
In [4]: mk.describe()
```

Out[4]:

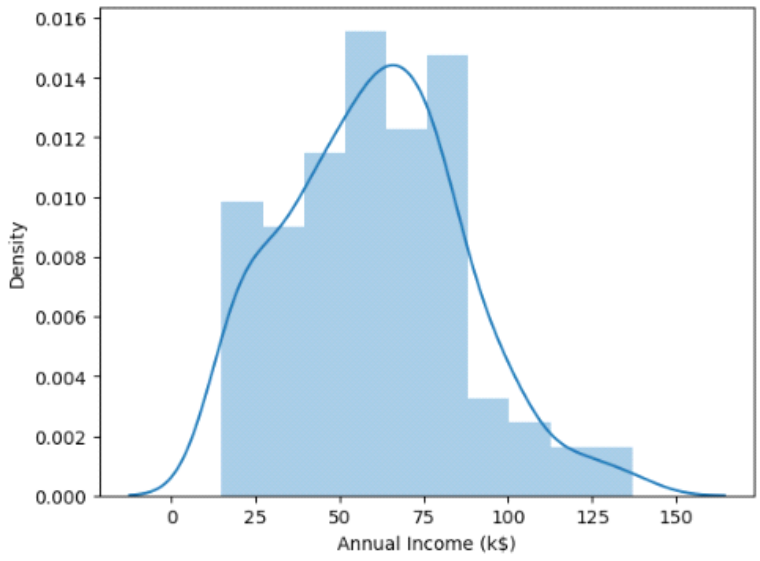
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Screen clipping taken: 10/03/2023 09:45

lets create a histogram to showcase the annual income and density distribution using seaborn distplot

```
In [5]: sns.distplot(mk['Annual Income (k$)'])
```

```
Out[5]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Density'>
```



Screen clipping taken: 10/03/2023 09:46

using for loop to loop through and create new visuals
using numerical variables to look through the univariate analysis
remember a for loop allows you to go go through each item in a list or any kind of structure
to get the columns we use a data frame variable (mk.columns)

```
In [6]: mk.columns
```

```
Out[6]: Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
               'Spending Score (1-100)'],
              dtype='object')
```

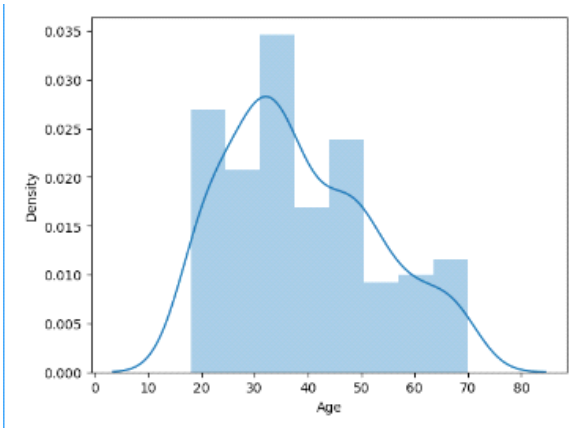
Screen clipping taken: 10/03/2023 09:47

so we would create a for loop that goes through specific columns saved in the variable called columns.

```
In [7]: columns = ['Age', 'Annual Income (k$)',
                   'Spending Score (1-100)']
        for e in columns:
            plt.figure()
            sns.distplot(mk[e])
```

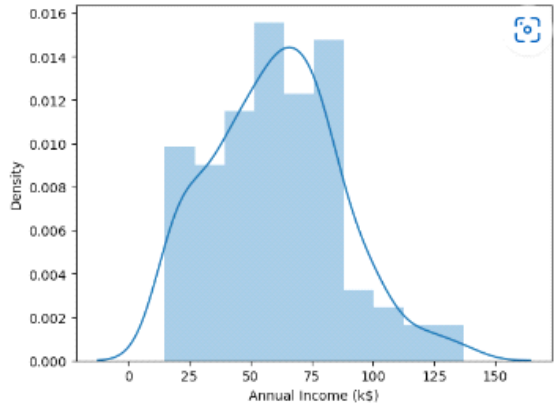
Screen clipping taken: 10/03/2023 09:47

Results :

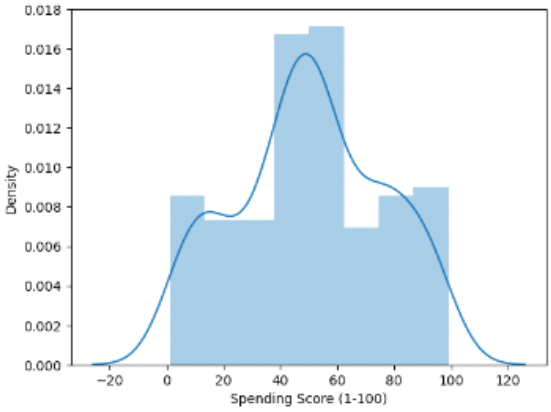


Screen clipping taken: 10/03/2023 09:48

this kde is a probability density plot, the probability is in the line



Screen clipping taken: 10/03/2023 09:48

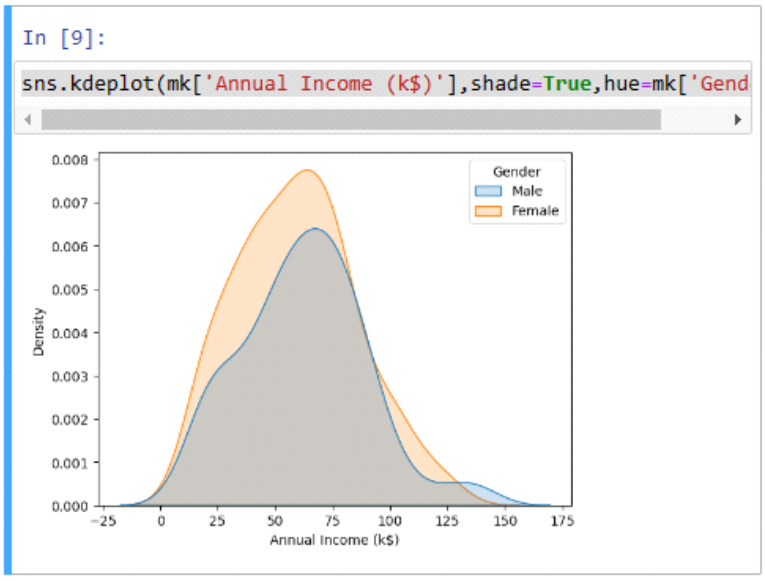


Screen clipping taken: 10/03/2023 09:49

from this we can see the spending score is isolated 40-50

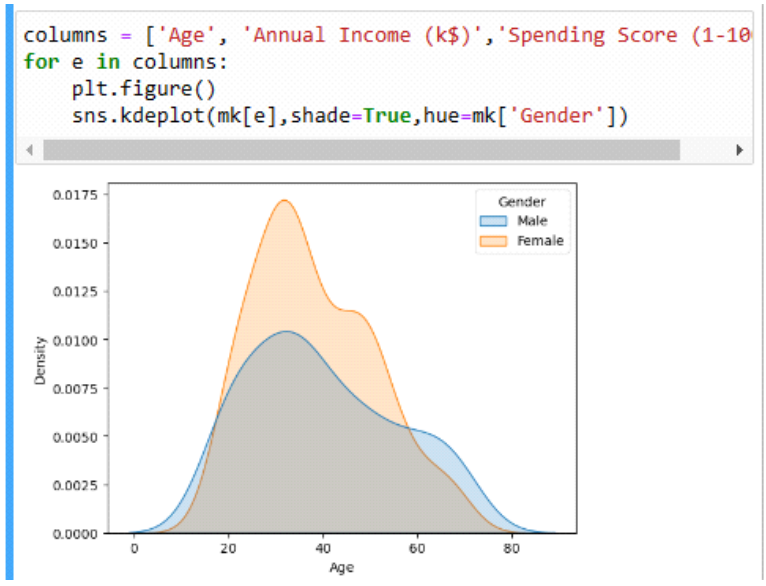
We can visualize the kde using split annual income ill visualize the KDE without the histogram. The beautiful thing about kde plot is not only a different looks but we can set different parameters(to see options press shift tab)

This is a kde with a parameter of hue

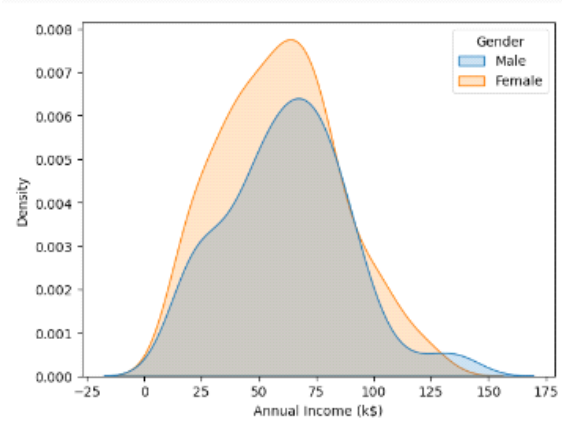


Screen clipping taken: 10/03/2023 09:53

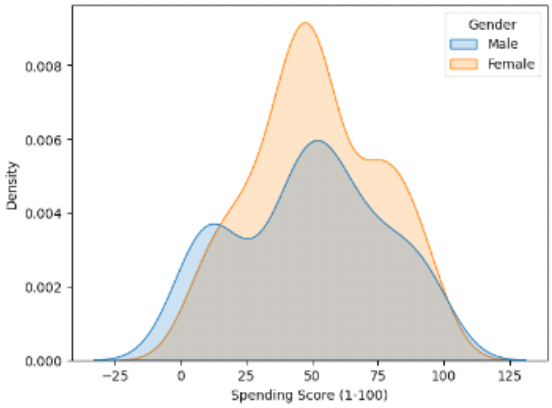
We can use a for loop too see how gender compares with age annual income and spending score



Screen clipping taken: 10/03/2023 09:55

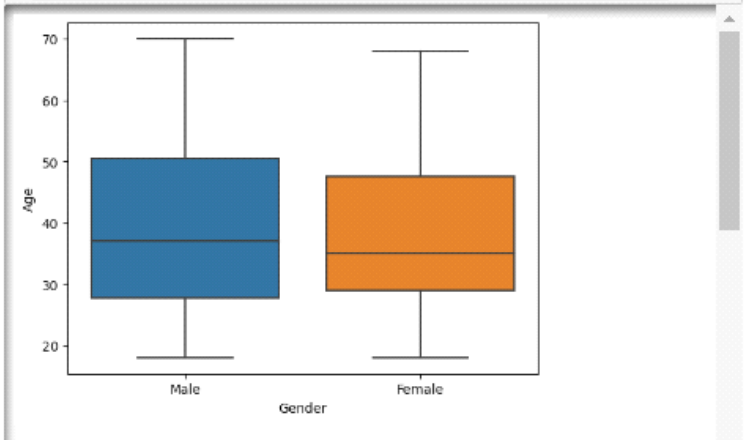


Screen clipping taken: 10/03/2023 09:55

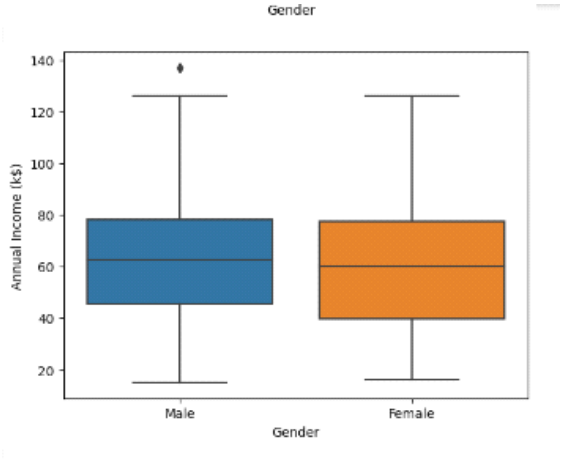


Screen clipping taken: 10/03/2023 09:55

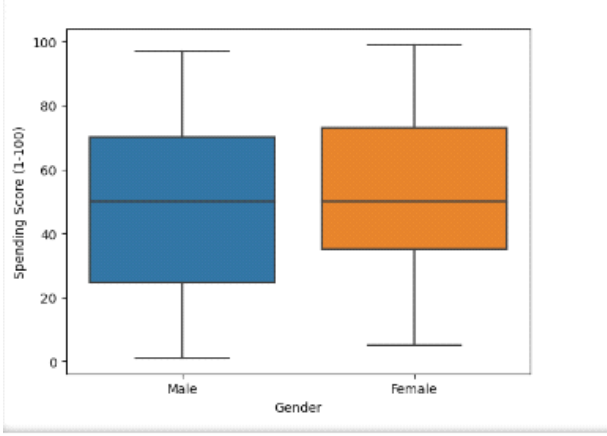
```
columns = ['Age', 'Annual Income (k$)','Spending Score (1-10
for e in columns:
    plt.figure()
    sns.boxplot(data=mk,x='Gender',y=mk[e])
```



Screen clipping taken: 10/03/2023 09:56



Screen clipping taken: 10/03/2023 09:56



Screen clipping taken: 10/03/2023 09:57

Using value count we count the values of the column ‘Gender’ to see how many are there

```
mk['Gender'].value_counts(normalize=True)
```

Out[21]:

```
Female    0.56
Male      0.44
Name: Gender, dtype: float64
```

Screen clipping taken: 10/03/2023 09:58

so through this we’ve discovered that 56% of customers are female and 44% percent are male, so using Univariate analysis we can explore our data and turn it into information we can understand.

Bivariate analysis works with two variables
lets look at a scatter plot using seaborn analysis

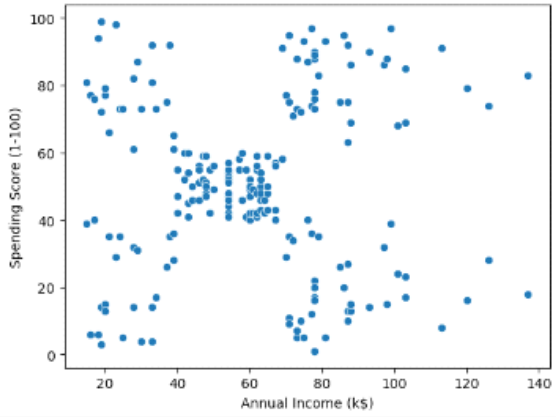
Bivariate Analysis

In [22]:

```
sns.scatterplot(data=mk,x='Annual Income (k$)', y='Spending
```

Out[22]:

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

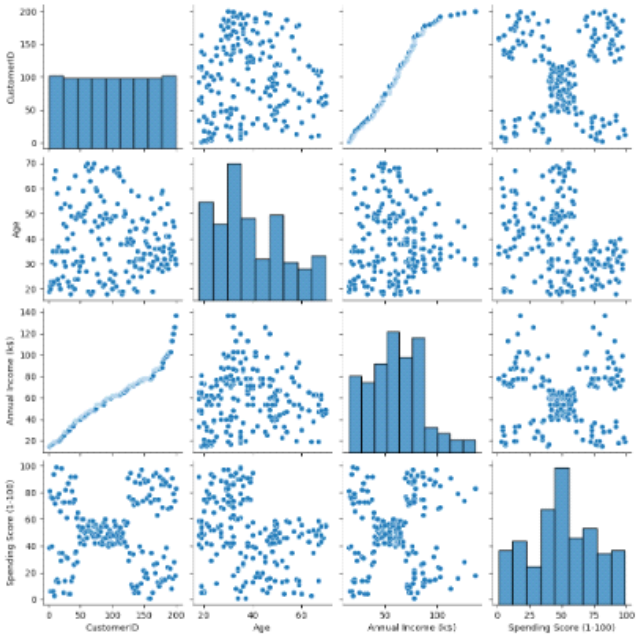


Screen clipping taken: 10/03/2023 09:58

In this scatter plot graphed by seaborn we set our X=Annual Income and Y=Spending Score
and we can already see we have some clusters between this two variables this is called bicluster variate we can see about 5-6 clusters.
We can use a pair plot to generate a scatter plot and histograms all together.

```
[12]: sns.pairplot(mk)
```

```
[12]: <seaborn.axisgrid.PairGrid at 0x12cc692c050>
```



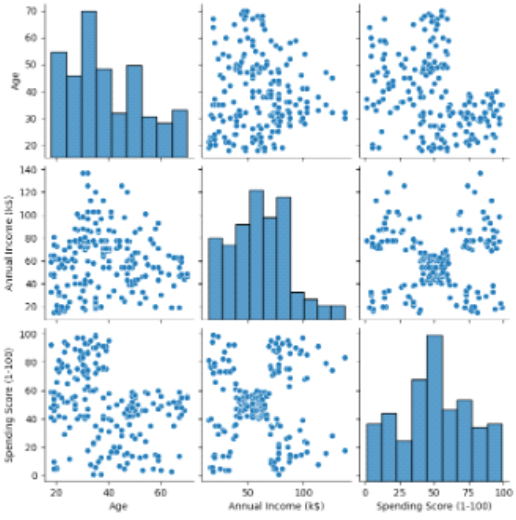
This is a lot of information gathered but it does help us to see where clusters are ongoing in a grand scale. However Customer ID right now is not an intricate additional information in our dataset so to remove it from the pair plot display, we set mk our variable equals mk to save the data frame over then drop the Customer ID then drop it on axis 1 because axis 0 is the rows and axis 1 is the columns

```
[14]: mk=mk.drop('CustomerID',axis=1)
      sns.pairplot(mk)
```

The result is we can see the pair plot better and have more intricate information

```
[14]: mk=mk.drop('CustomerID',axis=1)
      sns.pairplot(mk)
```

```
[14]: <seaborn.axisgrid.PairGrid at 0x12cdec79b10>
```

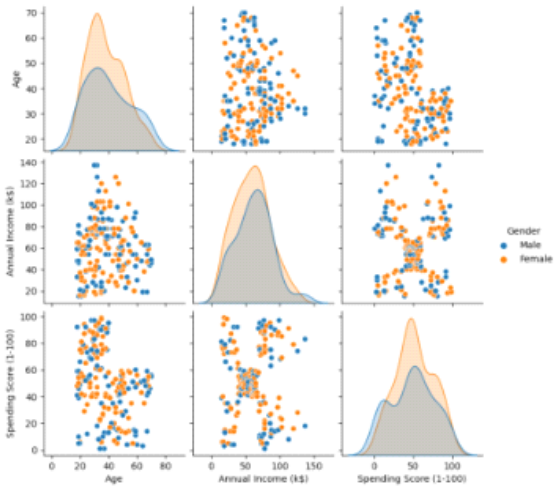


We would use hue to show the differences in relationship and clusters with gender

```
#mk=mk.drop('CustomerID',axis=1)
sns.pairplot(mk,hue='Gender')
```

Out[25]:

```
<seaborn.axisgrid.PairGrid at 0x246c19d8550>
```



Screen clipping taken: 10/03/2023 10:02

We commented out the drop because we don't need to run it anymore
We can see the mean values for our data using groupby to see mean values by gender.

In [26]:

```
mk.groupby(['Gender'])['Age', 'Annual Income (k$)',
                       'Spending Score (1-100)'].mean()
```

Out[26]:

Gender	Age	Annual Income (k$)	Spending Score (1-100)
Female	38.098214	59.250000	51.526786
Male	39.806818	62.227273	48.511364

Screen clipping taken: 10/03/2023 10:04

We can see the age is similar, the annual income for males is higher as well, the spending score for males is also higher.
We can look for the correlation for these two using correlation function(corr)

In [27]:

```
mk.corr()
```

Out[27]:

	Age	Annual Income (k$)	Spending Score (1-100)
Age	1.000000	-0.012398	-0.327227
Annual Income (k$)	-0.012398	1.000000	0.009903
Spending Score (1-100)	-0.327227	0.009903	1.000000

In [28]:

Screen clipping taken: 10/03/2023 10:04

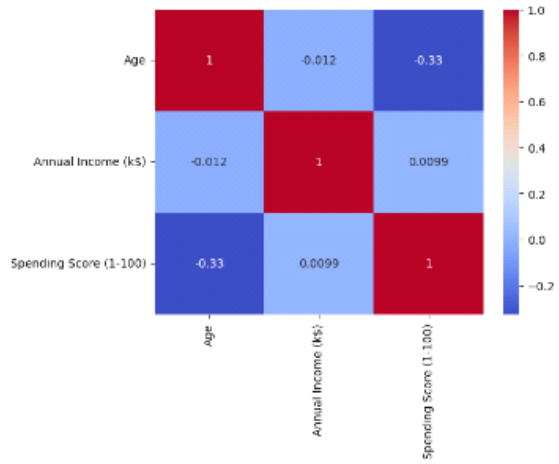
We can see with age annual income has a negative correlation, meaning as it increases it goes down.
Also spending score as well has a negative correlation with age.
We can also use seaborn to create a heat map, and for parameters we can use annotations as a parameter, and also use c map which is the color mapping using cool and warm

In [28]:

```
sns.heatmap(mk.corr(),annot=True,cmap='coolwarm')
```

Out[28]:

```
<AxesSubplot:>
```



Screen clipping taken: 10/03/2023 10:06

We've completed our Exploratory Data Analysis and now we will begin our K means algorithm clustering:
Univariate clustering, Bivariate clustering, Multivariate clustering. Then do some statistics around those clustering so we could get some insights about our customers from the data

Clustering - Univariate, Bivariate, Multivariate

Screen clipping taken: 10/03/2023 10:23

We want to first initiate our clustering algorithm, call it clustering 1 then get our KMeans algorithm
We can use shift to see parameters
Any machine learning algorithm using sk learn has 3 steps

1. The first step is to initiate and call the algorithm
2. Then we fit our data to that algorithm, which allows the algorithm to learn the data
3. Then we are going to either predict or grab some labels out of that fitted model

Step 1

In [29]:

```
clustering1 = KMeans()
```

Screen clipping taken: 10/03/2023 10:48

Step 2 we are going to fit only annual income in this case for our algorithm to learn

In [30]:

```
clustering1.fit(mk[['Annual Income (k$)']])
```

Out[30]:

```
KMeans()
```

Screen clipping taken: 10/03/2023 10:51

It has been initialized with no additional parameters.

Step3 lets look at the clustering labels from 0-7

```
In [31]:
clustering1.labels_
Out[31]:
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 6, 6, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 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, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 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, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 7, 7, 7, 7,
       7, 7])
```

Screen clipping taken: 10/03/2023 10:53

It doesn't mean much if we cannot compare it to our initial data so we are going to re-add these labels to our initial table/initial data frame

```
In [32]:
mk['Income Cluster'] = clustering1.labels_
mk.head()
Out[32]:
```

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

Screen clipping taken: 10/03/2023 10:57

Lets first see how many of our customers fall into each of our clusters using value count

```
In [33]: mk['Income Cluster'].value_counts()
Out[33]: 1    42
         4    36
         2    32
         0    28
         6    26
         5    16
         3    14
         7     6
         Name: Income Cluster, dtype: int64
```

Screen clipping taken: 10/03/2023 11:02

We can see the majority are in cluster 1

Lets find out the ideal amount of clusters we need, we currently have our default 8 amount of clusters. But we we can pass in any amount of clusters we want using the the parameter n_clusters	

We can perform summary statistics around the univariate cluster
39:34
Elbow method will show us the ideal number of clusters
There is a success metric called inertia it represents the distance between centroids it must be a low number

```
In [22]:
clustering1.inertia_
Out[22]:
5050.904761904763
```

Screen clipping taken: 11/03/2023 10:42

We need to use different numbers of clusters to check the inertia of each and find out when we should actually start our number of clusters. We can do that using a for loop.

We put inertia scores as the variable and pass all the scores in to the empty list.

In the for loop we use range and initiate it with a column we get our Kmeans and pss the number of clusters and equal it to our variable, we want the Kmeans to be fit to our data. We are still using annual income and finally we want to get the inertia score, we would append it to inertia

```
In [27]: intertia_scores=[]
         for e in range(1,11):
             kmeans=KMeans(n_clusters=e)
             kmeans.fit(mk[['Annual Income (k$)']])
             intertia_scores.append(kmeans.inertia_)
```

Screen clipping taken: 11/03/2023 10:59

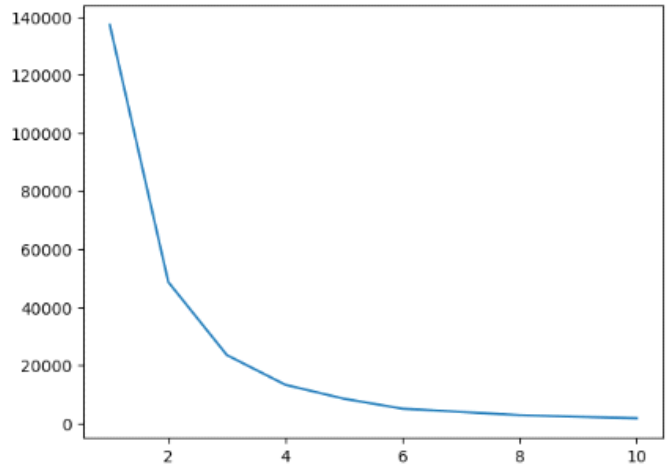
To see the scores recall the variable

```
In [28]: intertia_scores
Out[28]: [137277.28000000003,
          48660.88888888889,
          23517.330930930926,
          13278.112713472487,
          8481.496190476191,
          5050.904761904763,
          3949.2756132756135,
          2822.4996947496943,
          2274.165567765568,
          1767.6406204906204]
```

Screen clipping taken: 11/03/2023 11:00

We need to pair them with our range and a plot.

```
In [29]: plt.plot(range(1,11),intertia_scores)
Out[29]: [<matplotlib.lines.Line2D at 0x1d09c1dbac0>]
```



Screen clipping taken: 11/03/2023 11:02

So now we can see the elbow and try to find when the elbow actually starts which is around 3 so we can use 3 as the number of clusters we want

Multivariate

```
In [30]: clustering1 = KMeans(n_clusters=3)

In [31]: clustering1.fit(mk[['Annual Income (k$)']])
Out[31]: KMeans(n_clusters=3)
```

Screen clipping taken: 11/03/2023 11:05

```
In [34]: mk['Income Cluster'].value_counts()
Out[34]: 2    90
         0    74
         1    36
         Name: Income Cluster, dtype: int64
```

Screen clipping taken: 11/03/2023 11:05

Now we can begin doing analysis, we can see what the mean values of age and spending score for the particular cluster using group by

```
In [27]: mk.groupby(['Income Cluster'])['Age', 'Annual Income (k$)
                                       'Spending Score (1-100)'].mean()
```

Out[27]:

Income Cluster	Age	Annual Income (k$)	Spending Score (1-100)
0	39.500000	33.486486	50.229730
1	37.833333	99.888889	50.638889
2	38.722222	67.088889	50.000000

Screen clipping taken: 11/03/2023 11:10

The spending score is second lowest for the first cluster(0) and its annual income as well
Our second cluster(1) has the lowest for age but the highest annual income and spending score. Our

third cluster(2) has the second highest for age and for annual income but the lowest for spending score.
Now we are going to do a bivariate
47:33
We will not be passing in any clusters
We are going to fit the data and then pass in two variables Annual Income and spending score
And add it back to the data frame(mk) spending and income cluster Then we get our labels and add it to the data frame
Then we can take a look at the data frame head

Bivariate Clustering

In [27]:
```
clustering2 = KMeans()
clustering2.fit(mk[['Annual Income (k$)','Spending Score (1-100)']])
mk['Spending and Income Cluster'] =clustering2.labels_
mk.head()
```

Out[27]:

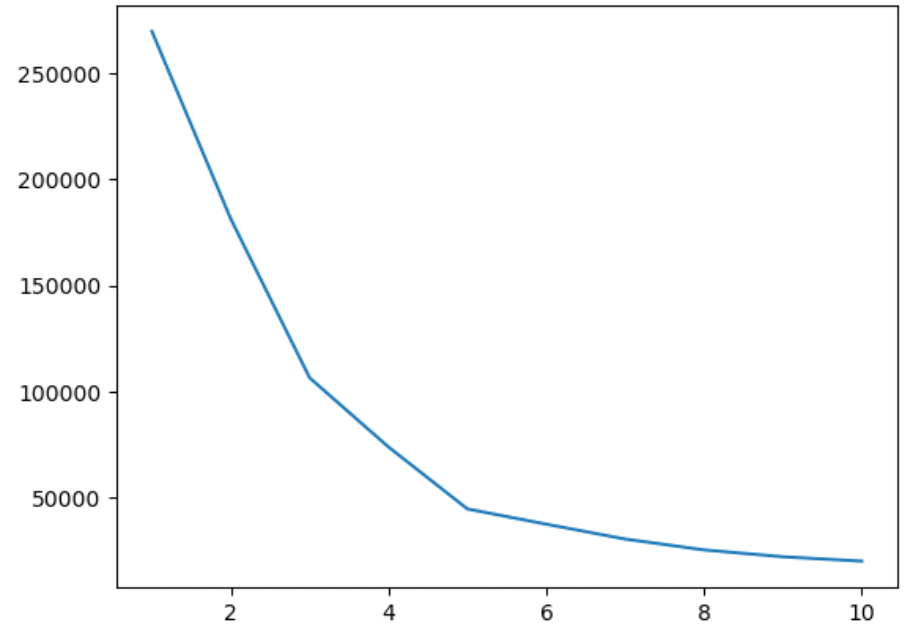
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Income Cluster	Spending and Income Cluster
0	1	Male	19	15	39	0	5
1	2	Male	21	15	81	0	3
2	3	Female	20	16	6	0	5
3	4	Female	23	16	77	0	3
4	5	Female	31	17	40	0	5

We can see another cluster so we to optimize our cluster
We also want to fit it on annual income and spending score
Then we exit the for loop and set out plot

In [29]:
```
intertia_scores2=[]
for e in range(1,11):
    kmeans2=KMeans(n_clusters=e)
    kmeans2.fit(mk[['Annual Income (k$)','Spending Score (1-100)']])
    intertia_scores2.append(kmeans2.inertia_)
plt.plot(range(1,11),intertia_scores2)
```

Screen clipping taken: 14/03/2023 06:09

Out[29]: [<matplotlib.lines.Line2D at 0x2035be55d30>]



Screen clipping taken: 14/03/2023 06:10

We can see that our elbow begins around 5 which would be our clusters
So we apply it

Bivariate Clustering

```
In [27]: clustering2 = KMeans(n_clusters=5)
         clustering2.fit(mk[['Annual Income (k$)','Spending Score (1-100)']])
         mk['Spending and Income Cluster'] =clustering2.labels_
         mk.head()
```

Out[27]:

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

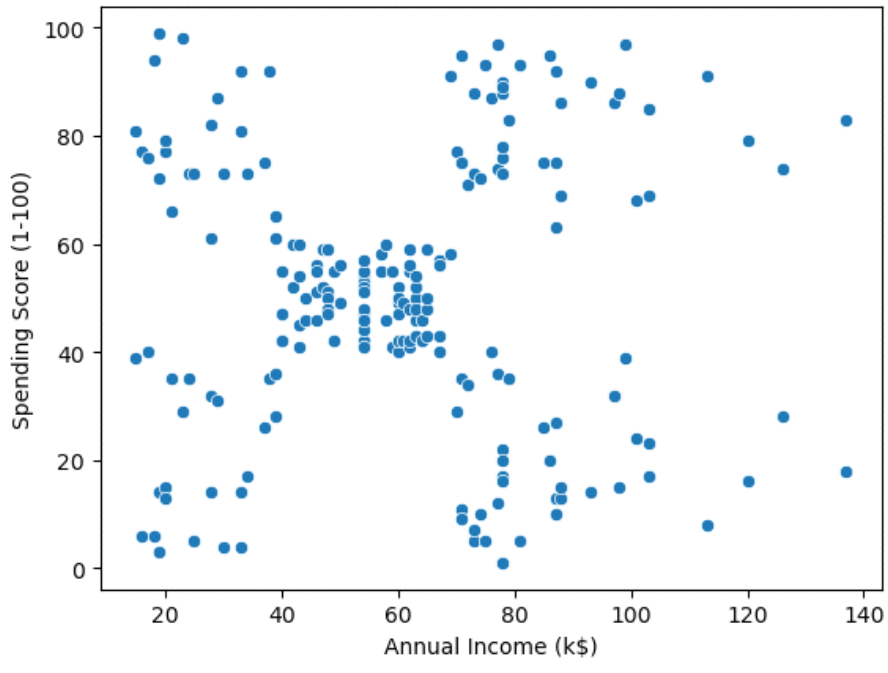
Screen clipping taken: 14/03/2023 06:13

We can see a visible change to our data so lets visualize our bivariate analysis using the best way to visualize a bivariate analysis which is a scatter plot

Using seaborn scatterplot set the data to the data frame, then set x to be annual income and set the y to be a spending score

```
In [29]: sns.scatterplot(data=mk, x='Annual Income (k$)',y='Spending Score (1-100)')
```

```
Out[29]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>
```

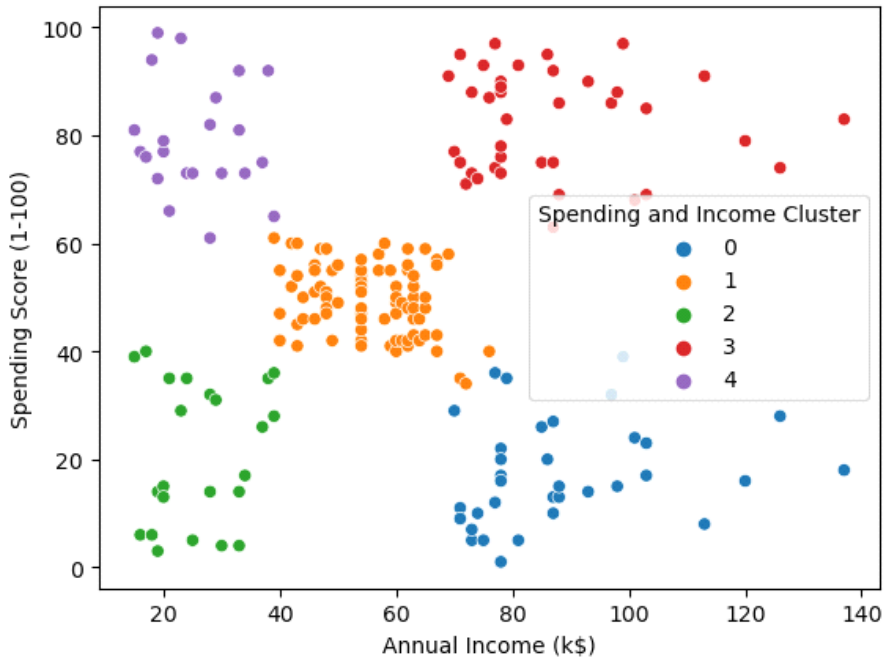


Screen clipping taken: 14/03/2023 06:20

We can actually see where our cluster are formed and since we chose to have 5 clusters the plot is showing us that it looks good with 5 clusters.
So we can pass our hue which would be our spending and income cluster then we change the color palette to tab10

```
In [32]: <, x='Annual Income (k$)',y='Spending Score (1-100)',hue='Spending and Income Cluster',palette='tab10')
```

```
Out[32]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>
```

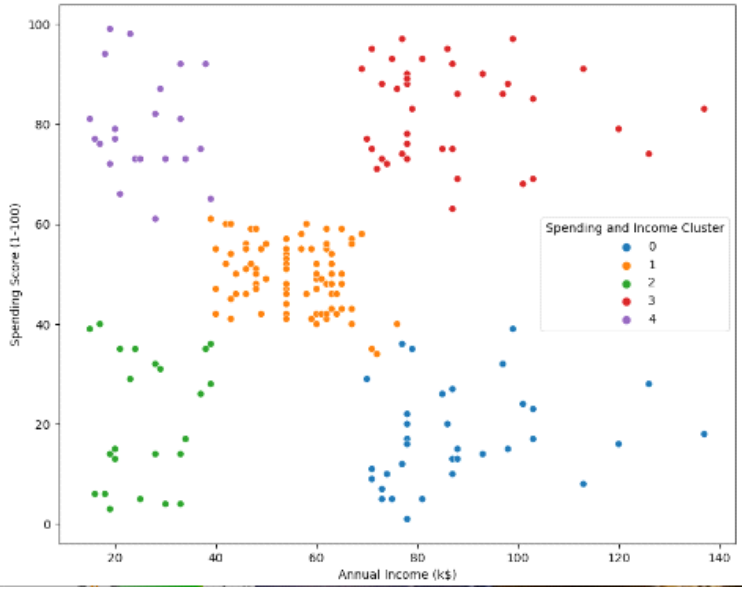


Screen clipping taken: 14/03/2023 06:28

We can clearly see our clusters but lets make the figure bigger using plt.figure set figsize to 10,8 turn it into a tupple

```
In [35]: plt.figure(figsize=(10,8))
         sns.scatterplot(data=mk, x='Annual Income (k$)',y='Spending Sc
```

```
Out[35]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Sc
         ore (1-100)'>
```



Screen clipping taken: 14/03/2023 06:32

Lets see our centers in clustering2

```
In [36]: clustering2.cluster_centers_

Out[36]: array([[88.2       , 17.11428571],
                [55.2962963 , 49.51851852],
                [26.30434783, 20.91304348],
                [86.53846154, 82.12820513],
                [25.72727273, 79.36363636]])
```

Screen clipping taken: 14/03/2023 06:39

These are x and y coordinates we want to add the x and y to our map, we can do that by turning our centers in to a data frame with pandas
Lets look at the list

```
In [38]: centers =pd.DataFrame(clustering2.cluster_centers_)
         centers
```

Out[38]:

	0	1
0	88.200000	17.114286
1	55.296296	49.518519
2	26.304348	20.913043
3	86.538462	82.128205
4	25.727273	79.363636

Screen clipping taken: 14/03/2023 06:42

Change the names to x and y

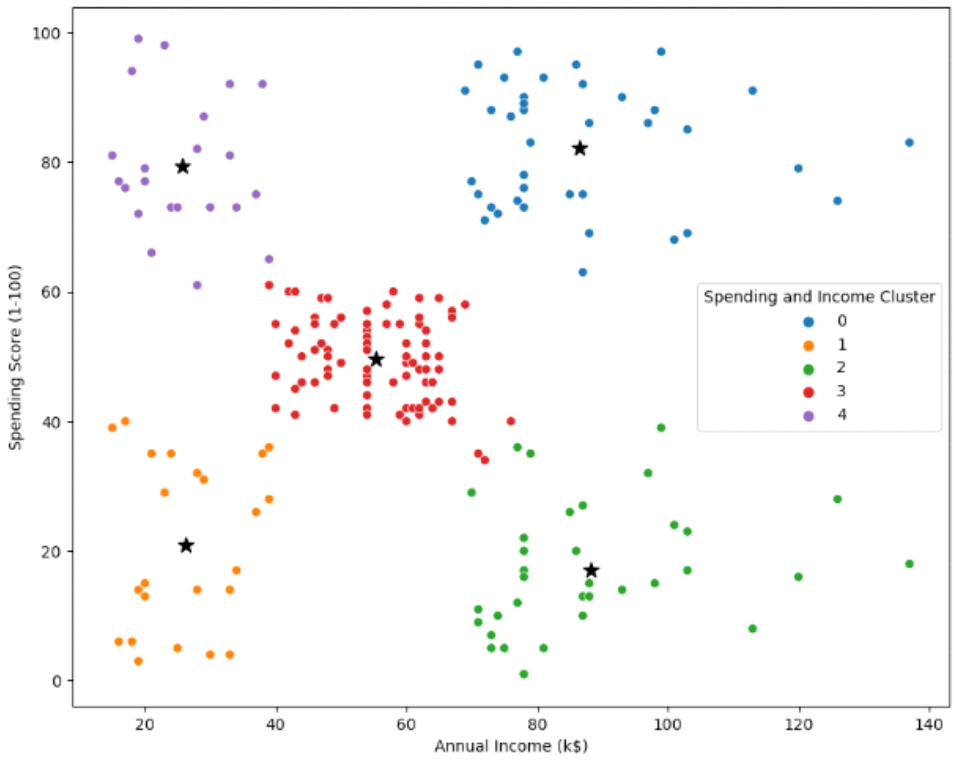
Them add it using plt.scatter set x = centers and y = centers then we set size to a 100 and color to black and marker to a star

```
In [31]: centers =pd.DataFrame(clustering2.cluster_centers_)
         centers.columns = ['x','y']
```

```
In [32]: plt.figure(figsize=(10,8))
         plt.scatter(x=centers['x'],y=centers['y'],s=100,c='black',marker='*')
         sns.scatterplot(data=mk, x='Annual Income (k$)',y='Spending Score (1-100)',hue='Spending and Income Clu
```

Screen clipping taken: 14/03/2023 06:52

Out[32]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>



Screen clipping taken: 14/03/2023 06:53

We can break it down by male and female to do further analysis

We can do a crosstab for the index we can set it to spending and income clusters pass it in to our data frame and since we wish to look at gender we can pass that in as well

In [33]:
```
pd.crosstab(mk['Spending and Income Cluster'],mk['Gender'])
```

Out[33]:

Spending and Income Cluster	Female	Male
0	21	18
1	14	9
2	16	19
3	48	33
4	13	9

Screen clipping taken: 14/03/2023 07:02

We can see where each cluster is and the percentages of the clusters we can use normalize the index to see the breakdown between male and female

In [34]:
```
rosstab(mk['Spending and Income Cluster'],mk['Gender'],normalize='index')
```

Out[34]:

Spending and Income Cluster	Female	Male
0	0.538462	0.461538
1	0.608696	0.391304
2	0.457143	0.542857
3	0.592593	0.407407
4	0.590909	0.409091

Screen clipping taken: 14/03/2023 07:05

We can see that cluster 0 which is the blue cluster has 53% female, cluster 1 the orange cluster with 59% is also dominated by females. The males in cluster 2 which is the green cluster with 54% are dominating. And in cluster 3 the red cluster is dominated by females with 59%. Finally the purple cluster which is 4 is al dominated by females 59%. From this analysis we can see that the males only dominated cluster 2 while the females dominated 4 clusters.
So our ideal cluster which would be the high spending score and annual income would be cluster 4 the purple cluster would be our target cluster because that cluster would bring the most money.

Lets look at the average age of each clusters using groupby.

```
In [35]: mk.groupby(['Spending and Income Cluster'])['Age', 'Annual Income (k$)',
                                'Spending Score (1-100)'].mean()
```

Out[35]:

Spending and Income Cluster	Age	Annual Income (k$)	Spending Score (1-100)
0	32.692308	86.538462	82.128205
1	45.217391	26.304348	20.913043
2	41.114286	88.200000	17.114286
3	42.716049	55.296296	49.518519
4	25.272727	25.727273	79.363636

Screen clipping taken: 14/03/2023 07:24

In cluster 0 we have high annual income and spending score with a low age but not the lowest age the lowest is cluster 4, they have a low annual income but a high spending score we can assume they are coming in for a big ticket item so we can build a campaign around that, using the customer id we can see the items purchased.
We can say from the data cluster 0 which is 53% female has a high spending score a high annual income and a low age of 32% is our ideal cluster to run campaigns on.