

Assignment Date	20 October 2022
Student Name	NAVEENA D
Student Roll Number	61771921030
Maximum Marks	2 Marks

Problem Statement: Customer Segmentation Analysis

Problem Statement :

Chronic Kidney Disease prediction is one of the most important in healthcare analytics. The most interesting and challenging tasks in day-to-day life is prediction in the medical field. 10% of the world is affected by chronic kidney disease (CKD), and millions die each year because they do not have access to affordable treatment. Chronic Kidney Disease can be cured, if treated in the early stages. The main aim of this project is to predict whether the patients have chronic kidney disease or not, in a more accurate and faster way based on certain diagnostic measurements like Blood Pressure(Bp), Albumin(AI).

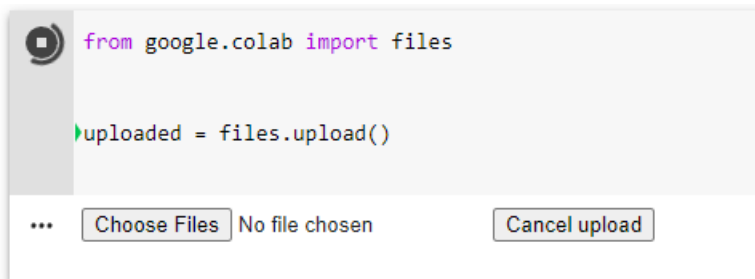
Clustering the data and performing classification algorithms

1. Download the dataset: Dataset
2. Load the dataset into the tool.
3. Perform Below Visualizations.
 - Univariate Analysis
 - Bi- Variate Analysis
 - Multi- Variate Analysis
4. Perform descriptive statistics on the dataset.
5. Check for Missing values and deal with them.
6. Find the outliers and replace them outliers
7. Check for Categorical columns and perform encoding.
8. Scaling the data
9. Perform any of the clustering algorithms
10. Add the cluster data with the primary dataset
11. Split the data into dependent and independent variables.
12. Split the data into training and testing

13. Build the Model
14. Train the Model
15. Test the Model
16. Measure the performance

TASK 1

DOWNLOAD AND LOAD THE DATASET



TASK 2

```
import numpy as np
```

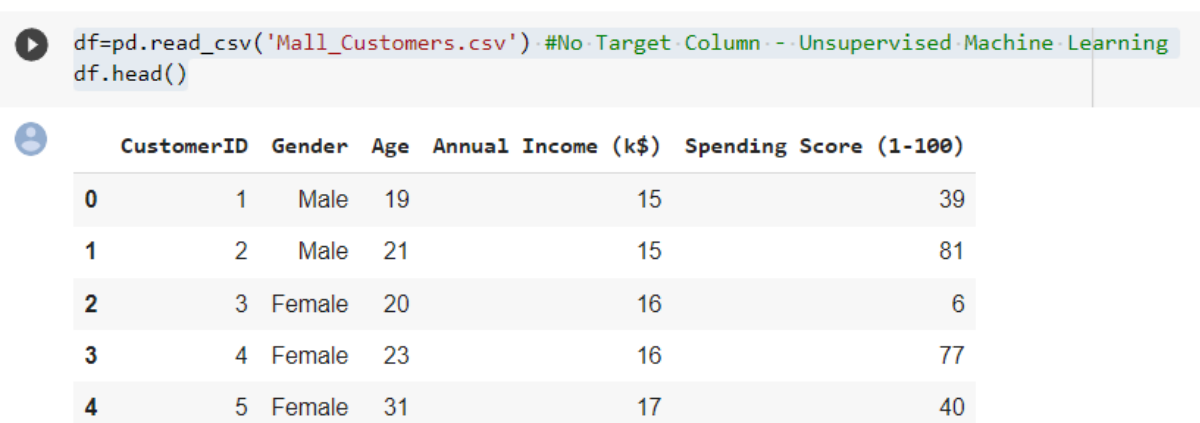
```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import matplotlib as rcParams
```

```
df=pd.read_csv('Mall_Customers.csv') #No Target Column - Unsupervised Machine Learning  
df.head()
```

A screenshot of the Google Colab interface showing the execution of the code from Task 2. The code cell contains:

```
df=pd.read_csv('Mall_Customers.csv') #No Target Column - Unsupervised Machine Learning  
df.head()
```

 Below the code cell, the output is displayed as a table with 6 columns: CustomerID, Gender, Age, Annual Income (k\$), and Spending Score (1-100). The table shows the first 5 rows of data.

	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

```
df = df.rename(columns = {'Annual Income (k$)': 'Annual_Income','Spending Score (1-100)':  
'Spending_Score'})  
df.head()
```

```
df = df.rename(columns = {'Annual_Income (k$)': 'Annual_Income', 'Spending_Score (1-100)': 'Spending_Score'})
df.head()
```

	CustomerID	Gender	Age	Annual_Income	Spending_Score
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

df.shape()

```
df.shape
```

```
(200, 5)
```

```
[ ] 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   200 non-null   int64
4   Spending_Score  200 non-null   int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
[ ] df.Gender.unique()
```

```
array(['Male', 'Female'], dtype=object)
```

df.Age.unique()

```
df.Age.unique()
```

```
array([19, 21, 20, 23, 31, 22, 35, 64, 30, 67, 58, 24, 37, 52, 25, 46, 54,
       29, 45, 40, 60, 53, 18, 49, 42, 36, 65, 48, 50, 27, 33, 59, 47, 51,
       69, 70, 63, 43, 68, 32, 26, 57, 38, 55, 34, 66, 39, 44, 28, 56, 41])
```

```
[ ] df.Gender.value_counts()
```

```
Female    112
Male       88
Name: Gender, dtype: int64
```

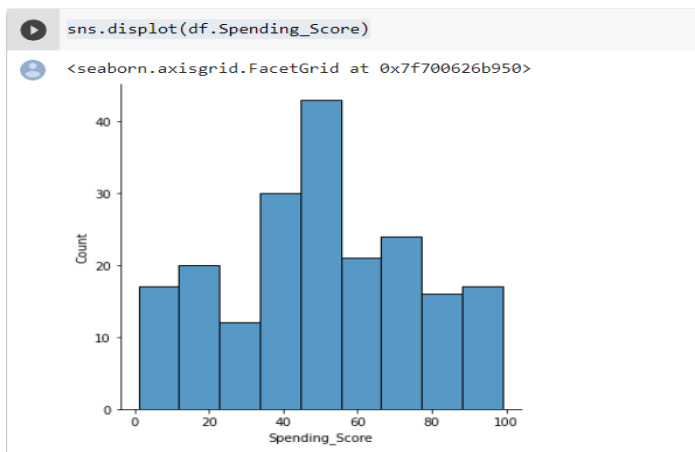
TASK 3

Perform Below Visualizations

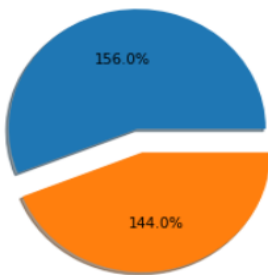
- ❖ Univariate Analysis
- ❖ Bi - Variate Analysis
- ❖ Multi - Variate Analysis

Univariate Analysis

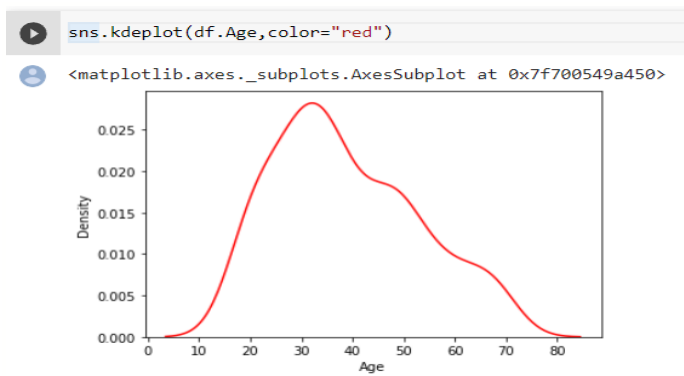
`sns.displot(df.Spending_Score)`



`plt.pie(df.Gender.value_counts(),[0,0.2],shadow='True',autopct="1%.1f%%")` #categorical column

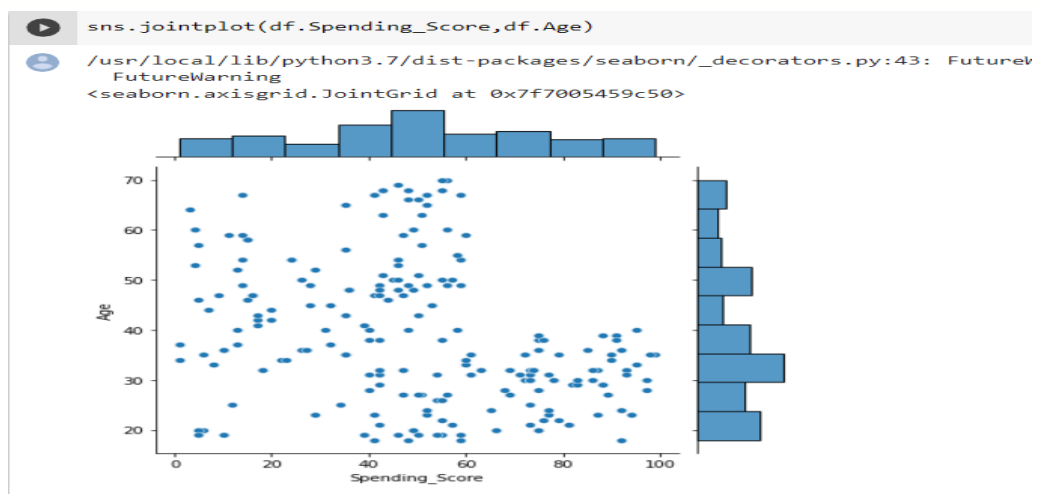


`sns.kdeplot(df.Age,color="red")`

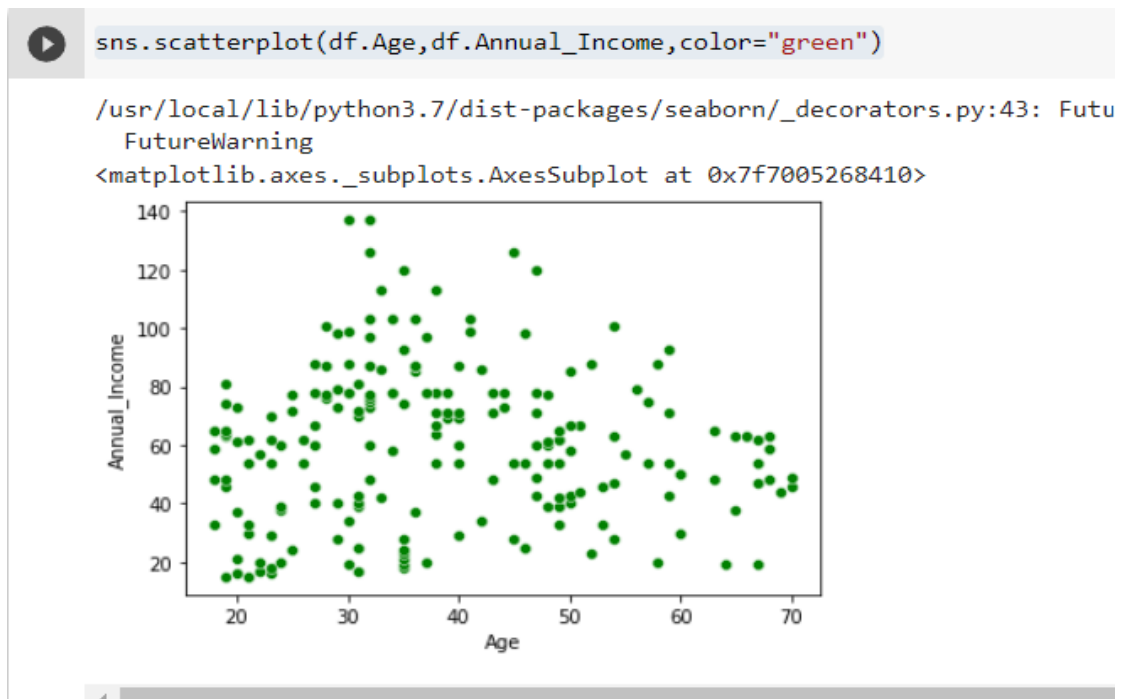


Bi-Variate Analysis

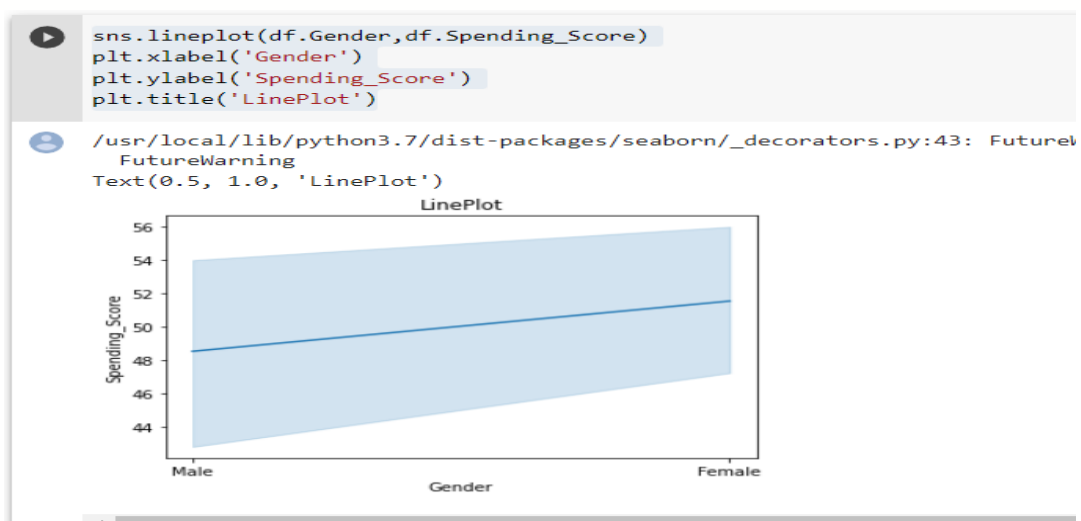
```
sns.jointplot(df.Spending_Score,df.Age)
```



```
sns.scatterplot(df.Age,df.Annual_Income,color="green")
```



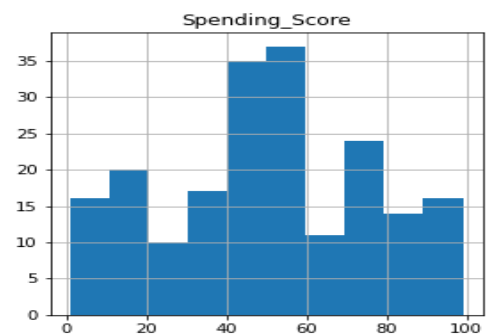
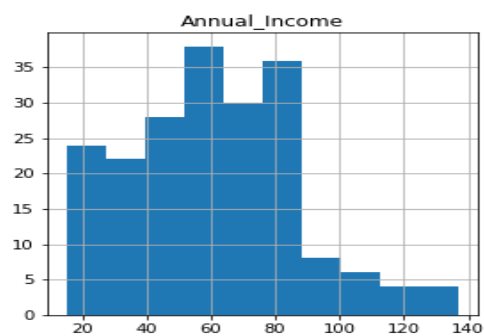
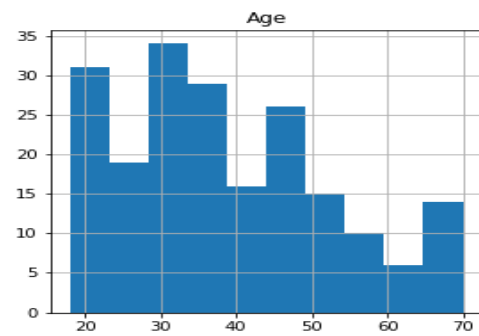
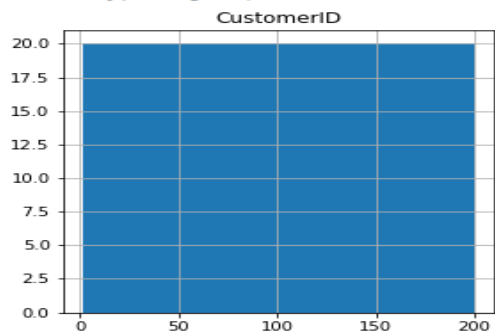
```
sns.lineplot(df.Gender,df.Spending_Score)  
plt.xlabel('Gender')  
plt.ylabel('Spending_Score')  
plt.title('LinePlot')
```



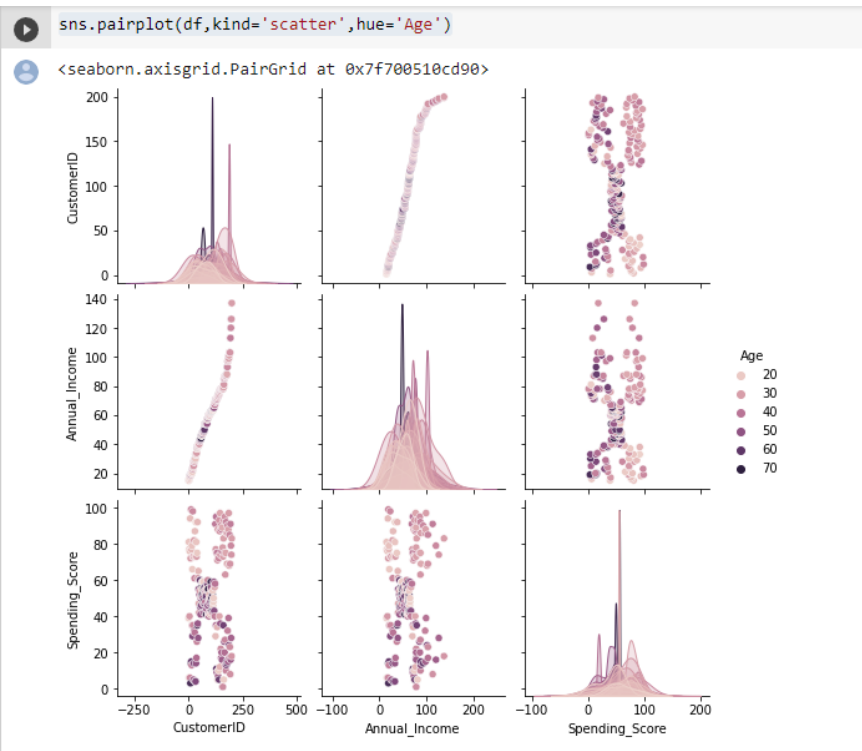
Multi-Variate Analysis

```
df.hist(figsize=(10,10))
```

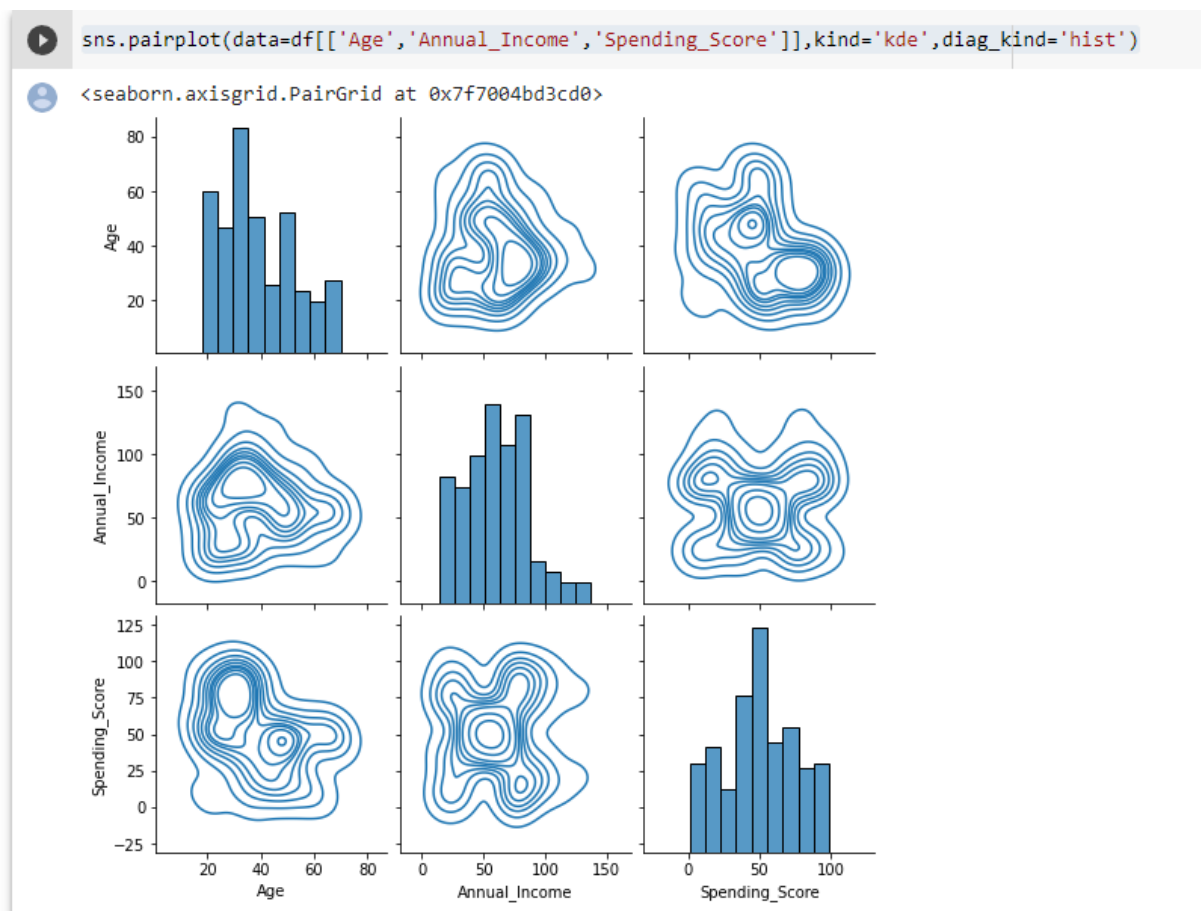
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f7005203910>,  
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f70051db810>],  
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f7005191c90>,  
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f70051541d0>]],  
      dtype=object)
```



```
sns.pairplot(df,kind='scatter',hue='Age')
```



```
sns.pairplot(data=df[['Age','Annual_Income','Spending_Score']],kind='kde',diag_kind='hist')
```



TASK 4

Descriptive statistics

df.describe()

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

	CustomerID	Age	Annual_Income	Spending_Score
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

TASK 5

Handle missing data

df.isnull().any() #no missing data

```
df.isnull().any() #no missing data  
CustomerID      False  
Gender           False  
Age             False  
Annual_Income   False  
Spending_Score  False  
dtype: bool
```

TASK 6

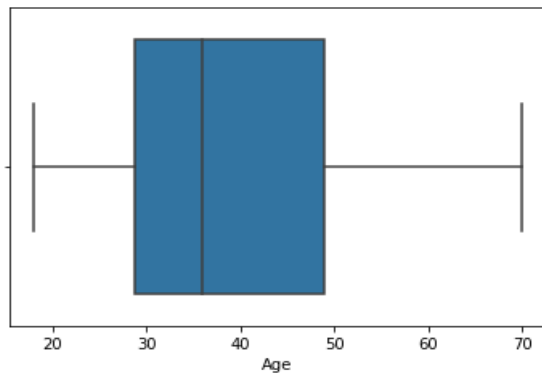
Outliers Replacement

sns.boxplot(df.Age) #no outliers



```
sns.boxplot(df.Age) #no outliers
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7004604090>
```



TASK 7

Check for Categorical column and perform encoding

```
[ ] from sklearn.preprocessing import LabelEncoder
```

```
[ ] le = LabelEncoder()
```

```
[ ] df.Gender=le.fit_transform(df.Gender)
```

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

	CustomerID	Gender	Age	Annual_Income	Spending_Score
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

TASK 8

Scaling the data

```
[ ] from sklearn.preprocessing import scale
```

```
[ ] data=pd.DataFrame(scale(X),columns=X.columns)
data.head()
```

	CustomerID	Gender	Age	Annual_Income	Spending_Score
0	-1.723412	1.128152	-1.424569	-1.738999	-0.434801
1	-1.706091	1.128152	-1.281035	-1.738999	1.195704
2	-1.688771	-0.886405	-1.352802	-1.700830	-1.715913
3	-1.671450	-0.886405	-1.137502	-1.700830	1.040418
4	-1.654129	-0.886405	-0.563369	-1.662660	-0.395980

TASK 9

Perform any of the clustering algorithms

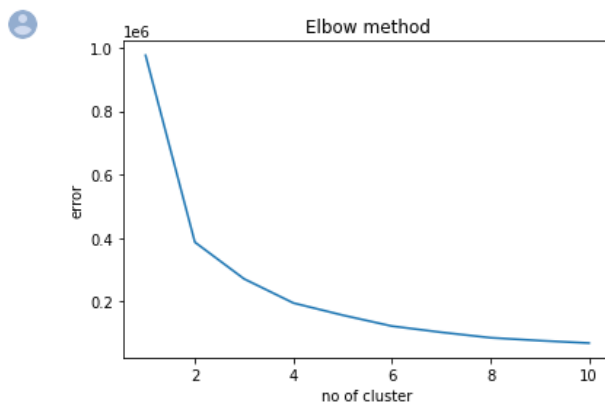
from sklearn import cluster

```
error = []
for i in range(1,11):
    kmeans=cluster.KMeans(n_clusters=i,init='k-means++',random_state=0)
    kmeans.fit(df)
    error.append(kmeans.inertia_)
```

```
[ ] error

[975512.0600000003,
 387065.71377137717,
 271384.508782868,
 195401.19855991466,
 157157.7579059829,
 122625.19813553878,
 103233.01724386725,
 86053.67444777445,
 76938.97565600359,
 69231.33607611558]
```

```
import matplotlib.pyplot as plt
plt.plot(range(1,11),error)
plt.title('Elbow method')
plt.xlabel('no of cluster')
plt.ylabel('error')
plt.show()
```



```
[ ] k_means_model=cluster.KMeans(n_clusters=3,init='k-means++',random_state=0)
```

```
[ ] k_means_model.fit(df)
```

```
KMeans(n_clusters=3, random_state=0)
```

```
[ ] clustered_data =k_means_model.predict(df)
```

TASK 10

Add the cluster data with the primary dataset

```
[ ] df['Clustered_data'] = pd.Series(clustered_data)
df.head()
```

	CustomerID	Gender	Age	Annual_Income	Spending_Score	Clustered_data
0	1	1	19	15	39	0
1	2	1	21	15	81	0
2	3	0	20	16	6	0
3	4	0	23	16	77	0
4	5	0	31	17	40	0

TASK 11

Split the data into dependent and independent variables

```
▶ y=df['Clustered_data']
y #y - target columns
```

```
0 0
1 0
2 0
3 0
4 0
..
195 2
196 2
197 2
198 2
199 2
Name: Clustered_data, Length: 200, dtype: int32
```

```
[ ] X=df.drop(columns=['Clustered_data'],axis=1)
X.head() #X - predicting columns
```

	CustomerID	Gender	Age	Annual_Income	Spending_Score
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

✓ Scale the independent variables

```
from sklearn.preprocessing import scale
```

```
[ ] data=pd.DataFrame(scale(X),columns=X.columns)
data.head()
```

	CustomerID	Gender	Age	Annual_Income	Spending_Score
0	-1.723412	1.128152	-1.424569	-1.738999	-0.434801
1	-1.706091	1.128152	-1.281035	-1.738999	1.195704
2	-1.688771	-0.886405	-1.352802	-1.700830	-1.715913
3	-1.671450	-0.886405	-1.137502	-1.700830	1.040418
4	-1.654129	-0.886405	-0.563369	-1.662660	-0.395980

TASK 12

Split the data into training and testing

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(data,y,test_size=0.3,random_state=1)
```

```
[ ] X_train.shape,X_test.shape

((140, 5), (60, 5))
```

```
[ ] y_train.shape,y_test.shape

((140,), (60,))
```

TASK 13

Build the model

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

```
[ ] model.fit(X_train,y_train) # K - Nearest Neighbour model (KNN)

KNeighborsClassifier()
```

TASK 14

Train the model

Loading...


```
▶ pred_train = model.predict(X_train)
pred_train
```


```
array([1, 1, 1, 0, 0, 0, 2, 1, 0, 1, 0, 1, 2, 2, 2, 1, 0, 1, 1, 1, 2, 1,
1, 1, 2, 0, 1, 1, 2, 0, 1, 0, 2, 2, 2, 1, 2, 2, 2, 2, 1, 0, 1, 2,
0, 1, 1, 2, 0, 1, 0, 2, 1, 1, 1, 2, 1, 2, 0, 1, 1, 1, 2, 2, 2, 1,
2, 2, 2, 0, 0, 1, 2, 1, 2, 0, 2, 0, 2, 1, 2, 2, 1, 2, 1, 0, 0, 2,
1, 1, 0, 0, 1, 0, 0, 0, 2, 0, 2, 1, 2, 0, 1, 1, 2, 0, 1, 2, 0, 1,
0, 1, 1, 0, 2, 2, 1, 1, 1, 0, 2, 2, 2, 2, 2, 1, 0, 2, 0, 2, 1, 2,
2, 2, 1, 2, 2, 1, 2, 0], dtype=int32)
```


TASK 15


Test the model

y_test

 y_test

 58 0
40 0
34 0
102 1
184 2
198 2
95 1
4 0
29 0
168 2
171 2
18 0
11 0
89 1
110 1
118 1
159 2
35 0
136 2
59 0
51 0
16 0
44 0
94 1
31 0
162 2
38 0
28 0
193 2
27 0
47 0
165 2
194 2
177 2
176 2
97 1

 `pred_test=model.predict(X_test)`
`pred_test`

 `array([0, 1, 0, 1, 2, 2, 1, 0, 0, 2, 2, 0, 0, 1, 1, 1, 2, 0, 2, 1, 1, 0,
0, 1, 0, 2, 0, 0, 2, 0, 0, 2, 2, 2, 2, 1, 2, 1, 0, 2, 1, 1, 2, 0,
0, 0, 1, 0, 2, 1, 1, 1, 1, 1, 0, 2, 2, 1, 2, 2], dtype=int32)`

```
[ ] pred = pd.DataFrame({'Actual_value':y_test,'Predicted_value_using_KNN':pred_test})  
pred.head()
```

	Actual_value	Predicted_value_using_KNN
58	0	0
40	0	1
34	0	0
102	1	1
184	2	2

TASK 16

Measure the performance using metrics

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
[ ] #Accuracy Score
print('Training accuracy: ', accuracy_score(y_train, pred_train))
print('Testing accuracy: ', accuracy_score(y_test, pred_test))
```

```
Training accuracy:  0.9214285714285714
Testing accuracy:  0.9166666666666666
```

```
[ ] #Confusion Matrix
pd.crosstab(y_test, pred_test)
```

	col_0	0	1	2
Clustered_data				
0		19	4	0
1		1	16	0
2		0	0	20

```
[ ] #Classification Report
print(classification_report(y_test, pred_test))
```

	precision	recall	f1-score	support
0	0.95	0.83	0.88	23
1	0.80	0.94	0.86	17
2	1.00	1.00	1.00	20
accuracy			0.92	60
macro avg	0.92	0.92	0.92	60
weighted avg	0.92	0.92	0.92	60