Assignment -1

Python Programming

Assignment Date	12 September 2022
Student Name	Ms. Monica V
Student Roll Number	19CS079
Maximum Marks	2 Marks

Question-1:

Split this string

s = "Hi there Sam!"

Solution:

```
s = "Hi there Sam!"
x = s.split()
print(x)
```

```
In [1]: s = "Hi there Sam!"
        x = s.split()
        print(x)

        ['Hi', 'there', 'Sam!']
```

Question-2:

Use .format() to print the following string.

Output should be: The diameter of Earth is 12742 kilometers.

Solution:

```
planet = "Earth"
```

diameter = 12742

s1="The diameter of {}"

s2=" is {} kilometers."

s3=s1+s2

print(s1.format(planet)+s2.forma

t(diameter))

```
[n [2]: planet = "Earth"
        diameter = 12742
        s1="The diameter of {}"
        s2=" is {} kilometers."
        s3=s1+s2
        print(s1.format(planet)+s2.format(diameter))

        The diameter of Earth is 12742 kilometers.
```

Question 3:

In this nest dictionary grab the word "hello"

Solution:

d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}
d['k1'][3]['tricky'][3]['target'][3]

```
3]: d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}
    d['k1'][3]['tricky'][3]['target'][3]
3]: 'hello'
```

Question 4.1:

Create an array of 10 zeros?

Solution:

import numpy as np

array=np.zeros(10)
print("An array of 10 zeros:")
print(array)

```
import numpy as np
array=np.zeros(10)
print("An array of 10 zeros:")
print(array)
```

```
An array of 10 zeros:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Question 4.2:

Create an array of 10 fives?

Solution:

import numpy as np
array=np.ones(10)*5
print("An array of 10 fives:")
print(array)

```
import numpy as np
array=np.ones(10)*5
print("An array of 10 fives:")
print(array)
```

```
An array of 10 fives:
[5. 5. 5. 5. 5. 5. 5. 5. 5. 5.]
```

Question 5:

Create an array of all the even integers from 20 to 35

Solution:

import numpy as np
array=np.arange(20,36,2)
print("Array of all the even integers from 20 to 35")
print (array)

```
import numpy as np
array=np.arange(20,36,2)
print("Array of all the even integers from 20 to 35")
print (array)
```

```
Array of all the even integers from 20 to 35
[20 22 24 26 28 30 32 34]
```

Question 6:

Create a 3x3 matrix with values ranging from 0 to 8

Solution:

import numpy as np
x = np.arange(0,9).reshape(3,3)
print(x)

```
In [7]: import numpy as np
        x = np.arange(0,9).reshape(3,3)
        print(x)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Question 7:

Concatenate a and b

a = np.array([1, 2, 3]), b = np.array([4, 5, 6])

Solution:

```
import numpy as np
a=np.array([1,2,3])
b=np.array([4,5,6])
c=np.concatenate((a,b))
print(c)
```

```
In [8]: import numpy as np
        a=np.array([1,2,3])
        b=np.array([4,5,6])
        c=np.concatenate((a,b))
        print(c)

        [1 2 3 4 5 6]
```

Question 8:

Create a dataframe with 3 rows and 2 columns

Solution:

```
import pandas as pd
data=[10,20,30,40,50,60]
df=pd.DataFrame(data,columns=['Numbers'])
print(df)
```

```
In [9]: import pandas as pd
        data=[10,20,30,40,50,60]
        df=pd.DataFrame(data,columns=['Numbers'])
        print(df)
```

```
   Numbers
0       10
1       20
2       30
3       40
4       50
5       60
```

Question 9:

Generate the series of dates from 1st Jan, 2023 to 10th Feb, 2023

Solution:

```
import pandas as pd
from datetime import timedelta, date
def get_date_range(start,end):
   return[start+ timedelta(n) for n in range(int((end-start).days))]
print(get_date_range(date(2023,1,1), date(2023,2,11)))
```

```
In [10]: import pandas as pd
         from datetime import timedelta, date
         def get_date_range(start,end):
             return[start+ timedelta(n) for n in range(int((end-start).days))]
         print(get_date_range(date(2023,1,1), date(2023,2,11)))
```

```
[datetime.date(2023, 1, 1), datetime.date(2023, 1, 2), datetime.date(2023, 1, 3), datetime.date(2023, 1, 4), datetime.date(2023, 1, 5), datetime.date(2023, 1, 6), datetime.date(2023, 1, 7), datetime.date(2023, 1, 8), datetime.date(2023, 1, 9), datetime.date(2023, 1, 10), datetime.date(2023, 1, 11), datetime.date(2023, 1, 12), datetime.date(2023, 1, 13), datetime.date(2023, 1, 14), datetime.date(2023, 1, 15), datetime.date(2023, 1, 16), datetime.date(2023, 1, 17), datetime.date(2023, 1, 18), datetime.date(2023, 1, 19), datetime.date(2023, 1, 20), datetime.date(2023, 1, 21), datetime.date(2023, 1, 22), datetime.date(2023, 1, 23), datetime.date(2023, 1, 24), datetime.date(2023, 1, 25), datetime.date(2023, 1, 26), datetime.date(2023, 1, 27), datetime.date(2023, 1, 28), datetime.date(2023, 1, 29), datetime.date(2023, 1, 30), datetime.date(2023, 1, 31), datetime.date(2023, 2, 1), datetime.date(2023, 2, 2), datetime.date(2023, 2, 3), datetime.date(2023, 2, 4), datetime.date(2023, 2, 5), datetime.date(2023, 2, 6), datetime.date(2023, 2, 7), datetime.date(2023, 2, 8), datetime.date(2023, 2, 9), datetime.date(2023, 2, 10)]
```

Question 10:

Create 2D list to DataFrame

Solution:

```
import pandas as pd
lists = [[1, 'aaa', 22], [2, 'bbb', 25], [3, 'ccc', 24]]
df=pd.DataFrame(lists,columns=['S.no', 'name' , 'value'])
print(df)
```

In [11]:
```
import pandas as pd
lists = [[1, 'aaa', 22], [2, 'bbb', 25], [3, 'ccc', 24]]
df=pd.DataFrame(lists,columns=['S.no', 'name' , 'value'])
print(df)
```

```
   S.no name  value
0     1  aaa     22
1     2  bbb     25
2     3  ccc     24
```

Assignment -2
Data Visualization and Pre-processing

Assignment Date	22 September 2022
Student Name	Ms. Monica V
Student Roll Number	19CS079
Maximum Marks	2 Marks

Perform Below Tasks to complete the assignment:-

Tasks:-

Question 1:

1. Download the dataset: Dataset

Solution:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Question 2:

2. Load the dataset.

Solution:

```
df=pd.read_csv("LCFS.csv")
```

LCFS_dataset

```
In [8]: LCFS_dataset
```

8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52

df.head()

```
In [9]: df.head()
```

Out[9]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

df.shape

```
In [10]: df.shape
```

```
Out[10]: (10000, 14)
```

Question 3:

3. Perform Below Visualizations.

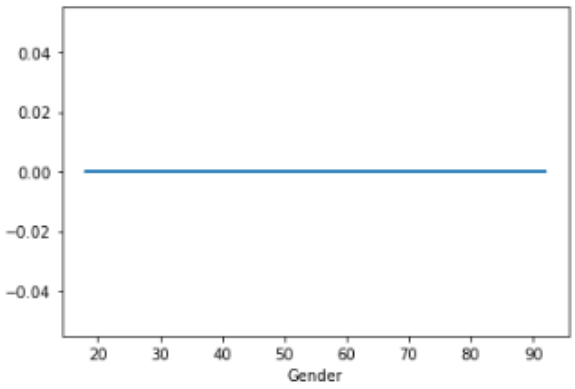
- Univariate Analysis
- Bi - Variate Analysis
- Multi - Variate Analysis

Solution:

Univariate Analysis

```
df_France=df.loc[df['Geography']=='France']

plt.plot(df_France['Age'], np.zeros_like(df_France['Age']))
plt.xlabel('Gender')
plt.show()
```

```
In [15]: plt.plot(df_France['Age'], np.zeros_like(df_France['Age']))
         plt.xlabel('Gender')
         plt.show()
```

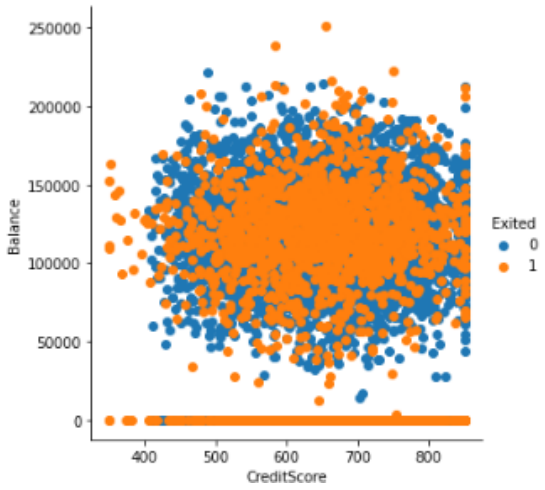


Bi-variate Analysis

sns.FacetGrid(df,hue="Exited",size=5).map(plt.scatter,"CreditScore" , "Balance").add_legend();
plt.show()

```
In [25]: sns.FacetGrid(df,hue="Exited",size=5).map(plt.scatter,"CreditScore" , "Balance").add_legend();
         plt.show()
```

```
C:\Users\MONICA\Python\lib\site-packages\seaborn\axisgrid.py:316: UserWarning: The `size` parameter
`; please update your code.
  warnings.warn(msg, UserWarning)
```



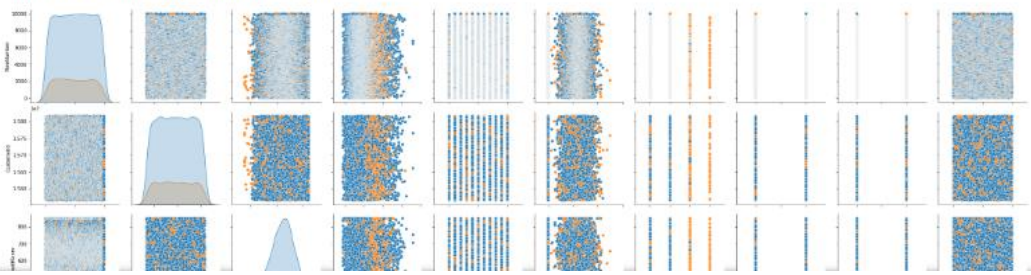
Multivariate Analysis

sns.pairplot(df,hue="Exited",size=3)

```
In [28]: sns.pairplot(df,hue="Exited",size=3)
```

```
C:\Users\MONICA\Python\lib\site-packages\seaborn\axisgrid.py:1969: UserWarning: The `size` parameter has been renamed to `h
eight`; please update your code.
  warnings.warn(msg, UserWarning)
```

```
Out[28]: <seaborn.axisgrid.PairGrid at 0x1fa53b6f6d0>
```



Question 4:

4. Perform descriptive statistics on the dataset.

Solution:

Descriptive Statistic Analysis

df.describe()

```
In [29]: df.describe()
```

Out[29]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000

df['CreditScore'].value_counts().to_frame()

```
In [33]: df['CreditScore'].value_counts().to_frame()
```

Out[33]:

	CreditScore
850	233
678	63
655	54
667	53
705	53
...	...
412	1
351	1
365	1
373	1
423	1

460 rows × 1 columns

```
Creditscore_counts=df['CreditScore'].value_counts().to_frame()
Creditscore_counts.rename(columns={'CreditScore':'value counts'},inplace=True)
Creditscore_counts
Creditscore_counts.index.name='Model'
Creditscore_counts
```

```
In [36]: Creditscore_counts=df['CreditScore'].value_counts().to_frame()
         Creditscore_counts.rename(columns={'CreditScore':'value counts'},inplace=True)
         Creditscore_counts
         Creditscore_counts.index.name='Model'
         Creditscore_counts
```

Out[36]:

Model	value counts
850	233
678	63
655	54
667	53
705	53
...	...
412	1
351	1
365	1
373	1
423	1

460 rows × 1 columns

Question 5:

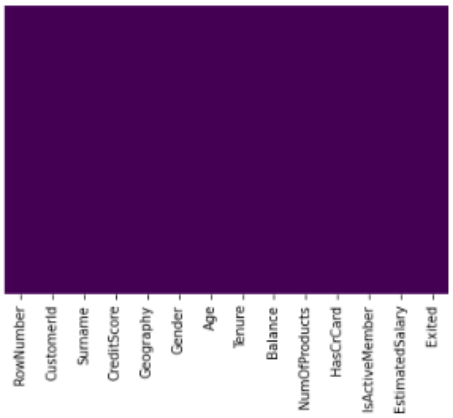
5. Handle the Missing values.

Solution:

Handle Missing Data

sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')

```
In [38]: sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
Out[38]: <AxesSubplot:>
```



Question 6:

6. Find the outliers and replace the outliers

Solution:

Detecting and replacing outliers

```
max_thresold=df['CreditScore'].quantile(0.95)
max_thresold
df[df['CreditScore']>max_thresold]
min_thresold=df['CreditScore'].quantile(0.05)
min_thresold
df[df['CreditScore']<min_thresold]
```

```
In [44]: max_thresold=df['CreditScore'].quantile(0.95)
         max_thresold
         df[df['CreditScore']>max_thresold]
         min_thresold=df['CreditScore'].quantile(0.05)
         min_thresold
         df[df['CreditScore']<min_thresold]
```

Out[44]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedS
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	1193
12	13	15632264	Kay	476	France	Female	34	10	0.00	2	1	0	262
29	30	15656300	Lucciano	411	France	Male	29	0	59697.17	2	1	1	534
35	36	15794171	Lombardo	475	France	Female	45	0	134264.04	1	1	0	278
40	41	15619360	Hsiao	472	Spain	Male	40	4	0.00	1	1	0	701
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9879	9880	15669414	Pisano	486	Germany	Male	62	9	118356.89	2	1	0	1680
9907	9908	15611247	McKenzie	481	France	Female	28	10	0.00	2	1	0	1452
9930	9931	15713604	Rossi	425	Germany	Male	40	9	166776.60	2	0	1	1726
9964	9965	15642785	Douglas	479	France	Male	34	5	117593.48	2	0	0	1133
9966	9967	15590213	Ch'en	479	Spain	Male	35	4	125920.98	1	1	1	203

490 rows × 14 columns

df[(df['CreditScore']<max_thresold) & (df['CreditScore']>min_thresold)]

```
In [45]: df[(df['CreditScore']<max_thresold) & (df['CreditScore']>min_thresold)]
```

Out[45]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedS
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	10134
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	11254
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	11393
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	9382
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	14975
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	9627
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	10169
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	4208
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	9288
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	3819

8987 rows × 14 columns

Question 7:

7. Check for Categorical columns and perform encoding.

Solution:

Categorical column and perform Encoding

```
df=pd.read_csv("LCFS.csv" , usecols=['CustomerId' , 'CreditScore'])
df.head()
```

```
In [47]: df=pd.read_csv("LCFS.csv" , usecols=['CustomerId' , 'CreditScore'])
         df.head()
```

Out[47]:

	CustomerId	CreditScore
0	15634602	619
1	15647311	608
2	15619304	502
3	15701354	699
4	15737888	850

pd.get_dummies(df).shape

```
In [49]: pd.get_dummies(df).shape
Out[49]: (10000, 2)
```

len(df['CustomerId'].unique())

```
In [50]: len(df['CustomerId'].unique())
Out[50]: 10000
```

len(df['CreditScore'].unique())

```
In [51]: len(df['CreditScore'].unique())
Out[51]: 460
```

```
for col in df.columns[0:]:
  print(col, ':' , len(df[col].unique()),'labels')
```

```
In [52]: for col in df.columns[0:]:
             print(col, ':' , len(df[col].unique()),'labels')
         CustomerId : 10000 labels
         CreditScore : 460 labels
```

df.CreditScore.value_counts().to_dict()

```
In [54]: df.CreditScore.value_counts().to_dict()
Out[54]: {850: 233,
          678: 63,
          655: 54,
          667: 53,
          705: 53,
          684: 52,
          651: 50,
          670: 50,
          683: 48,
          648: 48,
          652: 48,
          660: 48,
          682: 47,
          640: 47,
          663: 47,
          637: 46,
          714: 45,
          687: 45,
          645: 45,
```

```
df_frequency_map=df.CreditScore.to_dict()
df.CreditScore=df.CreditScore.map(df_frequency_map)
df.head()
```

```
In [57]: df.CreditScore=df.CreditScore.map(df_frequency_map)
         df.head()
```

Out[57]:

	CustomerId	CreditScore
0	15634602	673
1	15647311	699
2	15619304	774
3	15701354	742
4	15737888	646

Question 8:

8. Split the data into dependent and independent variables.

Solution:

Split Data into Dependent and Independent

```
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
print(X)
```

```
In [63]: X=df.iloc[:,:-1].values
         y=df.iloc[:,-1].values
         print(X)
```

```
[[15634602]
 [15647311]
 [15619304]
 ...
 [15584532]
 [15682355]
 [15628319]]
```

Question 9:

9. Scale the independent variables

Solution:

Scale the Independent Variables

```
from sklearn.preprocessing import StandardScaler
from sklearn import datasets
from sklearn.model_selection import train_test_split

digits=datasets.load_digits()

X=digits.data
print();print(X.shape)

Y=digits.target
print();print(Y.shape)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.33,random_state=42)

print(); print(X_train.shape)
print(); print(X_test.shape)
print(); print(Y_train.shape)
print(); print(Y_test.shape)
```

In [102]:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split

digits=datasets.load_digits()

X=digits.data
print();print(X.shape)

Y=digits.target
print();print(Y.shape)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.33,random_state=42)

print(); print(X_train.shape)
print(); print(X_test.shape)
print(); print(Y_train.shape)
print(); print(Y_test.shape)
```

```
(1797, 64)

(1797,)

(1203, 64)

(594, 64)

(1203,)

(594,)
```

Assignment -3
ABALONE AGE PREDICTION

Assignment Date	4 October 2022
Student Name	Ms. Monica V
Student Roll Number	19CS079
Maximum Marks	2 Marks

Problem Statement: Abalone Age Prediction

Description:- Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through
a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

Attribute Information:
Given is the attribute name, attribute type, measurement unit, and a brief description. The number of
rings is the value to predict: either as a continuous value or as a classification problem.

Name / Data Type / Measurement Unit / Description
1- Sex / nominal / -- / M, F, and I (infant)
2- Length / continuous / mm / Longest shell measurement
3- Diameter / continuous / mm / perpendicular to length
4- Height / continuous / mm / with meat in shell
5- Whole weight / continuous / grams / whole abalone
6- Shucked weight / continuous / grams / weight of meat
7- Viscera weight / continuous / grams / gut weight (after bleeding)
8- Shell weight / continuous / grams / after being dried
9- Rings / integer / -- / +1.5 gives the age in years

Building a Regression Model

Question 1:

1. Download the dataset: Dataset

Solution:

IMPORT AND LOAD DATASET

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sma
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

Question 2:

2. Load the dataset into the tool.

Solution:

```
data = pd.read_csv("../input/abalone/abalone.csv")
```

Question 3:

3. Perform Below Visualizations.
· Univariate Analysis
· Bi-Variate Analysis
· Multi-Variate Analysis

Solution:

Perform 1. UNIVARIATE ANALYSIS 2. BI-VARIATE ANALYSIS 3. MULTI VARIATE ANALYSIS Visualizations.

```
#rename output variable
data.rename(columns={"Sex":"sex", "Length":"length", "Diameter":"diameter",
                     "Height":"height", "Whole weight":"whole_weight",
```

"Shucked weight":"shucked_weight", "Viscera weight":"viscera_weight",
"Shell weight":"shell_weight", "Rings":"rings"}, inplace = True)

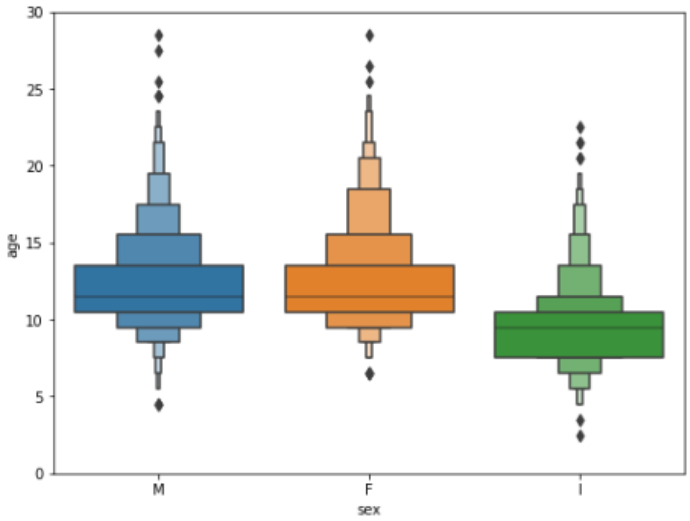
```
data[data['height'] == 0]  #need to drop these rows.
data.drop(index=[1257,3996], inplace = True)
data.shape
```

```
In [ ]: data[data['height'] == 0]  #need to drop these rows.
        data.drop(index=[1257,3996], inplace = True)
        data.shape
```

```
Out[141]: (4175, 9)
```

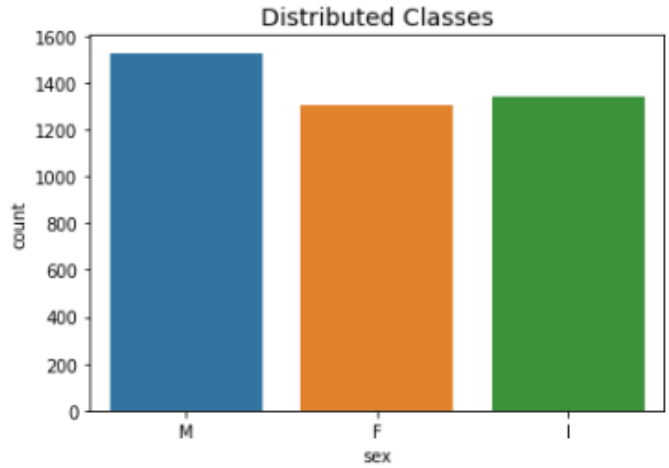
```
data['age'] = data['rings']+1.5 #AS per the problem statement
data.drop('rings', axis = 1, inplace = True)
data.head()
#categorical features
temp = pd.concat([df['age'], df['sex']], axis=1)

f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxenplot(x='sex', y="age", data=data)
fig.axis(ymin=0, ymax=30);
```

```
In [ ]: data['age'] = data['rings']+1.5 #AS per the problem statement
        data.drop('rings', axis = 1, inplace = True)
        data.head()
        #categorical features
        temp = pd.concat([df['age'], df['sex']], axis=1)

        f, ax = plt.subplots(figsize=(8, 6))
        fig = sns.boxenplot(x='sex', y="age", data=data)
        fig.axis(ymin=0, ymax=30);
```




```
sns.countplot('sex', data=data)
plt.title('Distributed Classes', fontsize=14)
plt.show()
```

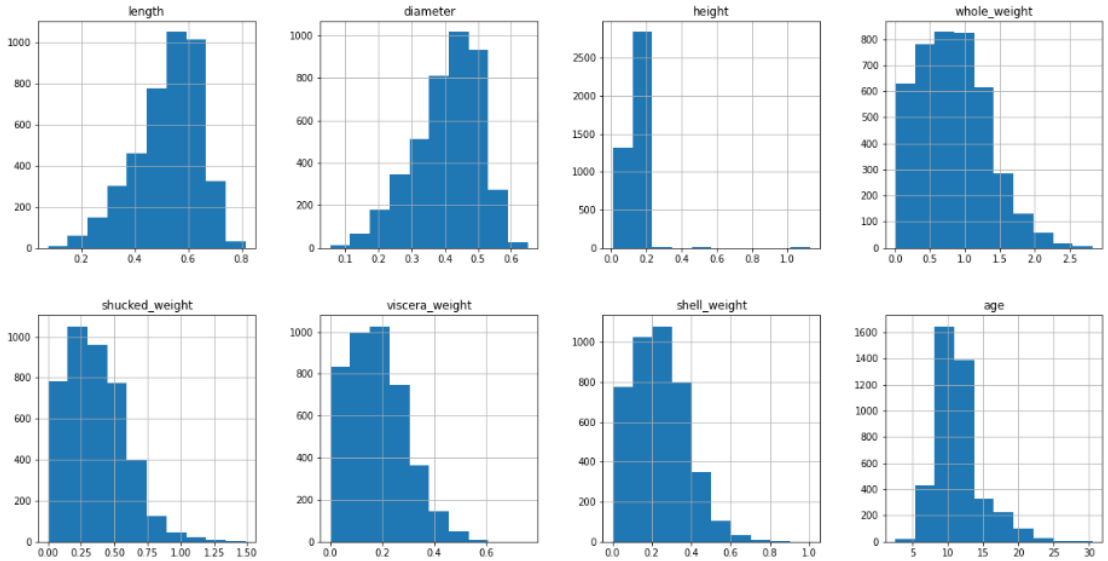
```
In [ ]: sns.countplot('sex', data=data)
        plt.title('Distributed Classes', fontsize=14)
        plt.show()
```



data.hist(figsize = (20,10), layout = (2,4))

```
In [ ]: data.hist(figsize = (20,10), layout = (2,4))
```

```
Out[144]: array([[<AxesSubplot:title={'center':'length'}>,
                  <AxesSubplot:title={'center':'diameter'}>,
                  <AxesSubplot:title={'center':'height'}>,
                  <AxesSubplot:title={'center':'whole_weight'}>],
                 [<AxesSubplot:title={'center':'shucked_weight'}>,
                  <AxesSubplot:title={'center':'viscera_weight'}>,
                  <AxesSubplot:title={'center':'shell_weight'}>,
                  <AxesSubplot:title={'center':'age'}>]], dtype=object)
```



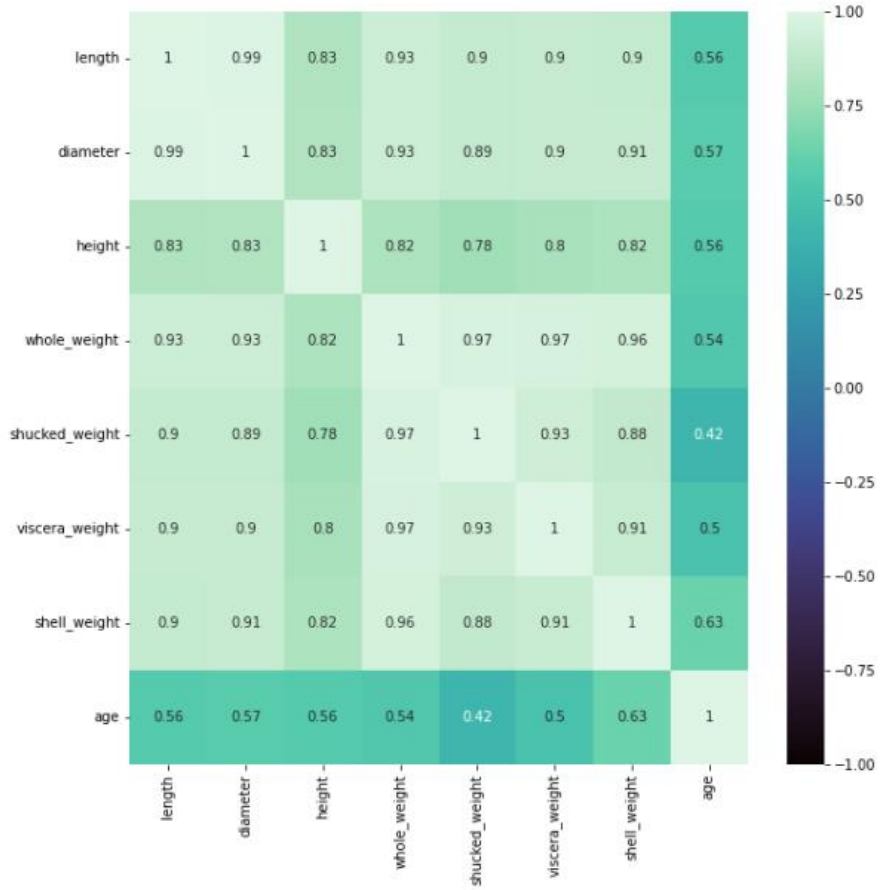
data.skew().sort_values(ascending = False)

```
In [ ]: data.skew().sort_values(ascending = False)
```

```
Out[145]: height            3.166364
          age               1.113754
          shucked_weight    0.718735
          shell_weight      0.621081
          viscera_weight    0.591455
          whole_weight      0.530549
          diameter         -0.610182
          length           -0.640993
          dtype: float64
```

```
corr = data.corr()
plt.figure(figsize = (10,10))
ax = sns.heatmap(corr, vmin = -1, center = 0, annot = True, cmap = 'mako')
```

```
In [ ]: corr = data.corr()
        plt.figure(figsize = (10,10))
        ax = sns.heatmap(corr, vmin = -1, center = 0, annot = True, cmap = 'mako')
```




```
upper_tri = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool))
columns_to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.95)]
#highly correlated variables to be removed.

print("Columns to drop:\n", columns_to_drop)
```

```
In [ ]: upper_tri = corr.where(np.triu(np.ones(corr.shape),k=1).astype(np.bool))
        columns_to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.95)] #highly correlated variables

        print("Columns to drop:\n", columns_to_drop)
```

```
Columns to drop:
 ['diameter', 'shucked_weight', 'viscera_weight', 'shell_weight']
```

Question 4:

4. Perform descriptive statistics on the dataset.

Solution:

DESCRIPTIVE STATISTICS

data.head()

```
In [ ]: data.head()
```

Out[148]:

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

data.shape

```
In [ ]: data.shape
```

```
Out[149]: (4175, 9)
```

data.describe()

```
In [ ]: data.describe()
```

Out[150]:

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age
count	4175.000000	4175.00000	4175.000000	4175.000000	4175.000000	4175.000000	4175.000000	4175.000000
mean	0.524065	0.40794	0.139583	0.829005	0.359476	0.180653	0.238834	11.435090
std	0.120069	0.09922	0.041725	0.490349	0.221954	0.109605	0.139212	3.224227
min	0.075000	0.05500	0.010000	0.002000	0.001000	0.000500	0.001500	2.500000
25%	0.450000	0.35000	0.115000	0.442250	0.186250	0.093500	0.130000	9.500000
50%	0.545000	0.42500	0.140000	0.800000	0.336000	0.171000	0.234000	10.500000
75%	0.615000	0.48000	0.165000	1.153500	0.502000	0.253000	0.328750	12.500000
max	0.815000	0.65000	1.130000	2.825500	1.488000	0.760000	1.005000	30.500000

data.info()

```
In [ ]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4175 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sex             4175 non-null   object
 1   length          4175 non-null   float64
 2   diameter        4175 non-null   float64
 3   height          4175 non-null   float64
 4   whole_weight    4175 non-null   float64
 5   shucked_weight  4175 non-null   float64
 6   viscera_weight  4175 non-null   float64
 7   shell_weight    4175 non-null   float64
 8   age             4175 non-null   float64
dtypes: float64(8), object(1)
memory usage: 455.2+ KB
```

Question 5:

5. Check for Missing values and deal with them.

Solution:

MISSING VALUES

data[data.duplicated()]

```
In [ ]: data[data.duplicated()]
```

Out[152]:

sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age

data.isna().sum()

```
In [ ]: data.isna().sum()
```

```
Out[153]: sex               0
          length            0
          diameter          0
          height            0
          whole_weight      0
          shucked_weight    0
          viscera_weight    0
          shell_weight      0
          age               0
          dtype: int64
```

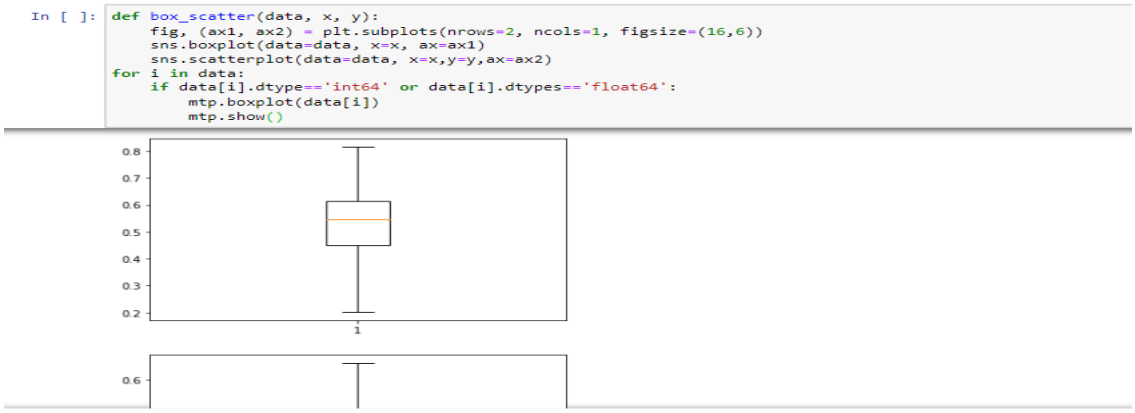
Question 6:

6. Find the outliers and replace them outliers

Solution:

REPLACE THE OUTLIERS

```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        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])
import matplotlib.pyplot as mtp
def box_scatter(data, x, y):
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))
    sns.boxplot(data=data, x=x, ax=ax1)
    sns.scatterplot(data=data, x=x,y=y,ax=ax2)
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        mtp.boxplot(data[i])
        mtp.show()
```



Question 7:

7. Check for Categorical columns and perform encoding

Solution:

ENCODING

data.head()

```
In [ ]: data.head()
```

Out[157]:

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
data['sex']=encoder.fit_transform(data['sex'])
data.head()

```
In [ ]: from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
data['sex']=encoder.fit_transform(data['sex'])
data.head()
```

Out[158]:

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

x=data.iloc[:,:-1]
x.head()

```
In [ ]: x=data.iloc[:,:-1]
x.head()
```

Out[159]:

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055

Question 8:

8. Split the data into dependent and independent variables.

Solution:

DEPENDENT AND INDEPENDENT VARIABLES

y=data.iloc[:,-1]

y.head()

```
In [ ]: y=data.iloc[:,-1]
        y.head()
```

```
Out[160]: 0    16.5
          1     8.5
          2    10.5
          3    11.5
          4     8.5
          Name: age, dtype: float64
```

Question 9:

9. Scale the independent variables

Solution:

INDEPENDENT VARIABLE SCALING

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x=scaler.fit_transform(x)
```

Question 10:

10. Split the data into training and testing

Solution:

SPLITING DATA

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
x_train.shape
```

```
In [ ]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33)
        x_train.shape
```

```
Out[162]: (2797, 8)
```

x_test.shape

```
In [ ]: x_test.shape
```

```
Out[163]: (1378, 8)
```

Question 11:

11. Build the Model

Solution:

BUILD THE MODEL

```
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
```

Question 12:

12. Train the Model

Solution:

TRAIN THE MODEL

```
reg.fit(x_train,y_train)
```

```
In [ ]: reg.fit(x_train,y_train)
Out[165]: RandomForestRegressor()
```

Question 13:

13. Test the Model

Solution:

TEST THE MODEL

```
y_pred=reg.predict(x_test)
```

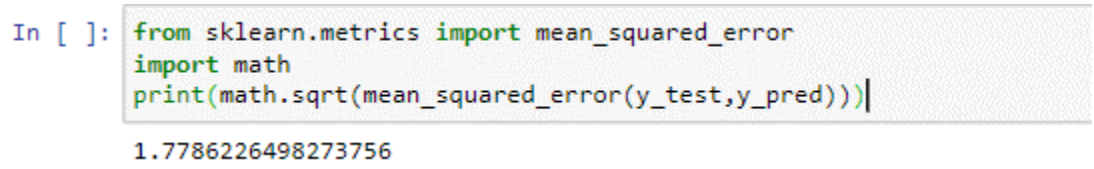
Question 14:

14. Measure the performance using Metrics

Solution:

PERFORMANCE MEASUREMENT USING METRICS

```
from sklearn.metrics import mean_squared_error
import math
print(math.sqrt(mean_squared_error(y_test,y_pred)))
```



Assignment -4
Customer Segmentation Analysis

Assignment Date	26 October 2022
Student Name	Ms. Monica V
Student Roll Number	19CS079
Maximum Marks	2 Marks

Problem Statement: Customer Segmentation Analysis

Description:- You own the mall and want to understand the customers who can quickly converge [Target Customers] so that the insight can be given to the marketing team and plan the strategy accordingly.

Perform the below Tasks to complete the assignment:-

Clustering the data and performing classification algorithms

Question 1:

1. Download the dataset: Dataset

Solution:

IMPORT AND LOAD DATASET

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sma
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

Question 2:

2. Load the dataset into the tool.

Solution:

```
df = pd.read_csv("MCD.csv")
```

```
dataset.head()
```

```
In [14]: dataset.head()
```

Out[14]:

	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.shape
```

```
df.describe()
```

```
In [16]: df.describe()
```

Out[16]:

	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

Question 3:

3. Perform Below Visualizations.

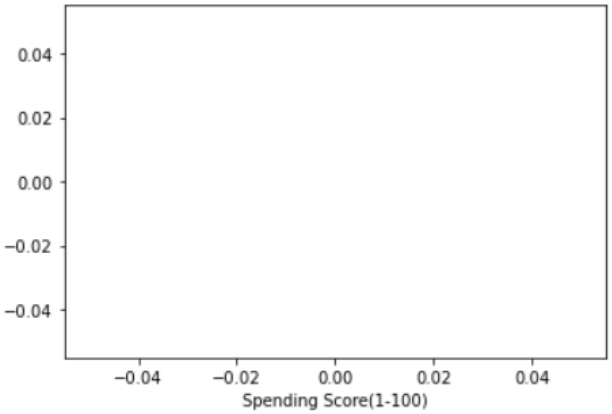
· Univariate Analysis
· Bi-Variate Analysis
· Multi-Variate Analysis

Solution:

Perform 1. UNIVARIATE ANALYSIS 2. BI-VARIATE ANALYSIS 3. MULTI VARIATE ANALYSIS Visualizations.

```
#Changing the name of some columns
df = df.rename(columns={'Annual Income (k$)': 'Annual_income', 'Spending Score (1-100)':
'Spending_score'})

#Replacing objects for numerical values
df['Gender'].replace(['Female','Male'], [0,1],inplace=True)

df_Gender=df.loc[df['Gender']=='Female']

plt.plot(df_Gender['Age'], np.zeros_like(df_Gender['Age']))
plt.xlabel('Spending Score(1-100)')
plt.show()
```

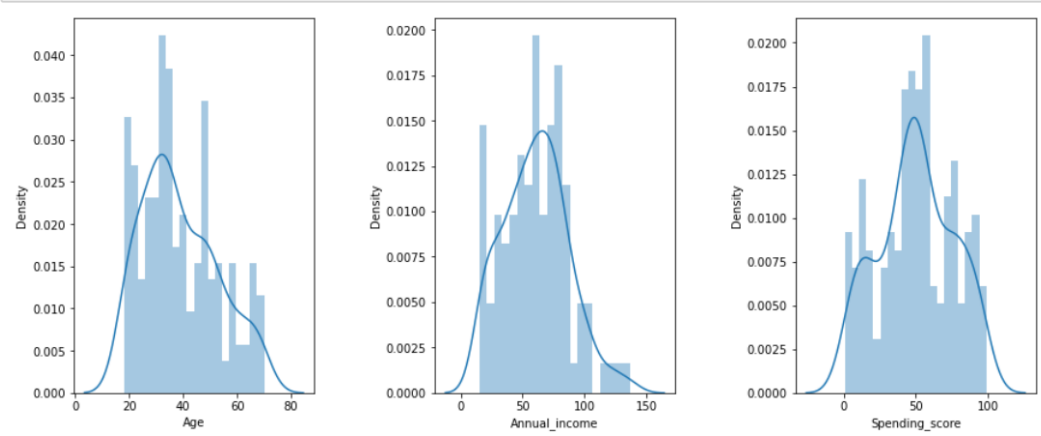
```
In [57]: plt.plot(df_Gender['Age'], np.zeros_like(df_Gender['Age']))
         plt.xlabel('Spending Score(1-100)')
         plt.show()
```




```
#Checking values have been replaced properly
df.Gender
```

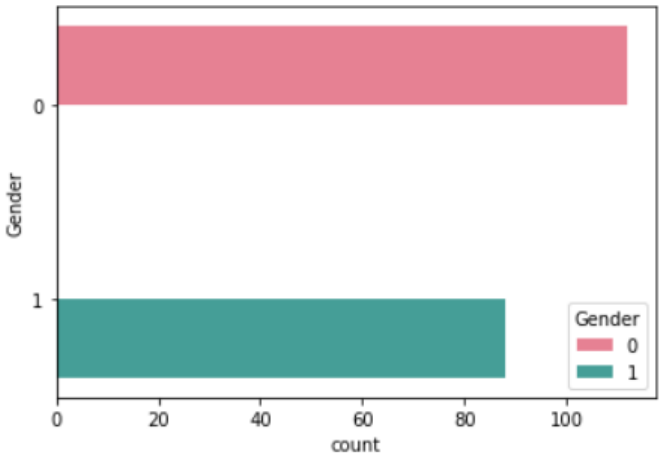
```
In [58]: #Checking values have been replaced properly
         df.Gender

Out[58]: 0      1.0
         1      1.0
         2      0.0
         3      0.0
         4      0.0
               ...
         195    0.0
         196    0.0
         197    1.0
         198    1.0
         199    1.0
         Name: Gender, Length: 200, dtype: float64
```

```
#Density estimation of values using distplot
plt.figure(1 , figsize = (15 , 6))
feature_list = ['Age','Annual_income', "Spending_score"]
feature_listt = ['Age','Annual_income', "Spending_score"]
pos = 1
for i in feature_list:
    plt.subplot(1 , 3 , pos)
    plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
    sns.distplot(df[i], bins=20, kde = True)
    pos = pos + 1
plt.show()
```




```
#Count and plot gender
sns.countplot(y = 'Gender', data = df, palette="husl", hue = "Gender")
df["Gender"].value_counts()
```

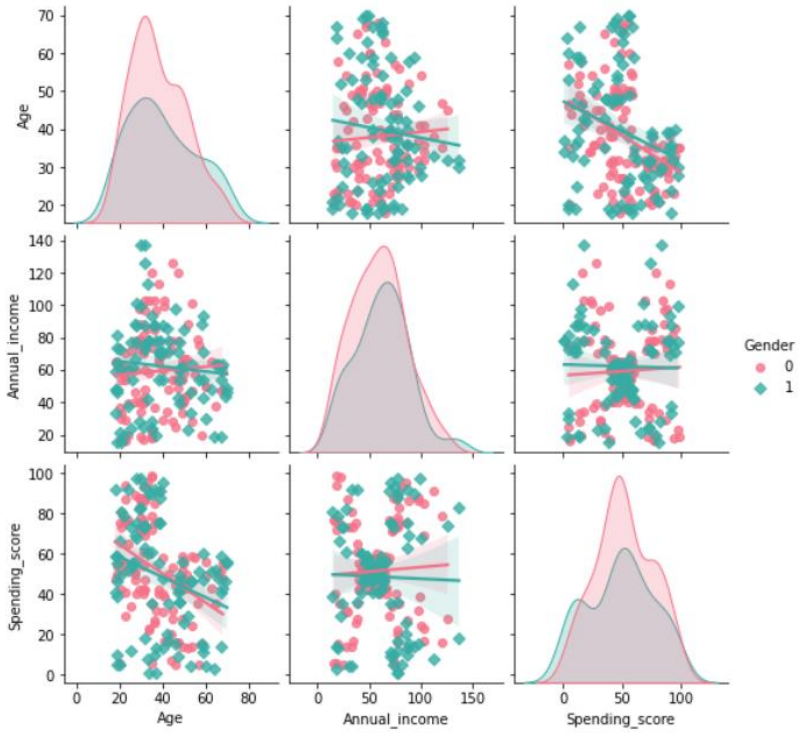
```
Out[23]: 0    112
         1     88
         Name: Gender, dtype: int64
```




```
#Pairplot with variables we want to study
```

```
sns.pairplot(df, vars=["Age", "Annual_income", "Spending_score"],  kind ="reg", hue = "Gender",
palette="husl", markers = ['o','D'])
```

```
Out[24]: <seaborn.axisgrid.PairGrid at 0x1a3d71e1640>
```



Question 4:

4. Perform descriptive statistics on the dataset.

Solution:

DESCRIPTIVE STATISTICS

```
df.head()
```

```
Out[25]:
```

	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

df.describe()

	CustomerID	Gender	Age	Annual_income	Spending_score
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	0.440000	38.850000	60.560000	50.200000
std	57.879185	0.497633	13.969007	26.264721	25.823522
min	1.000000	0.000000	18.000000	15.000000	1.000000
25%	50.750000	0.000000	28.750000	41.500000	34.750000
50%	100.500000	0.000000	36.000000	61.500000	50.000000
75%	150.250000	1.000000	49.000000	78.000000	73.000000
max	200.000000	1.000000	70.000000	137.000000	99.000000

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    int64
 2   Age             200 non-null    int64
 3   Annual_income   200 non-null    int64
 4   Spending_score  200 non-null    int64
dtypes: int64(5)
memory usage: 7.9 KB
```

df['Annual_income'].value_counts().to_frame()

Out[28]:

	Annual_income
54	12
78	12
48	6
71	6
63	6
...	...
58	2
59	2
16	2
64	2
137	2

64 rows × 1 columns

Question 5:

5. Check for Missing values and deal with them.

Solution:

MISSING VALUES

df[df.duplicated()]

CustomerID	Gender	Age	Annual_income	Spending_score

df.isna().sum()

```
CustomerID        0
Gender            0
Age               0
Annual_income     0
Spending_score    0
dtype: int64
```

Question 6:

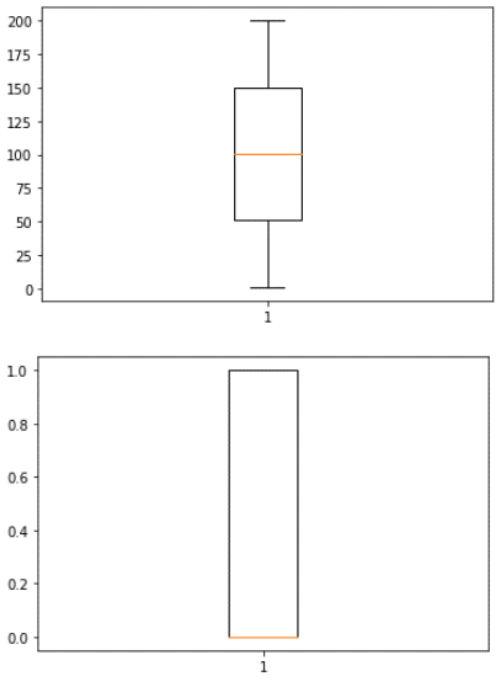
6. Find the outliers and replace them outliers

Solution:

REPLACE THE OUTLIERS

```
for i in df:
   if df[i].dtype=='int64' or df[i].dtypes=='float64':
      q1=df[i].quantile(0.25)
      q3=df[i].quantile(0.75)
      iqr=q3-q1
      upper=q3+1.5*iqr
      lower=q1-1.5*iqr
      df[i]=np.where(df[i] >upper, upper, df[i])
      df[i]=np.where(df[i] <lower, lower, df[i])
import matplotlib.pyplot as mtp
```

```
def box_scatter(df, x, y):
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))
    sns.boxplot(df=df, x=x, ax=ax1)
    sns.scatterplot(df=df, x=x,y=y,ax=ax2)
for i in df:
    if df[i].dtype=='int64' or df[i].dtypes=='float64':
        mtp.boxplot(df[i])
        mtp.show()
```



Question 7:

7. Check for Categorical columns and perform encoding

Solution:

ENCODING

```
data.head()
```

	CustomerID	Gender	Age	Annual_income	Spending_score
0	1.0	1.0	19.0	15.0	39.0
1	2.0	1.0	21.0	15.0	81.0
2	3.0	0.0	20.0	16.0	6.0
3	4.0	0.0	23.0	16.0	77.0
4	5.0	0.0	31.0	17.0	40.0

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df['Age']=encoder.fit_transform(df['Age'])
df.head()
```

	CustomerID	Gender	Age	Annual_income	Spending_score
0	1.0	1.0	1	15.0	39.0
1	2.0	1.0	3	15.0	81.0
2	3.0	0.0	2	16.0	6.0
3	4.0	0.0	5	16.0	77.0
4	5.0	0.0	13	17.0	40.0

```
x=df.iloc[:,:-1]
x.head()
```

	CustomerID	Gender	Age	Annual_income
0	1.0	1.0	1	15.0
1	2.0	1.0	3	15.0
2	3.0	0.0	2	16.0
3	4.0	0.0	5	16.0
4	5.0	0.0	13	17.0

Question 8:

8. Scaling the data

Solution:

DEPENDENT AND INDEPENDENT VARIABLES

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

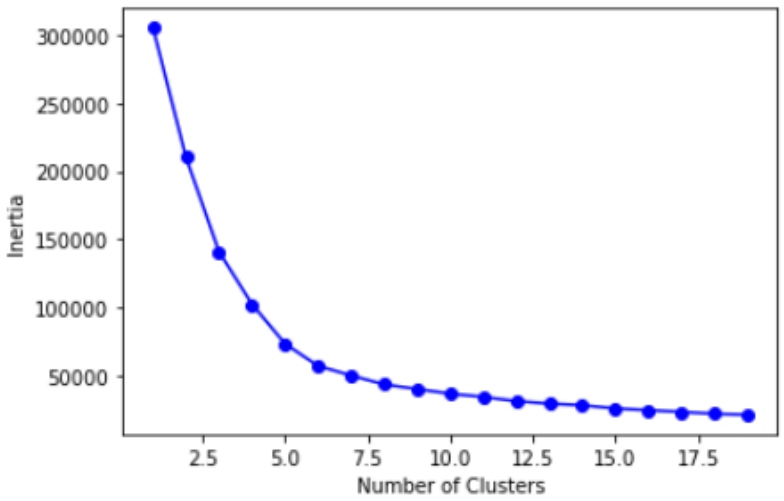
```
x=scaler.fit_transform(x)
```

Question 9:

9. Perform any of the clustering algorithms

Solution:

```
#Creating values for the elbow
from sklearn.cluster import KMeans
X = df.loc[:,["Age", "Annual_income", "Spending_score"]]
inertia = []
k = range(1,20)
for i in k:
   means_k = KMeans(n_clusters=i, random_state=0)
   means_k.fit(X)
   inertia.append(means_k.inertia_)

#Plotting the elbow
plt.plot(k , inertia , 'bo-')
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```



Question 10:

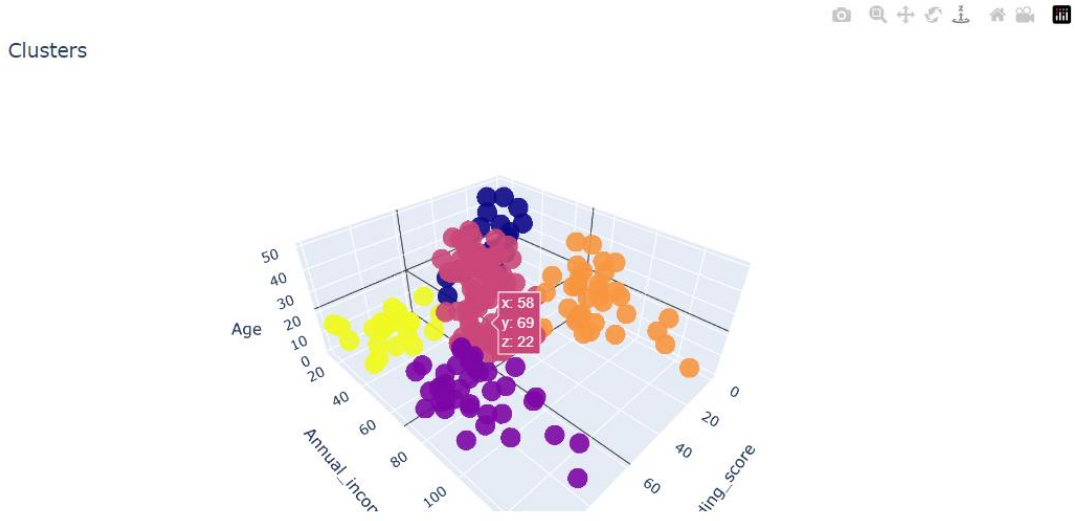
10. Add the cluster data with the primary dataset

Solution:

Clustering

```
#Training kmeans with 5 clusters
means_k = KMeans(n_clusters=5, random_state=0)
means_k.fit(X)
labels = means_k.labels_
centroids = means_k.cluster_centers_

pip install plotly

#Create a 3d plot to view the data sepparation made by Kmeans




import plotly as py
import plotly.express as px
import plotly.graph_objs as go



trace1 = go.Scatter3d(
   x= X['Spending_score'],
   y= X['Annual_income'],
   z= X['Age'],
   mode='markers',
    marker=dict(
      color = labels,
      size= 10,
      line=dict(
         color= labels,
      ),
      opacity = 0.9
    )
)
layout = go.Layout(
   title= 'Clusters',
   scene = dict(
         xaxis = dict(title  = 'Spending_score'),
         yaxis = dict(title  = 'Annual_income'),
         zaxis = dict(title  = 'Age')
      )
)
fig = go.Figure(data=trace1, layout=layout)
```

py.offline.iplot(fig)



Question 11:

11. Split the data into dependent and independent variables.

Solution:

BUILD THE MODEL

```
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
```

Question 12:

12. Split the data into training and testing

Solution:

```
from sklearn.model_selection import train_test_split

from sklearn import datasets
from sklearn.model_selection import train_test_split

digits=datasets.load_digits()

x=digits.data
print();print(x.shape)
```

```
y=digits.target
print();print(y.shape)

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_state=42)

print(); print(x_train.shape)
print(); print(x_test.shape)
print(); print(y_train.shape)
print(); print(y_test.shape)
```

```
(1797, 64)

(1797,)

(1203, 64)

(594, 64)

(1203,)

(594,)
```

Question 13:

13. Build the Model

Solution:

```
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
```

Question 14:

14. Train the Model

Solution:

```
reg.fit(x_train,y_train)
```

```
RandomForestRegressor()
```

Question 15:

15. Test the Model

Solution:

```
y_pred=reg.predict(x_test)
```

Question 16:

16. Measure the performance using Evaluation Metrics.

Solution:

```
from sklearn.metrics import mean_squared_error
import math
print(math.sqrt(mean_squared_error(y_test,y_pred)))
```

```
1.1693740591597244
```