ASSIGNMENT

APPLIED DATA SCIENCE

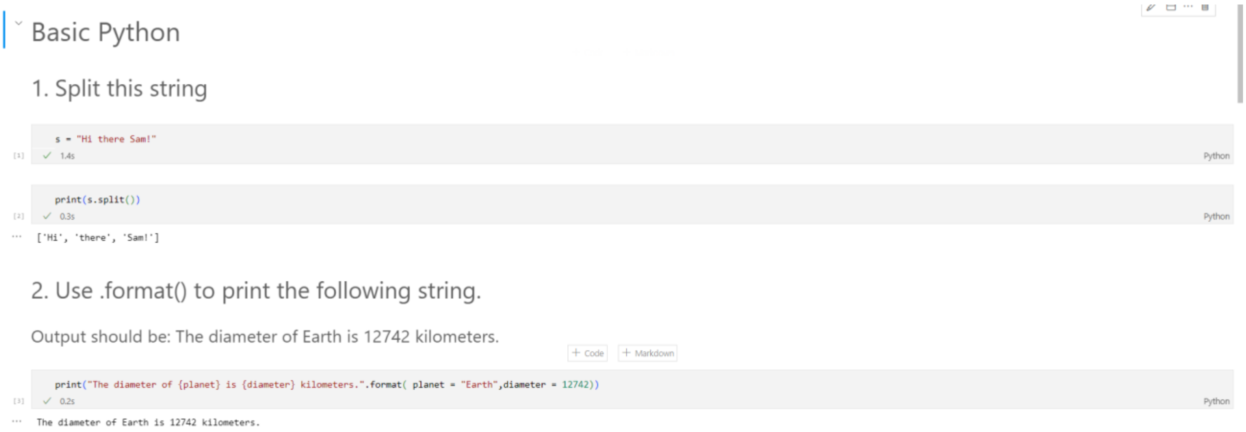
ASSIGNMENT 1

Assignment Date	12 September 2022
Student Name	Ms. Dharshana R
Student Roll Number	713319EC024
Maximum Marks	2 Marks

Basic Python

1. Split the String
2. Use .format() to print the following string.
   Output should be: The diameter of Earth is 12742 kilometers.
3. In this nest dictionary grab the word “hello”
4. **NUMPY**
   4.1 Create an array of 10 Zeros
   4.2 Create an array of 10 fives
5. Create an array of all the even integers from 20 to 35
6. Create a 3x3 matrix with values ranging from 0 to 8
7. Concatenate a and b
   a = np.array([1,2,3]), b = np.array([4,5,6])
   PANDAS
8. Create a dataframe with 3 rows and 2 columns
9. Generate the series of dates from 1st Jan, 2023 to 10th Feb, 2023
10. Create 2D list to DataFrame
    Lists = [[1, ‘aaa’, 22],[2,’bbb’,25],[3,’ccc’,24]]

SOLUTION:



Basic Python

1. Split this string

```
s = "Hi there Sam!"
```

[1] ✓ 1.4s Python

```
print(s.split())
```

[2] ✓ 0.3s Python

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

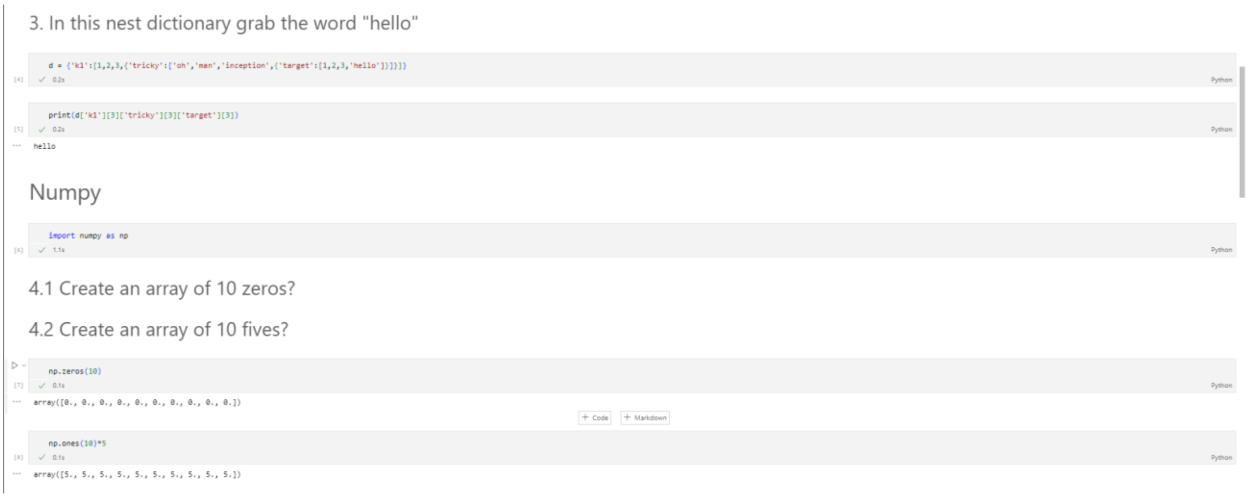
2. Use .format() to print the following string.

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

```
print("The diameter of {planet} is {diameter} kilometers.".format( planet = "Earth",diameter = 12742))
```

[3] ✓ 0.2s Python

```
The diameter of Earth is 12742 kilometers.
```



3. In this nest dictionary grab the word "hello"

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

```
print(d['k1'][3]['tricky'][3]['target'][3])
```

```
hello
```

Numpy

```
import numpy as np
```

4.1 Create an array of 10 zeros?

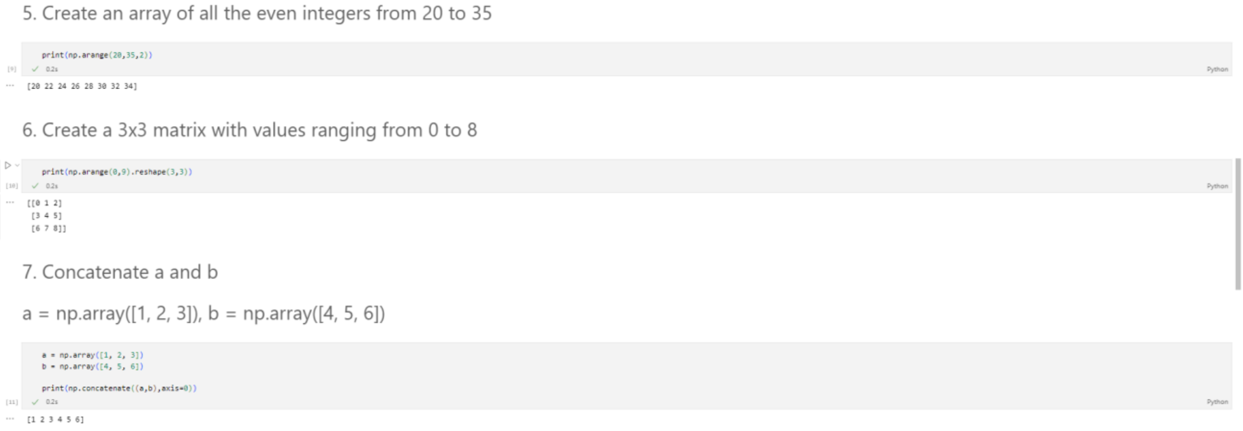
4.2 Create an array of 10 fives?

```
np.zeros(10)
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```
np.ones(10)*5
```

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```



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

```
print(np.arange(20,35,2))
```

```
[20 22 24 26 28 30 32 34]
```

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

```
print(np.arange(0,9).reshape(3,3))
```

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

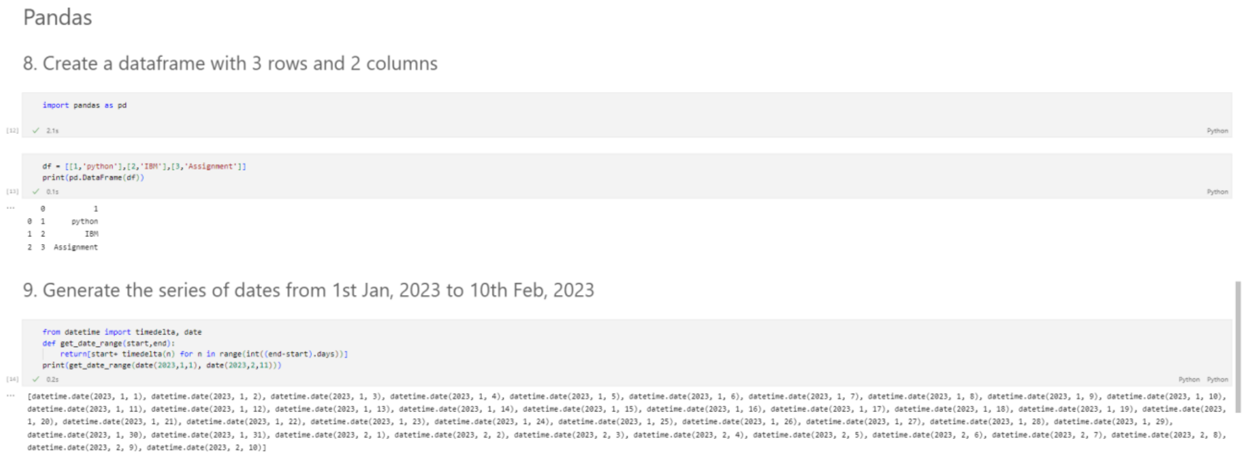
7. Concatenate a and b

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

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

print(np.concatenate((a,b),axis=0))
```

```
[1 2 3 4 5 6]
```



Pandas

8. Create a dataframe with 3 rows and 2 columns

```
import pandas as pd
```

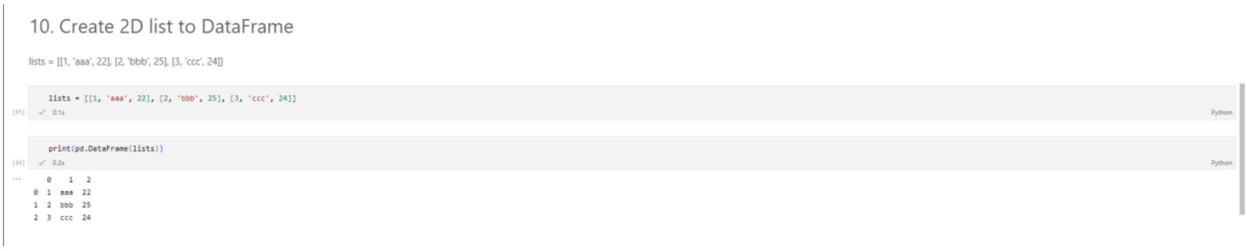
```
df = [[1,'python'],[2,'IBM'],[3,'Assignment']]
print(pd.DataFrame(df))
```

```
   0           1
0  1      python
1  2         IBM
2  3  Assignment
```

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

```
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)]
```



10. Create 2D list to DataFrame

lists = [[1, 'aaa', 22], [2, 'bbb', 25], [3, 'ccc', 24]]

```
lists = [[1, 'aaa', 22], [2, 'bbb', 25], [3, 'ccc', 24]]
```

```
print(pd.DataFrame(lists))
```

```
   0    1   2
0  1  aaa  22
1  2  bbb  25
2  3  ccc  24
```

ASSIGNMENT – 2

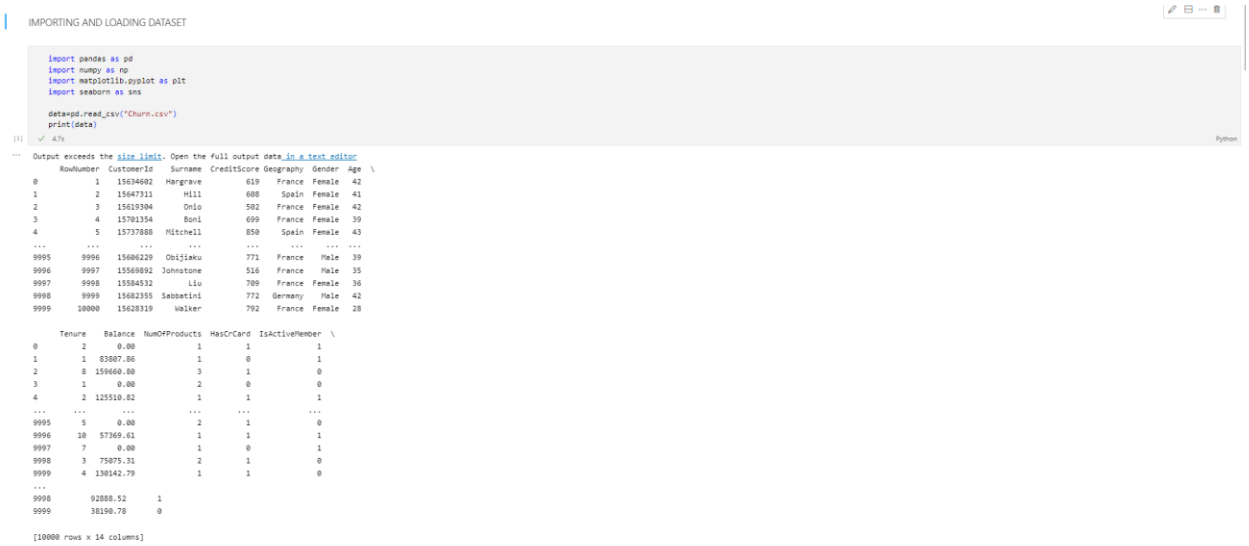
Assignment Date	22 September 2022
Student Name	Ms. Dharshana R
Student Roll Number	713319EC024
Maximum Marks	2 Marks

Data Visualization and Pre-Processing

Tasks:

1. Download the dataset
2. Load the dataset
3. Perform Below Visualizations.
   a. Univariate Analysis
   b. Bi – Variate Analysis
   c. Multi – Variate Analysis
4. Perform descriptive statistics on the dataset
5. Handle the Missing values
6. Find the outliers and replace the outliers
7. Check for Categorical columns and perform encoding
8. Split the data into dependent and independent variables
9. Scale the independent variables
10. Split the data into training and testing

SOLUTIONS:



IMPORTING AND LOADING DATASET

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

data=pd.read_csv("Churn.csv")
print(data)
```

[1] ✓ 4.7s

Python

```
Output exceeds the size limit. Open the full output data in a text editor
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender  Age  \
0             1    15634602   Hargrave          619    France  Female   42
1             2    15647311       Hill          608     Spain  Female   41
2             3    15619304       Onio          502    France  Female   42
3             4    15701354       Boni          699    France  Female   39
4             5    15737888   Mitchell          850     Spain  Female   43
...         ...         ...        ...          ...       ...     ...  ...
9995       9996    15606229   Obijiaku          771    France    Male   39
9996       9997    15569892  Johnstone          516    France    Male   35
9997       9998    15584532        Liu          709    France  Female   36
9998       9999    15682355  Sabbatini          772   Germany    Male   42
9999      10000    15628319     Walker          792    France  Female   28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0          2       0.00              1          1               1
1          1   83807.86              1          0               1
2          8  159660.80              3          1               0
3          1       0.00              2          0               0
4          2  125510.82              1          1               1
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0
...
9998        92888.52       1
9999        38190.78       0

[10000 rows x 14 columns]
```

```
data.head()
```

[2] ✓ 0.1s Python

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

```
data.shape
```

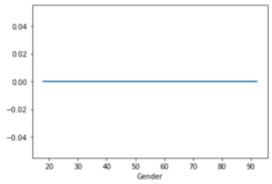
[3] ✓ 0.4s Python

```
(10000, 14)
```

UNIVARIATE ANALYSIS

```
data_France=data.loc[data['Geography']=='France']
plt.plot(data_France['Age'], np.zeros_like(data_France['Age']))
plt.xlabel('Gender')
plt.show()
```

[4] ✓ 0.6s Python

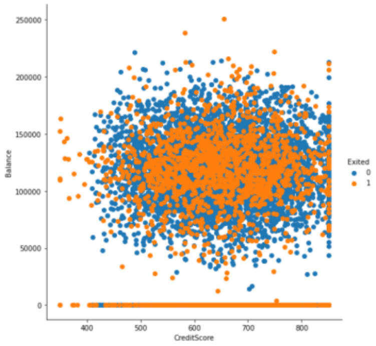


BI-VARIATE ANALYSIS

```
sns.FacetGrid(data,hue="Exited",height=7).map(plt.scatter,"CreditScore" , "Balance").add_legend();
plt.show()
```

[5] ✓ 1.1s Python

```
c:\Python310\lib\site-packages\seaborn\axisgrid.py:745: FutureWarning: iteritems is deprecated and will be removed in a future version. Use .items instead.
  plot_args = [v for k, v in plot_data.iteritems()]
c:\Python310\lib\site-packages\seaborn\axisgrid.py:745: FutureWarning: iteritems is deprecated and will be removed in a future version. Use .items instead.
  plot_args = [v for k, v in plot_data.iteritems()]
```

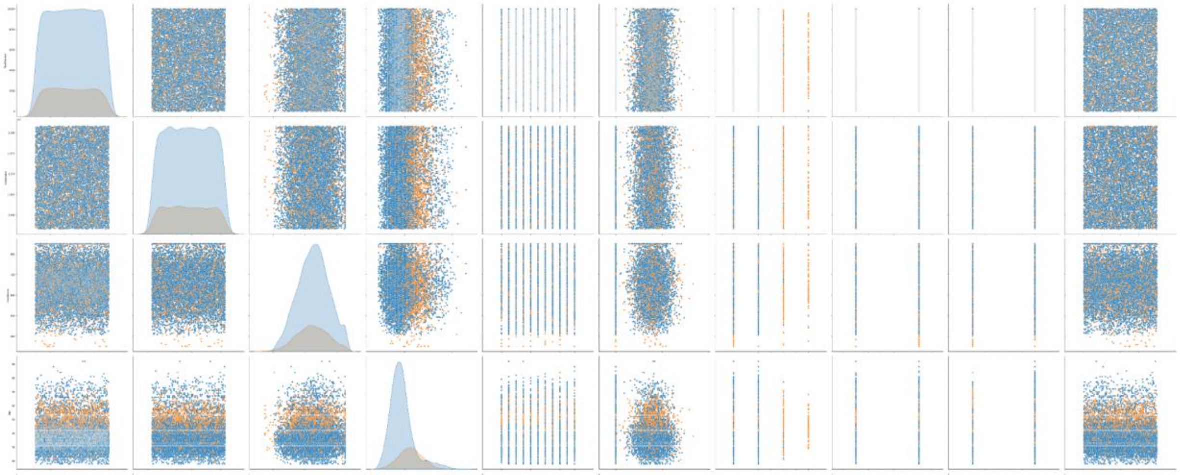


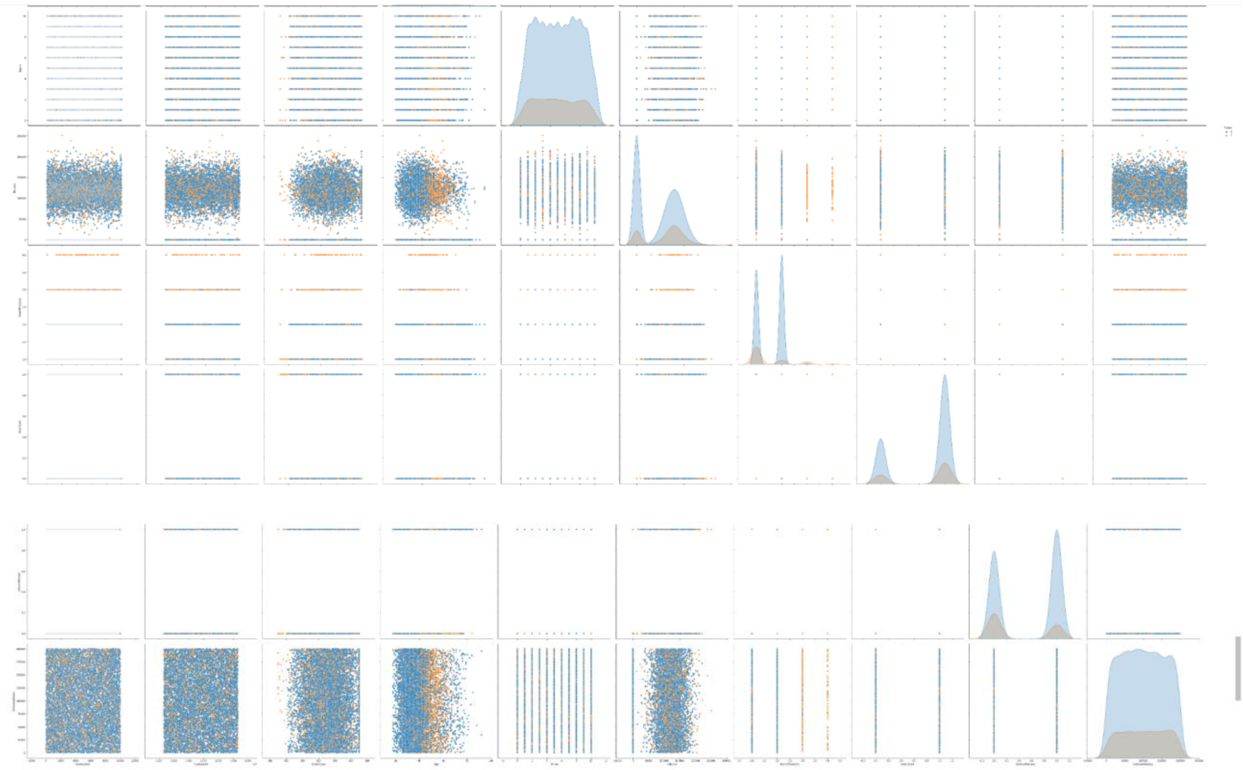
MULTIVARIATE ANALYSIS

```
sns.pairplot(data,hue="Exited",height=7)
```

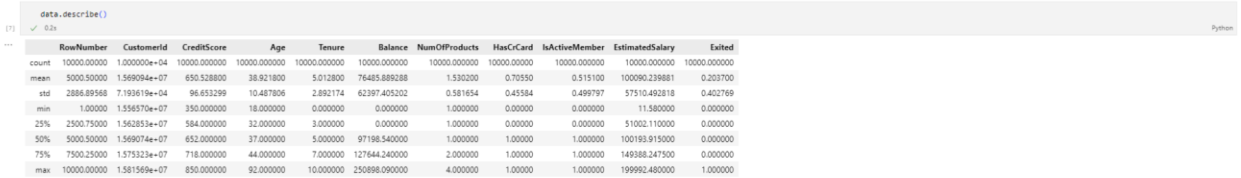
[6] ✓ 1m 20.6s Python

```
<seaborn.axisgrid.PairGrid at 0x1d9a01577f0>
```





DESCRIPTIVE STATISTIC ANALYSIS

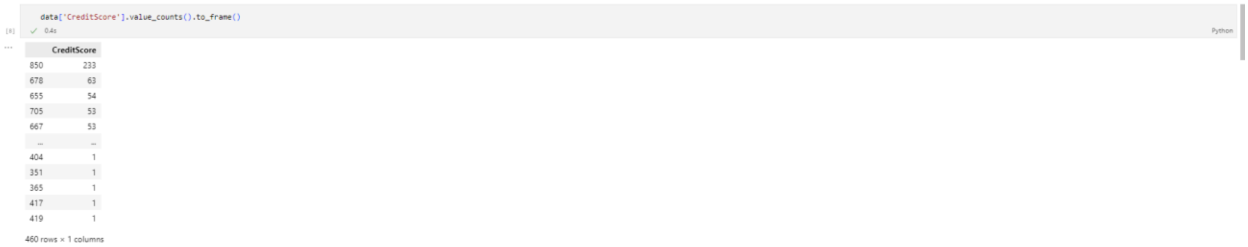


```
data.describe()
```

[7] ✓ 0.2s

Python

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	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	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000



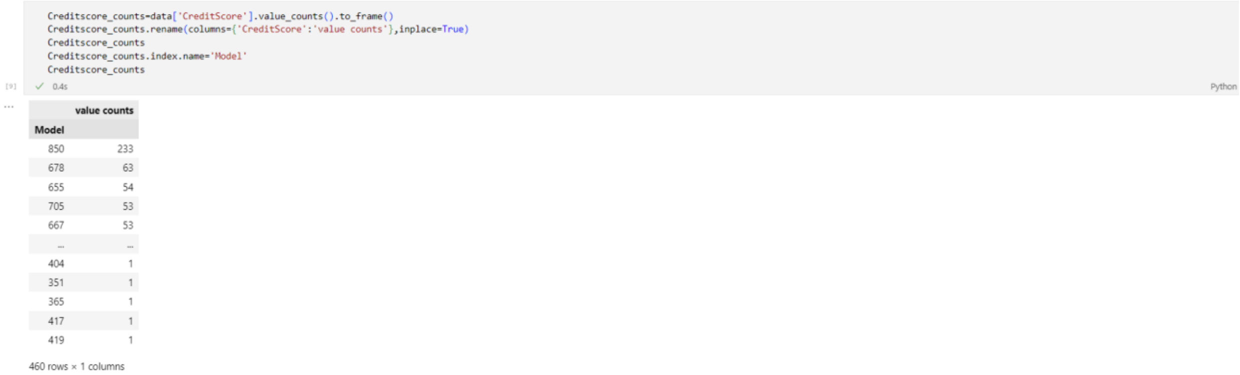
```
data['CreditScore'].value_counts().to_frame()
```

[8] ✓ 0.4s

Python

	CreditScore
850	233
678	63
655	54
705	53
667	53
...	...
404	1
351	1
365	1
417	1
419	1

460 rows × 1 columns



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

[9] ✓ 0.4s

Python

	value counts
Model	
850	233
678	63
655	54
705	53
667	53
...	...
404	1
351	1
365	1
417	1
419	1

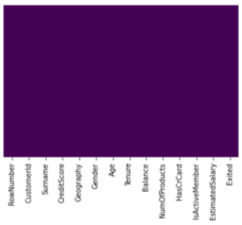
460 rows × 1 columns

HANDLE MISSING DATA

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

[10] ✓ 0.3s Python

```
<AxesSubplot: >
```



DETECTING AND REPLACING OUTLIERS

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

[11] ✓ 0.6s Python

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
12	13	15632264	Kay	476	France	Female	34	10	0.00	2	1	0	26260.98	0
29	30	15656300	Lucciano	411	France	Male	29	0	59697.17	2	1	1	53483.21	0
35	36	15794171	Lombardo	475	France	Female	45	0	134264.04	1	1	0	27822.99	1
40	41	15619360	Hsiao	472	Spain	Male	40	4	0.00	1	1	0	70154.22	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9879	9880	15669414	Pisano	486	Germany	Male	62	9	118356.89	2	1	0	168034.83	1
9907	9908	15611247	McKenzie	481	France	Female	28	10	0.00	2	1	0	145215.96	0
9930	9931	15713604	Rossi	425	Germany	Male	40	9	166776.60	2	0	1	172646.88	0
9964	9965	15642785	Douglas	479	France	Male	34	5	117593.48	2	0	0	113308.29	0
9966	9967	15590213	Ch'en	479	Spain	Male	35	4	125920.98	1	1	1	20393.44	0

490 rows × 14 columns

```
data[(data['CreditScore']<max_thresold) & (data['CreditScore']>min_thresold)]
```

[12] ✓ 0.4s Python

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

8987 rows × 14 columns

ASSIGNMENT – 3

Assignment Date	4 October 2022
Student Name	Ms. Dharshana R
Student Roll Number	713319EC024
Maximum Marks	2 Marks

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 locations (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

1. Download the 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. Split the data into dependent and independent variables.
9. Scale the independent variables
10. Split the data into training and testing
11. Build the Model
12. Train the Model
13. Test the Model
14. Measure the performance using Metrics.

SOLUTIONS:

IMPORT AND LOAD DATASET

```python
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')
```

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

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

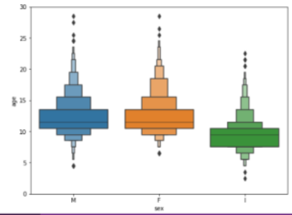
```python
#rename output variable
data.rename(columns={"Sex":"sex", "Length":"length", "Diameter":"diameter",
                     "Height":"height", "Whole weight":"whole_weight",
                     "Shucked weight":"shucked_weight", "Viscera weight":"viscera_weight",
                     "Shell weight":"shell_weight", "Rings":"rings"}, inplace = True)
```

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

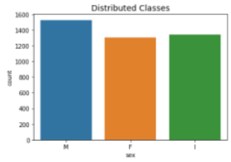
```
(4175, 9)
```

```python
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);
```

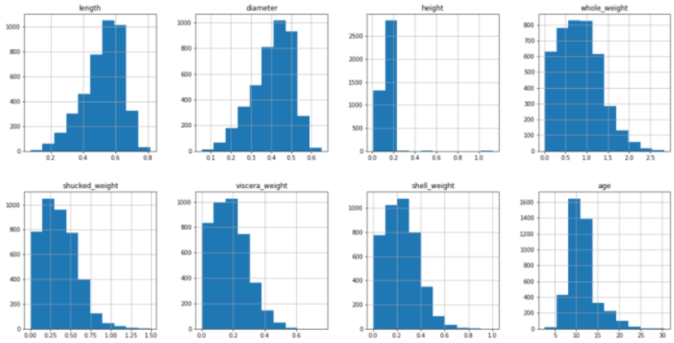


```python
sns.countplot('sex', data=data)
plt.title('Distributed Classes', fontsize=14)
plt.show()
```



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

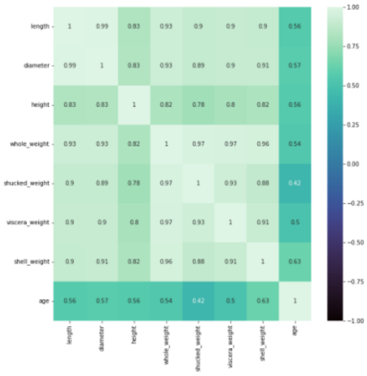
```
array([[<AxesSubplot:title={'center':'length'}>,
        <AxesSubplot:title={'center':'diameter'}>,
        <AxesSubplot:title={'center':'height'}>,
        <AxesSubplot:title={'center':'whole_weight'}>],
       [<AxesSubplot:title={'center':'shucked_weight'}>,
        <AxesSubplot:title={'center':'viscera_weight'}>,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



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

```
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')
```



```
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)
```

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

DESCRIPTIVE STATISTICS

```
data.head()
```

	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
```

```
(4175, 9)
```

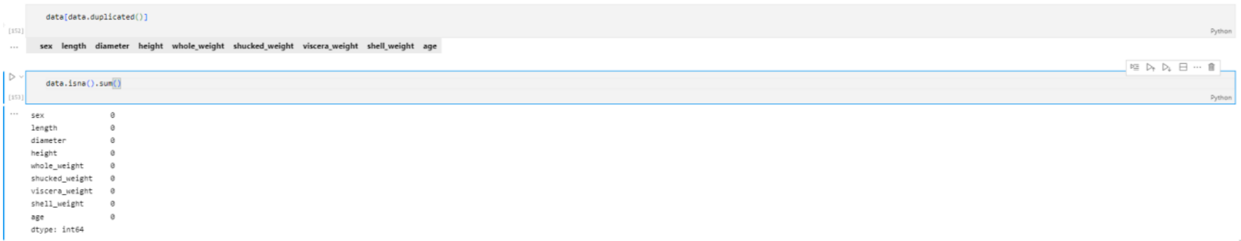
```
data.describe()
```

	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()
```

```
<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
```

MISSING VALUES



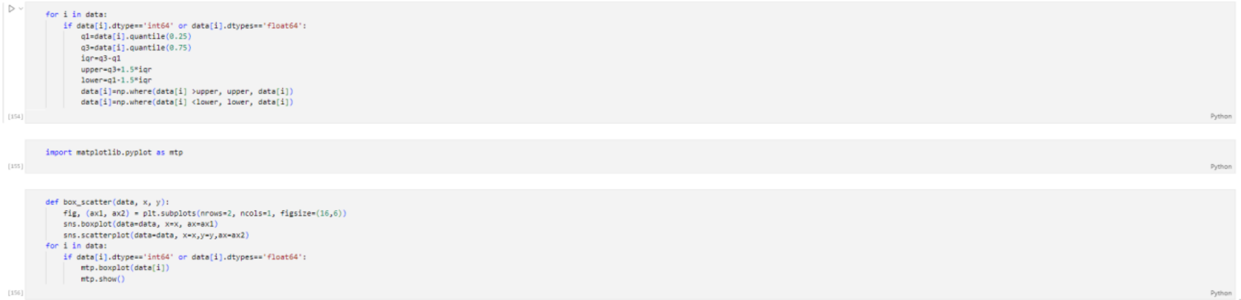
```
data[data.duplicated()]
```

sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age

```
data.isna().sum()
```

```
sex               0
length            0
diameter          0
height            0
whole_weight      0
shucked_weight    0
viscera_weight    0
shell_weight      0
age               0
dtype: int64
```

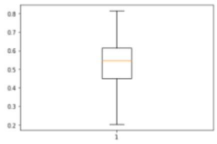
REPLACE THE OUTLIERS



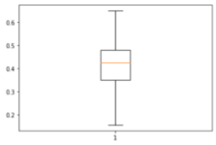
```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        data[i]=np.where(data[i] >upper, upper, data[i])
        data[i]=np.where(data[i]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

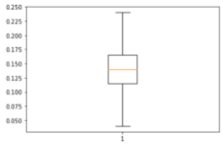
```
import matplotlib.pyplot as mtp
```

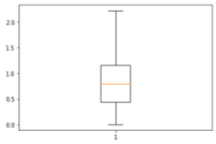
```
def box_scatter(data, x, y):
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))
    sns.boxplot(data=data, x=x, ax=ax1)
    sns.scatterplot(data=data, x=x,y=y,ax=ax2)
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        mtp.boxplot(data[i])
       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

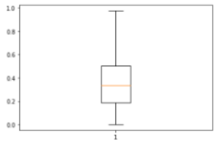


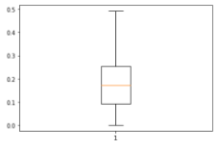
1

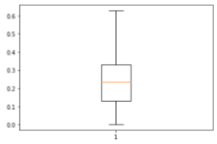


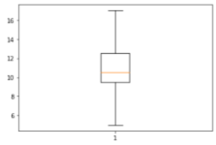












ENCODING

```
data.head()
```

	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()
```

	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()
```

	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

DEPENDENT AND INDEPENDENT VARIABLES

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

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

INDEPENDENT VARIABLE SCALING

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

SPLITING DATA

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

```
(2797, 8)
```

```
x_test.shape
```

```
(1378, 8)
```

BUILD THE MODEL

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

TRAIN THE MODEL

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

```
RandomForestRegressor()
```

TEST THE MODEL

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

PERFORMANCE MEASUREMENT USING METRICS

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

```
1.7786226498273756
```