ASSIGNMENT

APPLIED DATA SCIENCE

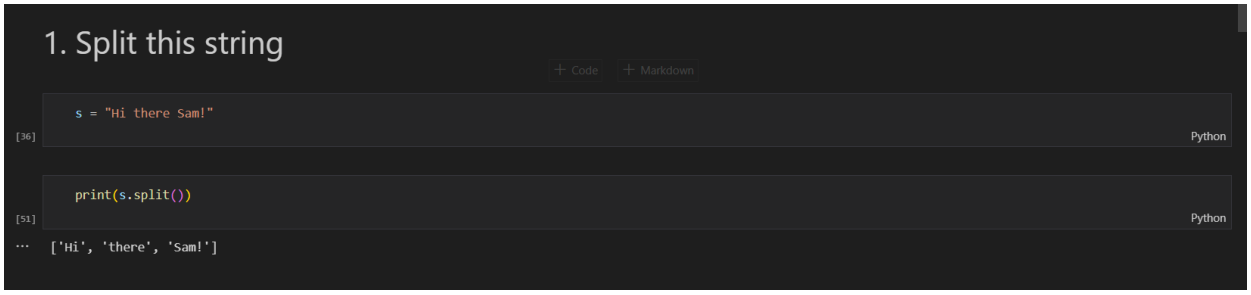
ASSIGNMENT 1

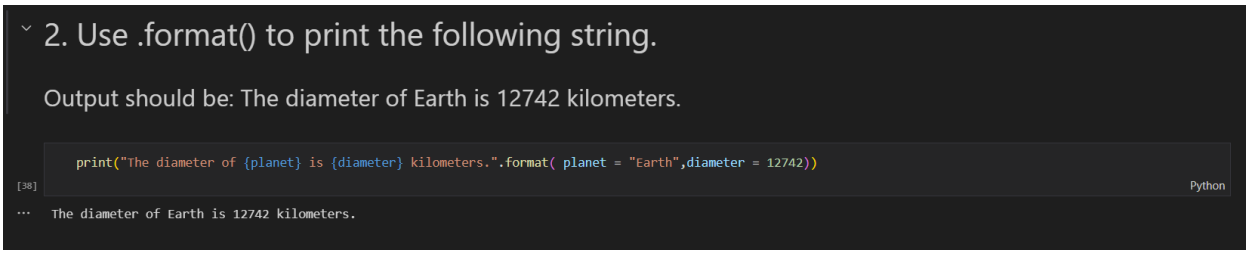
Assignment Date	19 September 2022
Student Name	Mr. Pranava Kailash S P
Student Roll Number	713319CS107
Maximum Marks	2 Marks

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:





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

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

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

```
hello
```

+ Code + Markdown

Numpy

```
[41] import numpy as np
```
Python

4.1 Create an array of 10 zeros?

4.2 Create an array of 10 fives?

```
[32] np.zeros(10)
```
Python

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

```
[33] np.ones(10)*5
```
Python

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

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

```
[34] print(np.arange(20,35,2))
```
Python

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

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

```
[35] print(np.arange(0,9).reshape(3,3))
```
Python

```
[[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])

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

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

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

Pandas

8. Create a dataframe with 3 rows and 2 columns

```
import pandas as pd
```

[45] Python

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

[46] Python

```
   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)))
```

[47] Python

```
[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]]
```

[49] Python

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

[50] Python

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

ASSIGNMENT – 2

Assignment Date	22 September 2022
Student Name	Mr. Pranava Kailash S P
Student Roll Number	713319CS107
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:

```
[1]
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```
Python

```
[2]
df = pd.read_csv('Churn_Modelling.csv')
```
Python

```
[3]
df.info()
```
Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
df.head()
```

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

```
df = df.drop(['RowNumber','CustomerId','Surname'],axis=1)
```

Python

```
df.head()
```

Python

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

+ Code + Markdown

```
df.describe()
```

Python

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

```
unique_count = []
for x in df.columns:
    unique_count.append([x,len(df[x].unique()),df[x].isnull().sum()])

pd.DataFrame(unique_count, columns=["Column","Unique","Missing"]).set_index("Column")
```

Python

Column	Unique	Missing
CreditScore	460	0
Geography	3	0
Gender	2	0
Age	70	0
Tenure	11	0
Balance	6382	0
NumOfProducts	4	0
HasCrCard	2	0
IsActiveMember	2	0
EstimatedSalary	9999	0
Exited	2	0

```
df.shape
```

Python

```
(10000, 11)
```

```
df.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   CreditScore      10000 non-null  int64  
 1   Geography        10000 non-null  object 
 2   Gender           10000 non-null  object 
 3   Age              10000 non-null  int64  
 4   Tenure           10000 non-null  int64  
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64  
 7   HasCrCard        10000 non-null  int64  
 8   IsActiveMember   10000 non-null  int64  
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  int64  
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

```
df_numerical = df.select_dtypes('int64')
df_cat = df.select_dtypes('object')
df_numerical.append(df[['Balance','EstimatedSalary']])

print(f"Numerical: {df_numerical.columns} \nCatergorical: {df_cat.columns}")
```

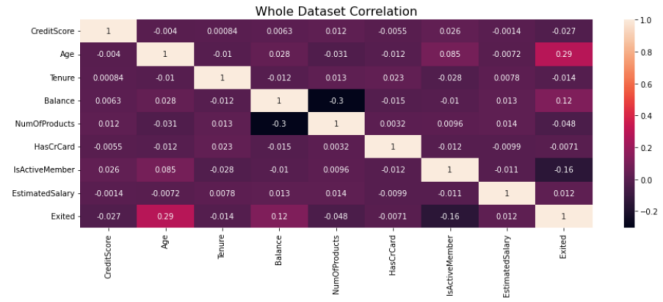
[11] Python

```
Numerical: Index(['CreditScore', 'Age', 'Tenure', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'Exited'],
      dtype='object')
Catergorical: Index(['Geography', 'Gender'], dtype='object')
```

```
plt.figure(figsize=(15,5))
sns.heatmap(df.corr(), annot=True,)
plt.title('Whole Dataset Correlation',fontsize = 16)
```

[12] Python

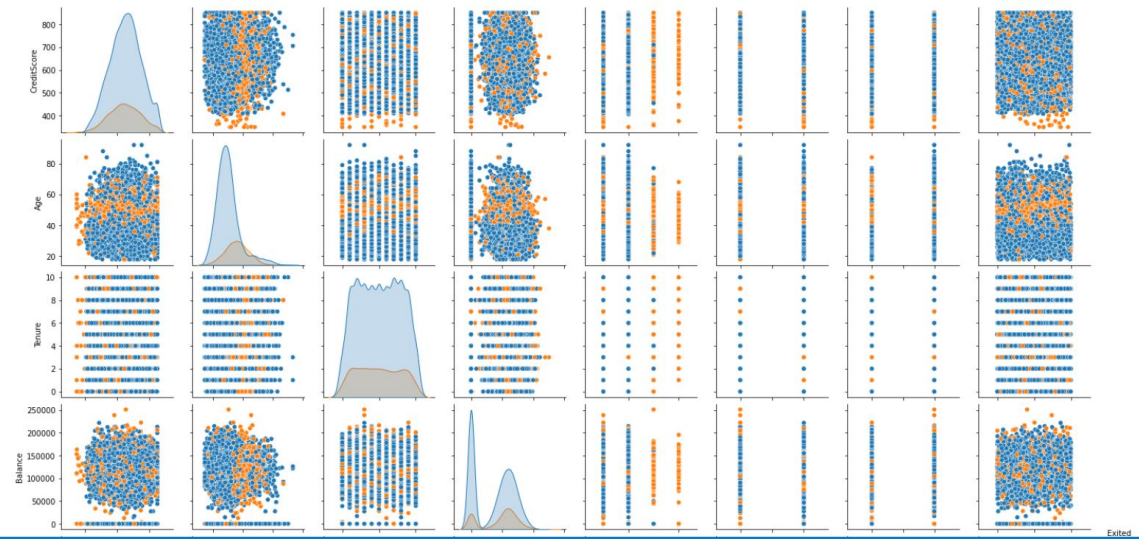
```
Text(0.5, 1.0, 'Whole Dataset Correlation')
```

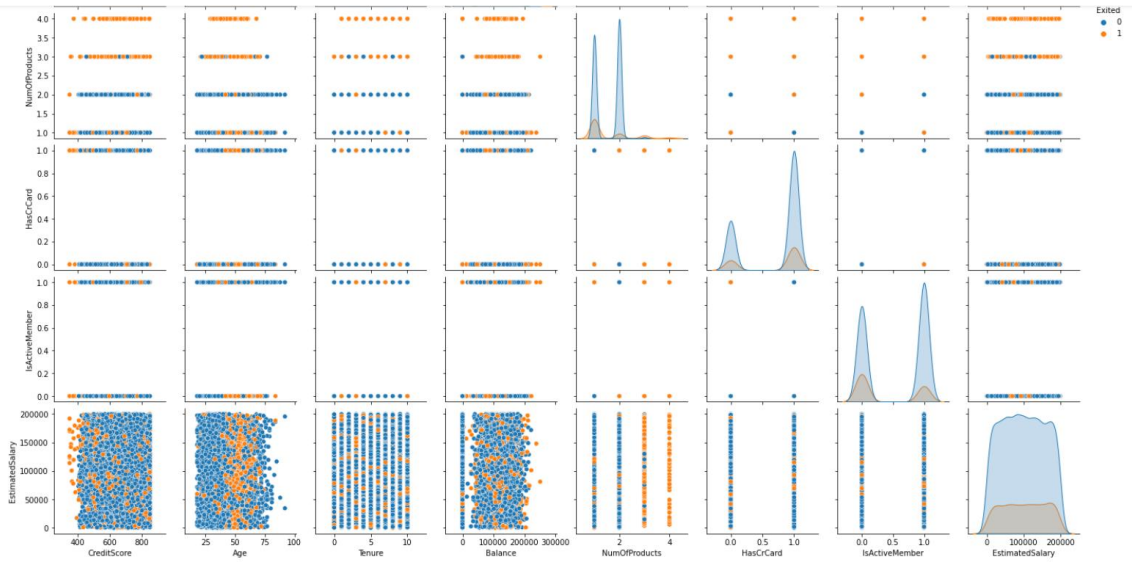



```
sns.pairplot(df, hue='Exited')
```

[13] Python

```
<seaborn.axisgrid.PairGrid at 0x15fff587d30>
```





Above Graph is Comapared with the Target variable, Exited (Iv'e taken it as the Target variable)

+ Code + Markdown

For numerical data - Histogram and Scatterplot

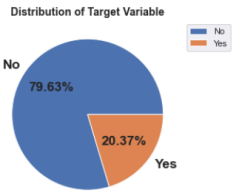
For categorical data - Pie chart and Bar chart

```
sns.set_theme(style="darkgrid")
```

[14] Python

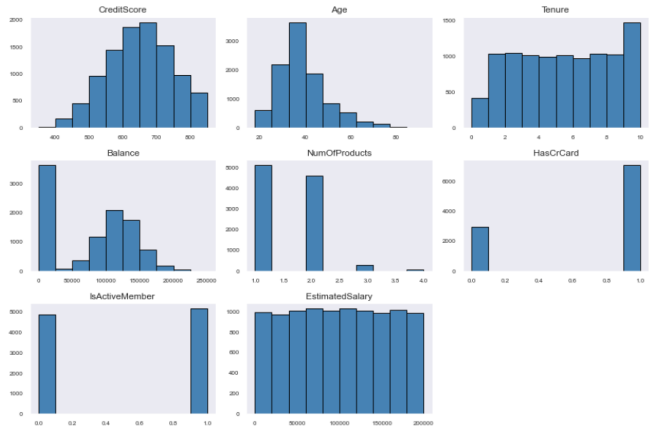
```
df.loc[df['Exited'] == 0, 'Exited'] = "No"
df.loc[df['Exited'] == 1, 'Exited'] = "Yes"
plt.figure(figsize = (6,4))
x = df['Exited'].value_counts()
labels = "No",'Yes'
plt.pie(x = x,  labels = labels,
        autopct = '%.2f%%',
        textprops = {'size' : 'x-large',
                     'fontweight' : 'bold'})
plt.title('Distribution of Target Variable', fontsize = 14, fontweight = 'bold')
plt.legend(labels, loc="upper left", bbox_to_anchor = (1,1))
plt.tight_layout()
plt.show()
```

[15] Python




```
df.hist(bins=10, color='steelblue', edgecolor='black', linewidth=1.0,
        xlabelsize=8, ylabelsize=8, grid=False)
plt.tight_layout(rect=(0, 0, 2, 2))
```

[16] Python



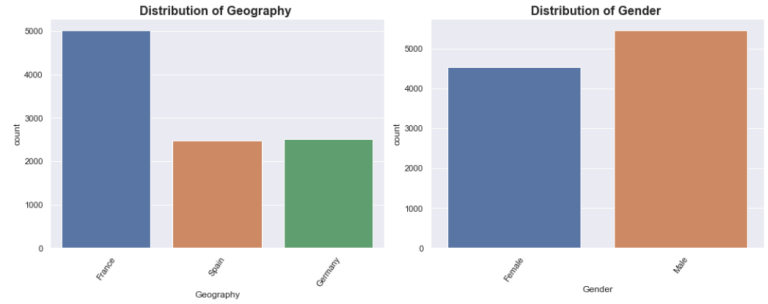
the above chart shows the following

the above chart shows the following

1. Credit Score is a little bit skewed right (Min Score to Passed Eligibilty is about 600 and starting would be 750)
2. Age is also skewed towards left
3. Most people prefer 10 month Tenure
4. Most have 0 balance (How is it possible)
5. No enough data for product 3 and 4
6. Estimated Salary is even in terms of the plot

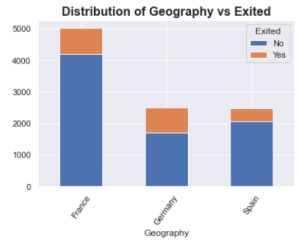
```
plt.figure(figsize=[14,10])
n=1
for x in df_cat:
    plt.subplot(2,2,n)
    sns.countplot(x=df[x],data=df)
    sns.despine()
    plt.title("Distribution of {} ".format(x), fontsize=16, fontweight='bold')
    plt.xticks(rotation=55)
    n=n+1
plt.tight_layout()
plt.show()
```

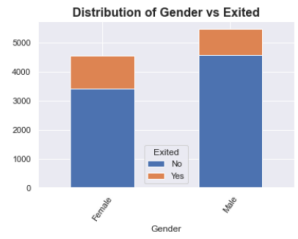
[17] Python




```
for x in df_cat:
    y = pd.crosstab(df[x],df['Exited'])
    y.div(y.sum(1).astype(float), axis=0)
    y.plot(kind='bar', stacked=True)
    plt.title("Distribution of {} vs Exited".format(x), fontsize=16, fontweight='bold')
    plt.xticks(rotation=55)
```

[18] Python





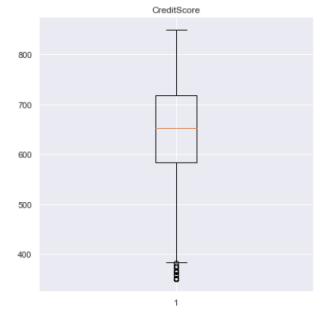
```
df.head()
```
[19] Python

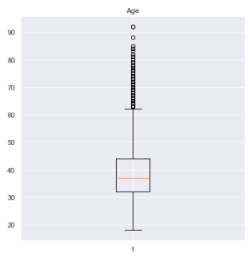
	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	Yes
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	No
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	Yes
3	699	France	Female	39	1	0.00	2	0	0	93826.63	No
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	No

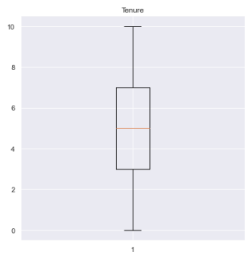
```
df_numerical.head()
```
[20] Python

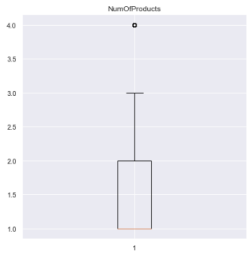
	CreditScore	Age	Tenure	NumOfProducts	HasCrCard	IsActiveMember	Exited
0	619	42	2	1	1	1	1
1	608	41	1	1	0	1	0
2	502	42	8	3	1	0	1
3	699	39	1	2	0	0	0
4	850	43	2	1	1	1	0

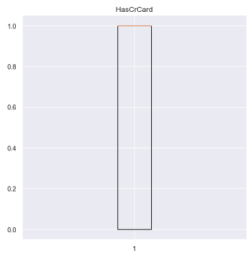
```
for x in df_numerical:
    fig = plt.figure(figsize =(5,5))
    ax = fig.add_axes([0, 0, 1, 1])
    bp = ax.boxplot(df_numerical[f'{x}'])
    plt.title(f'{x}')
    plt.show()
```
[21] Python

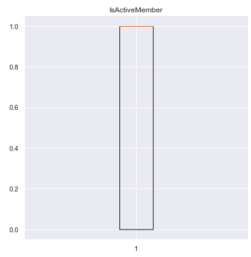


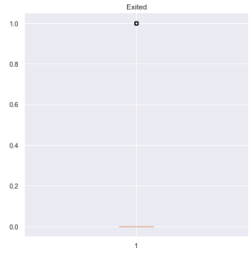












From the above we can find that , we have some outliers in Credit and in Age, which can be removed

```
def IQR_capping(df,cols,factor):

    for col in cols:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)

        iqr = q3 - q1

        up_limit = q3 + (factor*iqr)
        lower_limit = q1 - (factor*iqr)

        df[col] = np.where(df[col]>up_limit,up_limit,np.where(df[col]<lower_limit,lower_limit,df[col]))
    print("Removed Outliers")
```

[22] Python

```
outlier_data = df[['CreditScore','Age']]
```

[23] Python

```
IQR_capping(df_numerical,outlier_data,1.5)
```

[24] Python

```
Removed Outliers

C:\Users\prana\AppData\Local\Temp/ipykernel_10504/3905454356.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = np.where(df[col]>up_limit,up_limit,np.where(df[col]<lower_limit,lower_limit,df[col]))
```

Now to handle the cat values

```
df.head()
```

[25] Python

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	Yes
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	No
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	Yes
3	699	France	Female	39	1	0.00	2	0	0	93826.63	No
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	No

```
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()

df['Geography'] = Le.fit_transform(df['Geography'])
df['Gender'] = Le.fit_transform(df['Gender'])
```

[26] Python

```
df.head()
```

[27] Python

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	0	0	42	2	0.00	1	1	1	101348.88	Yes
1	608	2	0	41	1	83807.86	1	0	1	112542.58	No
2	502	0	0	42	8	159660.80	3	1	0	113931.57	Yes
3	699	0	0	39	1	0.00	2	0	0	93826.63	No
4	850	2	0	43	2	125510.82	1	1	1	79084.10	No

+ Code + Markdown

Splitting the Dependent and independent variables

```
features = df.iloc[:,:10]
```

[28] Python

```
labels = df['Exited']
```

[29] Python

Transforming the Label value to 1's and 0's

```
labels = Le.fit_transform(labels)
```

[30] Python

```
labels
```

[31] Python

```
array([1, 0, 1, ..., 1, 1, 0])
```

Scaling the data , using Standard Scalar

```
from sklearn.preprocessing import StandardScaler
sscaler = StandardScaler()
features = sscaler.fit_transform(features)
```

[33] Python

Train Test Split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(features,labels,test_size=0.2,shuffle=True)
```

[34] Python

```
print(x_test)
```

[36] Python

```
[[ 0.33597222 -0.90188624 -1.09598752 ... -1.54776799  0.97024255
   0.00707037]
 [-0.10893913  0.30659057 -1.09598752 ...  0.64609167 -1.03067011
   1.45921748]
 [ 1.37064979  0.30659057 -1.09598752 ...  0.64609167  0.97024255
  -0.06085889]
 ...
 [-0.29518109 -0.90188624  0.91241915 ...  0.64609167 -1.03067011
  -1.12474613]
 [ 0.09799638  0.30659057 -1.09598752 ...  0.64609167 -1.03067011
   1.27970456]
 [ 0.07730283  1.51506738 -1.09598752 ...  0.64609167  0.97024255
  -1.19298074]]
```

```
print(y_test)
```

[37] Python

```
[1 1 1 ... 0 1 0]
```

Ready for Model Training and Testing

+ Code + Markdown

ASSIGNMENT – 3

Assignment Date	4 October 2022
Student Name	Mr. Pranava Kailash S P
Student Roll Number	713319CS107
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

```
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')
```

[138] Python

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

[139] Python

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)
```

[140] Python

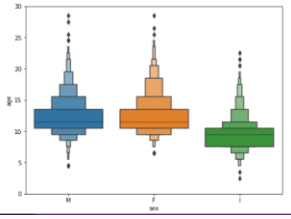
```
data[data['height'] == 0]  #need to drop these rows.
data.drop(index=[1257,3996], inplace = True)
data.shape
```

[141] Python

```
(4175, 9)
```

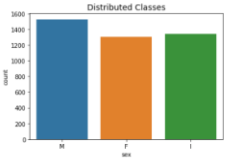
```
data['age'] = data['rings']+1.5 #AS per the problem statement
data.drop('rings', axis = 1, inplace = True)
data.head()
#categorical features
temp = pd.concat([df['age'], df['sex']], axis=1)

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

[142] Python




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

[143] Python

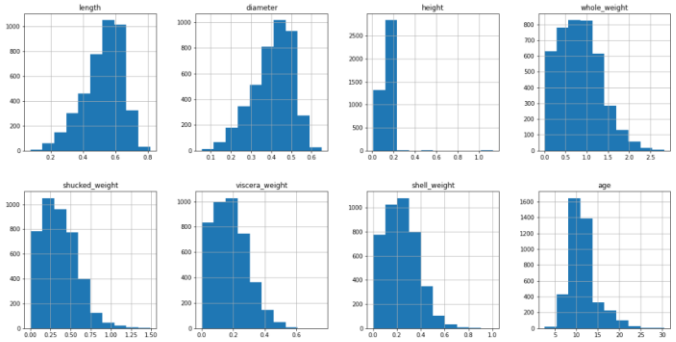



```
data.hist(figsize = (20,10), layout = (2,4))
```

[144] Python

```
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)
```

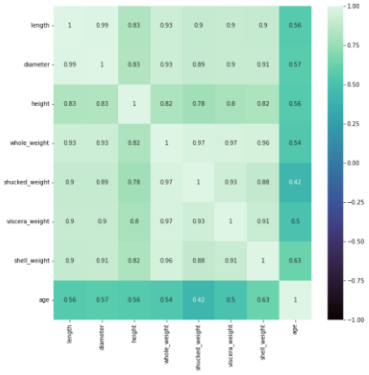
</>



```
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	4175.000000	4175.000000	4175.000000	4175.000000	4175.000000	4175.000000	4175.000000
mean	0.524065	0.407994	0.139583	0.829005	0.359476	0.180653	0.238834	11.435090
std	0.120069	0.099220	0.041725	0.490349	0.221954	0.109605	0.139212	3.224227
min	0.075000	0.055000	0.010000	0.002000	0.001000	0.000500	0.001500	2.500000
25%	0.450000	0.350000	0.115000	0.442250	0.186250	0.093500	0.130000	9.500000
50%	0.545000	0.425000	0.140000	0.800000	0.336000	0.171000	0.234000	10.500000
75%	0.615000	0.480000	0.165000	1.153500	0.502000	0.253000	0.328750	12.500000
max	0.815000	0.650000	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()
```

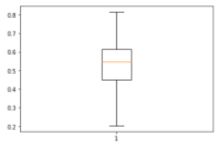
```
sex               0
length            0
diameter          0
height            0
whole_weight      0
shucked_weight    0
viscera_weight    0
shell_weight      0
age               0
dtype: int64
```

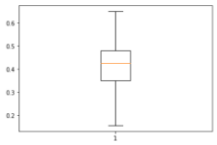
REPLACE THE OUTLIERS

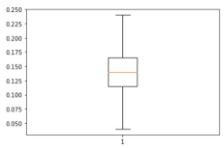
```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        data[i]=np.where(data[i] >upper, upper, data[i])
        data[i]=np.where(data[i]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

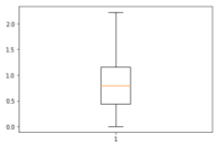
```
import matplotlib.pyplot as mtp
```

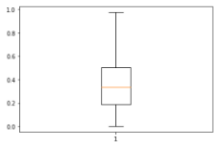
```
def box_scatter(data, x, y):
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))
    sns.boxplot(data=data, x=x, ax=ax1)
    sns.scatterplot(data=data, x=x,y=y,ax=ax2)
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        mtp.boxplot(data[i])
       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

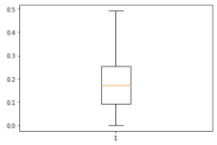


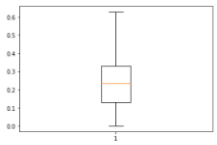


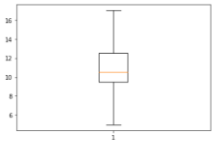












ENCODING

```
data.head()
```

Python

	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()
```

Python

	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()
```

Python

	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()
```

Python

```
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)
```

Python

SPLITING DATA

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

Python

```
(2797, 8)
```

```
x_test.shape
```

Python

```
(1378, 8)
```

BUILD THE MODEL

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

Python

TRAIN THE MODEL

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

Python

```
RandomForestRegressor()
```

TEST THE MODEL

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

Python

PERFORMANCE MEASUREMENT USING METRICS

+ Code + Markdown

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

Python

```
1.7786226498273756
```