PROJECT TITLE: Airlines Data Analytics for Avaition Industry

Team ID : PNT2022TMID29101

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

2.LOAD THE DATASET INTO COLLAB

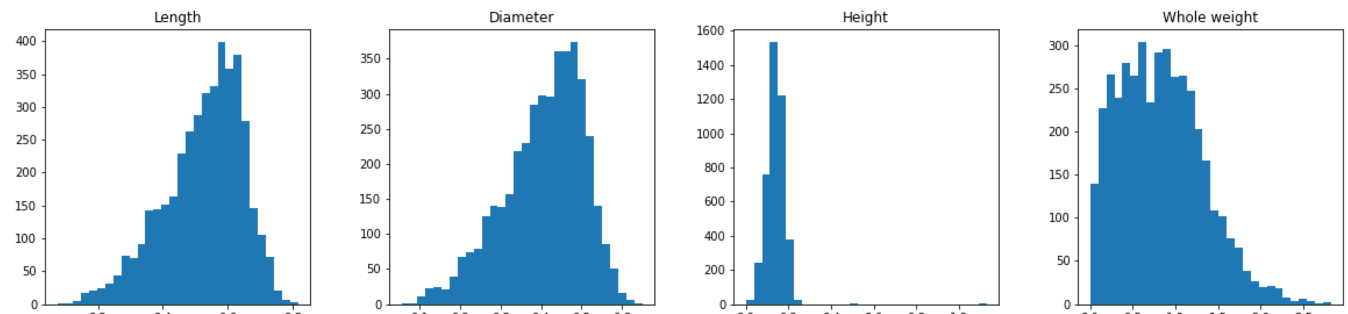
```
df=pd.read_csv("/content/abalone.csv")
```

```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

3.UNIVARIATE ANALYSIS

```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6654e8ffd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654e80490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654e37a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654dfa0d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f6654db06d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654d67cd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654d2b390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654ce28d0>]],
      dtype=object)
```



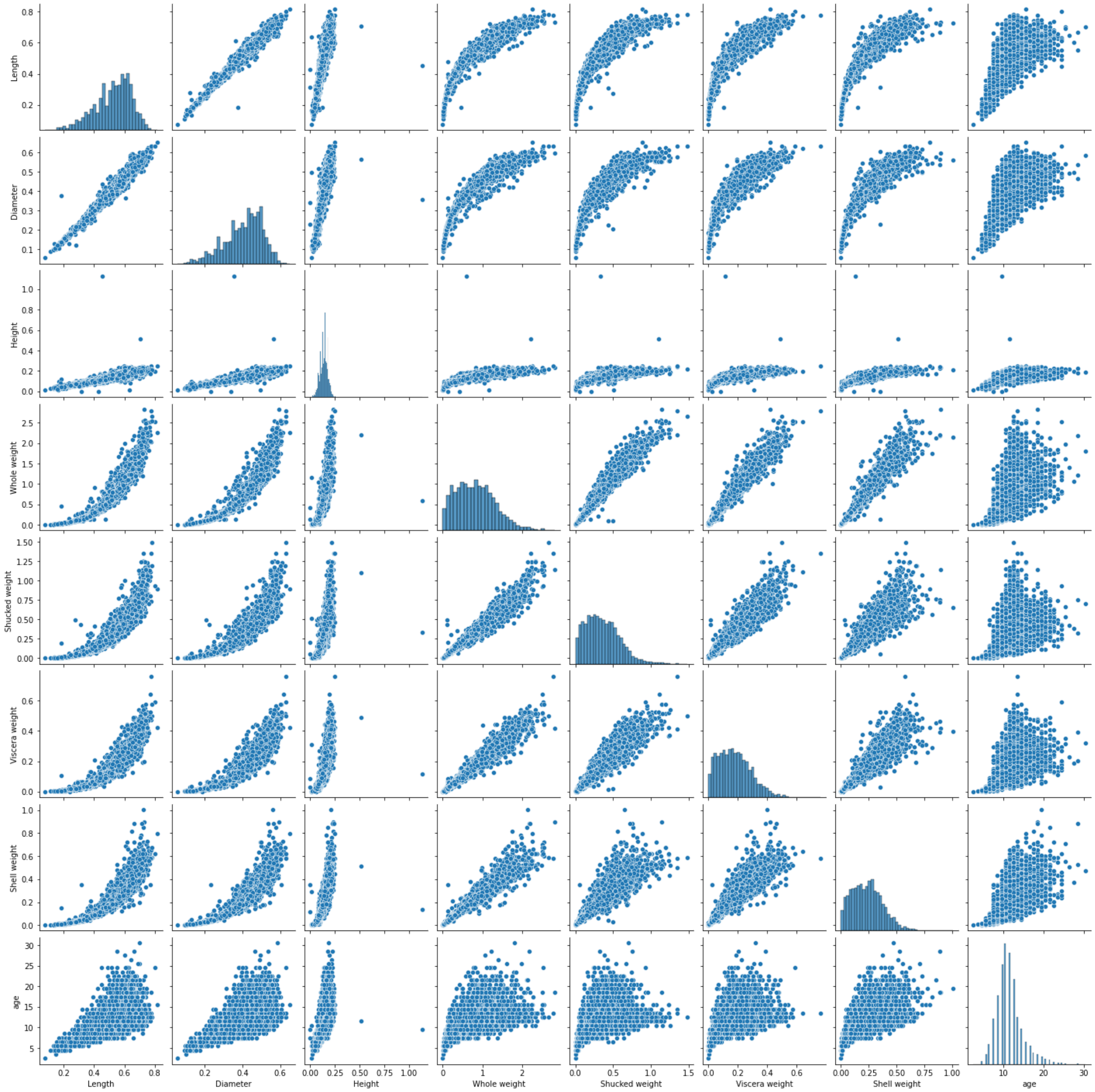

```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
I	0.427746	0.326494	0.107996	0.431363	0.191035	0.092010	0.128182	9.390462
M	0.561391	0.439287	0.151381	0.991459	0.432946	0.215545	0.281969	12.205497

3.BIVARIATE ANALYSIS & MULTIVARIATE ANALYSIS

```
numerical_features = df.select_dtypes(include = [np.number]).columns
sns.pairplot(df[numerical_features])
```

<seaborn.axisgrid.PairGrid at 0x7f6654826190>



Descriptive statistics

```
df.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000

5.Check for Missing Values

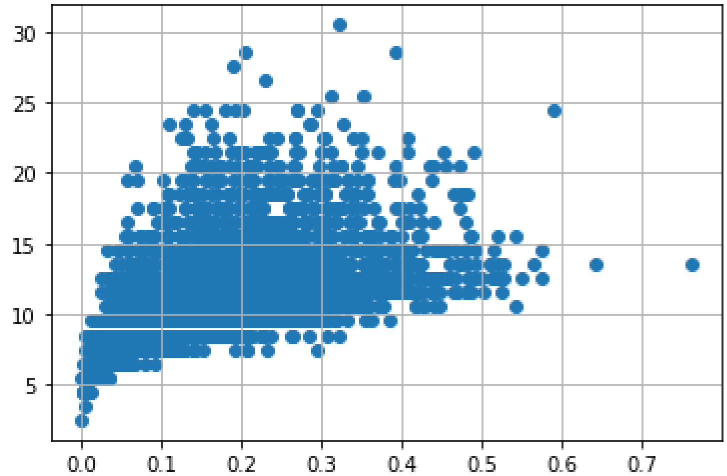
```
df.isnull().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

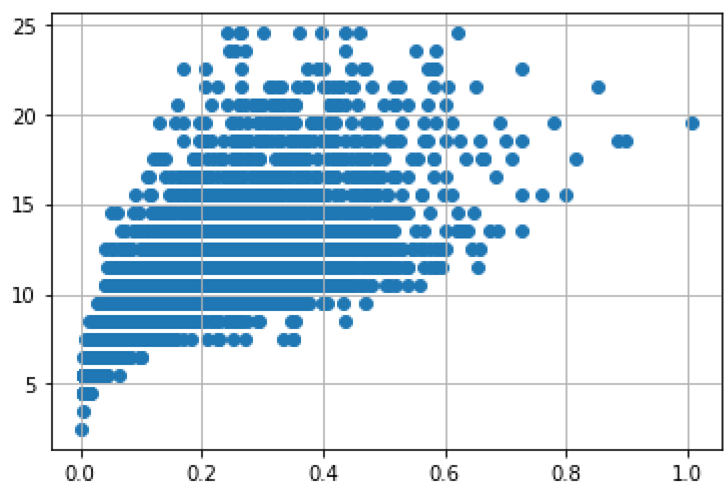
6.OUTLIER HANDLING

```
df = pd.get_dummies(df)
```

```
dummy_data = df.copy()

var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```

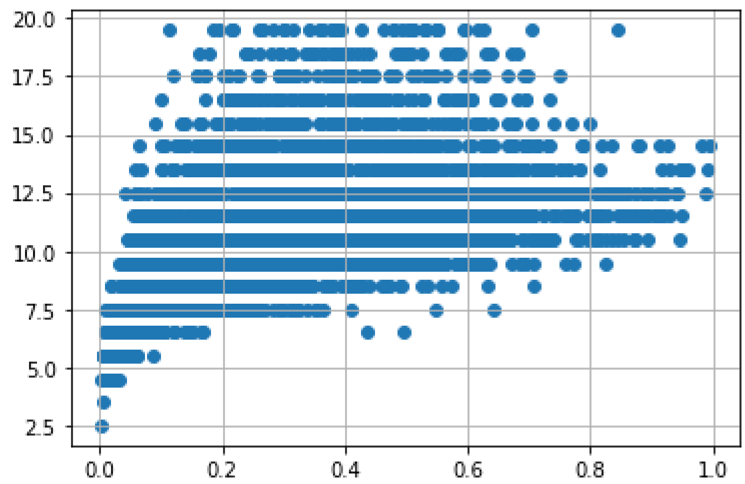



```
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)


var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
#Outliers removal
df.drop(df[(df['Shell weight']> 0.6) & (df['age'] < 25)].index, inplace=True)
df.drop(df[(df['Shell weight']<0.8) & (df['age'] > 25)].index, inplace=True)
```

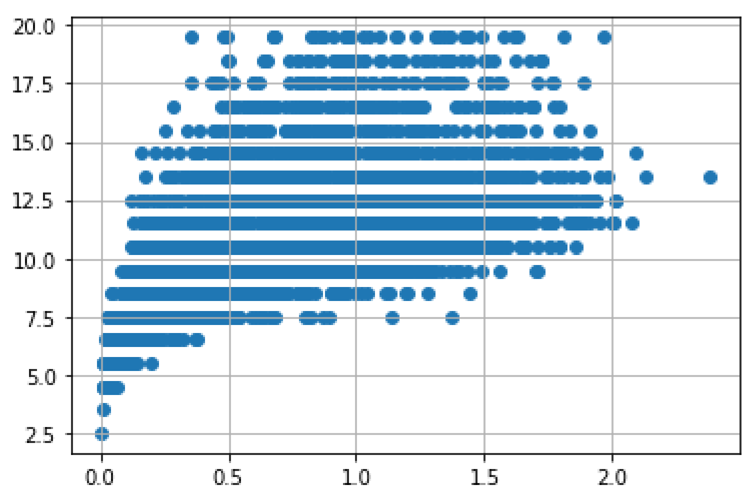



```
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)

#Outlier removal
```

```
df.drop(df[(df['Shucked weight']>= 1) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Shucked weight']<1) & (df['age'] > 20)].index, inplace=True)
```

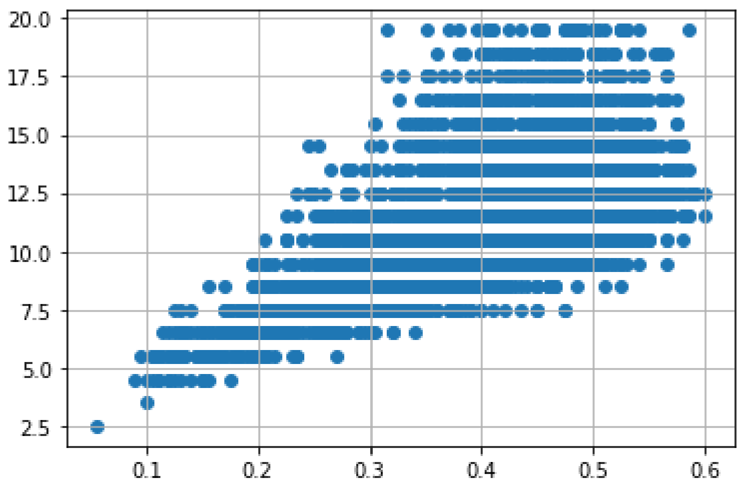


```
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Whole weight'] >= 2.5) &
          (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight']<2.5) & (
df['age'] > 25)].index, inplace = True)
```

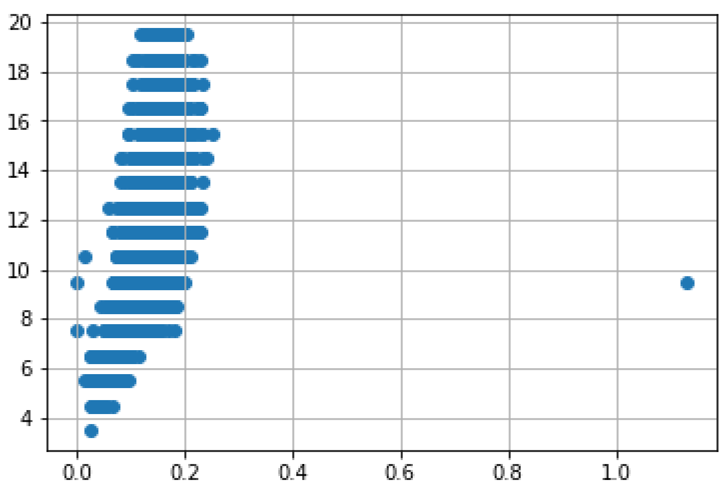


```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Diameter'] <0.1) &
          (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Diameter']<0.6) & (
df['age'] > 25)].index, inplace = True)
```

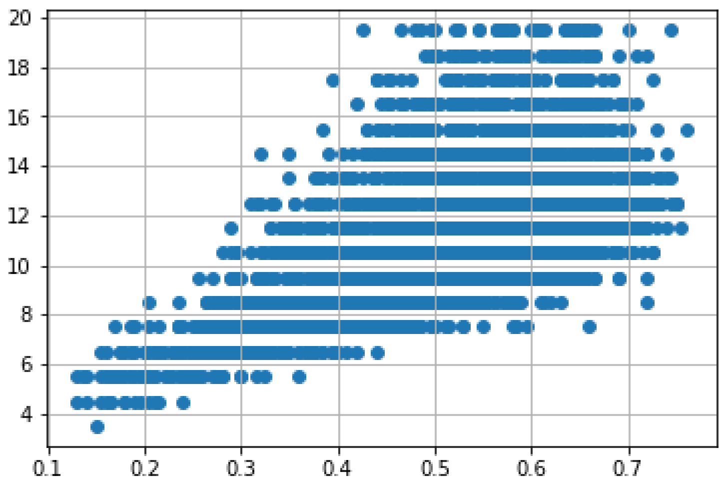
```
df.drop(df[(df['Diameter']>=0.6) & (
df['age'] < 25)].index, inplace = True)
```




```
var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
df.drop(df[(df['Height'] > 0.4) &
          (df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (
df['age'] > 25)].index, inplace = True)
```




```
var = 'Length'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

df.drop(df[(df['Length'] <0.1) &
          (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Length']<0.8) & (
df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Length']>=0.8) & (
df['age'] < 25)].index, inplace = True)
```



7.Categorical columns

```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `np
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
```

```
numerical_features
```

```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
      dtype='object')
```

```
categorical_features
```

```
Index([], dtype='object')
```

ENCODING

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
print(df.Length.value_counts())
```

```
0.575    93
0.625    91
0.580    89
0.550    89
0.620    83
         ..
0.220     2
0.150     1
```

```
0.755     1
0.135     1
0.760     1
Name: Length, Length: 126, dtype: int64
```

8.Split the dependent and independent variables

```
x=df.iloc[:,:5]
x
```

	Length	Diameter	Height	Whole weight	Shucked weight
0	0.455	0.365	0.095	0.5140	0.2245
1	0.350	0.265	0.090	0.2255	0.0995
2	0.530	0.420	0.135	0.6770	0.2565
3	0.440	0.365	0.125	0.5160	0.2155
4	0.330	0.255	0.080	0.2050	0.0895
...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700
4173	0.590	0.440	0.135	0.9660	0.4390
4174	0.600	0.475	0.205	1.1760	0.5255
4175	0.625	0.485	0.150	1.0945	0.5310
4176	0.710	0.555	0.195	1.9485	0.9455

3995 rows × 5 columns

```
y=df.iloc[:,5:]
y
```

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
0	0.1010	0.1500	16.5	0	0	1
1	0.0485	0.0700	8.5	0	0	1
2	0.1415	0.2100	10.5	1	0	0
3	0.1140	0.1550	11.5	0	0	1

9.Feature Scaling

```
#Scaling the Independent Variables
print ("\n ORIGINAL VALUES: \n\n", x,y)
```

```
 ORIGINAL VALUES: 

       Length  Diameter  Height  Whole weight  Shucked weight
0      0.455     0.365   0.095        0.5140          0.2245
1      0.350     0.265   0.090        0.2255          0.0995
2      0.530     0.420   0.135        0.6770          0.2565
3      0.440     0.365   0.125        0.5160          0.2155
4      0.330     0.255   0.080        0.2050          0.0895
...      ...       ...     ...           ...             ...
4172   0.565     0.450   0.165        0.8870          0.3700
4173   0.590     0.440   0.135        0.9660          0.4390
4174   0.600     0.475   0.205        1.1760          0.5255
4175   0.625     0.485   0.150        1.0945          0.5310
4176   0.710     0.555   0.195        1.9485          0.9455

[3995 rows x 5 columns]       Viscera weight  Shell weight   age  Sex_F  Sex_I  Sex_M
0             0.1010        0.1500  16.5      0      0      1
1             0.0485        0.0700   8.5      0      0      1
2             0.1415        0.2100  10.5      1      0      0
3             0.1140        0.1550  11.5      0      0      1
4             0.0395        0.0550   8.5      0      1      0
...              ...           ...   ...    ...    ...    ...
4172          0.2390        0.2490  12.5      1      0      0
4173          0.2145        0.2605  11.5      0      0      1
4174          0.2875        0.3080  10.5      0      0      1
4175          0.2610        0.2960  11.5      1      0      0
4176          0.3765        0.4950  13.5      0      0      1

[3995 rows x 6 columns]
```

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
new_y= min_max_scaler.fit_transform(x,y)
print ("\n VALUES AFTER MIN MAX SCALING: \n\n", new_y)
```

```
 VALUES AFTER MIN MAX SCALING:
```

```
[[0.51587302 0.54545455 0.38       0.21240245 0.22199798]
 [0.34920635 0.34343434 0.36      0.09069816 0.09586276]
 [0.63492063 0.65656566 0.54      0.28116431 0.2542886 ]
 ...
 [0.74603175 0.76767677 0.82      0.49166842 0.52573158]
 [0.78571429 0.78787879 0.6       0.45728749 0.53128153]
 [0.92063492 0.92929293 0.78      0.81754904 0.94954591]]
```

10.Split the data into training and testing

```
#Split the data into Training and Testing
X = df.drop('age', axis = 1)
y = df['age']
```