Importing libraries

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from google.colab import drive
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
```

Load the dataset

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.r
```

```
path='/content/drive/MyDrive/Colab Notebooks/miniproject/abalone.csv'
```

```
df=pd.read_csv(path)
```

```
df.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
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

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

50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000

Univariate Analysis

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

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6654da83d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654d40790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6654cecbd0>,
```

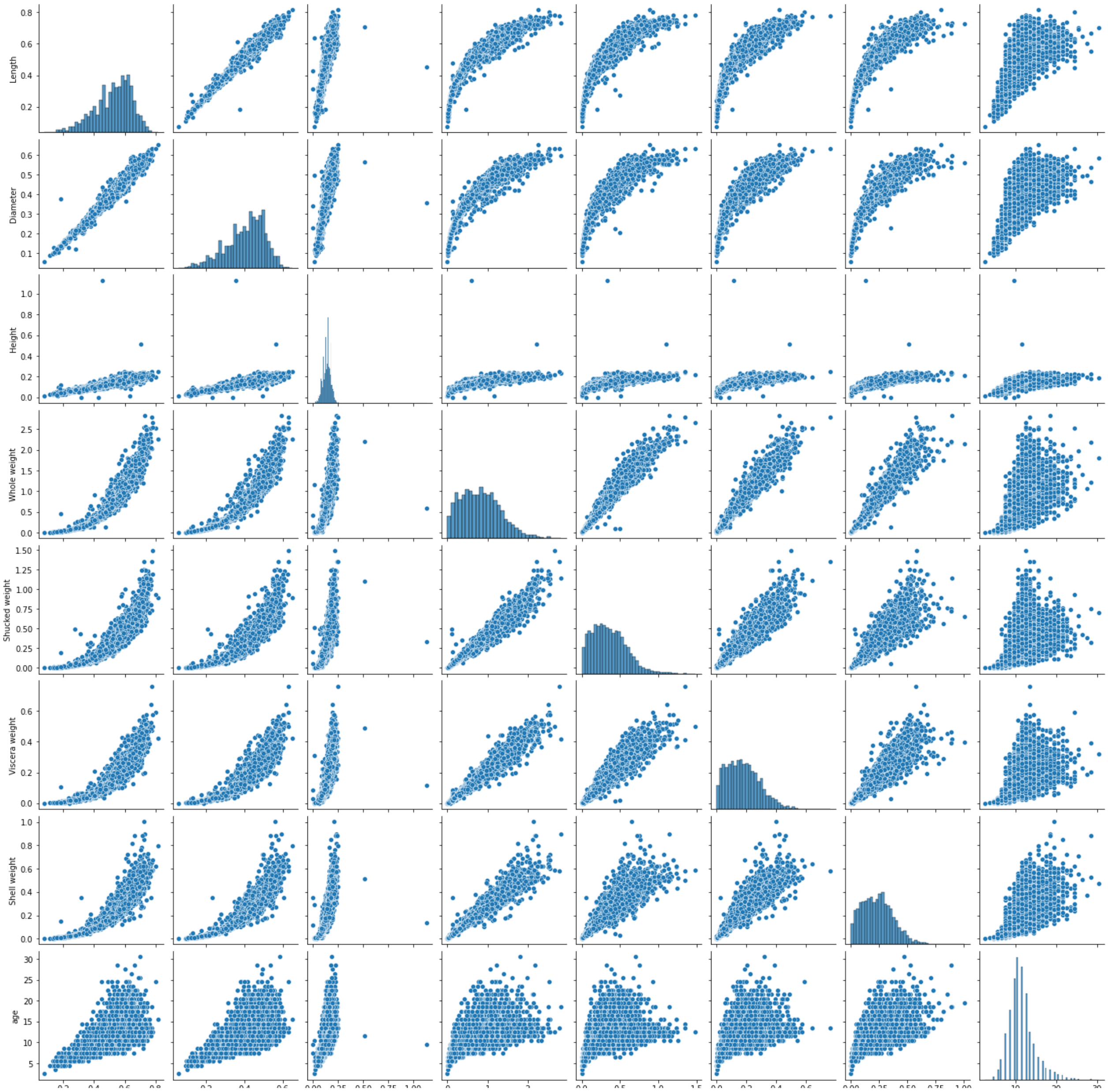
```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
Sex								
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

Bivariate and Multivariate Analysis

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

```
<seaborn.axisgrid.PairGrid at 0x7f66548d8910>
```



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

Check for missing values

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

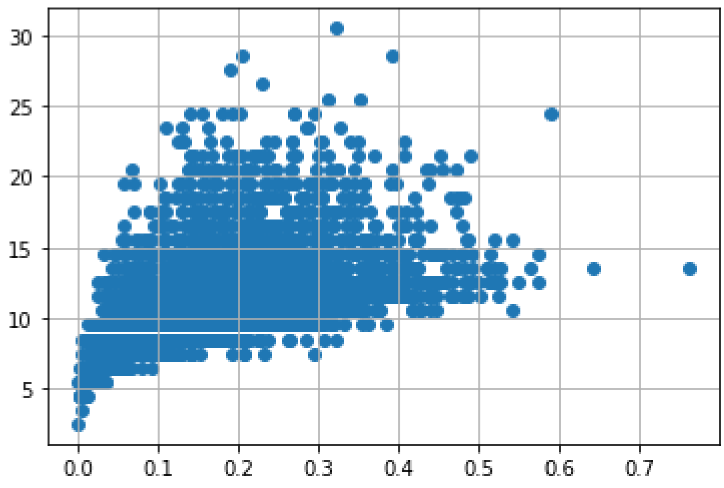
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

Outlier Handling

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

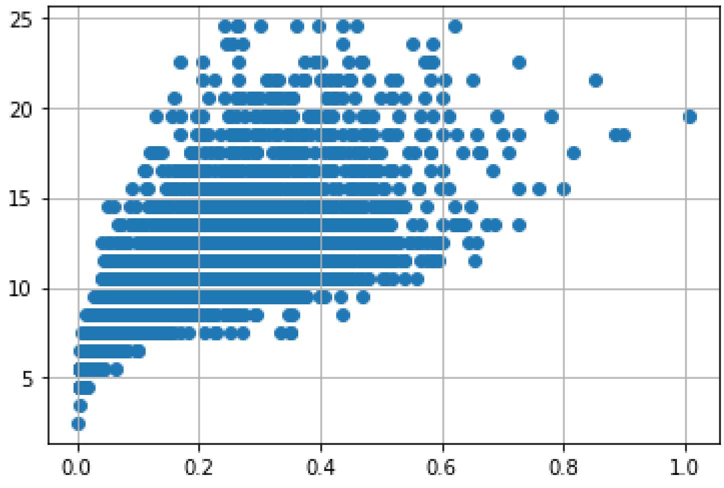
```
#outliers removal for viscera weight
```

```
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)
```



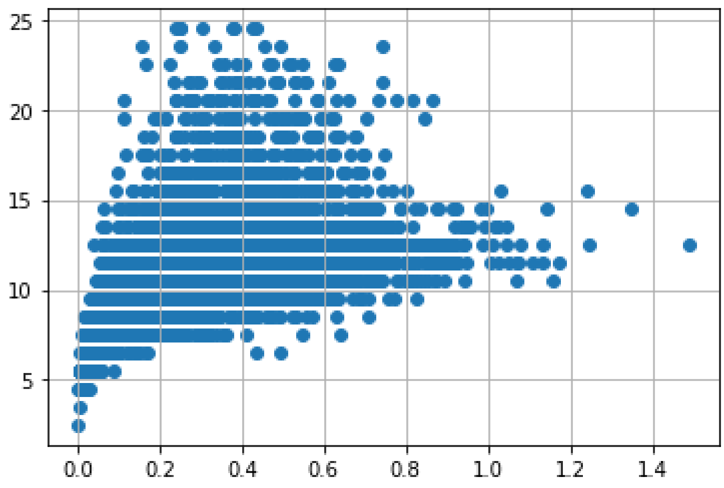

```
#outliers removal for shell weight
```

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




```
#Outliers removal for shuked weight
```

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

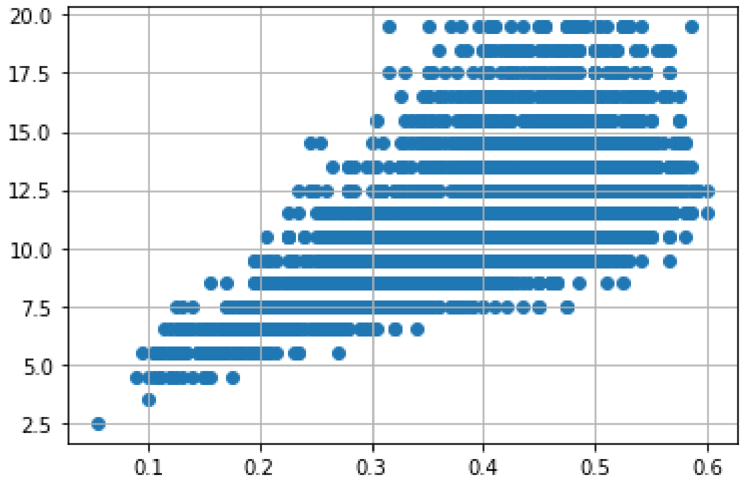



```
#outliers removal for whole weight
```

```
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)
```

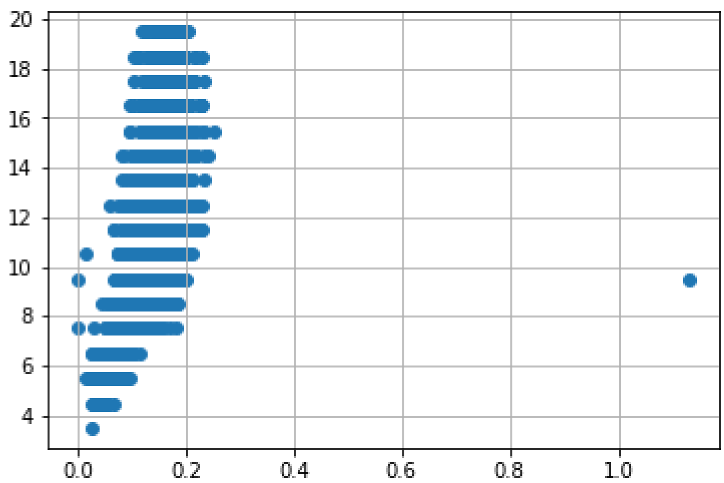

```
#outliers removal for diameters
```


```
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)
```



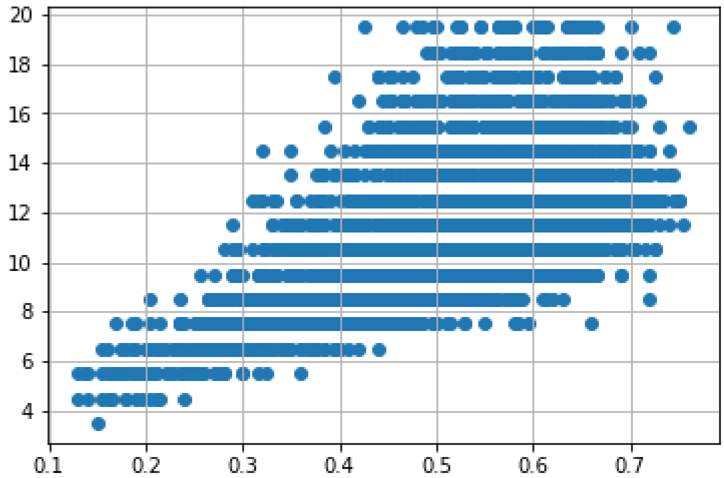

```
#outliers removal for height


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)
```



```
#outliers removal for length

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)
```



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: `
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
```

```
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')
```

Split the dependent and independent variables

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

	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 × 6 columns

split the data (train and test)

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Model Building

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

```
LinearRegression()
```

Train the model

```
x_train[0:4]
```

	Length	Diameter	Height	Whole weight	Shucked weight
2654	0.545	0.430	0.140	0.8320	0.4355
1927	0.615	0.470	0.150	1.0875	0.4975
3349	0.470	0.375	0.105	0.4680	0.1665
210	0.490	0.365	0.145	0.6345	0.1995

```
y_train[0:5]
```

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
2654	0.1700	0.2010	10.5	1	0	0
1927	0.2830	0.2685	10.5	0	0	1
3349	0.1080	0.1700	11.5	0	1	0
210	0.1625	0.2200	11.5	1	0	0
2337	0.1695	0.2450	12.5	0	0	1

Test the model

```
x_test[0:4]
```

	Length	Diameter	Height	Whole weight	Shucked weight
832	0.44	0.365	0.115	0.501	0.2435
3828	0.68	0.520	0.175	1.543	0.7525
4070	0.48	0.335	0.125	0.524	0.2460
1564	0.46	0.350	0.110	0.400	0.1760

```
y_test[0:5]
```

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
832	0.0840	0.1465	10.5	0	1	0
3828	0.3510	0.3740	12.5	0	0	1
4070	0.1095	0.1450	8.5	0	1	0
1564	0.0830	0.1205	8.5	0	1	0

```
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
```

```
lrpred=lr.predict(x_test[0:9])
```

```
lrpred
```

```
array([[1.03349797e-01, 1.46176709e-01, 9.94830412e+00, 2.07513223e-01,
        4.76563878e-01, 3.15922899e-01],
       [3.32292403e-01, 3.88571187e-01, 1.13642469e+01, 4.35538674e-01,
        9.09714235e-03, 5.55364184e-01],
       [1.18628384e-01, 1.44426541e-01, 9.74635777e+00, 1.86896709e-01,
        5.41056762e-01, 2.72046529e-01],
       [8.82063319e-02, 1.24706742e-01, 1.00390858e+01, 1.84922115e-01,
        5.62786902e-01, 2.52290983e-01],
       [1.05637045e-01, 1.47109060e-01, 1.05483729e+01, 2.21316254e-01,
        5.06681122e-01, 2.72002624e-01],
       [1.78444588e-01, 2.30724706e-01, 1.08998374e+01, 3.08914597e-01,
        2.89428215e-01, 4.01657189e-01],
       [2.11194701e-01, 2.71185034e-01, 1.13676751e+01, 3.53254514e-01,
        2.00968128e-01, 4.45777358e-01],
       [2.29246508e-01, 3.13521045e-01, 1.34259369e+01, 4.46728441e-01,
        1.23703244e-01, 4.29568315e-01],
       [2.48939082e-01, 3.11716127e-01, 1.07903974e+01, 3.79844823e-01,
        7.47301038e-02, 5.45425073e-01]])
```

Measure the performance using metrics

```
r2_score(lr.predict(x_test),y_test)
```

```
-3.372075449737968
```