Linear Regression

```
data = pd.read_csv('abalone.csv')
data = data.iloc[:,3:]
data%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
```

Load the dataset

Input:

```
data = pd.read_csv('abalone.csv')
data = data.iloc[:,3:]
data
```

Output:

Height	**Whole weight**	**Shucked weight**	**Viscera weight**	**Shell weight**	**Rings**	
0	0.095	0.5140	0.2245	0.1010	0.1500	15
1	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0.135	0.6770	0.2565	0.1415	0.2100	9
3	0.125	0.5160	0.2155	0.1140	0.1550	10
4	0.080	0.2050	0.0895	0.0395	0.0550	7

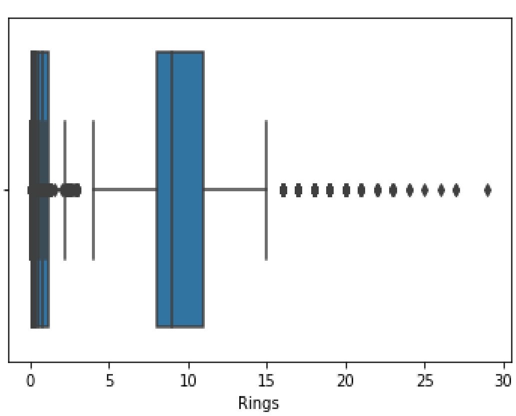
Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	
...	...	...	...	...	...	...
4172	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 6 columns

Visualization process

Input:

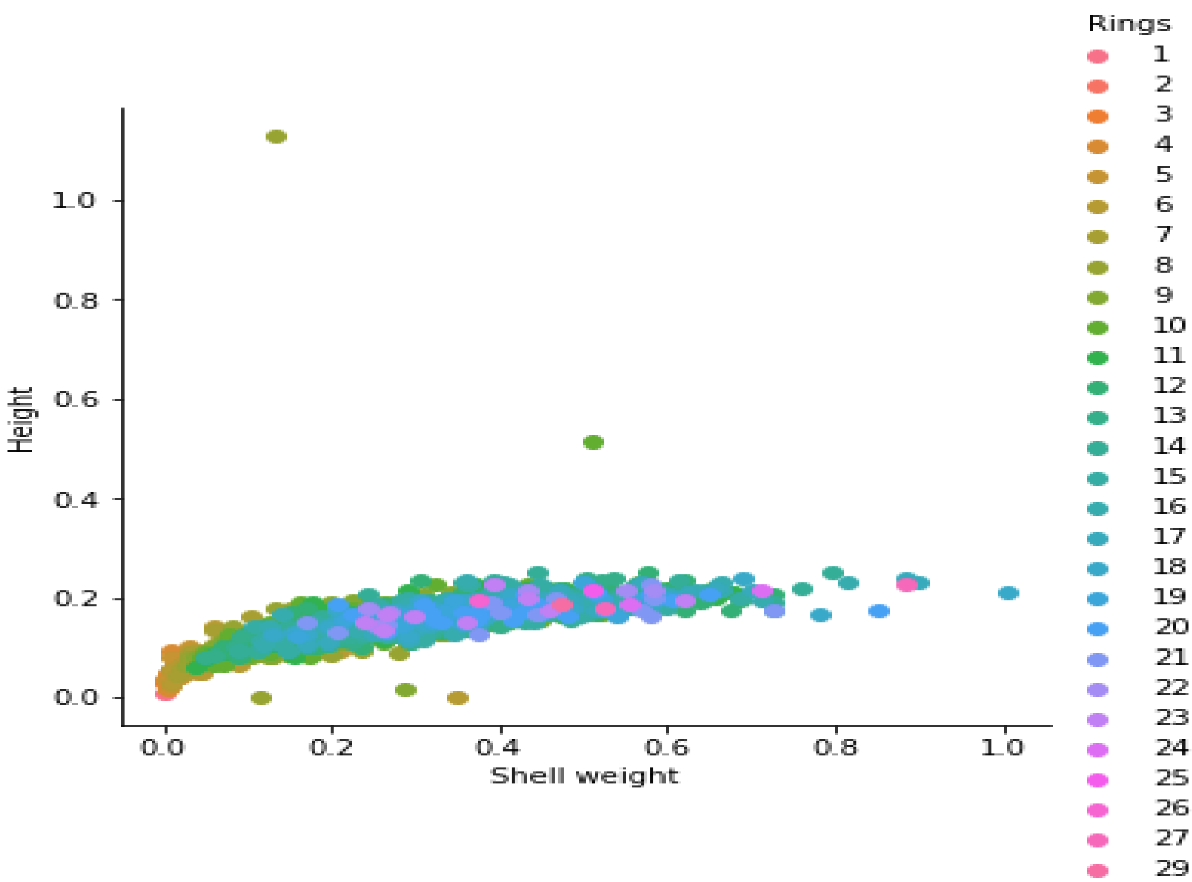
```
#univariate analysis
for col in data.columns:
  if data.dtypes[col]=='int64' or data.dtypes[col]=='float64':
    sns.boxplot(x=data[col]).set(xlabel=col)
```



Input:

#Bivariate analysis
sns.FacetGrid(data,hue='Rings',size=5).map(plt.scatter,"Shell weight","Height").add_legend()

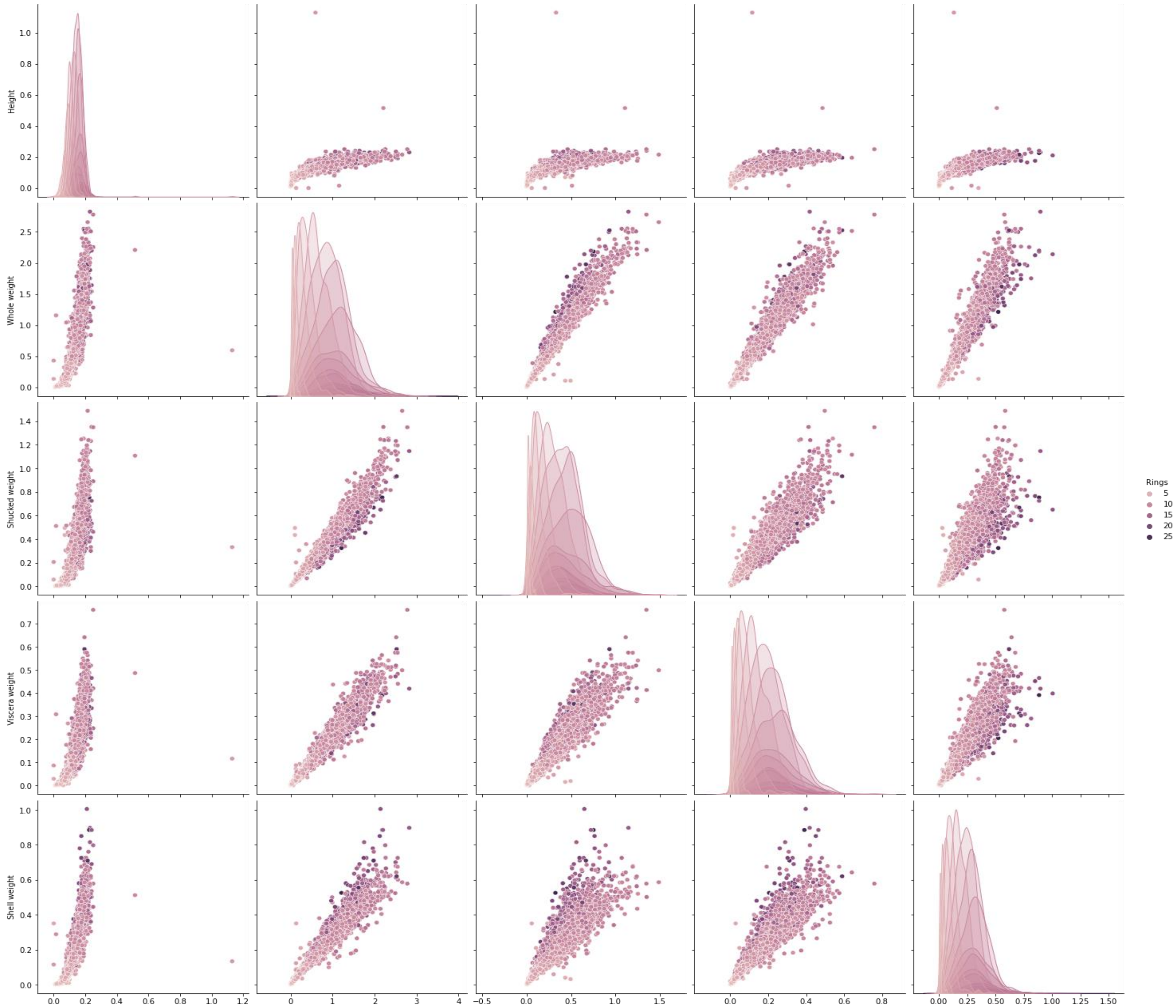
Output:



Input:

 # 3. Multi-Variate Analysis
sns.pairplot(data,hue='Rings',height=4)

Output:



Descriptive Statistics

Input:

data.describe()

Output:

	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

Handle the missing values

Input:

```
data.isnull().sum()
```

Output:

```
Height          0
Height          0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

Find and Replace the outliers

```
credit = data.loc[data['Height']<400, 'Height'].median()
prod = data.loc[data['Rings']>=3.5, 'Rings'].median()
data.loc[data.Height<400, 'Height']=np.nan
data.fillna(credit,inplace=True)
data.loc[data.Rings>3.5, 'Rings']=np.nan
data.fillna(prod,inplace=True)
```

Perform encoding for Categorical Columns

```
data['Whole weight'] = label.fit_transform(data['Whole weight'])
data['Shucked weight'] = label.fit_transform(data['Shucked weight'])
```

Dependent and Independent variables

```
dep = data.iloc[:,-1:]
indep = data.iloc[:,:-1]
```

Scale the independent variables

```
indep_var = MinMaxScaler()
show_indep = indep_var.fit_transform(indep)
```

Splitting Train and Test Data

```
xtrain,xtest,ytrain,ytest = train_test_split(show_indep, dep, test_size=0.3)
print(xtrain,xtest,ytrain,ytest)
```

```
[[0.         0.91680395 0.86856011 0.47399605 0.49775785]
 [0.         0.36985173 0.24438573 0.16063199 0.25759841]
 [0.         0.06342669 0.06340819 0.03620803 0.04235177]
 ...
 [0.         0.1630972  0.15323646 0.09282423 0.09367215]
 [0.         0.1985173  0.17701453 0.10533246 0.11559542]
 [0.         0.20551895 0.18295905 0.0757077  0.12306926]] [[0.         0.27182867 0.22457067 0.0
921659  0.16691579]
 [0.         0.24052718 0.23117569 0.15273206 0.11559542]
 [0.         0.28459638 0.28929987 0.13824885 0.12755356]
 ...
 [0.         0.17792422 0.167107   0.06649111 0.10413553]
 [0.         0.08401977 0.07529723 0.03291639 0.05829596]
 [0.         0.12067545 0.11889036 0.05069124 0.07573493]]       Rings
1981   10.0
2187   10.0
```

```
3632   10.0
1602   10.0
1930   10.0
...     ...
661    10.0
2860   10.0
1554   10.0
560    10.0
2638   10.0

[2923 rows x 1 columns]       Rings
613    10.0
1453   10.0
938    10.0
1131   10.0
2628   10.0
...     ...
306     3.0
2367   10.0
2378   10.0
319    10.0
3532   10.0

[1254 rows x 1 columns]
```

```
#visualize the first five row
data.head()
```

	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0.14	715	417	0.1010	0.150	10.0
1	0.14	285	178	0.0485	0.070	10.0
2	0.14	962	480	0.1415	0.210	10.0
3	0.14	718	400	0.1140	0.155	10.0
4	0.14	253	159	0.0395	0.055	10.0

```
data['Height'].unique()
array([0.14])
```

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
```

Build a model

```
import statsmodels.formula.api as smf

model=smf.ols("Height~Rings",data=data).fit()
```

Test model

```
model.params
```

Output:

```
Intercept    1.400000e-01
Rings        7.329207e-17
dtype: float64
```

```
model.rsquared , model.rsquared_adj
```

Output:

```
(-1151.283935839119, -1151.5599319914159)
```