

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

```
from google.colab import files
upload=files.upload()
df = pd.read_csv('abalone.csv')
```

abalone.csv

- **abalone.csv**(text/csv) - 191962 bytes, last modified: 10/29/2022 - 100% done
Saving abalone.csv to abalone (1).csv

```
df.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1

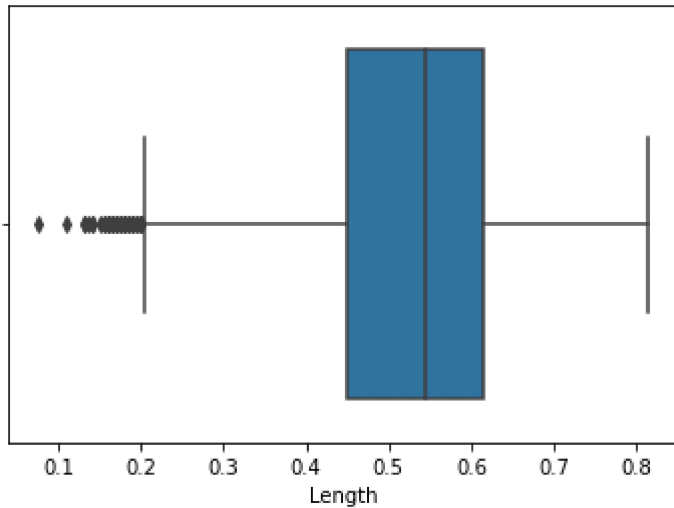
```
df.head()
```

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

```
sns.boxplot(df.Length)
```

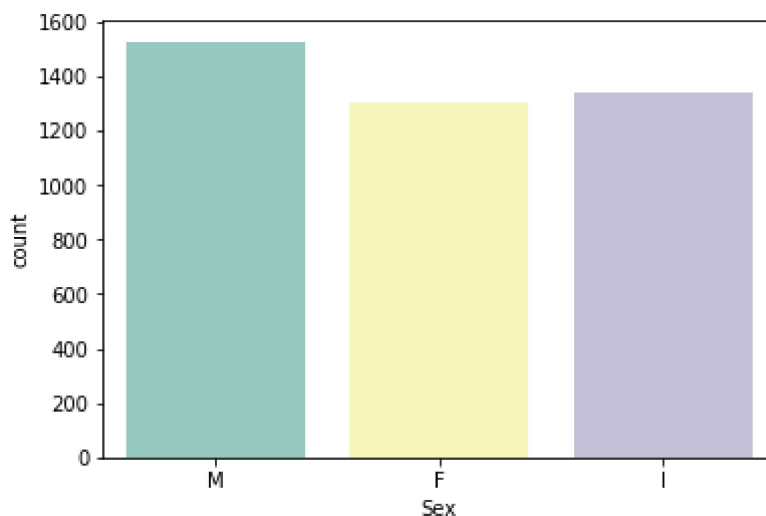
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff5877a5350>
```



```
sns.countplot(x = 'Sex', data = df, palette = 'Set3')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff586e42c10>
```

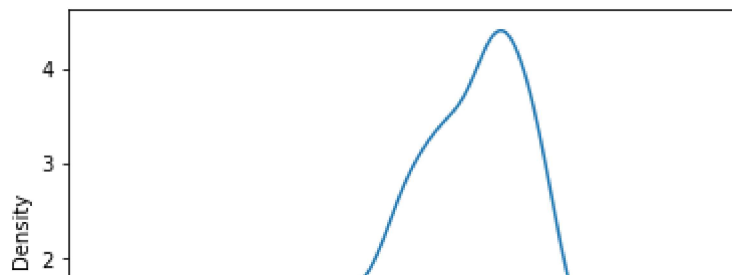


```
a = pd.read_csv('abalone.csv')
```

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

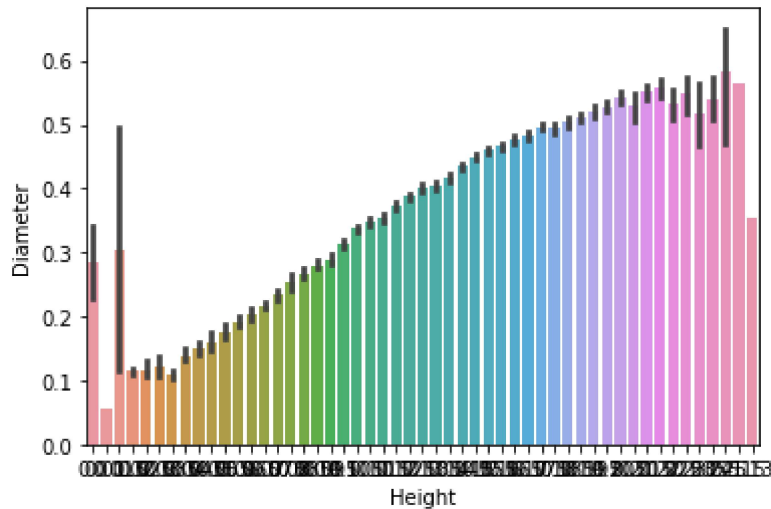
```
sns.kdeplot(a['Diameter'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff586d84350>
```



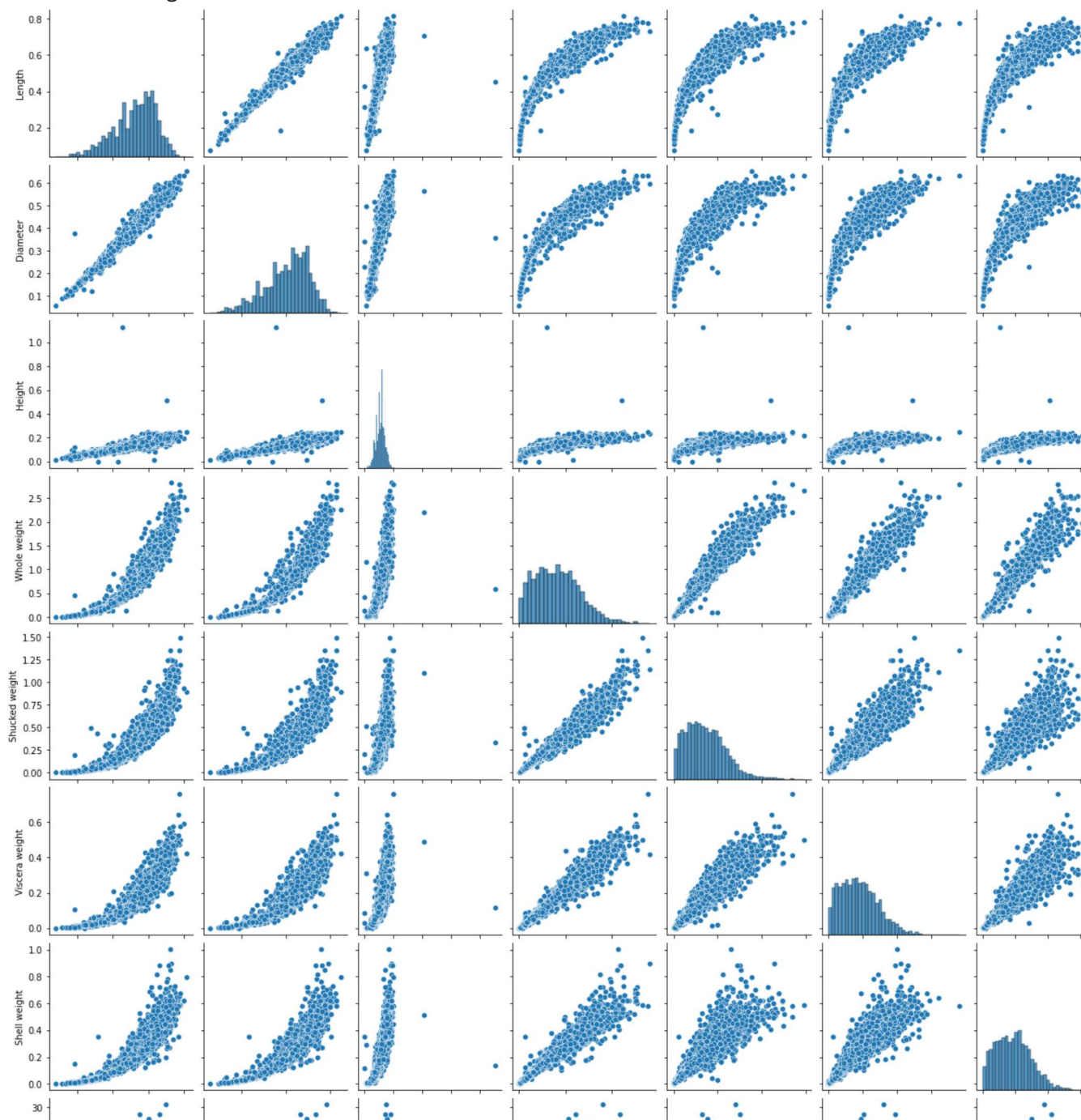
```
sns.barplot(x=df.Height,y=df.Diameter)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff5868893d0>
```



```
sns.pairplot(a)
```

```
<seaborn.axisgrid.PairGrid at 0x7ff5865d74d0>
```



```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 4177 entries, 0 to 4176
```

```
Data columns (total 9 columns):
```

#	Column	Non-Null Count	Dtype
0	Sex	4177 non-null	object
1	Length	4177 non-null	float64
2	Diameter	4177 non-null	float64
3	Height	4177 non-null	float64
4	Whole weight	4177 non-null	float64
5	Shucked weight	4177 non-null	float64
6	Viscera weight	4177 non-null	float64

```

7  Shell weight    4177 non-null    float64
8  age            4177 non-null    float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB

```

```
a['Diameter'].describe()
```

```

count    4177.000000
mean      0.407881
std       0.099240
min       0.055000
25%      0.350000
50%      0.425000
75%      0.480000
max       0.650000
Name: Diameter, dtype: float64

```

```
a['Sex'].value_counts()
```

```

M    1528
I    1342
F    1307
Name: Sex, dtype: int64

```

```
df['Height'].describe()
```

```

count    4177.000000
mean      0.139516
std       0.041827
min       0.000000
25%      0.115000
50%      0.140000
75%      0.165000
max       1.130000
Name: Height, dtype: float64

```

```
df[df.Height == 0]
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
1257	I	0.430	0.34	0.0	0.428	0.2065	0.0860
3996	I	0.315	0.23	0.0	0.134	0.0575	0.0285

```
df['Diameter'].median()
```

```
0.425
```

```
df['Shucked weight'].skew()
```

0.7190979217612694

```
missing_values = df.isnull().sum().sort_values(ascending = False)
percentage_missing_values = (missing_values/len(df))*100
pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Missing values', '%
```

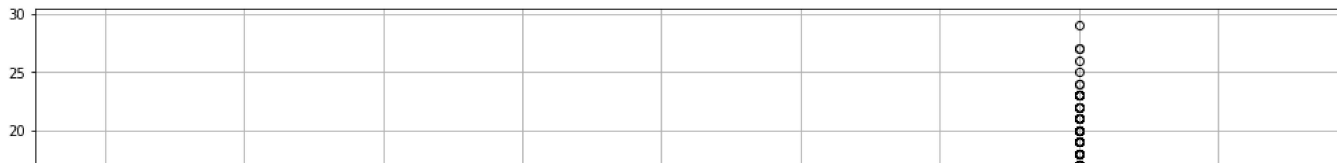
	Missing values	% Missing	
Sex	0	0.0	
Length	0	0.0	
Diameter	0	0.0	
Height	0	0.0	
Whole weight	0	0.0	
Shucked weight	0	0.0	
Viscera weight	0	0.0	
Shell weight	0	0.0	
Rings	0	0.0	

```
q1=df.Rings.quantile(0.25)
q2=df.Rings.quantile(0.75)
iqr=q2-q1
print(iqr)
```

3.0

```
df = pd.get_dummies(df)
dummy_df = df
df.boxplot( rot = 90, figsize=(20,5))
```

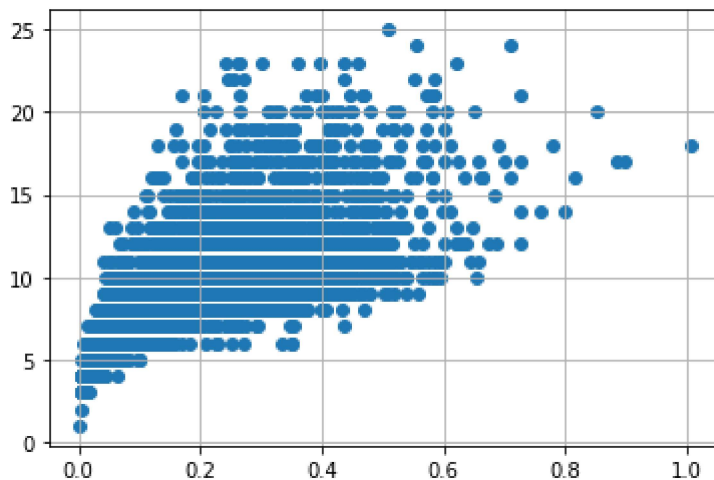
<matplotlib.axes._subplots.AxesSubplot at 0x7ff581e54fd0>



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

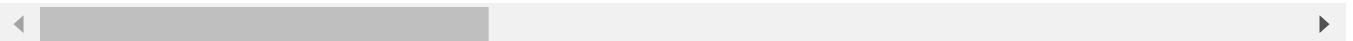
```
df.drop(df[(df['Viscera weight'] > 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight'] < 0.5) & (df['age'] > 25)].index, inplace=True)
```

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



```
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/rele>



```
abalone_numeric = df[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', 'Viscera weight', 'Shell weight']]
abalone_numeric.head()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age	Sex_
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	(

```
x = df.iloc[:, 0:1].values
```

```
y = df.iloc[:, 1]
```

```
y
```

```
0      0.365
```

```
1      0.265
```

```
2      0.420
```

```
3      0.365
```

```
4      0.255
```

```
...
```

```
4172    0.450
```

```
4173    0.440
```

```
4174    0.475
```

```
4175    0.485
```

```
4176    0.555
```

```
Name: Diameter, Length: 4150, dtype: float64
```

```
print ("\n ORIGINAL VALUES: \n\n", x,y)
```

```
ORIGINAL VALUES:
```

```
[[0.455]
```

```
[0.35 ]
```

```
[0.53 ]
```

```
...
```

```
[0.6 ]
```

```
[0.625]
```

```
[0.71 ]] 0      0.365
```

```
1      0.265
```

```
2      0.420
```

```
3      0.365
```

```
4      0.255
```

```
...
```

```
4172    0.450
```

```
4173    0.440
```

```
4174    0.475
```

```
4175    0.485
```

```
4176    0.555
```

```
Name: Diameter, Length: 4150, dtype: float64
```

```
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:
```



```

[[0.51351351]
 [0.37162162]
 [0.61486486]
 ...
 [0.70945946]
 [0.74324324]
 [0.85810811]]

```

```

X = df.drop('age', axis = 1)
y = df['age']

```

```

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
standardScale = StandardScaler()
standardScale.fit_transform(X)

```

```

selectkBest = SelectKBest()
X_new = selectkBest.fit_transform(X, y)

```

```

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)
X_train

```

```

array([[0.475, 0.355, 0.12 , ..., 0.    , 0.    , 1.    ],
       [0.5   , 0.365, 0.13 , ..., 0.    , 0.    , 1.    ],
       [0.58  , 0.43  , 0.125, ..., 0.    , 0.    , 1.    ],
       ...,
       [0.49  , 0.38  , 0.135, ..., 0.    , 0.    , 1.    ],
       [0.4   , 0.31  , 0.1  , ..., 0.    , 1.    , 0.    ],
       [0.5   , 0.37  , 0.115, ..., 0.    , 1.    , 0.    ]])

```

```

y_train

```

```

65      8
2826    9
3100   10
1753   10
503    13
..
1820   13
1902   11
12     11
2127    7
2980    7
Name: age, Length: 3112, dtype: int64

```

```

from sklearn import linear_model as lm
from sklearn.linear_model import LinearRegression

```

```

model=lm.LinearRegression()
results=model.fit(X_train,y_train)
accuracy = model.score(X_train, y_train)
print('Accuracy of the model:', accuracy)

```

Accuracy of the model: 0.5385553745257212

```

lm = LinearRegression()
lm.fit(X_train, y_train)
y_train_pred = lm.predict(X_train)
y_train_pred

```

```

array([ 8.7821132 ,  8.69223506,  7.91346477, ..., 10.66419868,
        6.49661756,  7.32689668])

```

X_train

```

array([[0.475, 0.355, 0.12 , ..., 0.    , 0.    , 1.    ],
       [0.5   , 0.365, 0.13 , ..., 0.    , 0.    , 1.    ],
       [0.58  , 0.43  , 0.125, ..., 0.    , 0.    , 1.    ],
       ...,
       [0.49  , 0.38  , 0.135, ..., 0.    , 0.    , 1.    ],
       [0.4   , 0.31  , 0.1   , ..., 0.    , 1.    , 0.    ],
       [0.5   , 0.37  , 0.115, ..., 0.    , 1.    , 0.    ]])

```

y_train

```

65      8
2826    9
3100   10
1753   10
503    13
..
1820   13
1902   11
12     11
2127    7
2980    7
Name: age, Length: 3112, dtype: int64

```

```

from sklearn.metrics import mean_absolute_error, mean_squared_error
s = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error of training set :%2f'%s)

```

Mean Squared error of training set :4.638811

```

y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
y_test_pred

```

```
array([ 9.65587378,  5.56256054, 12.55940421, ...,  8.63884103,
        7.17360886,  9.97959562])
```

X_test

```
array([[0.42 , 0.325, 0.115, ..., 0.    , 0.    , 1.    ],
       [0.31 , 0.225, 0.05 , ..., 0.    , 1.    , 0.    ],
       [0.52 , 0.415, 0.175, ..., 0.    , 0.    , 1.    ],
       ...,
       [0.385, 0.305, 0.105, ..., 1.    , 0.    , 0.    ],
       [0.635, 0.495, 0.015, ..., 1.    , 0.    , 0.    ],
       [0.55 , 0.43 , 0.145, ..., 0.    , 1.    , 0.    ]])
```

y_test

```
542      15
2116      6
3223      8
1974     10
2554      7
      ..
2525      9
1956     11
2287      7
1174      9
2760     10
```

Name: age, Length: 1038, dtype: int64

```
p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

Mean Squared error of testing set :5.150781

```
from sklearn.metrics import r2_score
s = r2_score(y_train, y_train_pred)
print('R2 Score of training set:%.2f'%s)
```

R2 Score of training set:0.54

[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 12:13 PM

