Task: Predicting the age of abalone from physical measurements

In [ ]:

```
## 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
```

In [77]:

```
### Importing The Required libraries


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
from scipy import stats

%matplotlib inline


### model process
from sklearn import linear_model as lm  # modelling

from sklearn.model_selection import train_test_split
```

In [2]:

```
df=pd.read_csv("C:/Users/nprav/Downloads/abalone.csv")
```

In [3]:

```
df
```

Out[3]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

In [4]:

```
### Changing column names for better understand
df.columns=["sex",'length','diameter',"height",'whole_weight',"shucked_weight","viscera_w
eight","shell_weight","rings"]
```

In [5]:

```
df.head()
```

Out[5]:

	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

In [6]:

```
df.tail()
```

Out[6]:

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

In [17]:

```
print("Shape of the colunm:")
print(df.shape)
print()
print("DataTypes of the column:")
print(df.dtypes)
```

```
Shape of the colunm:
(4177, 9)

DataTypes of the column:
sex                object
length            float64
diameter          float64
height            float64
whole_weight      float64
shucked_weight    float64
viscera_weight    float64
shell_weight      float64
rings               int64
dtype: object
```

In [7]:

```
df.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   rings           4177 non-null   int64  
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

In [9]:

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

### No null values here
```

Out[9]:

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

In [10]:

```
df.describe()
```

Out[10]:

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
count	4177.000000	4177.000000	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	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

In [11]:

```
df.corr()
```

Out[11]:

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
whole_weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
shucked_weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
viscera_weight	0.903818	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
shell_weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

In [12]:

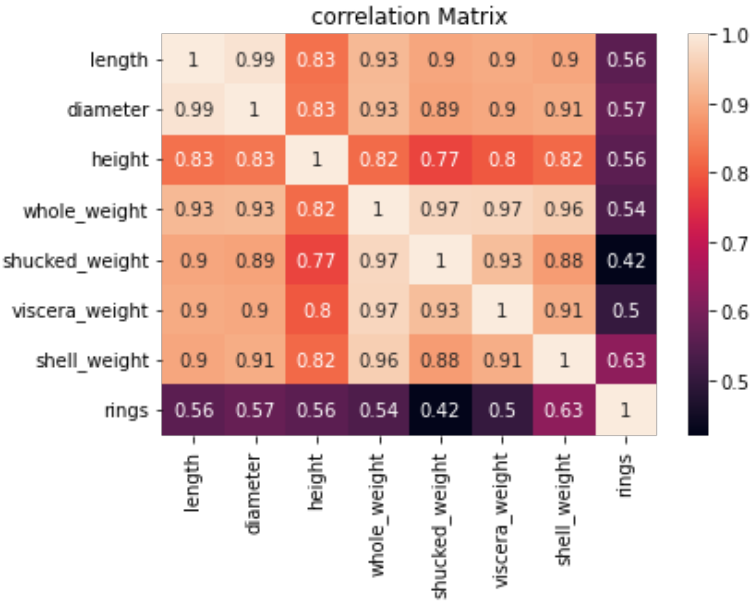
```
df.cov()
```

Out[12]:

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
length	0.014422	0.011761	0.004157	0.054491	0.023935	0.011887	0.015007	0.215562
diameter	0.011761	0.009849	0.003461	0.045038	0.019674	0.009787	0.012507	0.183872
height	0.004157	0.003461	0.001750	0.016803	0.007195	0.003660	0.004759	0.075179
whole_weight	0.054491	0.045038	0.016803	0.240481	0.105518	0.051946	0.065216	0.854409
shucked_weight	0.023935	0.019674	0.007195	0.105518	0.049268	0.022675	0.027271	0.301204
viscera_weight	0.011887	0.009787	0.003660	0.051946	0.022675	0.012015	0.013850	0.178057
shell_weight	0.015007	0.012507	0.004759	0.065216	0.027271	0.013850	0.019377	0.281663
rings	0.215562	0.183872	0.075179	0.854409	0.301204	0.178057	0.281663	10.395266

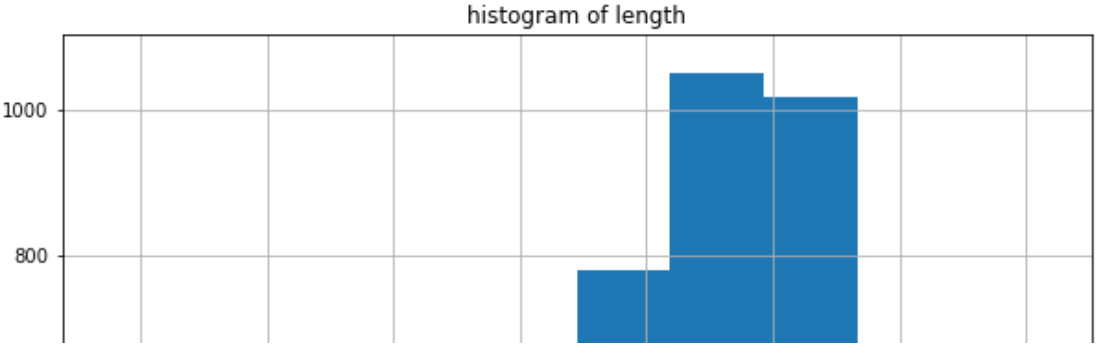
In [27]:

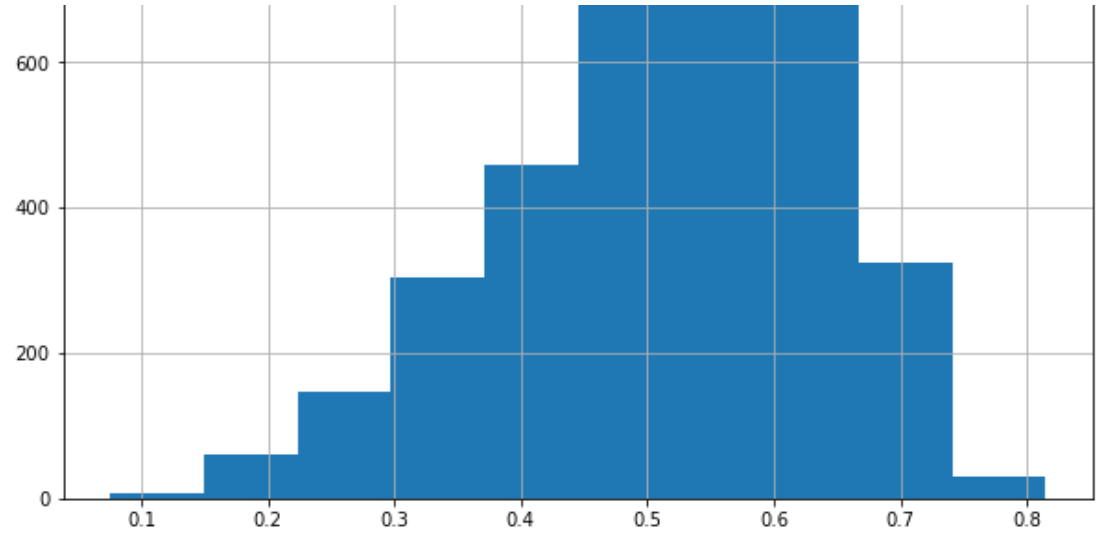
```
sns.heatmap(df.corr(),annot=True)
plt.title("correlation Matrix")
plt.show()
```



In [28]:

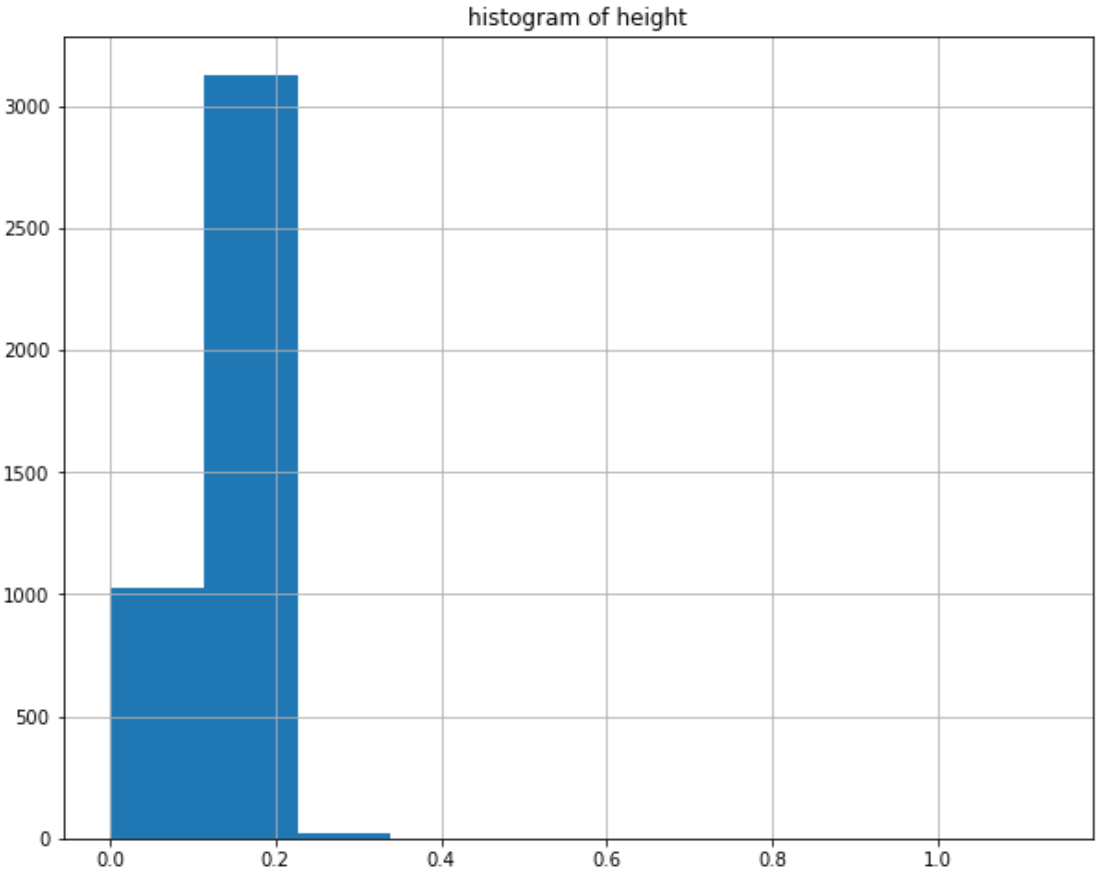
```
df['length'].hist(bins=10,figsize=(10,8))
plt.title("histogram of length")
plt.show()
```





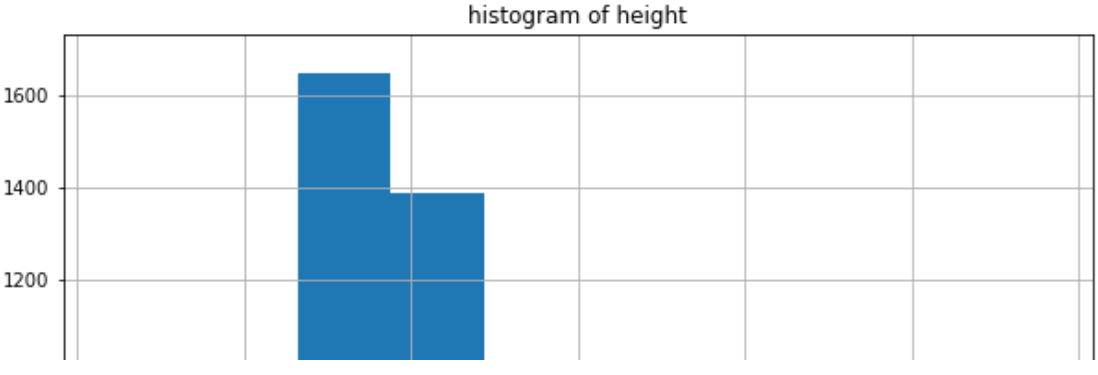
In [29]:

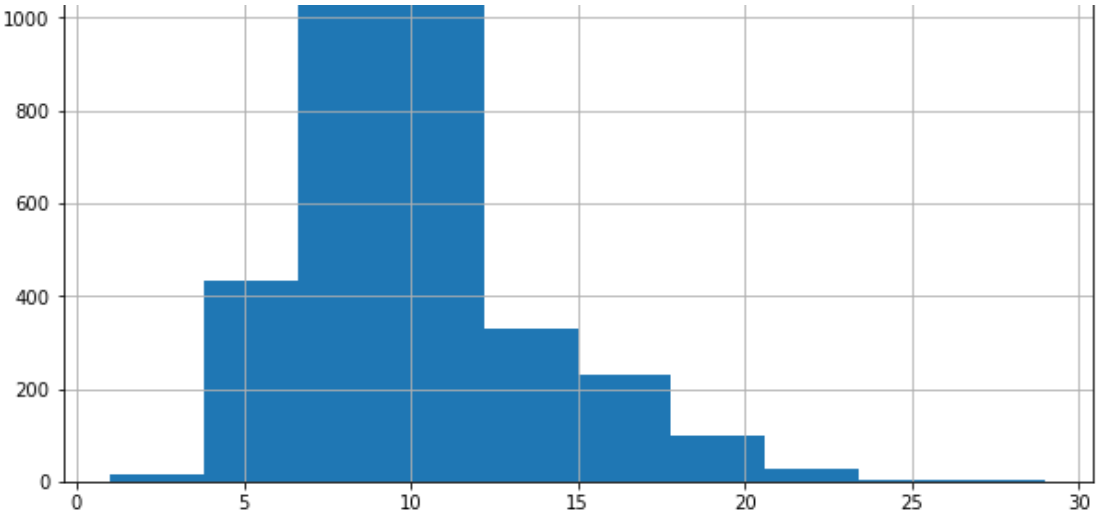
```
df['height'].hist(bins=10,figsize=(10,8))
plt.title("histogram of height")
plt.show()
```



In [30]:

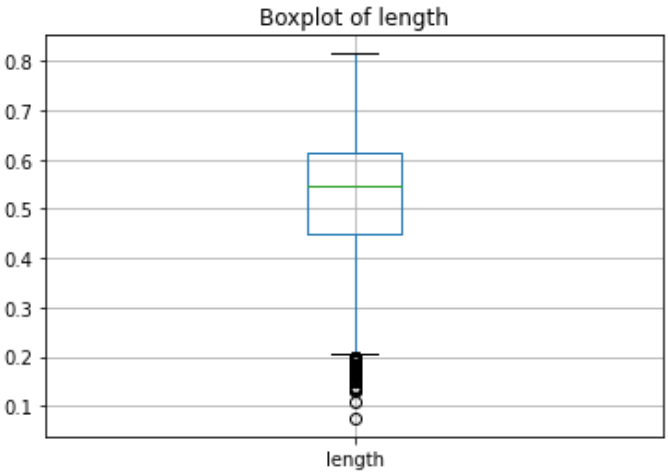
```
df['rings'].hist(bins=10,figsize=(10,8))
plt.title("histogram of height")
plt.show()
```





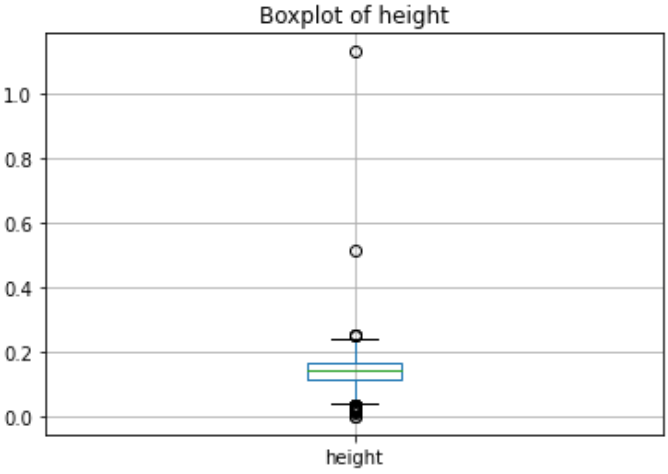
In [31]:

```
df.boxplot(column="length")
plt.title("Boxplot of length")
plt.show()
```



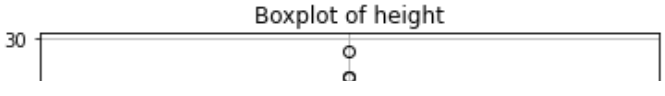
In [32]:

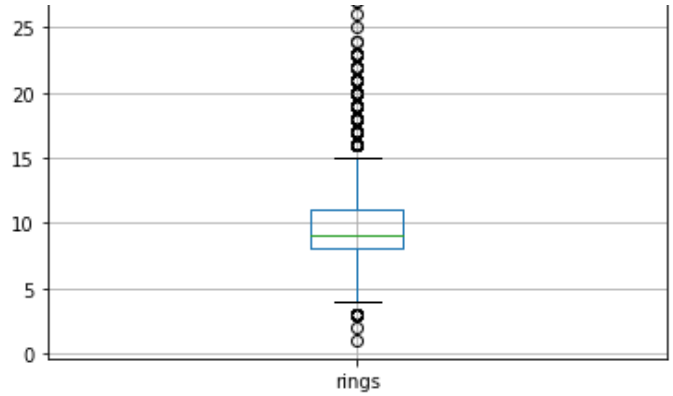
```
df.boxplot(column="height")
plt.title("Boxplot of height")
plt.show()
```



In [33]:

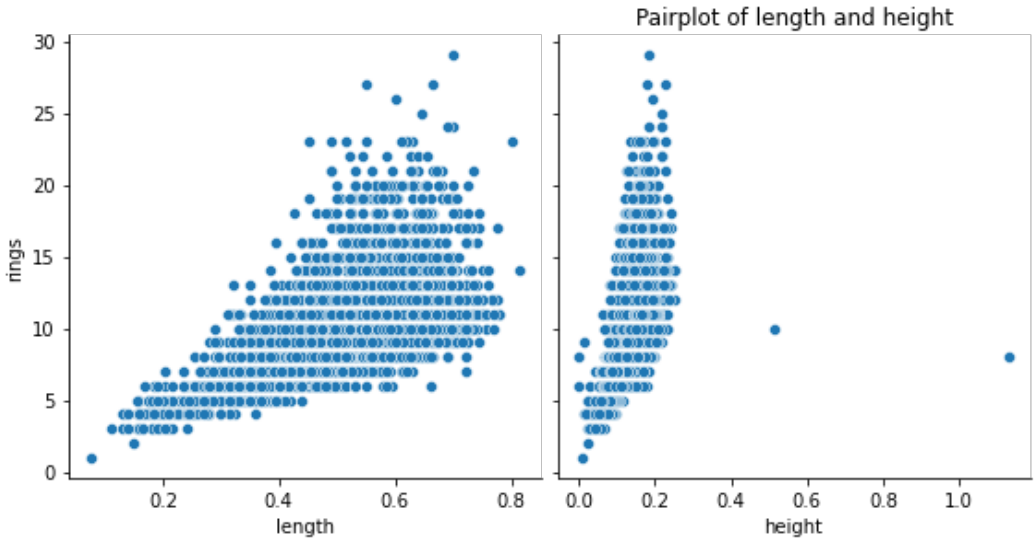
```
df.boxplot(column="rings")
plt.title("Boxplot of height")
plt.show()
```





In [36]:

```
sns.pairplot(df,x_vars=['length','height'],y_vars='rings',height=4)
plt.title("Pairplot of length and height")
plt.show()
```



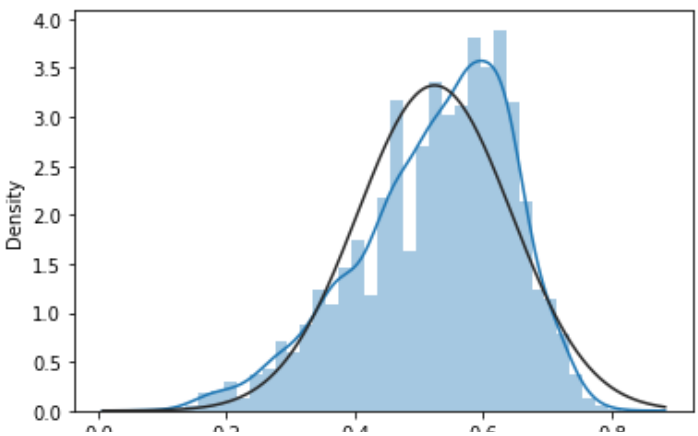
In [38]:

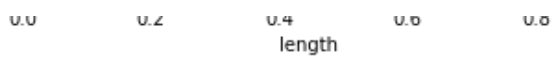
```
sns.distplot(df['length'],fit=stats.norm)
```

```
C:\Users\nprav\AppData\Local\Temp\ipykernel_15000\553287353.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['length'],fit=stats.norm)
```

Out[38]:

```
<AxesSubplot:xlabel='length', ylabel='Density'>
```





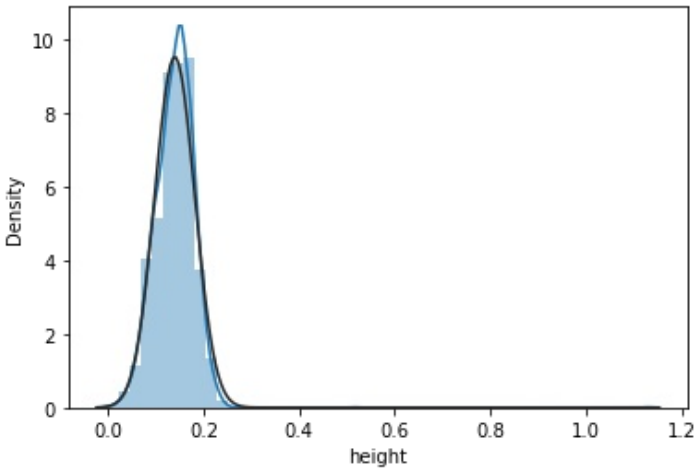
In [39]:

```
sns.distplot(df['height'],fit=stats.norm)
```

```
C:\Users\nprav\AppData\Local\Temp\ipykernel_15000\3887100128.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['height'],fit=stats.norm)
```

Out[39]:

```
<AxesSubplot:xlabel='height', ylabel='Density'>
```



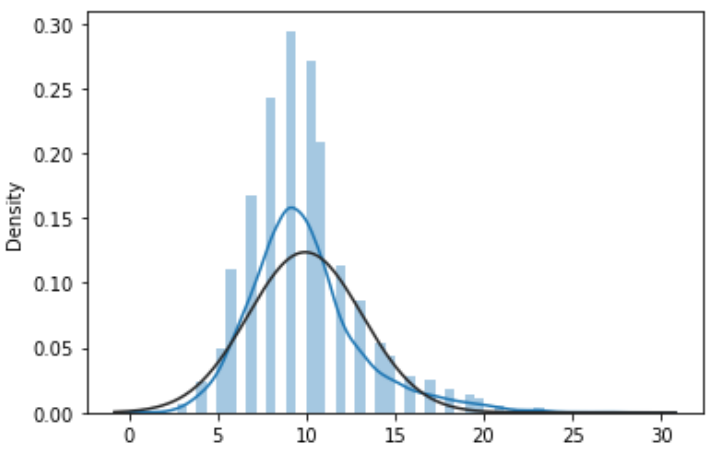
In [40]:

```
sns.distplot(df['rings'],fit=stats.norm)
```

```
C:\Users\nprav\AppData\Local\Temp\ipykernel_15000\579089518.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['rings'],fit=stats.norm)
```

Out[40]:

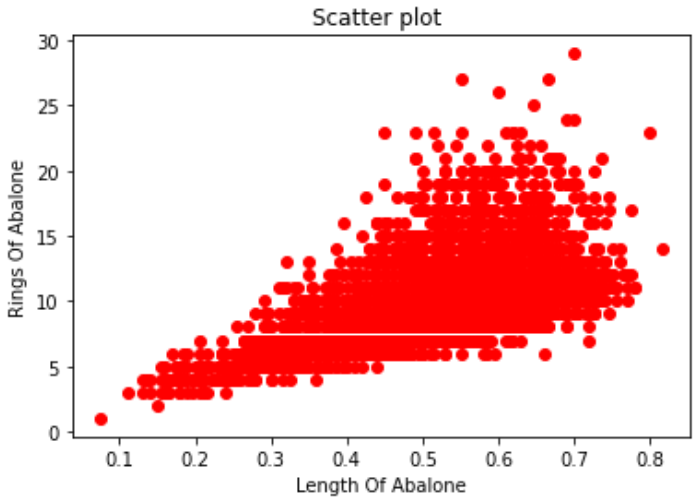
```
<AxesSubplot:xlabel='rings', ylabel='Density'>
```



rings

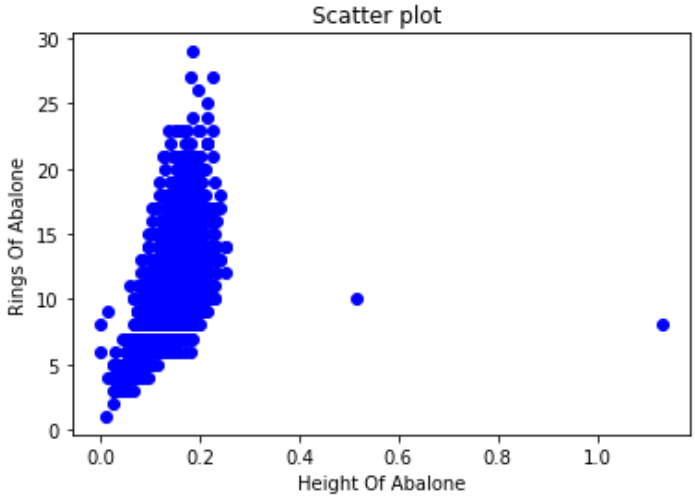
In [44]:

```
x=df['length']
y=df['rings']
plt.scatter(x,y,c="red")
plt.title("Scatter plot ")
plt.xlabel("Length Of Abalone")
plt.ylabel("Rings Of Abalone")
plt.show()
```



In [43]:

```
x=df['height']
y=df['rings']
plt.scatter(x,y,c="blue")
plt.title("Scatter plot ")
plt.xlabel("Height Of Abalone")
plt.ylabel("Rings Of Abalone")
plt.show()
```



In [45]:

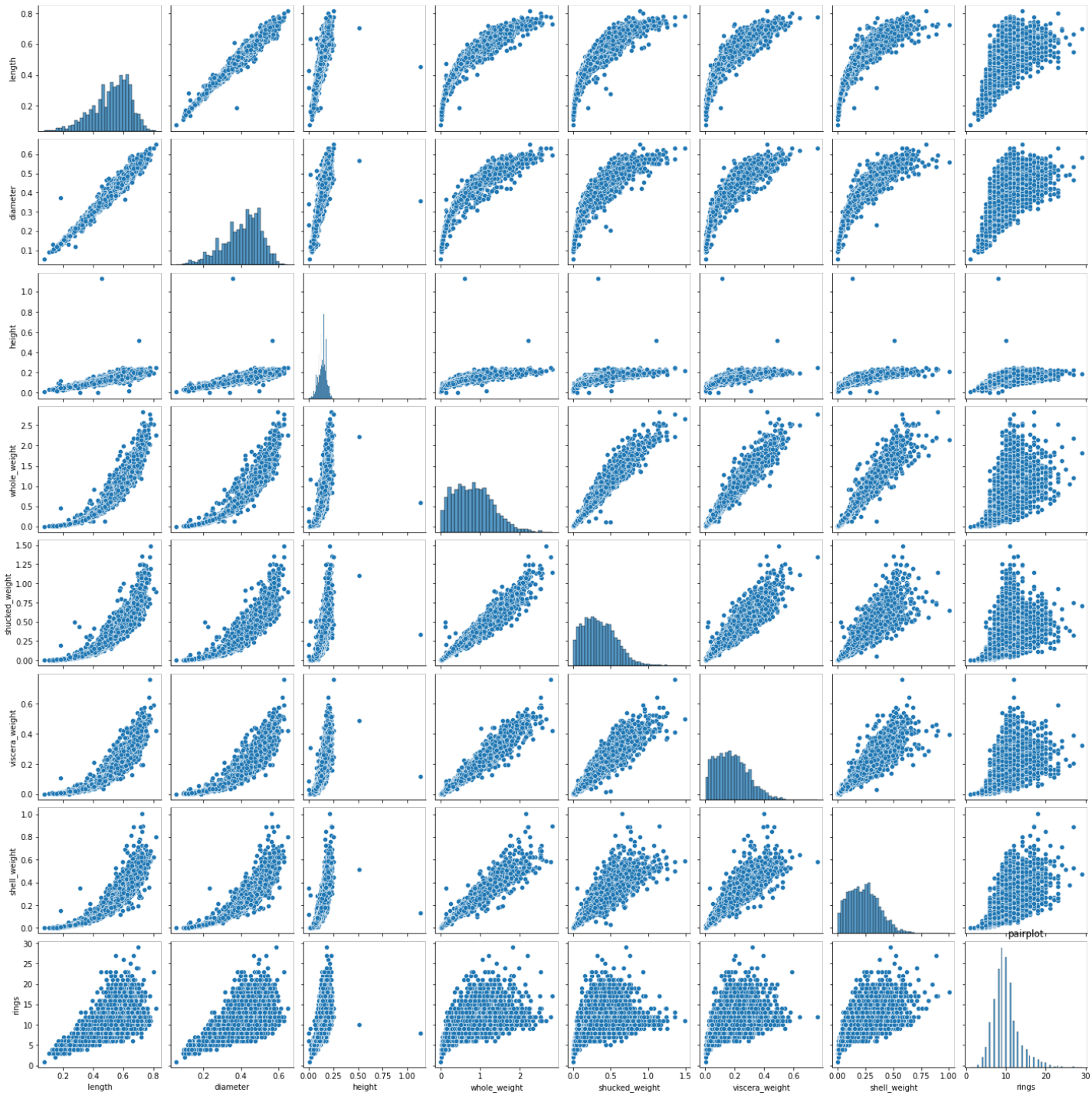
```
len=df["length"]

heg=df["height"]

ring=df["rings"]
```

In [53]:

```
sns.pairplot(df)
plt.title("pairplot")
```

Out[53]:

```
Text(0.5, 1.0, 'pairplot')
```



In [46]:

```
### Rgression model using simple basic stats
import statsmodels.api as sm
model=sm.OLS(ring,len).fit()
```

In [48]:

```
model.summary()
```

Out[48]:

OLS Regression Results

Dep. Variable:	rings	**R-squared (uncentered):**	0.932
Model:	OLS	**Adj. R-squared (uncentered):**	0.932
Method:	Least Squares	**F-statistic:**	5.744e+04
Date:	Tue, 25 Oct 2022	**Prob (F-statistic):**	0.00
Time:	20:54:40	**Log-Likelihood:**	-10105.
No. Observations:	4177	**AIC:**	2.021e+04

No. Observations:	4177	AIC:	2.021e+04
Df Residuals:	4176	BIC:	2.022e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
length	18.7576	0.078	239.669	0.000	18.604	18.911

Omnibus:	1183.909	Durbin-Watson:	0.885
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3337.687
Skew:	1.488	Prob(JB):	0.00
Kurtosis:	6.213	Cond. No.	1.00

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [59]:

```
### Encoding process:

### one Hot encoding :

data_encoded_df=pd.get_dummies(df)
```

In [60]:

```
data_encoded_df.head()
```

Out[60]:

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings	sex_F	sex_I	sex_M
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	0	1	0

In [61]:

```
data_encoded_df.tail()
```

Out[61]:

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings	sex_F	sex_I	sex_M
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11	1	0	0
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10	0	0	1
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9	0	0	1
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10	1	0	0
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12	0	0	1

In [62]:

```
data_encoded_df.shape
```

Out[62]:

```
(4177, 11)
```

In [63]:

```
### split the data:

train_data=data_encoded_df.iloc[:4095,:]
validate_data=data_encoded_df.iloc[4095:,:]

print(train_data.shape)
print(validate_data.shape)
```

```
(4095, 11)
(82, 11)
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [64]:

```
x=train_data[["length","height"]]
```

In [74]:

```
y=train_data["rings"].values.reshape(-1,1)

y
```

Out[74]:

```
array([[15],
       [ 7],
       [ 9],
       ...,
       [11],
       [11],
       [13]], dtype=int64)
```

In [66]:

```
x_val=validate_data[["length","height"]]
```

In [73]:

```
y_val=validate_data['rings'].values.reshape(-1,1)
y_val
```

Out[73]:

```
array([[11],
       [12],
       [ 9],
       [ 9],
       [ 9],
       [ 9],
       [11],
       [11],
       [10],
       [11],
       [ 9],
       [11],
       [ 7],
       [ 7],
       [ 8]
```

```
       [ 9],
       [ 8],
       [ 9],
       [ 8],
       [ 9],
       [10],
       [ 9],
       [ 9],
       [ 9],
       [ 4],
       [ 7],
       [ 9],
       [ 8],
       [ 8],
       [ 8],
       [ 9],
       [11],
       [10],
       [ 8],
       [10],
       [10],
       [11],
       [10],
       [11],
       [ 9],
       [11],
       [ 9],
       [11],
       [11],
       [10],
       [10],
       [11],
       [13],
       [13],
       [11],
       [11],
       [10],
       [11],
       [11],
       [ 6],
       [ 7],
       [ 6],
       [ 7],
       [ 8],
       [ 6],
       [ 6],
       [ 8],
       [ 8],
       [ 8],
       [ 9],
       [11],
       [11],
       [ 8],
       [ 7],
       [ 7],
       [ 7],
       [10],
       [ 9],
       [ 8],
       [10],
       [10],
       [ 8],
       [11],
       [10],
       [ 9],
       [10],
       [12]], dtype=int64)
```

In [105]:

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=2)
```

In [106]:

```
from sklearn import linear_model as lm
model=lm.LinearRegression()
results=model.fit(X_train,y_train)
```

In [116]:

```
# model

model
```

Out[116]:

```
LinearRegression()
```

In [108]:

```
accuracy = model.score(X_train, y_train) # SCORE is used to find the accuracy
print('Accuracy of the model:', accuracy)
```

```
Accuracy of the model: 0.36581011994528945
```

In [82]:

```
print('intercept:', model.intercept_)


print('slope:', model.coef_)
```

```
intercept: [2.39877558]
slope: [[ 8.52236136 22.18277048]]
```

In [83]:

```
### model testing

X_test
```

Out[83]:

	length	height
1298	0.530	0.155
2711	0.190	0.030
3543	0.460	0.090
3357	0.375	0.100
2865	0.315	0.075
...	...	...
263	0.245	0.060
3299	0.605	0.175
3954	0.485	0.105
680	0.370	0.100
1973	0.680	0.155

410 rows × 2 columns

In [84]:

```
X_test.shape
```

Out[84]:

```
(410, 2)
```

In [85]:

```
y_test.shape
```

Out[85]:

```
(410, 1)
```

In [87]:

```
X_test = X_test.values.reshape((-1,1))
```

In [118]:

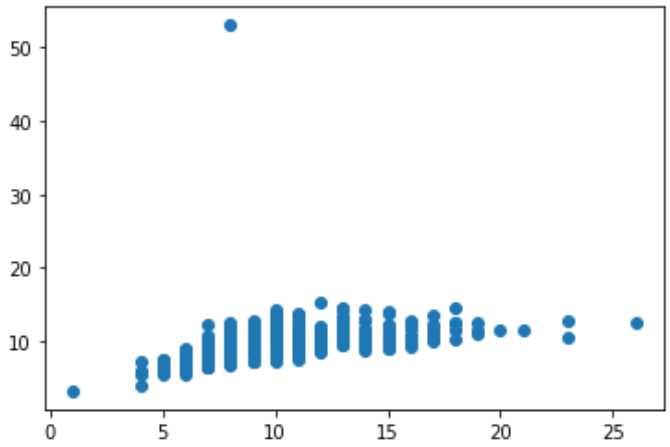
```
#### Predictions from the model
prediction = model.predict(X_test)
```

In [114]:

```
### visualizing the prediction

plt.scatter(y_test, prediction)
```

Out[114]:

```
<matplotlib.collections.PathCollection at 0x1a523cd09d0>
```



In [123]:

```
#### Giving new data values to the model to predict
X_new_val = [[30,45]]

#Make a Prediction
y_new_val = model.predict(X_new_val)

print(X_new_val ,y_new_val)
```

```
[[30, 45]] [[2037.49530562]]
```

```
C:\Users\nprav\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not h
ave valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Metrics

In [136]:

```
test_rmse=mean_squared_error(y_test,prediction,squared=False)
```

In [137]:

```
print(f"Test rmse:{test_rmse}")
```

```
Test rmse:3.467565501714804
```

In [138]:

```
lm_predict_val= model.predict(x_val)

val_rmse=mean_squared_error(y_val,lm_predict_val,squared=False)

print(f"validation of rmse:{val_rmse}")
```

```
validation of rmse:1.599376907868416
```