PROJECT NAME:- HOSING PRICE REGRESSION

SUBMITTED TO:- TECH-A-INTERN

SUBMITTED BY:- SHIVESH PANDEY

LEVEL-2 TASK-1 DATA SCIENTIST

```
import os
os.environ['KAGGLE_CONFIG_DIR']='/content'
```

```
! kaggle datasets download -d ashydv/housing-dataset
```

```
Downloading housing-dataset.zip to /content
  0% 0.00/4.63k [00:00<?, ?B/s]
100% 4.63k/4.63k [00:00<00:00, 10.6MB/s]
```

```
! chmod 600 /content/kaggle.json
```

```
!unzip \*.zip && rm *.zip
```

```
Archive:  housing-dataset.zip
replace Housing.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

```
import pandas as pd

df=pd.read_csv('Housing.csv')
```

```
df
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	base
0	13300000	7420	4	2	3	yes	no	
1	12250000	8960	4	4	4	yes	no	
2	12250000	9960	3	2	2	yes	no	
3	12215000	7500	4	2	2	yes	no	
4	11410000	7420	4	1	2	yes	yes	
...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	yes	no	
541	1767150	2400	3	1	1	no	no	
542	1750000	3620	2	1	1	yes	no	
543	1750000	2910	3	1	1	no	no	

544	1750000	3850	3	1	2	yes	no

545 rows × 13 columns

Next steps: Generate code with `df` View recommended plots

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

```
df.describe()
```

	price	area	bedrooms	bathrooms	stories	p
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.(
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.(
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.8
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.(
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.(
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.(
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.(
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.(

```
df.shape
```

```
(545, 13)
```

```
df.dropna()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	base

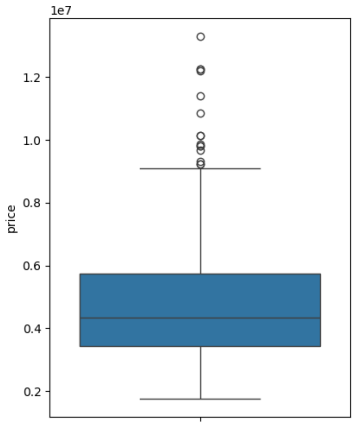
0	13300000	7420	4	2	3	yes	no
1	12250000	8960	4	4	4	yes	no
2	12250000	9960	3	2	2	yes	no
3	12215000	7500	4	2	2	yes	no
4	11410000	7420	4	1	2	yes	yes
...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	yes	no
541	1767150	2400	3	1	1	no	no
542	1750000	3620	2	1	1	yes	no
543	1750000	2910	3	1	1	no	no
544	1750000	3850	3	1	2	yes	no

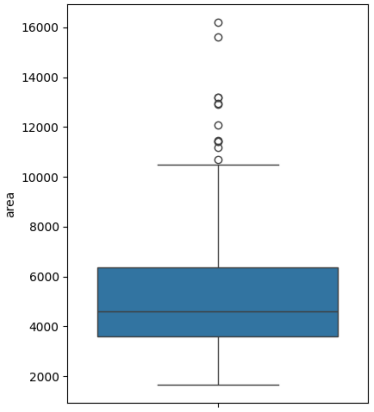
545 rows × 13 columns

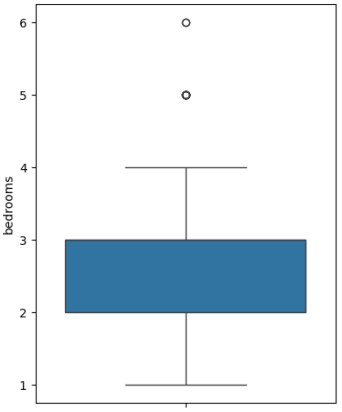
```
k= df.columns
k
```

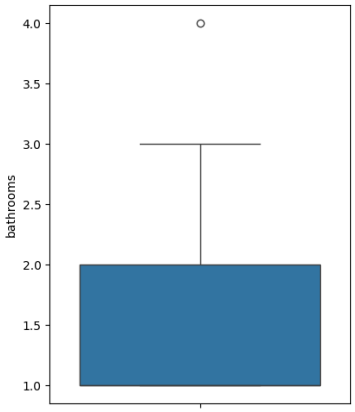
```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

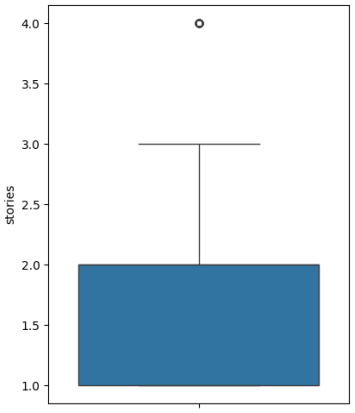
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(18,15))
n=0
for i in df[['price', 'area', 'bedrooms', 'bathrooms', 'stories','parking']]:
      n+=1
      plt.subplot(2,3,n)
      plt.subplots_adjust(hspace=0.5,wspace=0.5)
      sns.boxplot(df['{}'.format(i)])
```

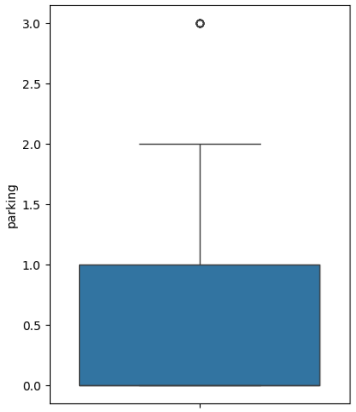






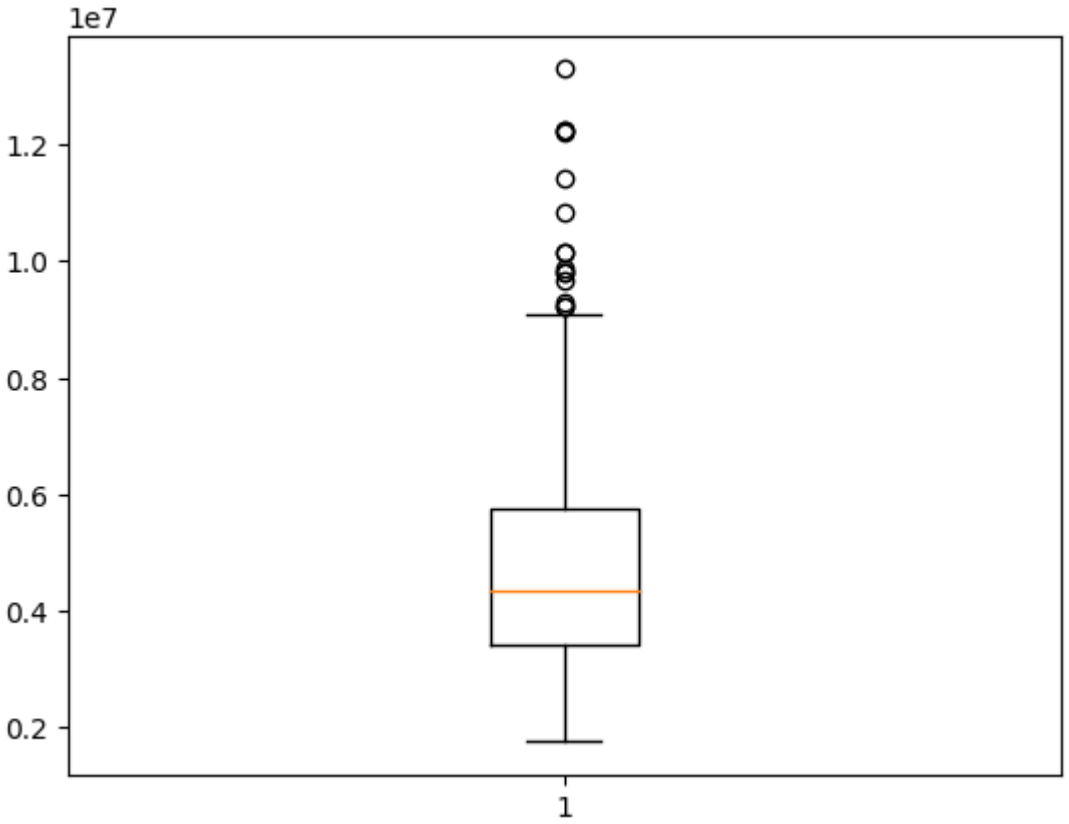




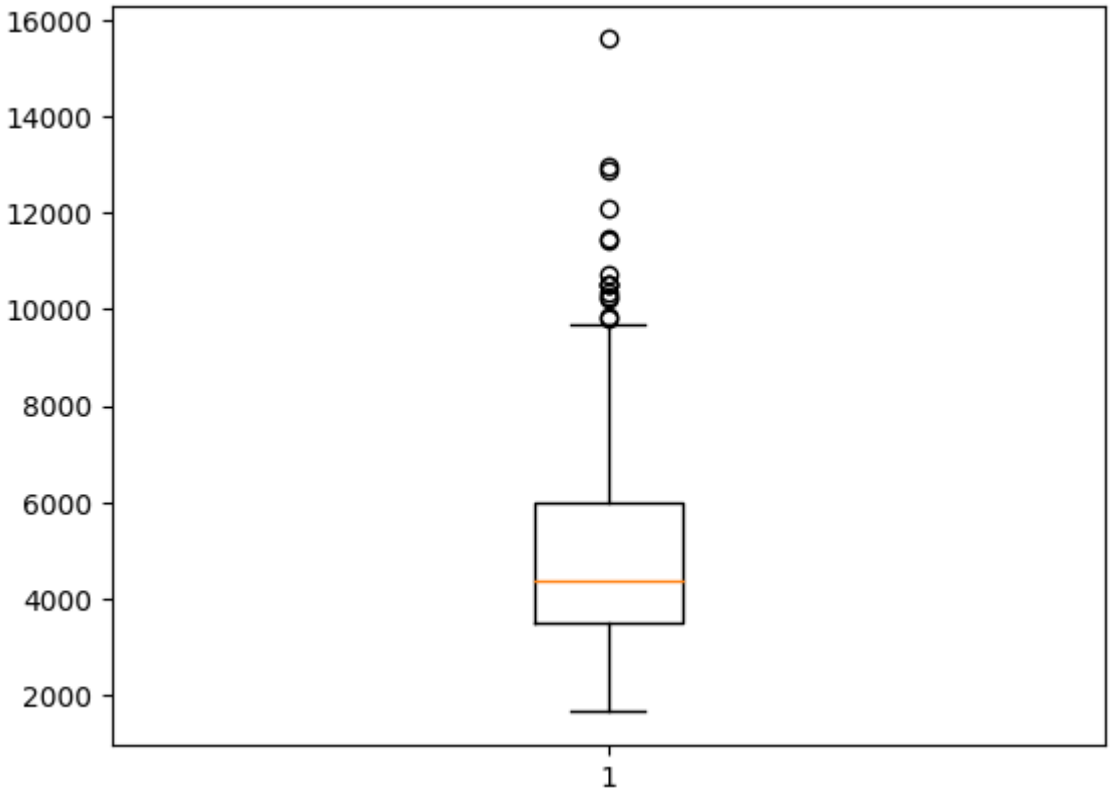



```
plt.boxplot(df.price)
q1=df.price.quantile(0.25)
q2=df.price.quantile(0.75)
iqr=q2-q1
```

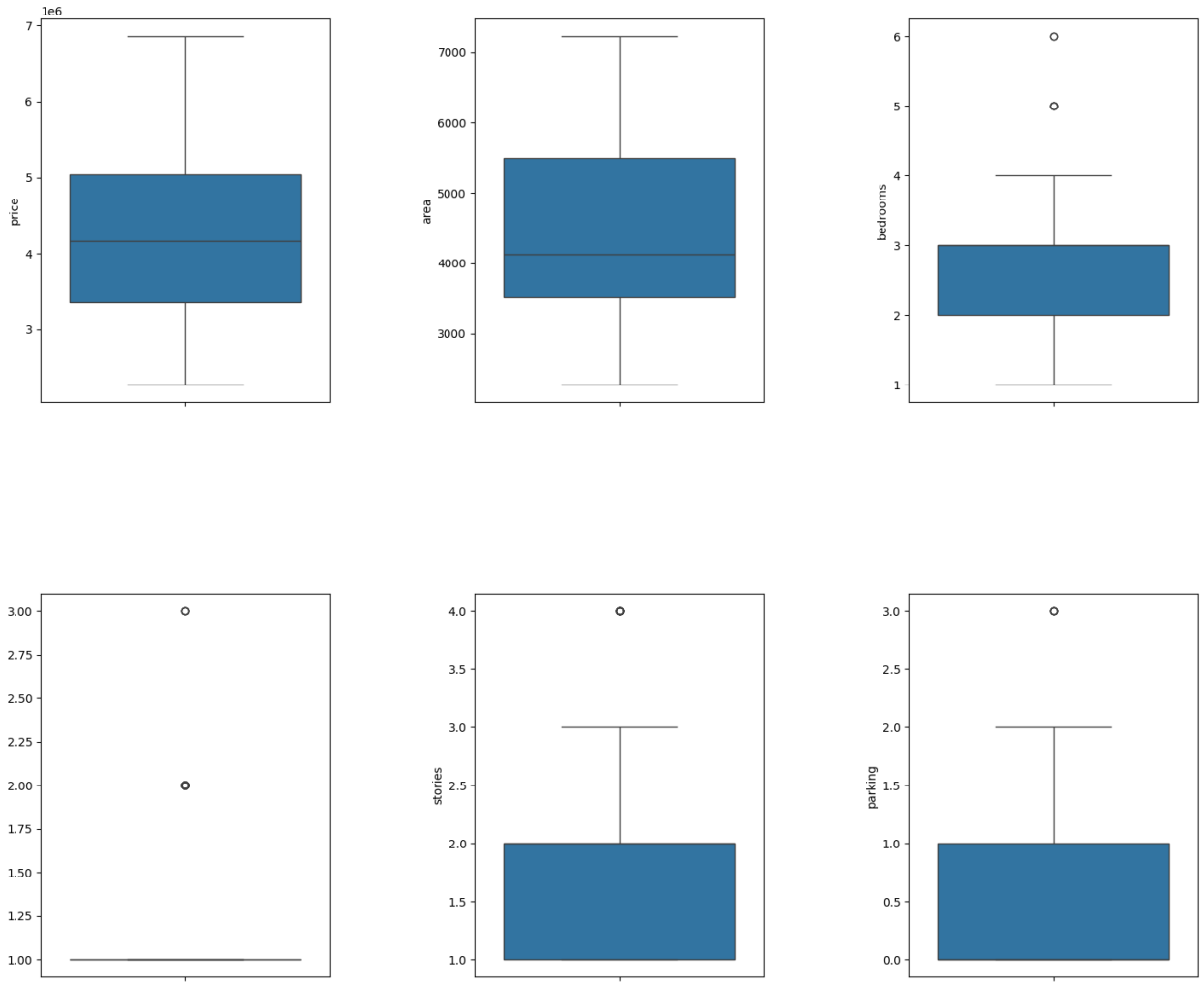
```
df=df[(df.price>=q2-iqr*1.5) & (df.price<=q1+iqr*1.5)]
```




```
plt.boxplot(df.area)
q1=df.area.quantile(0.25)
q2=df.area.quantile(0.75)
iqr=q2-q1
df=df[(df.area>=q2-iqr*1.5) & (df.area<=q1+iqr*1.5)]
```



```
plt.figure(figsize=(18,15))
n=0
for i in df[['price', 'area', 'bedrooms', 'bathrooms', 'stories','parking']]:
        n+=1
        plt.subplot(2,3,n)
        plt.subplots_adjust(hspace=0.5,wspace=0.5)
        sns.boxplot(df['{}'.format(i)])
```

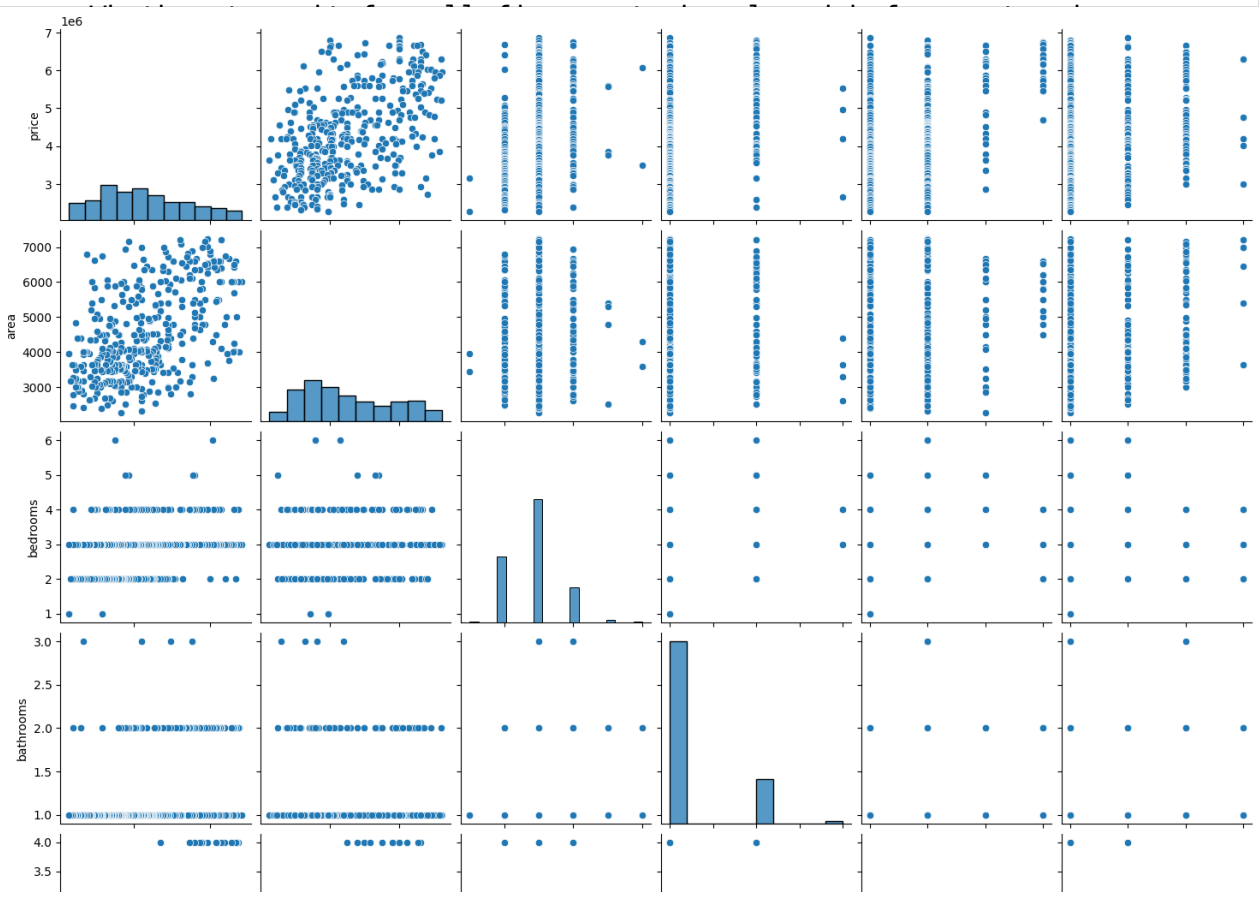


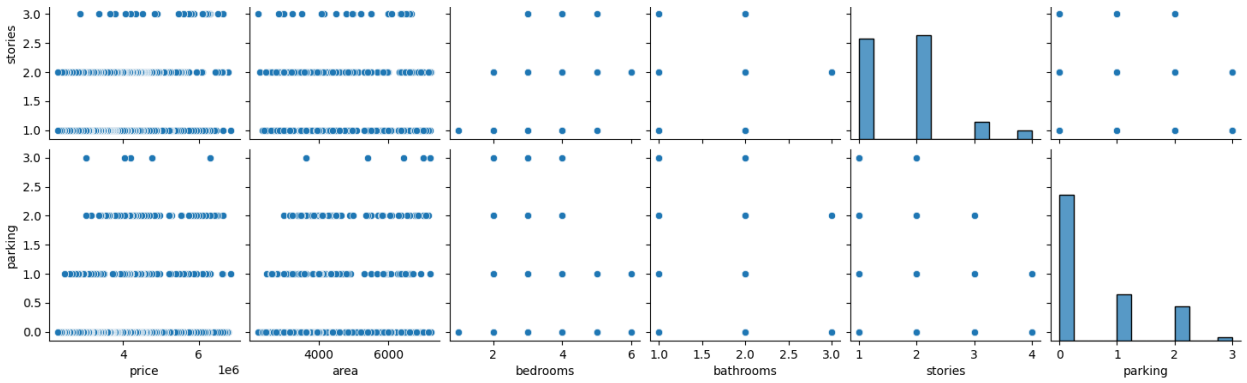
```
sns.pairplot(df)
plt.show
```

matplotlib.pyplot.show
def show(*args, **kwargs)

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py
Display all open figures.

Parameters
----
block : bool, optional




stories
parking
price
1e6
area
bedrooms
bathrooms
stories
parking

```
x=['mainroad','guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'p
mapping={'yes':1,'no':0}
df[x]=df[x].applymap(lambda x: mapping.get(x))
```

```
df.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	baseme
68	6860000	6000	3	1	1	1	0	
70	6790000	4000	3	2	2	1	0	
71	6755000	6000	4	2	4	1	0	
72	6720000	5020	3	1	4	1	0	
73	6685000	6600	2	2	4	1	0	

Next steps: Generate code with df View recommended plots

```
x=['furnishingstatus']
mapping={'furnished':2,'semi-furnished':1,'unfurnished':0}
df[x]=df[x].applymap(lambda x: mapping.get(x))

df.head(5)
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	baseme
68	6860000	6000	3	1	1	1	0	
70	6790000	4000	3	2	2	1	0	
71	6755000	6000	4	2	4	1	0	
72	6720000	5020	3	1	4	1	0	
73	6685000	6600	2	2	4	1	0	

Next steps: Generate code with df View recommended plots

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

import numpy as np

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

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.8, random_

df.shape
```

```
.
```

```
(397, 13)
```

```
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred= regressor.predict(x_test)
```

```
print('Train Score: ', regressor.score(x_train, y_train))
print('Test Score: ', regressor.score(x_test, y_test))
```

```
Train Score:  0.6808007036159243
Test Score:  0.513516874279074
```

Start coding or generate with AI.