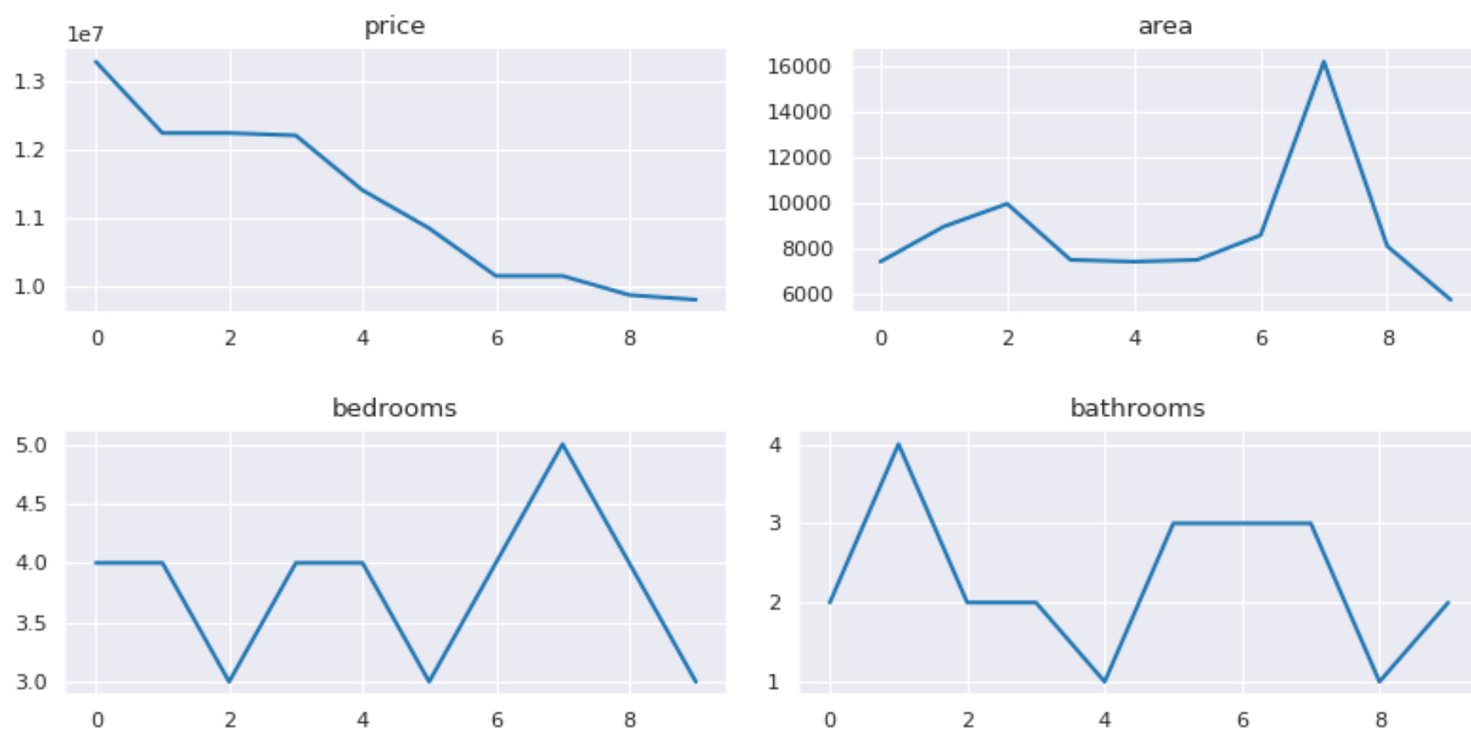
```
#House Price Prediction Task 1
#Importing the required libraries :

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

```
#Importing the dataset :
df = pd.read_csv(r"/content/Housing.csv")
```

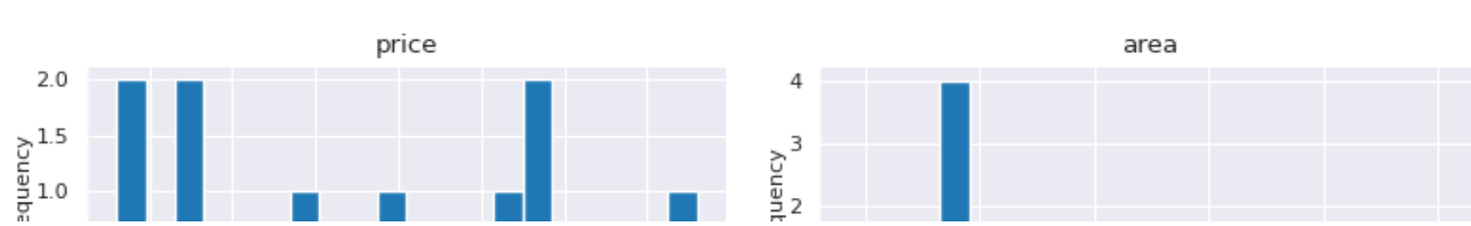
```
# FIRST 10 ROWS OF THE DATASET

df.head(10)
```

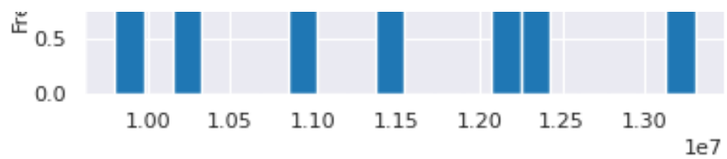
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	aircondit
0	13300000	7420	4	2	3	1	0	0	0	
1	12250000	8960	4	4	4	1	0	0	0	
2	12250000	9960	3	2	2	1	0	1	0	
3	12215000	7500	4	2	2	1	0	1	0	
4	11410000	7420	4	1	2	1	1	1	0	
5	10850000	7500	3	3	1	1	0	1	0	
6	10150000	8580	4	3	4	1	0	0	0	
7	10150000	16200	5	3	2	1	0	0	0	
8	9870000	8100	4	1	2	1	1	1	0	
9	9800000	5750	3	2	4	1	1	0	0	

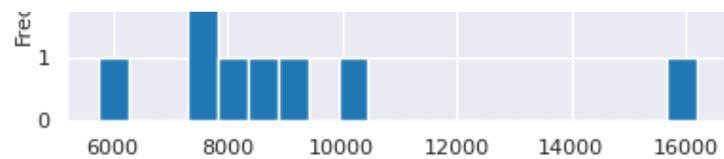
Values

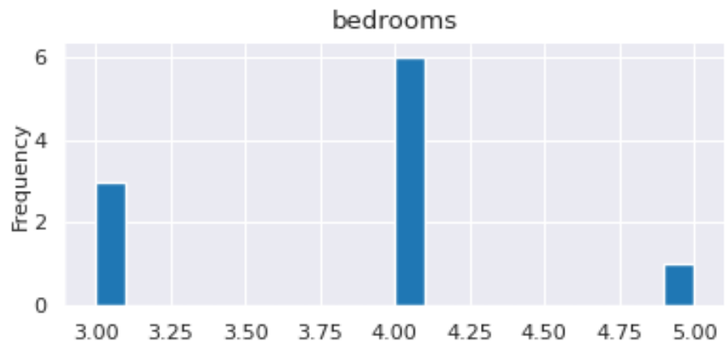


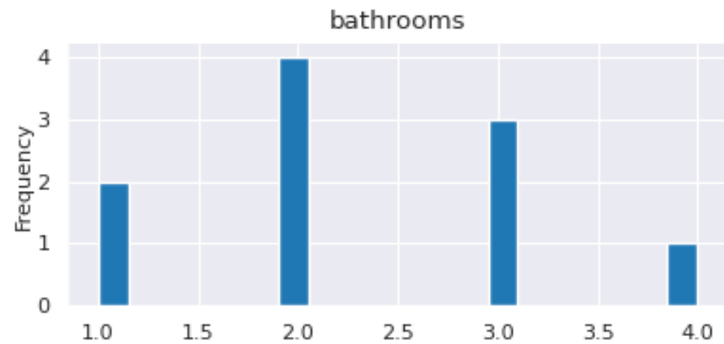
Distributions



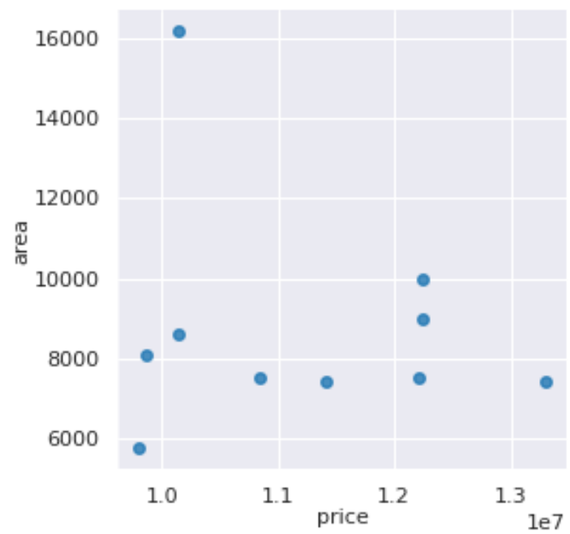


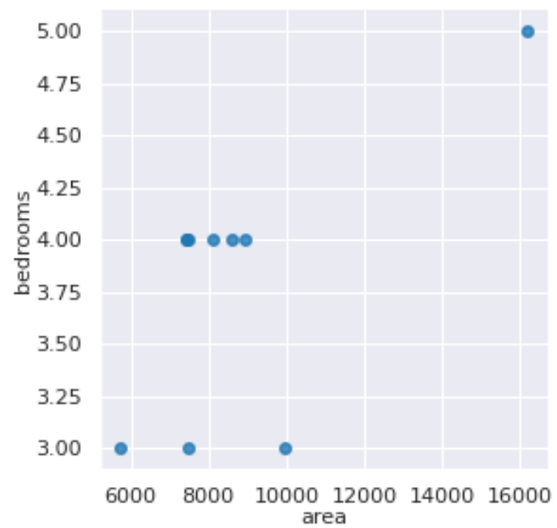


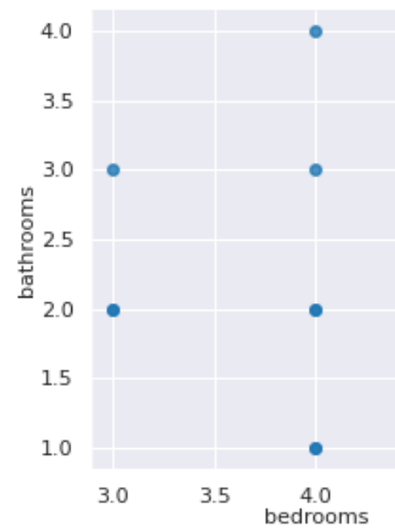




2-d distributions







Time series

```
# COLUMNS :

df.columns
```

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

```
# SIZE OF THE DATASET

df.shape
```

```
(545, 13)
```

```
# DATA TYPES OF THE COLUMNS OF THE DATASET

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
```

```
#Data Preprocessing :
# CHECKING FOR NULL VALUES

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

```
price              0
area               0
bedrooms           0
bathrooms          0
stories            0
mainroad           0
guestroom          0
basement           0
hotwaterheating    0
airconditioning    0
parking            0
prefarea           0
furnishingstatus   0
dtype: int64
```

```
# REMOVAL OF DUPLICATE VALUE
```

```
counter = 0
rs,cs = df.shape

df.drop_duplicates(inplace=True)

if df.shape==(rs,cs):
  print('\n\033[1mInference:\033[0m The dataset doesn\'t have any duplicates')
else:
  print(f'\n\033[1mInference:\033[0m Number of duplicates dropped/fixed ---> {rs-df.shape[0]}')
```

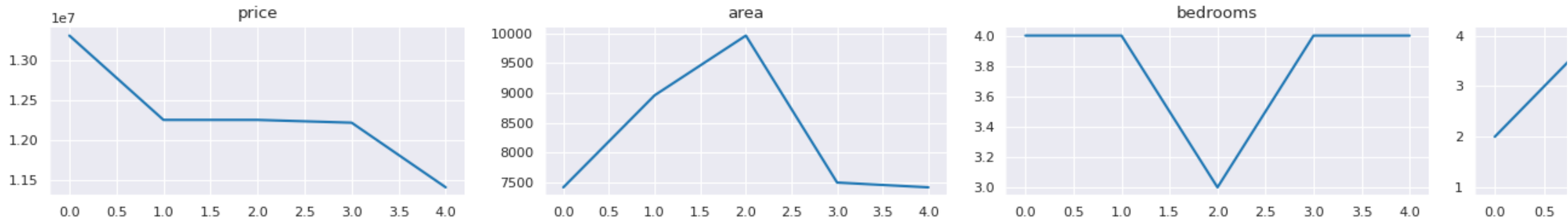
Inference: The dataset doesn't have any duplicates

```
# CONVERTING ALL OUR CATEGORICAL DATA COLUMNS TO NUMERIC FORM

from sklearn.preprocessing import LabelEncoder
categ = ["mainroad","guestroom","basement","hotwaterheating","airconditioning","prefarea","furnishingstatus"]

# Encode Categorical Columns
le = LabelEncoder()
df[categ] = df[categ].apply(le.fit_transform)
```

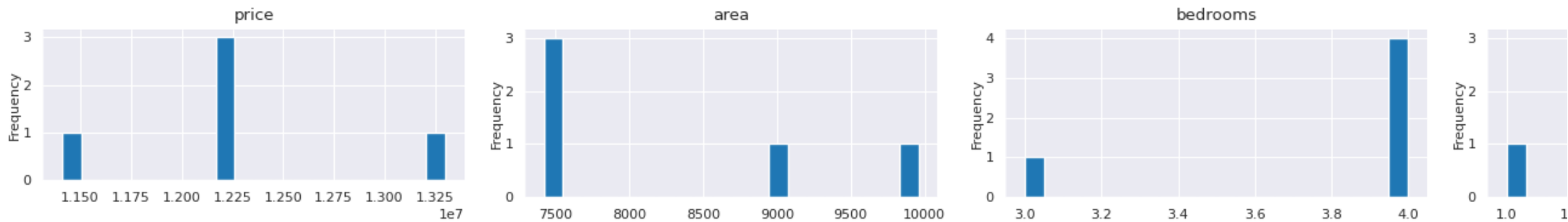
```
# AFTER CONVERTING OUR DATASET LOOKS LIKE THIS ...

df.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	0
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	0
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	1
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	0
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	0

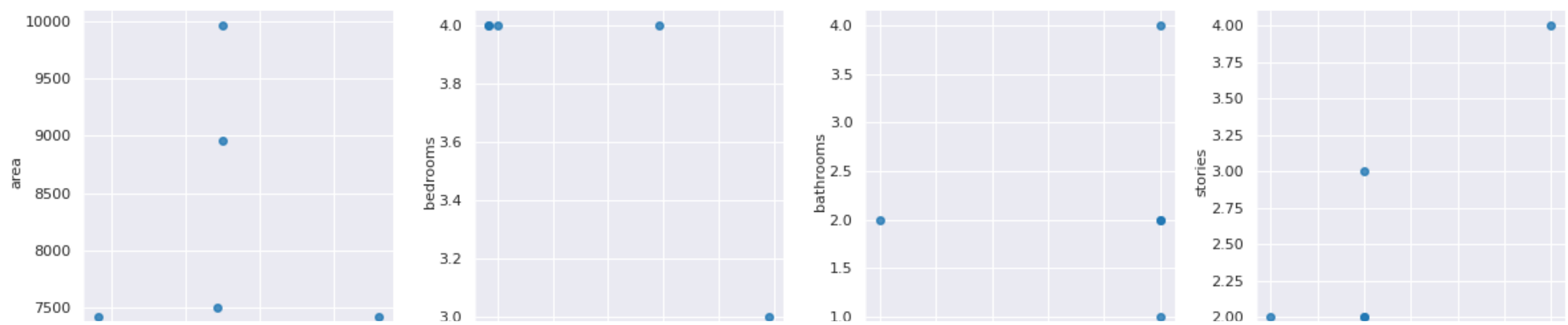
Values

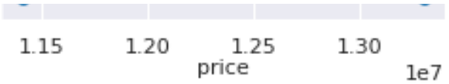


Distributions

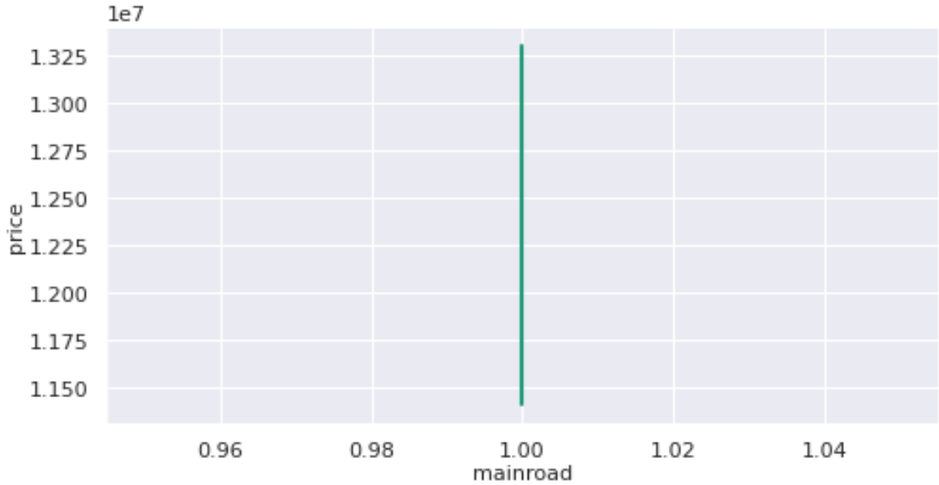


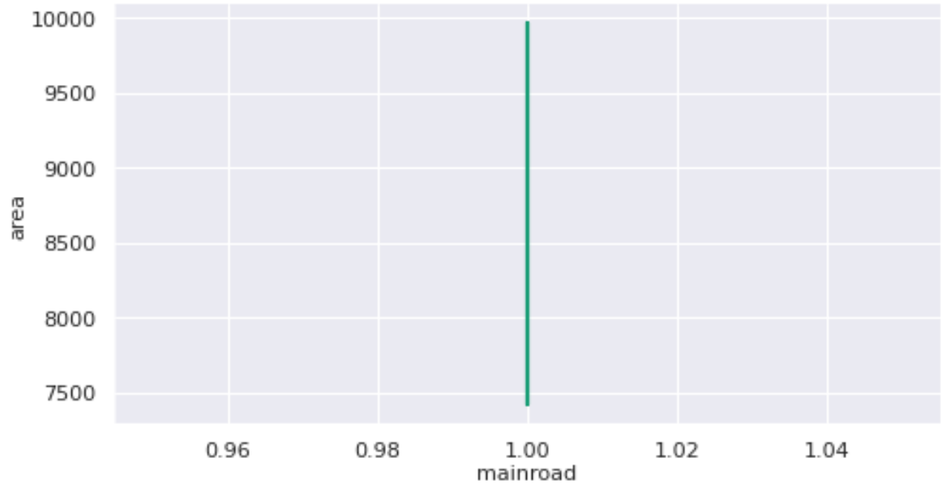
2-d distributions

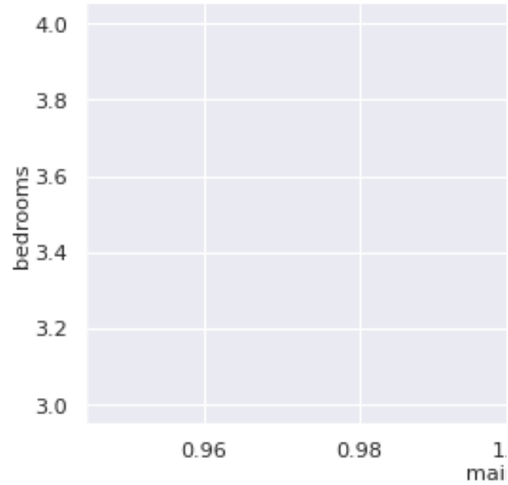


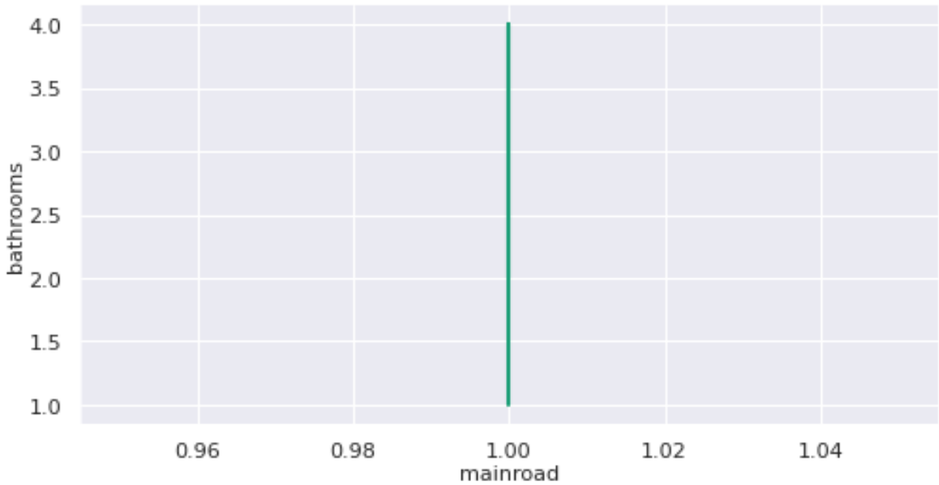


Time series



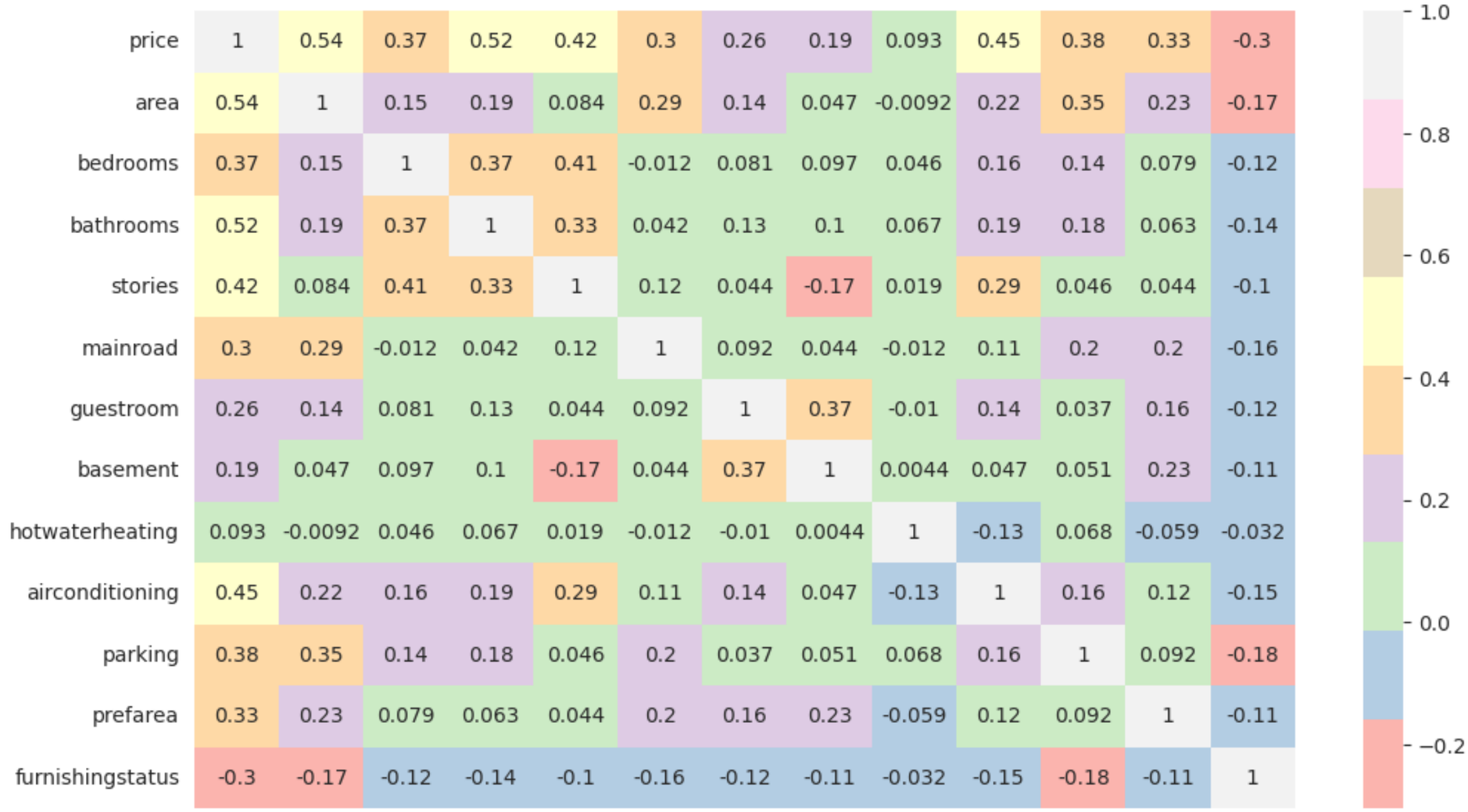







```
# CORRELATION BETWEEN THE COLUMNS

corr = df.corr()
plt.figure(figsize=(12,7))
sns.heatmap(corr,cmap='Pastel1',annot=True)
```

```
<Axes: >
```



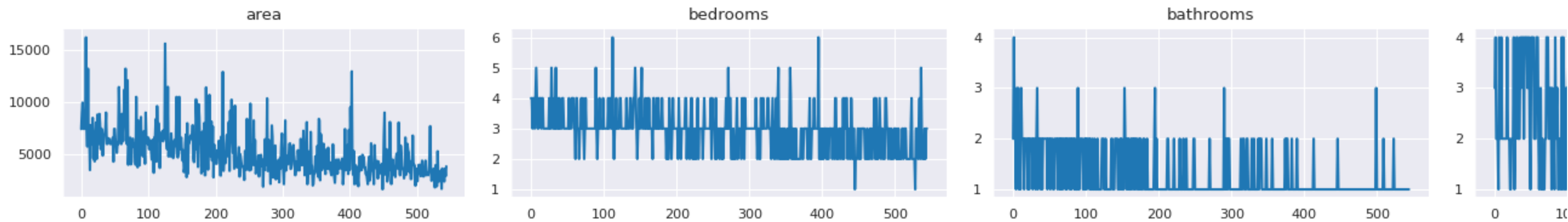

```
#Spliting The Dataset :
X = df.drop(['price'],axis=1)
y = df['price']
```

```
X
```

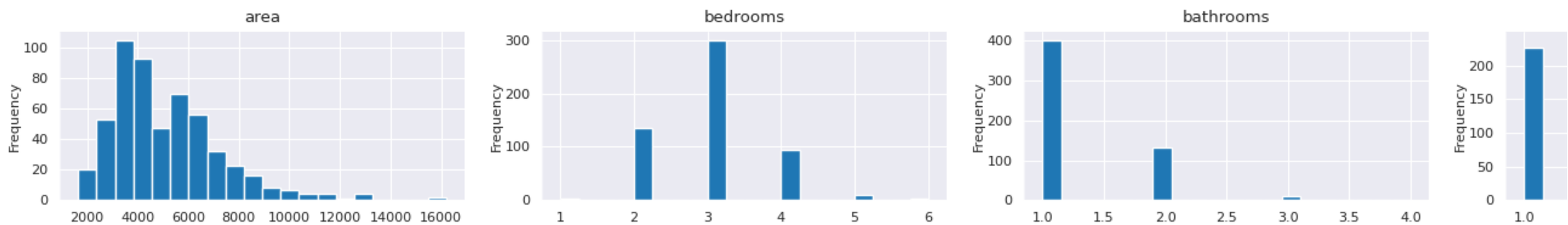
	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	7420	4	2	3	1	0	0	0	1	2	1	0
1	8960	4	4	4	1	0	0	0	1	3	0	0
2	9960	3	2	2	1	0	1	0	0	2	1	1
3	7500	4	2	2	1	0	1	0	1	3	1	0
4	7420	4	1	2	1	1	1	0	1	2	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
540	3000	2	1	1	1	0	1	0	0	2	0	2
541	2400	3	1	1	0	0	0	0	0	0	0	1
542	3620	2	1	1	1	0	0	0	0	0	0	2
543	2910	3	1	1	0	0	0	0	0	0	0	0
544	3850	3	1	2	1	0	0	0	0	0	0	2

545 rows × 12 columns

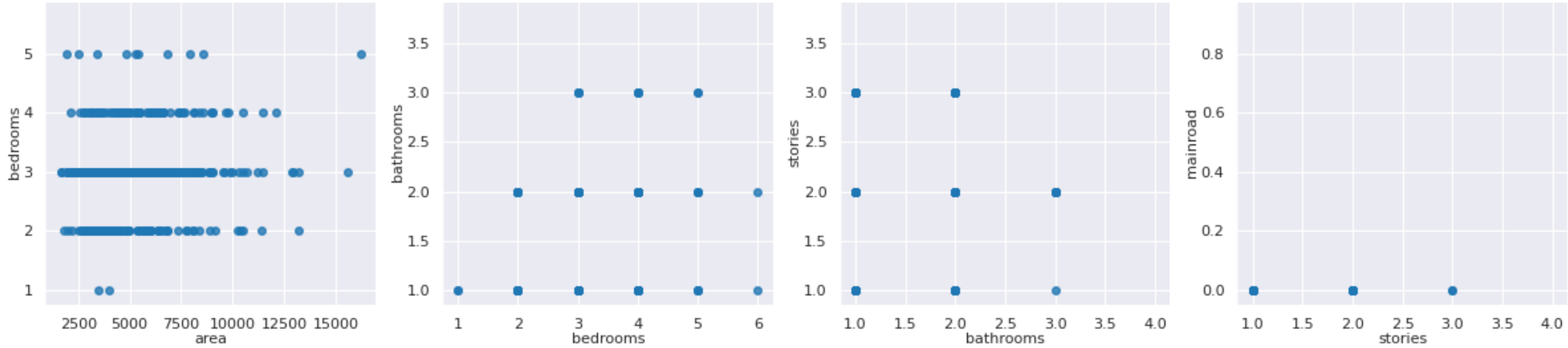
Values



Distributions



2-d distributions




```
y
```

```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

```
# LENGTH OF X_train AND X_test

len(X_train),len(X_test)
```

```
(381, 164)
```

```
#Linear Regression :

# IMPORTING THE MODULE

from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

```
# FITTING THE DATA INTO THE MODEL

model.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# PREDICTING THE OUTCOMES

y_predict = model.predict(X_test)
```

```
y_predict
```

```
array([5407508.87024418, 7097185.46706855, 3055462.44314053,
       4476945.19636315, 3315983.65663579, 3618373.03255259,
       5758111.46044028, 6466502.43909126, 2830273.16469119,
       2588804.65810568, 9649589.31414054, 2830606.51113843,
       3048137.62898116, 3392779.60203048, 3823232.9673009 ,
       5358170.87034032, 2955016.41578148, 4836054.53230682,
       4603068.47740645, 3551464.60674927, 5625018.82657786,
       5796938.54363456, 2758483.74755246, 4873266.20950521,
       5600804.93370716, 7772078.63540938, 3381536.16270183,
       5370732.06725797, 8352665.9587942 , 3406110.06934798,
       6335677.41367624, 3427228.10570008, 6740746.88053743,
       4205633.93578768, 3624702.80095917, 5797171.46441145,
       5080025.13346592, 4386055.52335342, 3070137.54474224,
       4635050.40917587, 4743419.55702888, 3433682.48420934,
       7076940.4807988 , 4096598.07073102, 3741261.35302813,
       4308416.36745432, 6678982.6364043 , 4092649.04459023,
       3872211.05471678, 3687383.17722361, 7462374.91109114,
       2898324.62920572, 4501494.58592862, 4427073.78264695,
       3822457.27350851, 2641947.74706375, 7510739.56714418,
       2940944.89650582, 4246613.52617664, 2796696.15913661,
       5048338.5881557 , 3582935.78133612, 5255053.95042848,
       4255235.59619518, 4807607.51782258, 4711232.00096705,
       7260531.24221025, 3577762.50878024, 6285890.98464552,
       6318431.41494409, 4758297.02517639, 5066345.6283754 ,
```

```
       4660923.85280286, 8066891.0835348 , 3498363.85433617,
       5429714.24082061, 4058399.55113095, 4441034.13232771,
       4816316.06017677, 4040152.44618202, 7752648.28119992,
       4172277.59889827, 6705184.89500426, 5431743.8402273 ,
       2763801.26664556, 7145185.10824616, 2626368.70533511,
       3708769.99925747, 8020894.34927404, 8360086.48106035,
       3255839.86876679, 6099608.09004818, 3610311.65133947,
       3666255.62947034, 7909304.88416764, 4708264.41070268,
       5200345.04442612, 6430338.84553524, 4954759.03354751,
       5980566.29859787, 3959705.24306595, 6578815.16383058,
       3699022.87139188, 6086458.26574741, 5225872.35919378,
       4461872.63697472, 7065795.84374117, 6430992.76558986,
       6577488.25056947, 7181092.29436555, 7132469.27644317,
       4969951.60846033, 3988509.66451833, 3282579.03980776,
       4284449.11400227, 3307885.27264494, 2794018.09992771,
       2314211.92533476, 3639260.72788702, 4017775.19574567,
       3783639.23978665, 4997528.13318576, 3920039.36114687,
       4086116.99024392, 6051867.67437964, 2314854.63439944,
       5648580.65216443, 3388208.44694471, 8106887.379355  ,
       6945211.4619021 , 3290813.6187236 , 4943358.78455695,
       7021438.17367802, 6633943.09586481, 3257272.80697058,
       2346620.41017362, 3128074.82457597, 5602713.80008827,
       3052041.70375378, 3931847.49328003, 2574372.82187877,
       4011784.5084254 , 5232639.95388041, 4105734.9218601 ,
       4188942.50390193, 2421273.7970433 , 6767186.36585123,
       5302303.58194732, 6807894.55657294, 7246680.49265498,
       5468336.15502   , 4606361.04221867, 3769887.60312207,
       3860336.84228787, 3308041.40432699, 7268912.41470939,
       8146912.63449662, 4013563.22451009, 7210577.89436119,
       3796902.64702948, 4130262.35836178, 7058248.99034354,
       5417344.25448756, 5180833.66432886])
```

```
#Evaluation :
from sklearn.metrics import r2_score,mean_absolute_error
score = r2_score(y_test,y_predict)
mae = mean_absolute_error(y_test,y_predict)
```

```
score
```

```
0.6435419628959105
```

```
mae
```

```
925543.5483156566
```