untitled-checkpoint

May 24, 2024

1 House Price Analysis

1.0.1 Importing the necessary libraries.

[8]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

2 Loading and reading the dataset

[2]:
```
df = pd.read_csv("HousingData.csv")
```

[3]:
```
df.head()
```

[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	\
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	

	B	LSTAT	MEDV
0	396.90	4.98	24.0
1	396.90	9.14	21.6
2	392.83	4.03	34.7
3	394.63	2.94	33.4
4	396.90	NaN	36.2

[4]:
```
df.tail()
```

[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	\
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	\
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	

```
          B  LSTAT  MEDV
501  391.99    NaN  22.4
502  396.90   9.08  20.6
503  396.90   5.64  23.9
504  393.45   6.48  22.0
505  396.90   7.88  11.9
```

3 Checking the dataset information

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

[6]:
```
df.describe()
```

[6]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  486.000000  486.000000  486.000000  486.000000  506.000000  506.000000   \
mean     3.611874   11.211934   11.083992    0.069959    0.554695    6.284634   
std      8.720192   23.388876    6.835896    0.255340    0.115878    0.702617   
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000   
25%      0.081900    0.000000    5.190000    0.000000    0.449000    5.885500   
50%      0.253715    0.000000    9.690000    0.000000    0.538000    6.208500   
75%      3.560263   12.500000   18.100000    0.000000    0.624000    6.623500   
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000   
```

```
              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  486.000000  506.000000  506.000000  506.000000  506.000000  506.000000   \
mean    68.518519    3.795043    9.549407  408.237154   18.455534  356.674032   
std     27.999513    2.105710    8.707259  168.537116    2.164946   91.294864   
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000   
25%     45.175000    2.100175    4.000000  279.000000   17.400000  375.377500   
50%     76.800000    3.207450    5.000000  330.000000   19.050000  391.440000   
75%     93.975000    5.188425   24.000000  666.000000   20.200000  396.225000   
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000   

            LSTAT        MEDV  
count  486.000000  506.000000  
mean    12.715432   22.532806  
std      7.155871    9.197104  
min      1.730000    5.000000  
25%      7.125000   17.025000  
50%     11.430000   21.200000  
75%     16.955000   25.000000  
max     37.970000   50.000000  
```

[7]:
```
df.isnull().sum()
```

[7]:
```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

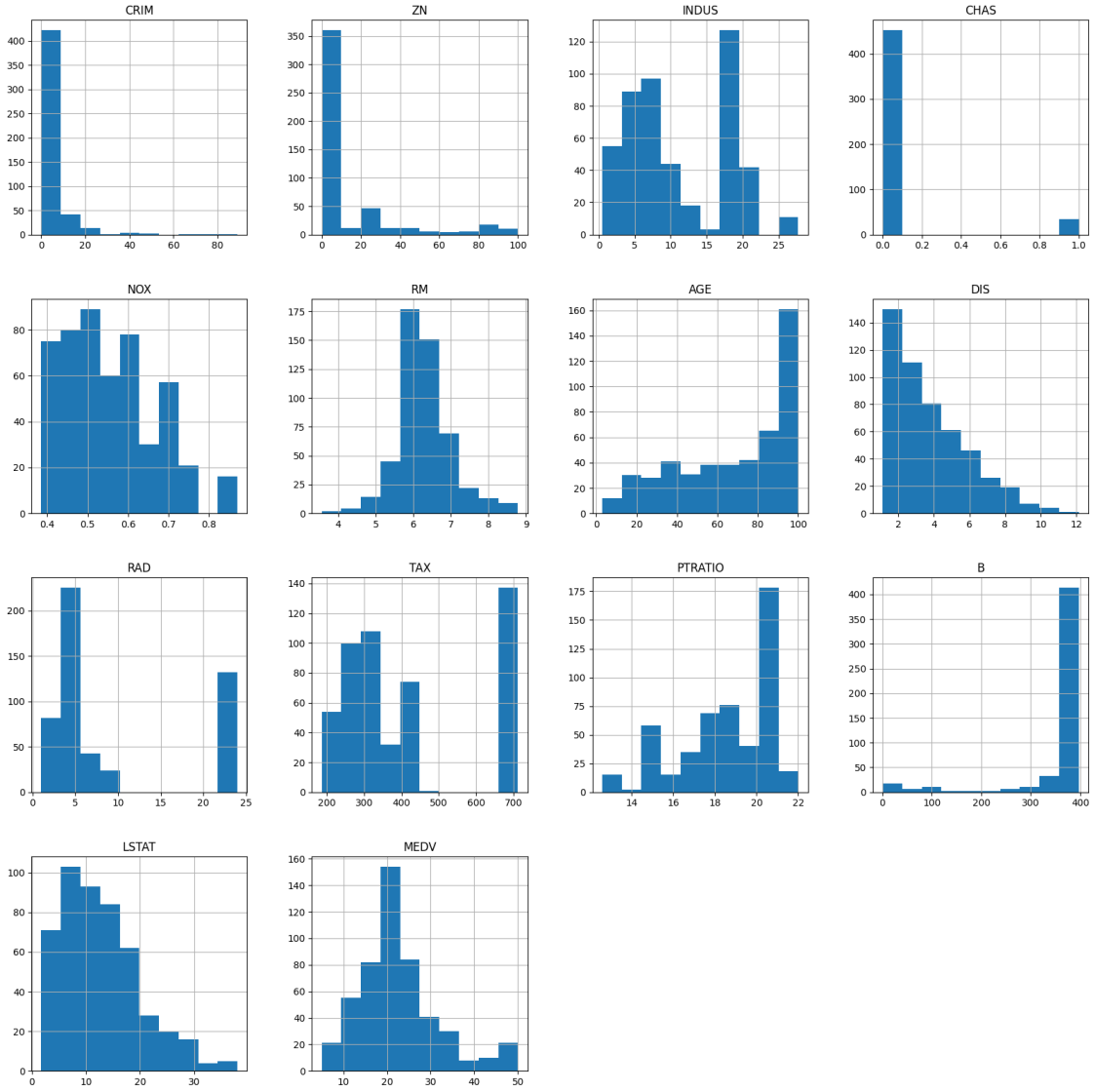
[10]:
```
df.duplicated().sum()
```

[10]:
```
0
```

4 Data Visualization

[11]:
```
df.hist(layout=(4,4),figsize=(20,20))
```

```
[11]: array([[<Axes: title={'center': 'CRIM'}>, <Axes: title={'center': 'ZN'}>,
              <Axes: title={'center': 'INDUS'}>,
              <Axes: title={'center': 'CHAS'}>],
             [<Axes: title={'center': 'NOX'}>, <Axes: title={'center': 'RM'}>,
              <Axes: title={'center': 'AGE'}>, <Axes: title={'center': 'DIS'}>],
             [<Axes: title={'center': 'RAD'}>, <Axes: title={'center': 'TAX'}>,
              <Axes: title={'center': 'PTRATIO'}>,
              <Axes: title={'center': 'B'}>],
             [<Axes: title={'center': 'LSTAT'}>,
              <Axes: title={'center': 'MEDV'}>, <Axes: >, <Axes: >]],
            dtype=object)
```



[17]:
```
from sklearn.utils import resample
df_majority = df[(df['CHAS']==0)]
df_minority = df[(df['CHAS']==1)]
df_minority_upsampled = resample(df_minority,
                                 replace = True,
                                 n_samples = 350,
                                 random_state = 42)
df = pd.concat([df_minority_upsampled, df_majority])
df.head()
```

[17]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
358	5.20177	0.0	18.10	1.0	0.770	6.127	83.4	2.7227	24	666	
220	0.35809	0.0	6.20	1.0	0.507	6.951	88.5	2.8617	8	307	
209	0.43571	0.0	10.59	1.0	0.489	5.344	100.0	3.8750	4	277	
273	0.22188	20.0	6.96	1.0	0.464	7.691	51.8	4.3665	3	223	
236	NaN	0.0	6.20	1.0	0.507	6.631	76.5	4.1480	8	307	

	PTRATIO	B	LSTAT	MEDV
358	20.2	395.43	11.48	22.7
220	17.4	391.70	9.71	26.7
209	18.6	396.90	23.09	20.0
273	18.6	390.77	6.58	35.2
236	17.4	388.45	9.54	25.1

[18]:
```
from sklearn.utils import resample
df_majority = df[(df['ZN']<10)]
df_minority = df[(df['ZN']>10)]
df_minority_upsampled = resample(df_minority,
                                 replace = True,
                                 n_samples = 350,
                                 random_state = 42)
df = pd.concat([df_minority_upsampled, df_minority])
df.head()
```

[18]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
61	0.17171	25.0	5.13	0.0	0.453	5.966	93.4	6.8185	8	284	
299	0.05561	70.0	2.24	0.0	0.400	7.041	10.0	7.8278	5	358	
51	0.04337	21.0	NaN	0.0	0.439	6.115	63.0	6.8147	4	243	
276	0.10469	40.0	6.41	1.0	0.447	7.267	49.0	4.7872	4	254	
65	0.03584	80.0	3.37	0.0	0.398	6.290	17.8	6.6115	4	337	

	PTRATIO	B	LSTAT	MEDV
61	19.7	378.08	14.44	16.0
299	14.8	371.58	4.74	29.0
51	16.8	393.97	9.43	20.5
276	17.6	389.25	6.05	33.2
65	16.1	396.90	4.67	23.5

5 Finding the factors affecting the price of various region

[23]:
```
plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),cmap = "YlGnBu",annot=True)
plt.show()
```

[24]:
```
df = df.drop(["LSTAT"],axis =1)
```

[25]:
```
df['MEDV'].unique()
```

[25]:
```
array([16. , 29. , 20.5, 33.2, 23.5, 32.4, 23.9, 33.1, 30.5, 18.9, 20.6,
       23.3, 22.2, 36. , 32.9, 44. , 36.5, 20.7, 34.9, 24.5, 32.7, 43.5,
       46. , 21.7, 32.2, 31.2, 24.8, 19.4, 50. , 35.1, 24.1, 30.8, 24. ,
       19.7, 20.9, 20.1, 24.4, 35.2, 48.5, 33. , 23.1, 21.1, 23.4, 33.8,
       31.1, 31. , 34.6, 23.7, 25.2, 36.4, 31.6, 26.6, 19.6, 22.5, 22. ,
       17.1, 16.5, 33.4, 30.3, 37. , 25. , 43.1, 32. , 28.2, 28.4, 15. ,
       29.6, 36.1, 24.7, 26.2, 35.4, 42.8, 18.6, 45.4, 27.9, 22.8, 28. ,
       17.6, 18.2, 29.1, 30.1, 17.4, 28.5, 22.9, 29.8, 26.4, 27.1, 18.7,
```

```
       33.3, 42.3, 24.3, 48.8, 23.2, 22.3, 37.3])
```

[26]:
```
df["MEDV"].value_counts()
```

[26]:
```
MEDV
50.0    54
33.1    39
33.2    34
32.4    34
46.0    28
        ..
24.3     1
48.8     1
23.2     1
22.3     1
37.3     1
Name: count, Length: 95, dtype: int64
```

[27]:
```
df["MEDV"].mean()
```

[27]:
```
31.753199268738573
```

[31]:
```
from xgboost import XGBRegressor
from sklearn import metrics as me
from sklearn.model_selection import train_test_split
```

[32]:
```
x = df.drop("MEDV",axis=1)
y=df["MEDV"]
```

[33]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=0.
↪2)
```

[34]:
```
model=XGBRegressor()
model.fit(x_train,y_train)
```

[34]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, …)
```

[36]:
```
y_predt = model.predict(x_test)
```

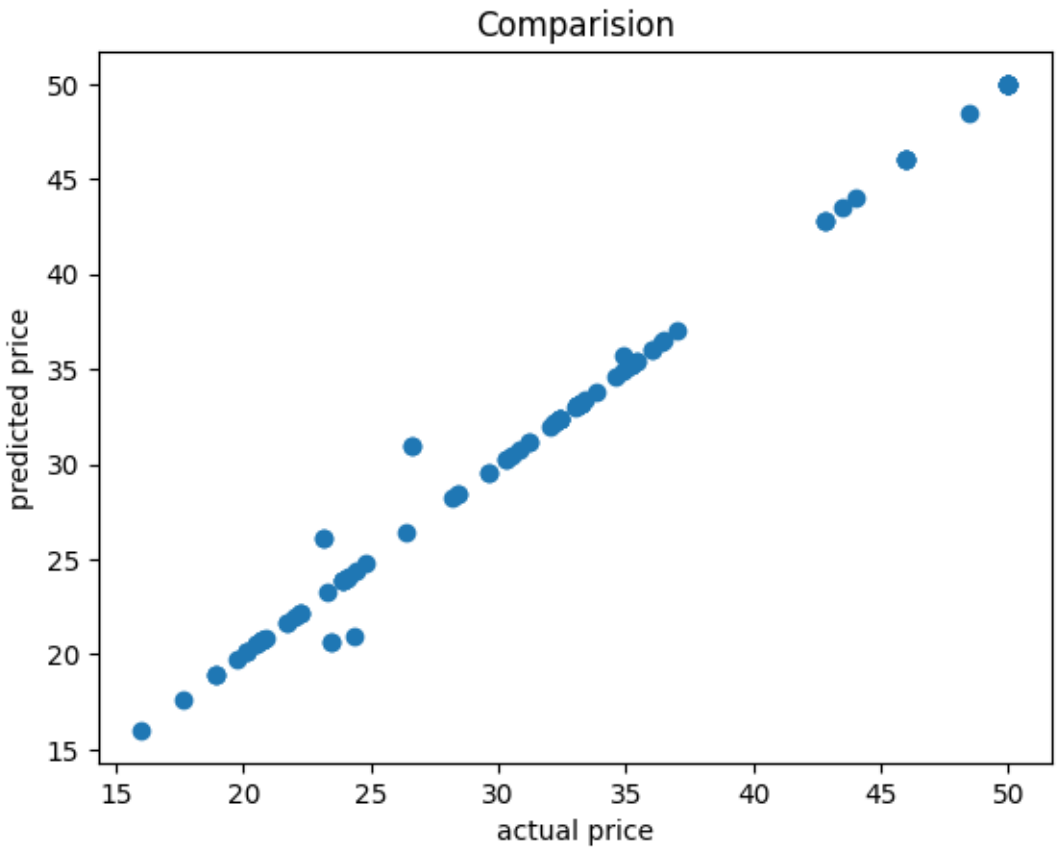
[37]:
```
r2_score=me.r2_score(y_test,y_predt)
print(r2_score)
```

```
0.9912446471508071
```

[39]:
```
mean_error = me.mean_absolute_error(y_test,y_predt)
print(mean_error)
```

```
0.22224800109863307
```

[40]:
```
plt.scatter(y_test,y_predt)
plt.xlabel("actual price")
plt.ylabel("predicted price")
plt.title("Comparision")
plt.show()
```



[ ]:
```

```