Boston Housing price prediction

Submission #2

Group6-Insightful 5

PRABHJOT KAUR #0779539,

HARSHDEEP JARNAIL SINGH THINDE #0775726,

JASKARAN SINGH #0781563,

GURJIT KAUR #0776146

KIRANPREET KAUR #0788217

1. Data cleaning and Transformation

1.1 Data cleaning

Data cleaning is the process of preparing raw data for analysis by removing duplicate data, incomplete raw data, and filling in the null values. For data cleaning, we will address and resolve issues identified during the EDA phase of descriptive statistics using PYTHON.

1.1.1 Missing values

Data summarization is the best way to identify the null and missing values and any another interesting fact about the dataset.

```
# Let's summarize the data to see the distribution of data
        print(data.describe())
```

```
              CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   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  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT
count  506.000000
mean    12.653063
std      7.141062
min      1.730000
25%      6.950000
50%     11.360000
75%     16.955000
max     37.970000
```

There are no missing values as showed from the data summary.

1.1.2 Null Values

Null values across all columns Can be determined using isnull() and sum() functions.

```
In [13]: boston.isnull().sum()

Out[13]: CRIM       0
         ZN         0
         INDUS      0
         CHAS       0
         NOX        0
         RM         0
         AGE        0
         DIS        0
         RAD        0
         TAX        0
         PTRATIO    0
         B          0
         LSTAT      0
         MEDV       0
         dtype: int64
```

There is no null values for all columns.

1.1.3 Duplicate Values

Duplicate values across all columns can be determined using nunique() function.

```
In [16]: boston.nunique()

Out[16]: CRIME_RATE              504
         ZONED                    26
         PROPORTION               76
         CHARLES_RIVER             2
         NITRIC_OXIDES            81
         AVERAGE_ROOMS           446
         AGE                     356
         WEIGHTED_DISTANCE       412
         RADIAL_HIGHWAY            9
         PROPERTY-TAX             66
         PUPIL-TEACHER_RATIO      46
         PROPORTION_OF_BLACKS    357
         POPULATION              455
         MEDV                    229
         dtype: int64
```

There is no null values for any variables.

1.2 Data Transformation

Data transformation is the process of converting data from one format to another such as merging data into a single data Frame or renaming columns.

1.2.1 Data Merge

Merging data into a single data frame as following-

```
In [7]: from sklearn.datasets import load_boston
        boston = load_boston()
        data = pd.DataFrame(boston.data)
        data.head()
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

1.2.2 Renaming Column

```
In [15]: boston.columns=["CRIME_RATE","ZONED","PROPORTION","CHARLES_RIVER","NITRIC_OXIDES","AVERAGE_ROOMS","AGE","WEIGHTED_DISTANCE","RADI
         boston.head()
```

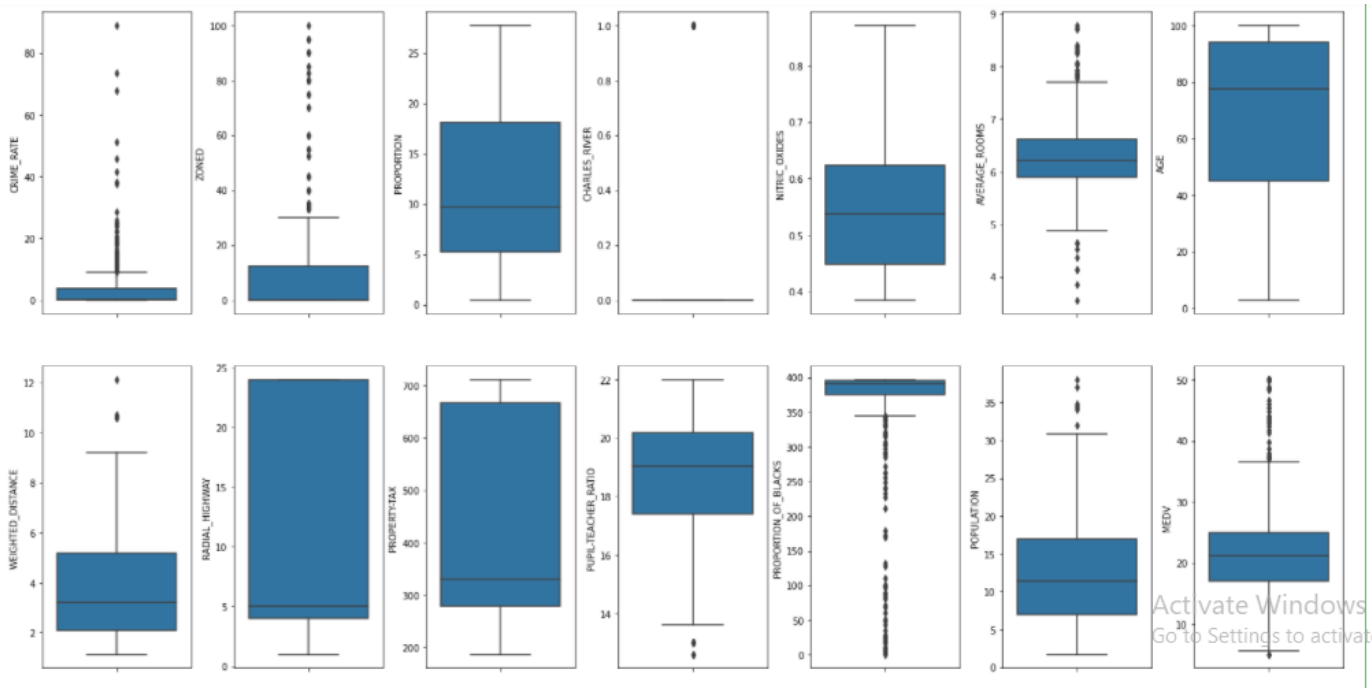
Out[15]:

	CRIME_RATE	ZONED	PROPORTION	CHARLES_RIVER	NITRIC_OXIDES	AVERAGE_ROOMS	AGE	WEIGHTED_DISTANCE	RADIAL_HIGHWAY	PROPERTY-TAX	T
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

1.2.3 Outliers

An **outlier** is an observation that lies an abnormal distance from other values in a random sample from a population. In order to perform accurate analysis, it is important to detect and remove outliers from the dataset.

Let's plot the boxplots for detecting outliers for all variables and check for outliers.



Columns like MEDV, CRIM, ZN, RM, B seems to have outliers as seen from the boxplots for all variables .Let's calculate the percentage of outliers present in every variable and remove the outliers with more than 10 % presence in the variables.

In [19]:
```
for k, v in data.items():
        q1 = v.quantile(0.25)
        q3 = v.quantile(0.75)
        irq = q3 - q1
        v_col = v[(v <= q1 - 1.5 * irq) | (v >= q3 + 1.5 * irq)]
        perc = np.shape(v_col)[0] * 100.0 / np.shape(boston)[0]
        print("Column %s outliers = %.2f%%" % (k, perc))
```

```
Column CRIM outliers = 13.04%
Column ZN outliers = 13.44%
Column INDUS outliers = 0.00%
Column CHAS outliers = 100.00%
Column NOX outliers = 0.00%
Column RM outliers = 5.93%
Column AGE outliers = 0.00%
Column DIS outliers = 0.99%
Column RAD outliers = 0.00%
Column TAX outliers = 0.00%
Column PTRATIO outliers = 2.96%
Column B outliers = 15.22%
Column LSTAT outliers = 1.38%
Column MEDV outliers = 7.91%
```

We need to remove response variable MEDV outliers (MEDV = 50.0) for better prediction .

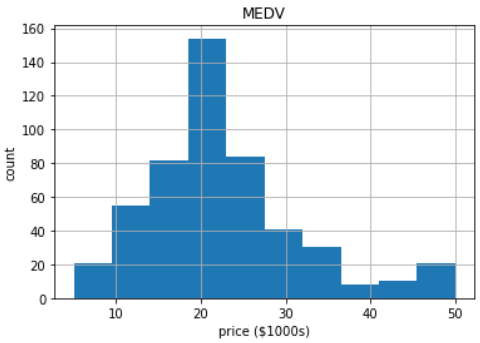
2. DATA ANALYSIS

This section deals with the answering the questions from the problem statement by Identifying the key patterns by using various required visualizations.

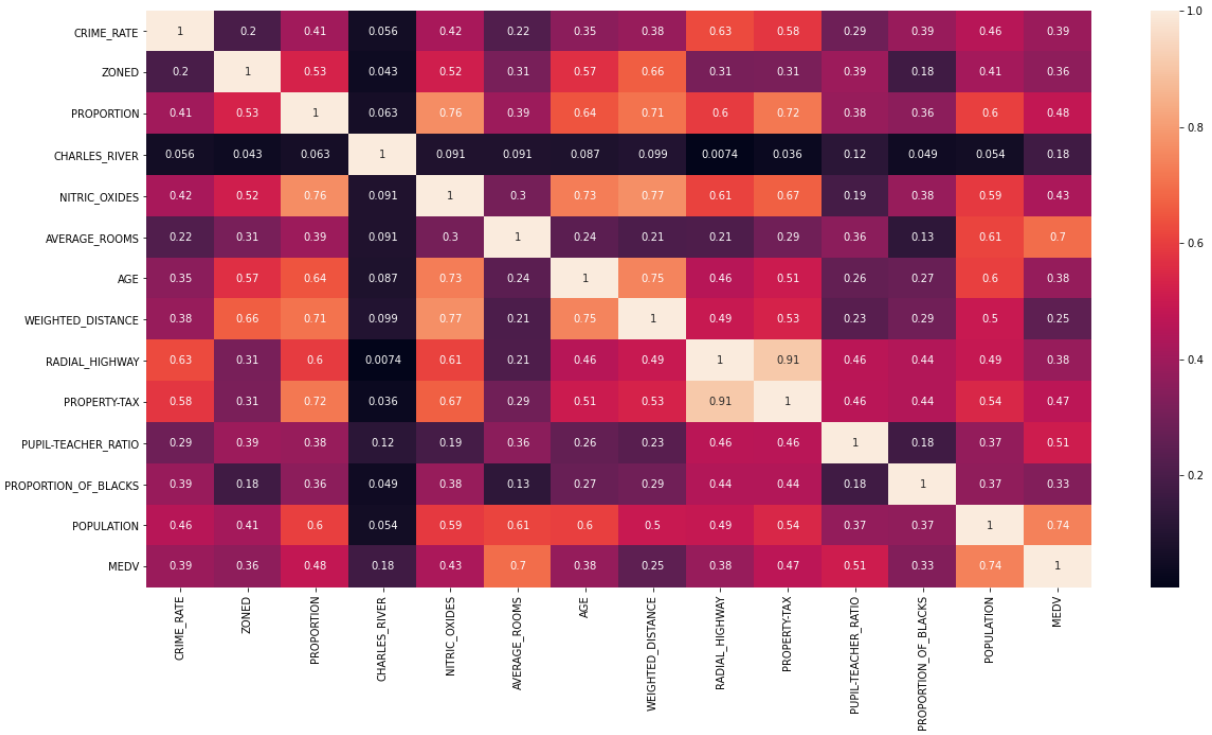
First step is create a histogram of the quantity median price MEDV to check the distribution pattern for the response variable.The distribution appears to be normally distributed after removing outliers as ween can see from the histogram plot.

```
In [28]: boston.hist(column='MEDV')
         plt.xlabel('price ($1000s)')
         plt.ylabel('count')

Out[28]: Text(0, 0.5, 'count')
```

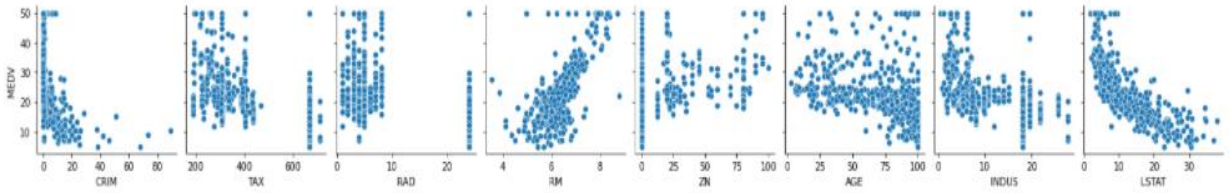


Another step for identifying key pattern is find the correlation of response variable with the variables by plotting the pairwise correlation on data and plotting an heatmap for correlation matrix for all variables which gives numerical representation of correlation values for all pairs of variables.



From correlation matrix, we see TAX and RAD are highly correlated features. The columns LSTAT, INDUS, RM, TAX, NOX, PTRAIO has a correlation score above 0.5 with MEDV which is a good indication of using as predictors. Let's plot these columns against MEDV.

```
In [49]: pp = sns.pairplot(data=data,
                           y_vars=['MEDV'],
                           x_vars=['CRIM', 'TAX', 'RAD','RM','ZN','AGE','INDUS','LSTAT'])
```



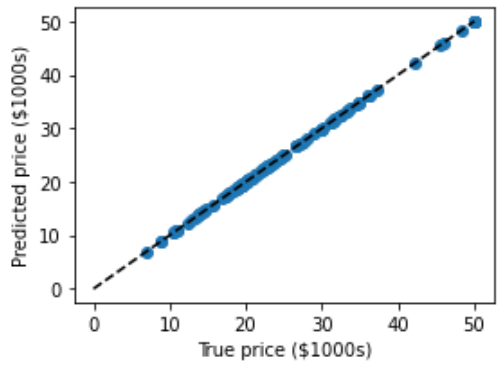
So with these analysis, we may try predict MEDV with 'LSTAT', 'INDUS', 'NOX', 'PTRATIO', 'RM', 'TAX', 'DIS', 'AGE' features

Now, next step is to make a predictive model using simple linear regression model and check the how accurate is model in predicting the prices.

```
In [54]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(boston, boston['MEDV'])

         from sklearn.linear_model import LinearRegression
         clf = LinearRegression()
         clf.fit(X_train, y_train)
         predicted = clf.predict(X_test)
         expected = y_test

         plt.figure(figsize=(4, 3))
         plt.scatter(expected, predicted)
         plt.plot([0, 50], [0, 50], '--k')
         plt.axis('tight')
         plt.xlabel('True price ($1000s)')
         plt.ylabel('Predicted price ($1000s)')
         plt.tight_layout()
```



As we can see from the plot of predicted price and true prices, there are hardly few residuals which is considered as the difference between the observed value of the dependent variable (y) and the predicted value (ŷ) and pattern occurs to be linear which concludes that model is efficient enough to predict the true values i.e median prices of the houses based on various variables.

3. Conclusions

After performing data cleaning and data transformation on the dataset for an efficient data analysis, we have been able to successfully answer many business problems related to the Boston house price prediction as such –

1. When looking for the house, clients looks after certain other things besides the price and location such as crime rate ratio of the location, proportion of residential land zoned for lots over 25,000 sq.ft., proportion of non-retail business acres per town, accessibility to highways from the location, population of the area (% lower status of the population-LSAT) , average number of rooms per dwelling and full-value property-tax rate per $10,000.

2. The model with the selected variables or factors was highly efficient in predicting the correct prices which help us in concluding that this model can be used by clients to help clients and sellers to determine the estimated prices while buying a property in Boston using certain factors features.