Boston House Price Prediction

By SABAL SHARMA

THE PROBLEM

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's data-set proves that much more influences price negotiations than the number of bedrooms or a white-picket fence

ABOUT THE DATASET

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. In this project, house prices will be predicted given explanatory variables that cover many aspects of residential houses. The goal of this project is to create a regression model that is able to accurately estimate the price of the house given the features. In this dataset made for predicting the Boston House Price Prediction. Here I just show the all of the feature for each house separately. Such as Number of Rooms, Crime rate of the House's Area and so on. We'll show in the upcoming part.

DATA OVERVIEW

```
df_x.head()
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

1. **CRIM** per capital crime rate by town

2. **ZN** proportion of residential land zoned for lots over 25,000 sq.ft.

3. **INDUS** proportion of non-retail business acres per town

4. **CHAS** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

5. **NOX** nitric oxides concentration (parts per 10 million)

6. **RM** average number of rooms per dwelling

7. **AGE** proportion of owner-occupied units built prior to 1940

8. **DIS** weighted distances to five Boston employment centers

9. **RAD** index of accessibility to radial highways

10.**TAX** full-value property-tax rate per 10,000 USD

11. **PTRATIO** pupil-teacher ratio by town

12. **Black** 1000(Bk — 0.63)² where Bk is the proportion of blacks by town

13. **LSTAT** % lower status of the population

About the Algorithms used in

The major aim of in this project is to predict the house prices based on the features using some of the regression techniques and algorithms.

1. **Linear Regression**

Machine Learning Packages are used for in this Project

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

This Dataset consist several features such as Number of Rooms, Crime Rate, and Tax and so on. Let's know about how to read the dataset into the Jupyter Notebook. You can download the dataset from Kaggle in csv file format.

As well we can also able to get the dataset from the sklearn datasets.

```
df_x=pd.read_csv("Boston_Train.csv.csv")
df_y=pd.read_csv("Boston_Test.csv.csv")
```

Learning about the dataset

```
df_x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 351 entries, 0 to 350
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  351 non-null    int64
 1   crim        351 non-null    float64
 2   zn          351 non-null    float64
 3   indus       351 non-null    float64
 4   chas        351 non-null    int64
 5   nox         351 non-null    float64
 6   rm          351 non-null    float64
 7   age         351 non-null    float64
 8   dis         351 non-null    float64
 9   rad         351 non-null    int64
 10  tax         351 non-null    int64
 11  ptratio     351 non-null    float64
 12  black       351 non-null    float64
 13  lstat       351 non-null    float64
 14  medv        351 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 41.2 KB
```

Data Preprocessing

Checking for null values:

```
df_x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 351 entries, 0 to 350
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Unnamed: 0   351 non-null    int64  
 1   crim         351 non-null    float64
 2   zn           351 non-null    float64
 3   indus        351 non-null    float64
 4   chas         351 non-null    int64  
 5   nox          351 non-null    float64
 6   rm           351 non-null    float64
 7   age          351 non-null    float64
 8   dis          351 non-null    float64
 9   rad          351 non-null    int64  
 10  tax          351 non-null    int64  
 11  ptratio      351 non-null    float64
 12  black        351 non-null    float64
 13  lstat        351 non-null    float64
 14  medv         351 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 41.2 KB
```

We find that the dataset is already cleaned

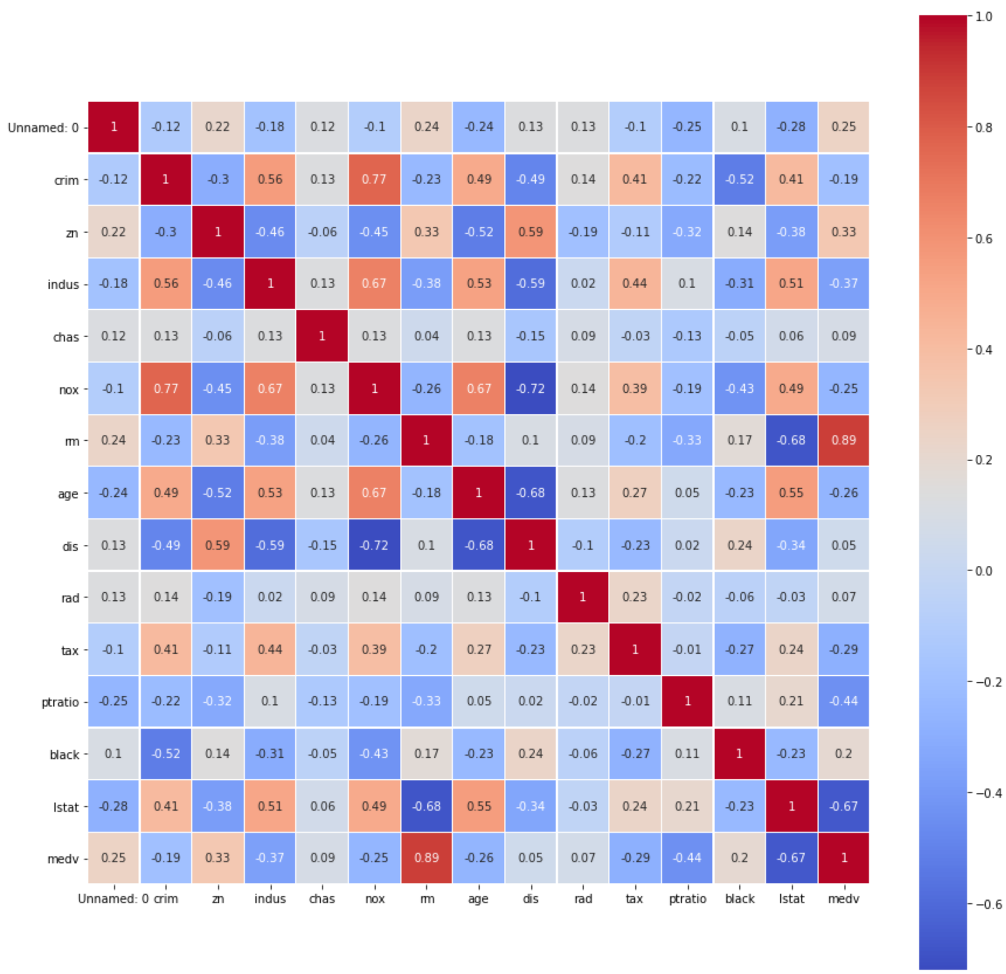
Also Here our target variable is medv

Exploratory Data Analysis

In statistics, exploratory data analysis (***EDA***) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily ***EDA*** is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

First Understanding the correlation of features between target and other features.

```
plt.figure(figsize = (15,15))
sns.heatmap(data = df_x.corr().round(2),annot=True,cmap='coolwarm',linewidths=0.2,square=True)
```



The Big colorful picture above which is called Heatmap helps us to understand how features are correlated to each other.

1.Postive sign implies postive correlation between two features whereas Negative sign implies negative correlation between two features.

2.I am here interested to know which features have good correlation with our dependent variable MEDV and can help in having good predictions.

3.I observed that INDUS, RM, TAX, PTRATIO and LSTAT shows some good correaltion with MEDV and I am interested to know more about them.

4.However I noticed that INDUS shows good correlation with TAX and LSAT which is a pain point because it leads to Multicollinearity. So I decided NOT to consider this feature and do further analysis with other 5 remaining features.

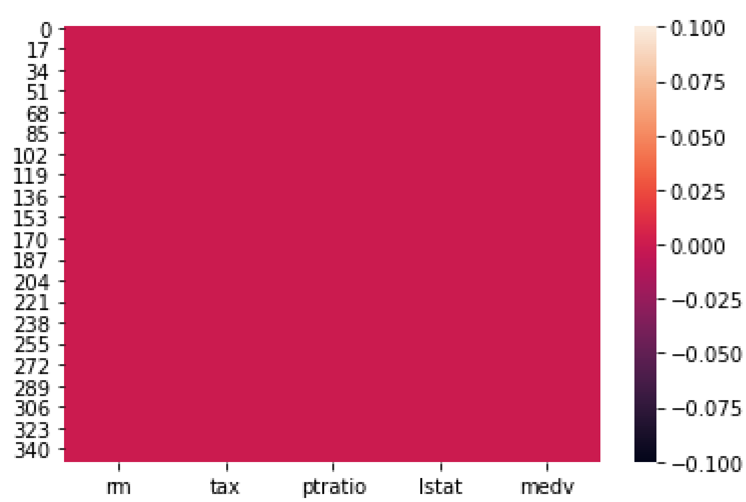
```
df1 = df_x[['rm','tax','ptratio','lstat','medv']]
df1.head()
```

	rm	tax	ptratio	lstat	medv
0	6.575	296	15.3	4.98	24.0
1	6.421	242	17.8	9.14	21.6
2	7.185	242	17.8	4.03	34.7
3	6.998	222	18.7	2.94	33.4
4	7.147	222	18.7	5.33	36.2

Checking null values in df1

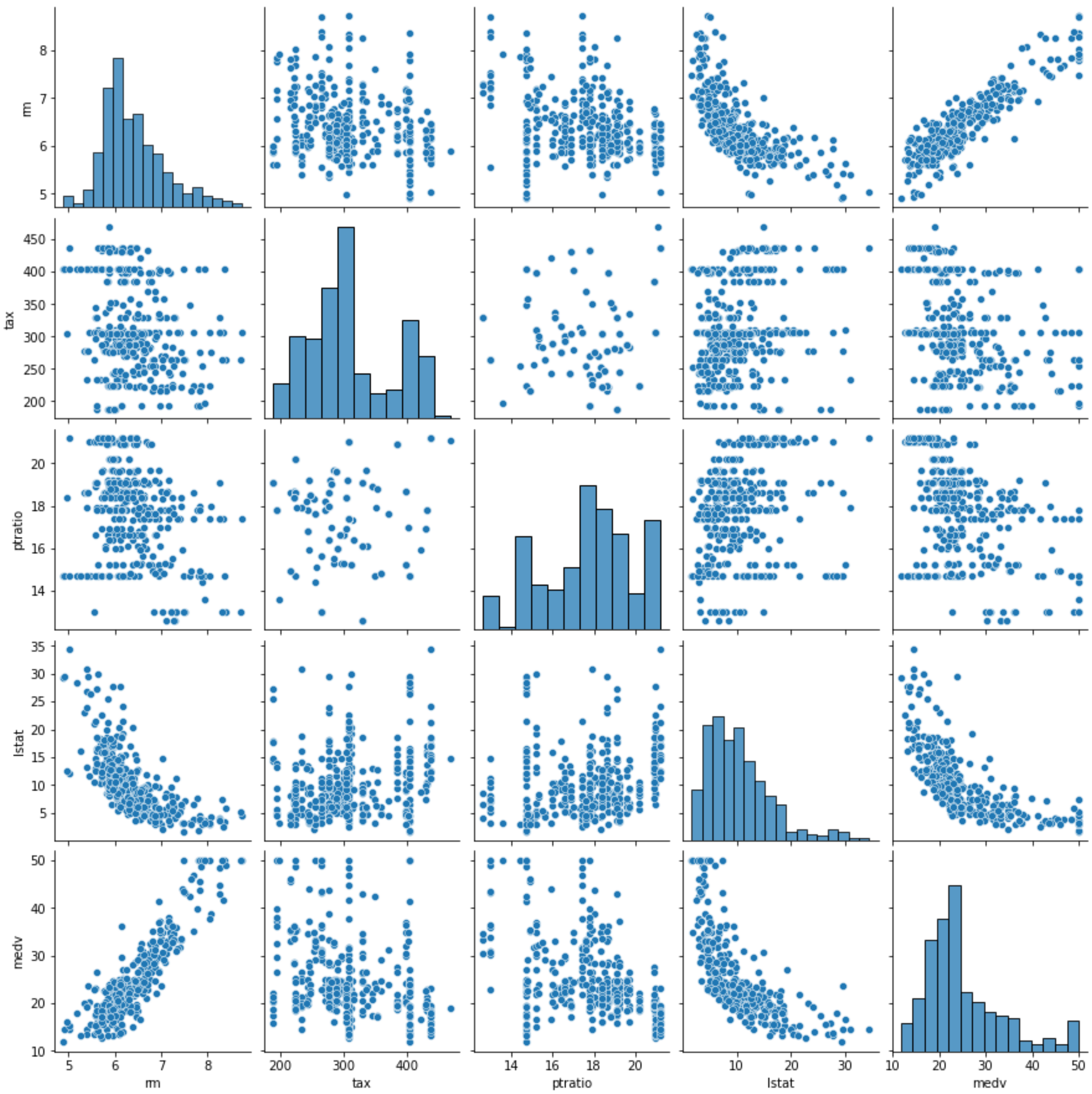
```
sns.heatmap(df1.isna())
```

```
<AxesSubplot:>
```



Let us try and plot all the co-relation

```
sns.pairplot(data=df1)
```



Now since MEDV is our target variable let's explore it more with statistical analysis.

```
prices = df1['medv']
```

```
features = df1.drop('medv',axis=1)
```

Statistic calculation

```
#Min price of the data
min_price = np.amin(prices)
```

```
min_price
```

```
11.8
```

```
# Max price of the data
max_price = np.amax(prices)
```

```
max_price
```

```
50.0
```

```
# Mean price of the data
mean_price = np.mean(prices)
```

```
mean_price
```

```
25.062678062678092
```

```
# Median price of the data
median_price = np.median(prices)
```

```
median_price
```

```
22.9
```

```
# Standard deviation of prices of the data
std_price = np.std(prices)
```

```
std_price
```

```
8.449855623881836
```

Model Fitting

Here we will use Linear Regression Model

```
y= df1['medv']
X= df1.drop('medv',axis=1)
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

```
# Import library for Linear Regression
from sklearn.linear_model import LinearRegression

# Create a Linear regressor
lm = LinearRegression()

# Train the model using the training sets
lm.fit(X_train, y_train)
```

```
LinearRegression()
```

```
# Model prediction on train data
y_pred = lm.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
lm.score(X_test,y_test)
```

```
0.7967322690509783
```

Here the model gave us an accuracy of 0.7967322690509783

i.e. 79.67%

Output & Conclusion

From the Exploratory Data Analysis, we could generate insight from the data. How each of the features relates to the target. Also, the evalution of the linear model can be seen.

Business Recommendation

Using this idea we can create a user friendly application to deduce the required capital amount for a dream house based on its features.