

AN INTERNSHIP PROJECT REPORT ON
Boston House Data Analysis
A Project Report for Internship Programme

Submitted By

Sounak Sarkar

in partial fulfillment for the award of the degree of
Bachelors in Computer Applications



At



DECLARATION

I undersigned, hereby declare that the project titled “Boston House Data Analysis” submitted in partial fulfillment for the award of Degree of Bachelors in Computer Application of MAKAUT is a bonafide record of work done by me under the guidance of Animesh Ojha. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any University.

06/01/2022

Sounak Sarkar

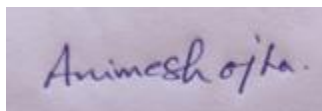
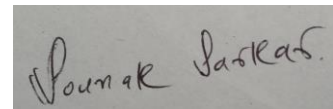
BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision Development of a feature-rich, Boston House Data Analysis is the bonafide work of

Student Name

Signature

Sounak Sarkar



SIGNATURE

Name:

ANIMESH OJHA

CERTIFICATE

This is to certify that the report titled 'Boston House Data Analysis' being submitted by Sounak Sarkar (30601219083) in partial fulfilment of the requirements for the award of the Degree of Bachelors in Computer Application, is a bonafide record of the project work done by Sounak Sarkar of department of CA, Narula Institute of Technology.



ACKNOWLEDGEMENT

Through this acknowledgement I express my sincere gratitude towards all those people who helped me in this project, which has been a learning experience. This space wouldn't be enough to extend my warm gratitude towards my project guide Mr. Animesh Ojha for his efforts in coordinating with the work and guiding in right direction. It would be injustice to proceed without acknowledging those vital supports. I would like to thank Mr. Anirban Banerjee, for helping me to get such an excellent opportunity. I also use this space to offer our sincere love to my parents and all others who had been there, helping me walk through this work.

Sounak Sarkar

LIST OF TABLES

Table No	Title Of The Table	Page No
1	Abstract	4

2	Introduction	4
3	Research Methodology	8
4	Data Analysis (Boston House Data)	9
5	Implementation	21
6	Conclusion	21
7	References	22

1. ABSTRACT

The machine learning field, which can be briefly defined as enabling computers make successful predictions using past experiences, has exhibited an impressive development recently with the help of the rapid increase in the storage capacity and processing power of computers. Together with many other disciplines, machine learning methods have been widely employed in bioinformatics. The difficulties and cost of biological analyses have led to the development of sophisticated machine learning approaches for this application area. In this paper, we were asked to experiment with a dataset, and to explore how machine learning algorithms can be used to find the patterns in data. We were expected to gain experience using a common data-mining and machine learning library, and were expected to submit a report about the dataset and the algorithms used. After performing the required tasks on a dataset of my choice, herein lies my final report.

2. INTRODUCTION

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly

programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Five basic steps for a machine learning task:

1. Data Collection and Preparation

The first step of machine learning is gathering or collecting data. This is extremely important because the amount of data you collect and the quality of that data will have a huge impact on the results.

After this data is collected, we have to prepare it for use in our machine learning training. We also need to divide the data into 2 parts. The first part, and the majority of the data, will be used for training, and the second part will be used for evaluating the model's performance.

2. Type of Model

The next step is to choose a model. There are many models' data scientists have created over the years, for example, logistic regressions, linear models, decision trees, and much more.

3. Training

Training is the most important step in machine learning. In training, you pass the prepared data to your machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

4. Evaluation and Parameter Tuning

Once the training is complete, we need to see if the model actually works. We need to use the second data set that we kept aside for this part. Evaluation allows us to test the model against data that it has never seen before. The way the model performs is representative of how it is going to perform in the real world.

Once the evaluation is done, we need to see if we can still improve our training. We can do this by tuning our parameters. We can now show our model the full dataset so that it can finetune its predictions. We can also show it the dataset multiple times, because it may make the predictions more accurately.

5. Prediction

We are finally at the last step of machine learning. Prediction is the step where we actually get to answer the question.

Types of machine learning algorithms–

Supervised vs Unsupervised Learning vs Reinforcement Learning

- Supervised Learning deals with two main tasks Regression and Classification. Unsupervised Learning deals with clustering and associative rule mining problems. Whereas Reinforcement Learning deals with exploitation or exploration, Markov's decision processes, Policy Learning, Deep Learning and value learning.
- Supervised Learning works with the labelled data and here the output data patterns are known to the system. But, the unsupervised learning deals with unlabeled data where the output is based on the collection of perceptions. Whereas in Reinforcement Learning Markov's Decision process- the agent interacts with the environment in discrete steps.
- The name itself says, Supervised Learning is highly supervised. And Unsupervised Learning is not supervised. As against, Reinforcement Learning is less supervised which depends on the agent in determining the output.
- The input data in Supervised Learning is labelled data. Whereas, in Unsupervised Learning the data is unlabeled. The data is not predefined in Reinforcement Learning.
- Supervised Learning predicts based on a class type. Unsupervised Learning discovers underlying patterns. And in Reinforcement Learning, the learning agent works as a reward and action system.
- Supervised learning maps labelled data to known output. Whereas, Unsupervised Learning explore patterns and predict the output. Reinforcement Learning follows a trial-and-error method.
- To sum up, in Supervised Learning, the goal is to generate formula based on input and output values. In Unsupervised Learning, we find an association between input values and group them. In Reinforcement Learning an agent learn through delayed feedback by interacting with the environment.

2.1. BACKGROUND OF THE STUDY

The data comprised in this dataset was collected by the U.S Census Service, and it first appeared in the history of statistical analysis in a paper by David Harrison Jr. and Daniel L. Rubinfeld, called Hedonic housing prices and the demand for clean air. Researchers had a hypothesis that people were willing to pay more for clean air —hence the term “hedonic pricing” which in this case is used to describe the monetary value that people assign to factors not inherent to the property but to its surrounding area— but there was debate on how to measure it. Harrison and Rubinfeld were concerned:

While several studies have used [the housing market approach] to estimate the demand for air quality improvements, they have paid little attention to the sensitivity of the results to the assumptions embedded in the procedures.

To make their point, they needed a clean database with lots of variables and a precise measure of air pollution. They found this in the “data for census tracts in the Boston Standard Metropolitan Statistical Area (SMSA) in 1970”.

2.2 NEED AND SIGNIFICANCE OF THE STUDY

The Boston housing prices dataset has an ethical problem: as investigated, the authors of this dataset engineered a non-invertible variable “B” assuming that racial self-segregation had a positive impact on house prices. Furthermore, the goal of the research that led to the creation of this dataset was to study the impact of air quality but it did not give adequate demonstration of the validity of this assumption.

The scikit-learn maintainers therefore strongly discourage the use of this dataset unless the purpose of the code is to study and educate about ethical issues in data science and machine learning.

2.3. STATEMENT OF 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.

2.4. OBJECTIVE OF THE STUDY

In this project, we will evaluate the performance and predictive power of a model that has been trained and tested on data collected from homes in suburbs of Boston, Massachusetts. A model trained on this data that is seen as a good fit could then be used to make certain predictions about a home — in particular, its monetary value. This model would prove to be invaluable for someone like a real estate agent who could make use of such information on a daily basis.

2.5. ORGANIZATION OF THE STUDY

The dataset in this section is the so-called Boston House dataset. It contains data concerning make certain predictions about a home — in particular, its monetary value in Boston.

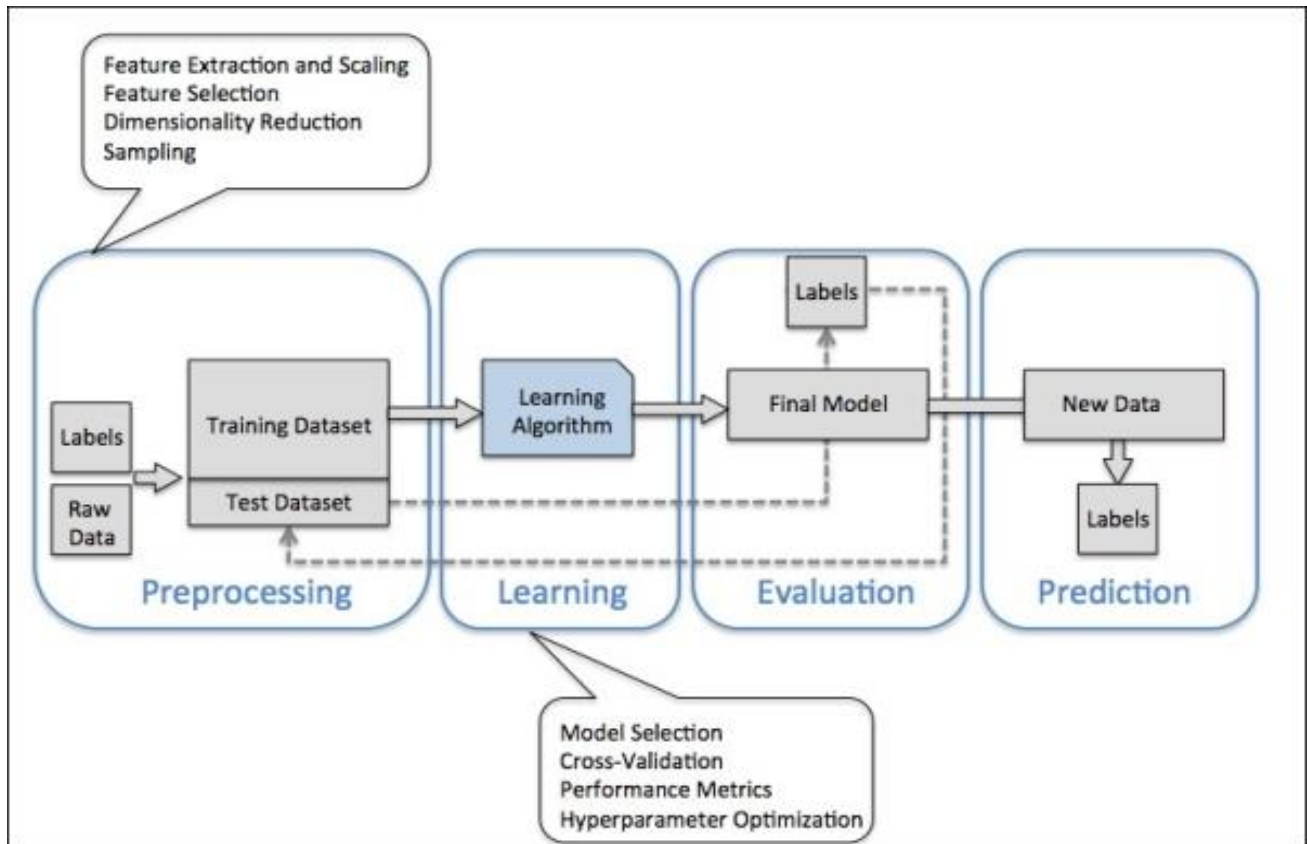
3. RESEARCH METHODOLOGY

3.1. OBJECTIVES

The prime objective of this project is to construct a working model which has the capability of predicting the value of houses, we will need to separate the dataset into features and the target variable. The features, 'RM', 'LSTAT', and 'PTRATIO', give us quantitative information about each data point. The target variable, 'MEDV', will be the variable we seek to predict. These are stored in features and prices, respectively.

3.2. RESEARCH DESIGN

To pick a specific model, there is a step-by-step technique. It is important to determine if the model to be implemented is appropriate for the given issue. In the below diagram, flow of processes can be seen acting in accordance here.



3.3. SOFTWARE REQUIREMENT

This project uses the following tools for its implementation:

- 1) **Python**: the programming language used to implement this project
- 2) **Jupyter notebook**: used to provide an interactive environment for python, for the implementation of this project
- 3) **Scikit-Learn**: used to implement the several machine learning models used in this project as view their accuracies.
- 4) **Matplotlib**: used to plot graphs for us to understand the trend in accuracy/performance of the various machine learning algorithm implemented.
- 5) **Seaborn**: use for visualization of the statistical behavior of data set.
- 6) **Pandas**: - used to analyze and clean our dataset,
- 7) **NumPy**: for mathematical Calculations.

4. DATA ANALYSIS (BOSTON HOUSE DATA)

Importing the libraries

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn import metrics
6 from sklearn.model_selection import train_test_split
7 from sklearn.linear_model import LinearRegression
8 import warnings
9 warnings.filterwarnings('ignore')
10 %matplotlib inline
```

Importing the Boston Housing dataset

Initializing the dataframe

```
1 from sklearn.datasets import load_boston
2 boston = load_boston()
```

```
1 data=pd.DataFrame(boston.data)
2 data
```

	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
...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 13 columns

#Adding the feature names to the dataframe

```
1 data.columns = boston.feature_names
2 data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
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

CRIM per capita crime rate by town

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

INDUS proportion of non-retail business acres per town

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

NOX nitric oxides concentration (parts per 10 million)

RM average number of rooms per dwelling

AGE proportion of owner-occupied units built prior to 1940

DIS weighted distances to five Boston employment centers.

RAD index of accessibility to radial highways
TAX full-value property-tax rate per 10,000usd
PTRATIO pupil-teacher ratio by town
B $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town
LSTAT % lower status of the population

Each record in the database describes a Boston suburb or town.

Adding target variable to dataframe

Median value of owner-occupied homes in \$1000s

```
1 data['Price']=boston.target
```

#Check the shape of dataframe

```
1 data.shape
```

```
(506, 14)
```

```
1 data.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',  
       'PTRATIO', 'B', 'LSTAT', 'Price'],  
      dtype='object')
```

```
1 data.dtypes
```

```
CRIM    float64  
ZN      float64  
INDUS   float64  
CHAS    float64  
NOX     float64  
RM      float64  
AGE     float64  
DIS     float64  
RAD     float64  
TAX     float64  
PTRATIO float64  
B       float64  
LSTAT   float64  
Price   float64  
dtype: object
```

```
1 data.info()
```

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

Identifying the unique number of values in the dataset

Check for missing values

```
1 data.nunique()
CRIM      504
ZN        26
INDUS     76
CHAS       2
NOX       81
RM       446
AGE      356
DIS      412
RAD        9
TAX       66
PTRATIO   46
B        357
LSTAT    455
Price    229
dtype: int64
```

```
1 data.isnull().sum()
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
Price     0
dtype: int64
```

Viewing the data statistics

1	data.describe()												
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37

Finding out the correlation between the features

Plotting the heatmap of correlation between features

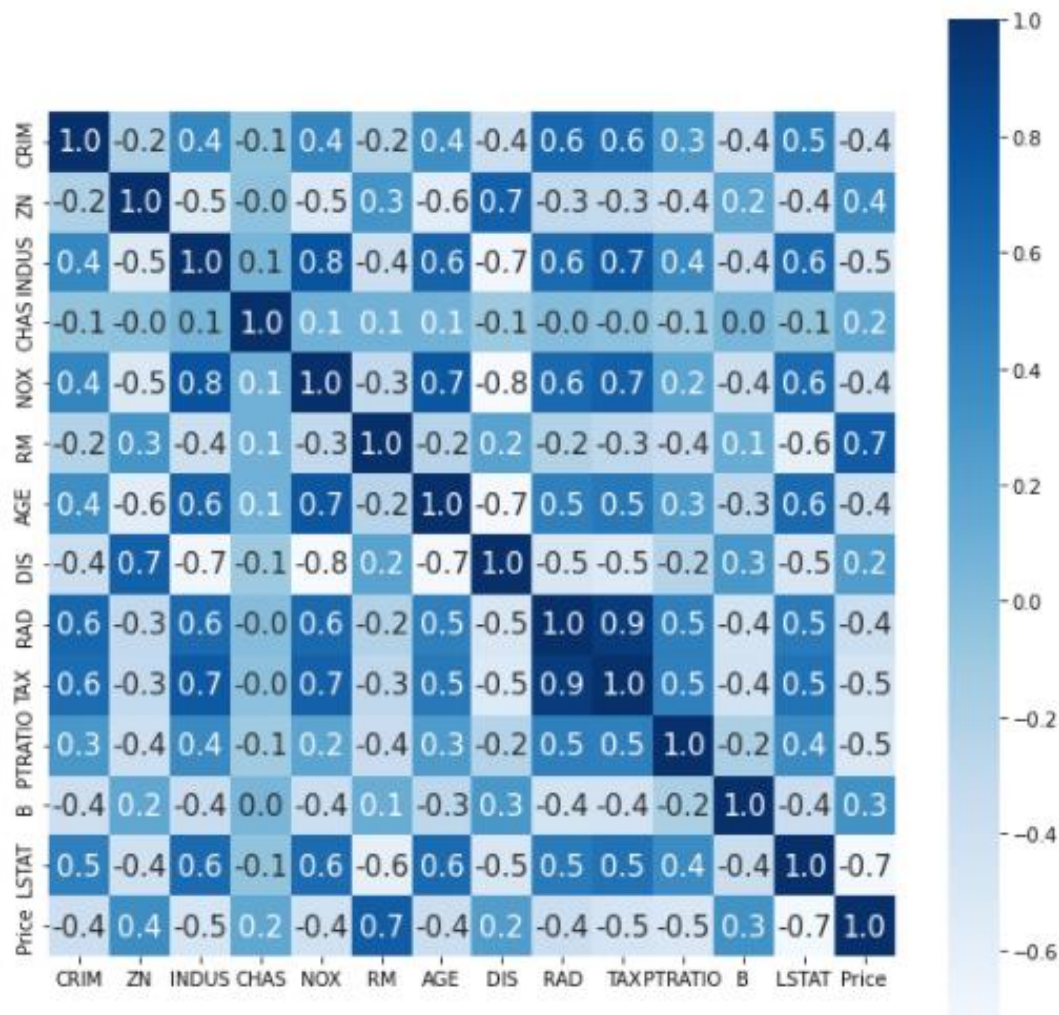
1	corr=data.corr()														
2	corr														
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price	
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.388305	
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.360445	
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.483725	
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.175260	
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.427321	
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.695360	
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.376955	
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.249929	
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.381626	
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.468536	
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	-0.507787	
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	0.333461	
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	-0.737663	
Price	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663	1.000000	

```

1 plt.figure(figsize=(10,10))
2 sns.heatmap(corr, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':15}, cmap='Blues')

```

<AxesSubplot:>



Splitting target variable and independent variables

Splitting to training and testing data

```

1 X = data.drop(['Price'], axis = 1)
2 y = data['Price']

1 X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 4)

```

Linear regression

Training the model

Create a Linear regressor

Train the model using the training sets

1	lm=LinearRegression()
2	lm.fit(X_train,y_train)

LinearRegression()

1	lm.intercept_
---	---------------

36.357041376595205

Value of y intercept

Converting the coefficient values to a dataframe


```
1 lm.intercept_
```

36.357041376595205

```
1 coefficients = pd.DataFrame([X_train.columns,lm.coef_]).T
2 coefficients = coefficients.rename(columns={0: 'Attribute', 1: 'Coefficients'})
3 coefficients
```

	Attribute	Coefficients
0	CRIM	-0.12257
1	ZN	0.055678
2	INDUS	-0.008834
3	CHAS	4.693448
4	NOX	-14.435783
5	RM	3.28008
6	AGE	-0.003448
7	DIS	-1.552144
8	RAD	0.32625
9	TAX	-0.014067
10	PTRATIO	-0.803275
11	B	0.009354
12	LSTAT	-0.523478

Model Evaluation

Model prediction on train data

Model Evaluation

```
1 y_pred = lm.predict(X_train)
2 print("R^2:",metrics.r2_score(y_train, y_pred))
3 print("Adjusted R^2:",1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(len(y_train)-X_train.shape[1]-1))
4 print("MAE:",metrics.mean_absolute_error(y_train, y_pred))
5 print("MSE:",metrics.mean_squared_error(y_train, y_pred))
6 print("RMSE:",np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

R^2: 0.7465991966746854

Adjusted R^2: 0.736910342429894

MAE: 3.08986109497113

MSE: 19.07368870346903

RMSE: 4.367343437774162

R^2 : It is a measure of the linear relationship between X and Y. It is interpreted as the proportion of the variance in the dependent variable that is predictable from the independent variable.

Adjusted R^2 : The adjusted R-squared compares the explanatory power of regression models that contain different numbers of predictors.

MAE: It is the mean of the absolute value of the errors. It measures the difference between two continuous variables, here actual and predicted values of y.

MSE: The mean square error (MSE) is just like the MAE, but squares the difference before summing them all instead of using the absolute value.

RMSE: The mean square error (MSE) is just like the MAE, but squares the difference before summing them all instead of using the absolute value.

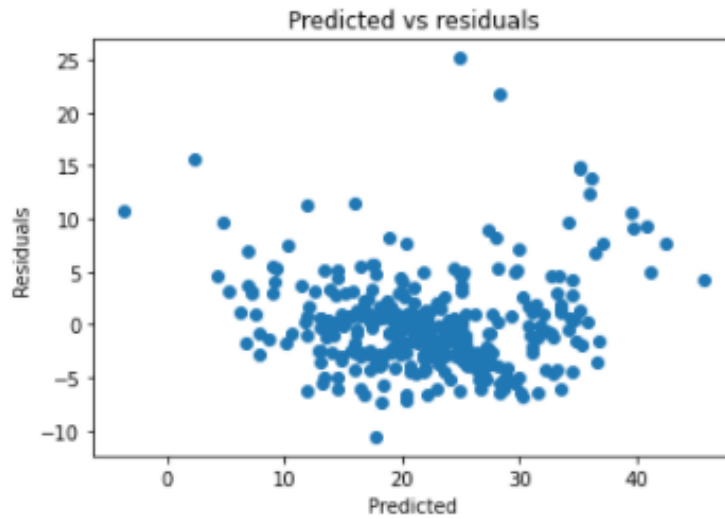
Visualizing the differences between actual prices and predicted values

Checking residuals

```
1 plt.scatter(y_train, y_pred)
2 plt.xlabel("Prices")
3 plt.ylabel("Predicted prices")
4 plt.title("Prices vs Predicted prices")
5 plt.show()
```



```
1 plt.scatter(y_pred,y_train-y_pred)
2 plt.title("Predicted vs residuals")
3 plt.xlabel("Predicted")
4 plt.ylabel("Residuals")
5 plt.show()
```



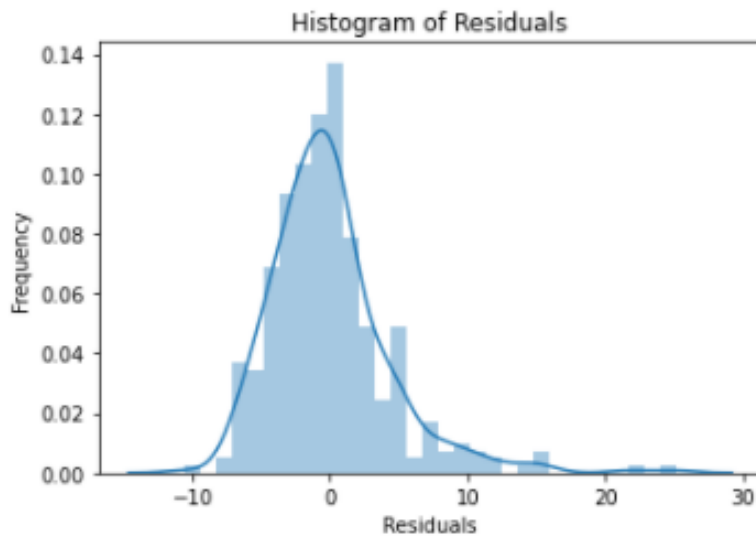
There is no pattern visible in this plot and values are distributed equally around zero. So Linearity assumption is satisfied.

Checking Normality of errors

```

1 sns.distplot(y_train-y_pred)
2 plt.title("Histogram of Residuals")
3 plt.xlabel("Residuals")
4 plt.ylabel("Frequency")
5 plt.show()

```



Here the residuals are normally distributed. So, normality assumption is satisfied.

For test data

```

1 y_test_pred = lm.predict(X_test)
2 acc_linreg = metrics.r2_score(y_test, y_test_pred)
3 print('R^2:', acc_linreg)
4 print('Adjusted R^2:', 1 - (1-metrics.r2_score(y_test, y_test_pred))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
5 print('MAE:', metrics.mean_absolute_error(y_test, y_test_pred))
6 print('MSE:', metrics.mean_squared_error(y_test, y_test_pred))
7 print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))

```

```

R^2: 0.7121818377409195
Adjusted R^2: 0.6850685326005713
MAE: 3.8590055923707407
MSE: 30.053993307124127
RMSE: 5.482152251362974

```

Here the model evaluations scores are almost matching with that of train data. So, the model is not overfitting.

5. IMPLEMENTATION

The implementation of this project involved the following steps: -

- *Import the necessary packages*
- *Import the required dataset and clean the dataset*
- *Visualize the dataset*
- *Split the dataset into a training and test set*
- *Apply the algorithm*
- *Generate graphs comparing train and test accuracies*
- *Model performance using training and testing set by Linear Regression*

6. CONCLUSION

The model performance for training set

R²: 0.7465991966746854

Adjusted R²: 0.736910342429894

MAE: 3.08986109497113

MSE: 19.07368870346903

RMSE: 4.367343437774162

The model performance for testing set

R²: 0.7121818377409195

Adjusted R²: 0.6850685326005713

MAE: 3.8590055923707407

MSE: 30.053993307124127

RMSE: 5.482152251362974

7. REFERENCES

- ❖ <https://www.kaggle.com/prasadperera/the-boston-housing-dataset>
- ❖ <https://towardsdatascience.com/machine-learning-project-predicting-boston-house-prices-with-regression-b4e47493633d>
- ❖ <https://realpython.com/best-python-books/>
- ❖ <https://www.geeksforgeeks.org/how-to-start-learning-machine-learning>
- ❖ <https://machinelearningmastery.com/linear-regression-for-machine-learning/>