A Satyam Roychowdhury initiative

SNU SISTER NIVEDITA UNIVERSITY

Boston Housing Price Prediction

Susanta Dhurua

Department Of Statistics

Roll no : 1811108010005

21

INDEX

Contents

Introduction :

What is bostonhousing ?

The dataset contains information collected by U.S. Census Service concerning housing in the area of Boston Mass.It was obtained from the StatLib archive(http://lib.stat.cmu.edu/datasets/boston) and has been used extensively throughout the literature to benchnmark algorithms.

Why we predict house price ?

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 are able to accurately estimate the price of the house given the features.

What is the problem we deal with ?

The problem that we are going to solve here is that given set of features that describes a house in Boston, our machine learning model must predict the house price .

Data set collection:

Our Dataset is collected from kaggale machine learning(https://www.kaggle.com/datasets)In Dataset , each row describe a boston town or suburb.

506 rows and 13 attributes (features) with a target column (price).

Linear regression:

Linear Regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable ? (2) Which variables in particular are significant predictors of the outcome variable , and in what way do they-indicated by the magnitude and sign of the bela estimates – impact the outcome variable ?

These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y=c+bx where y=estimated dependent variable score, c=constant, b= regression coefficients, and x=score on the independent variable .

Problem statement:

Our aim is to predict housing price by using Linear Regression using python.

Libraries Used :

In [1]:
```
# Importing the libraries
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Pandas : An open source , BSD – licensed library providing high performance ,easy to use data structures and data analysistools for the python programming language.

Numpy: The fundamental package for scientific computing with python. .

Sklearn: is a free software machine learning library for the Python programming language.[2] It features various classification, regression and clustering algorithms .

Matplotlib.pyplot : is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

Seaborn:Seaborn is a python data visualization library based on matplotlib. It provides a high level interface for drawing attractive and informative statistical graphics.

Sklearn.metrices :

- Accuracy score: In multilevel classification, this function computes subset accuracy, the set of labels predicted for a sample must exactly match the corresponding set of labels in y true.
- Classification report : Build a report showing the main classification metrices .
- Confusion matrix : Compute confusion matrix to evaluate the accuracy of classification.
- It allows us to perform a range of evaluation techniques to evaluate regression model.

And the Dataset contain 506 rows and 13 columns.

Importing Dataset :

```
In [2]: # Importing the Boston Housing dataset
        from sklearn.datasets import load_boston
        boston = load_boston()
```

```
In [3]: # Initializing the dataframe
        data = pd.DataFrame(boston.data)
```

```
In [4]: # See head of the dataset
        data.head()
```

Now we import data from the disk to python by using the library sklearn.datasets and after that we locate the value of boston housing and we keep this in the DataFrame because DataFrame used to alinged the data in rows and column form.

Out[4]:

	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

And here we got the DataFrame in systematic ways and now we have to add features names .

Attributes :

In [5]:

```
#Adding the feature names to the dataframe
data.columns = boston.feature_names
data.head()
```

Out[5]:

	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

Parameters:

```
In [6]: # information about info
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 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
dtypes: float64(13)
memory usage: 51.5 KB
```

And there are 12 parameters and now we describe them one by one.

CRIM : per capita crime rate by town .

ZN : proportion of residential land zoned for lots over 25000 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 employementcentres

RAD :index of accessibility to radial highways

TAX : full –value property-tax rate per 10,000usd

PTRATIO : pupil teacher ratio by own

B : 1000 (Bk-0.63)^2 where Bk is the proportion of blacks by town.

LSTAT : % lower status of the population

Target value ["PRICE"]

```
In [7]: #Adding target variable to dataframe
        data['PRICE'] = boston.target
        # Median value of owner-occupied homes in $1000s
```

After that we look for the shape of the data and columns.

```
In [8]: #Check the shape of dataframe
        data.shape

Out[8]: (506, 14)

In [9]: data.columns

Out[9]: Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
               'PTRATIO', 'B', 'LSTAT', 'PRICE'],
              dtype='object')
```

Data Types :

```
In [10]: data.dtypes

Out[10]: 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
```

Identify Unique Number :

```
In [11]: # Identifying the unique number of values in the dataset
         data.nunique()
```

```
Out[11]: 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
```

Checking for Missing Values :

```
In [12]: # Check for missing values
         data.isnull().sum()
```

```
Out[12]: 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
```

There are no Missing values.

Row Missing Values :

```
In [13]: # See rows with missing values
         data[data.isnull().any(axis=1)]
```

Out[13]:

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

Data Statistics :

```
In [14]: # Viewing the data statistics
         data.describe()
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
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	506.000000	506.000000
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	12.653063	22.532806
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	7.141062	9.197104
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	1.730000	5.000000
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	6.950000	17.025000
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	11.360000	21.200000
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	16.955000	25.000000
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	37.970000	50.000000

Correlation Between Features :

```
In [15]: # Finding out the correlation between the features
         corr = data.corr()
         corr.shape
```

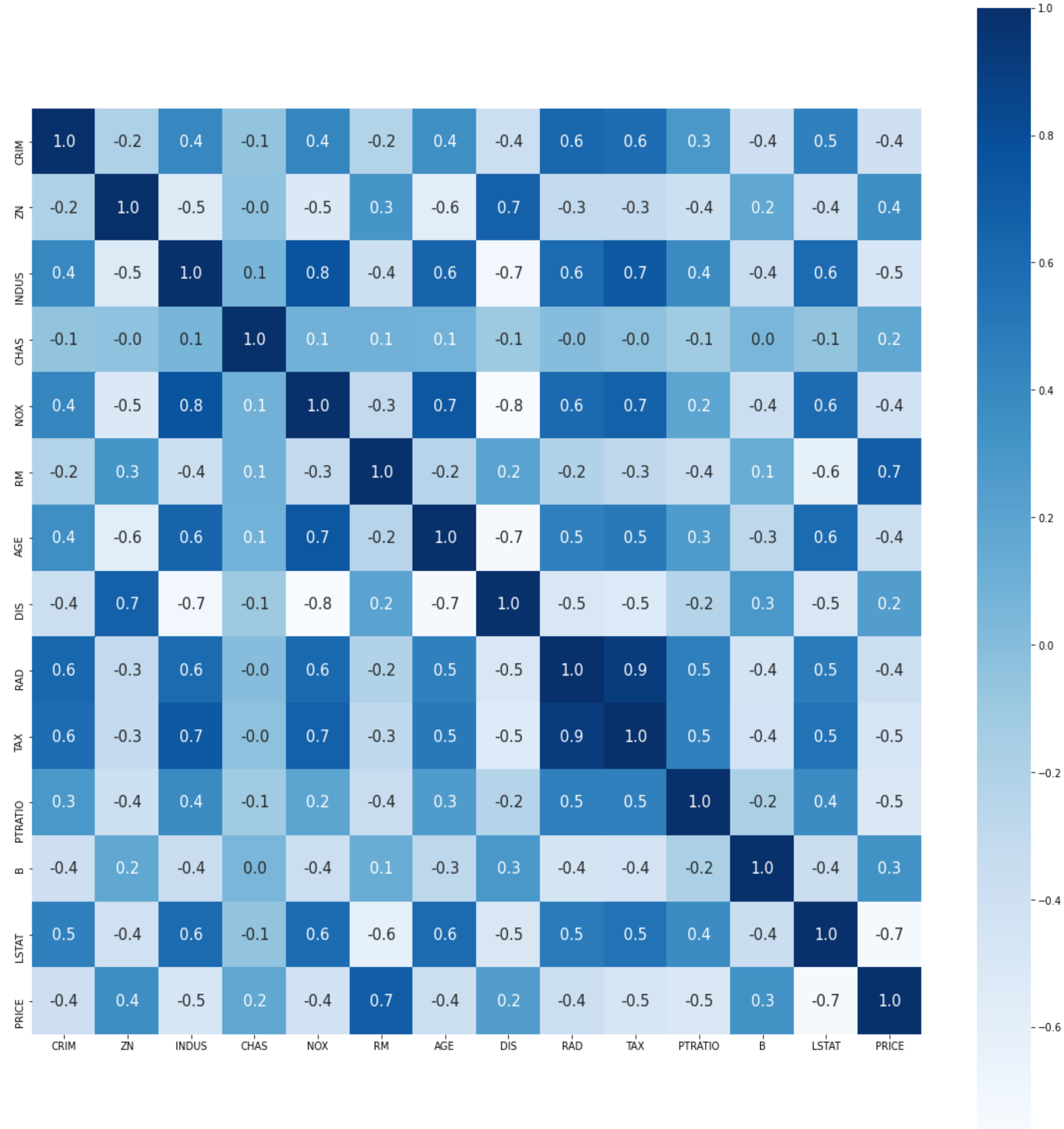
Out[15]: (14, 14)

HeatMap :

is a two dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors . The Heatmap is comig from the seaborn python package allows the creation of annotated Heatmaps.

```
n [17]: # Plotting the heatmap of correlation between features
        plt.figure(figsize=(20,20))
        sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws={'size':15}, cmap='Blues')

ut[17]: <AxesSubplot:>
```



From the above diagram we saw that there is (n-1) Rows and Column. Heatmap help us to show that the individual value in matrix way. The heat plot visualized the correlation between each pair of attributes in the dataset.

And now we spilit the target variables and independent variable.

```
In [19]: # Spliting target variable and independent variables
X = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

Here the Target variable is "PRICE."

Now we spiliting to training and testing data.

```
In [20]: # Splitting to training and testing data

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 = 4)
```

LINEAR REGRESSION :

Now we move to the main part of the project which is the main glimpse of this project.

```
In [21]: # 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)
```

So before go to Linear Regression we have to use/ import some of the library.

So here is the common Question is arised that what is Trainning sets and Test data

So , basically the both Train/Test is a method to measure the accuracy of the data.

We can train the model by using the Training set.

We can test the model by using the Testing set.

It is called Train/Test because you split the data into two sets : a training set and testing set.

80% of the Training set and 20% of the testing set.

Train the model means Create the model.

Test the model means test the accuracy of the model.

And output of the above cells is

```
Out[21]: LinearRegression()
```

Value of Y intercept .

```
In [22]: # Value of y intercept
         lm.intercept_
```

```
Out[22]: 36.35704137659535
```

Here we have Attributes and Coefficients we make in the dataframe in the form of rows and columns.

```
In [23]: #Converting the coefficient values to a dataframe
         coeffcients = pd.DataFrame([X_train.columns,lm.coef_]).T
         coeffcients = coeffcients.rename(columns={0: 'Attribute', 1: 'Coefficients'})
         coeffcients
```

Out[23]:

	Attribute	Coefficients
0	CRIM	-0.12257
1	ZN	0.0556777
2	INDUS	-0.00883428
3	CHAS	4.69345
4	NOX	-14.4358
5	RM	3.28008
6	AGE	-0.00344778
7	DIS	-1.55214
8	RAD	0.32625
9	TAX	-0.0140666
10	PTRATIO	-0.803275
11	B	0.00935369
12	LSTAT	-0.523478

Model Evaluation :

Model Evaluation techniques help us to judge the performance of a model and also allows us to compare differebt models fitted on the same dataset . We not only evaluate the performance of the model on our train dataset but also on our test dataset.

```
In [24]: # Model prediction on train data
         y_pred = lm.predict(X_train)
```

```
In [25]: # Model Evaluation
         print('R^2:',metrics.r2_score(y_train, y_pred))
         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))
         print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
         print('MSE:',metrics.mean_squared_error(y_train, y_pred))
         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
```

Here are some Variables that we describe below :

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 variables that is prediction from the independent variables.

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

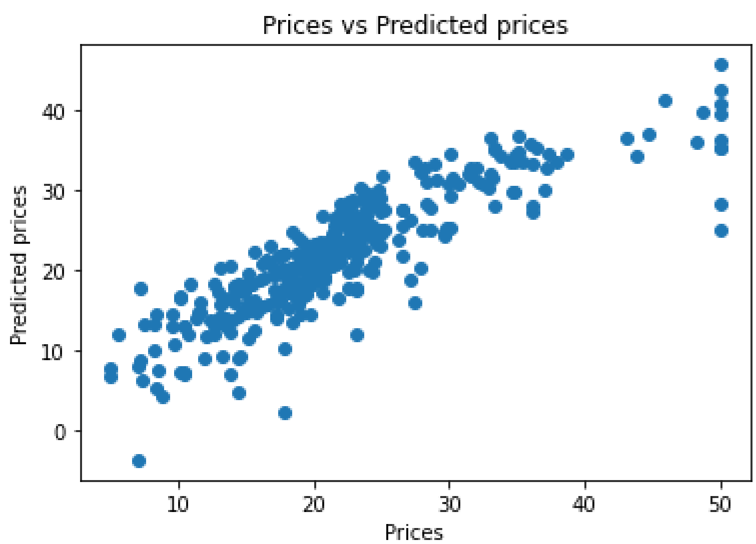
MAE : It is the Mean of the absolute value of the errors . It measures the difference between two continuous variables here actual and predicted value of y.

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

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

Now we look for the visualizing the actual price and predicted price.

```
In [26]: # Visualizing the differences between actual prices and predicted values
         plt.scatter(y_train, y_pred)
         plt.xlabel("Prices")
         plt.ylabel("Predicted prices")
         plt.title("Prices vs Predicted prices")
         plt.show()
```



This is a scatter plot where each value in the data are represented by dot in the plot.

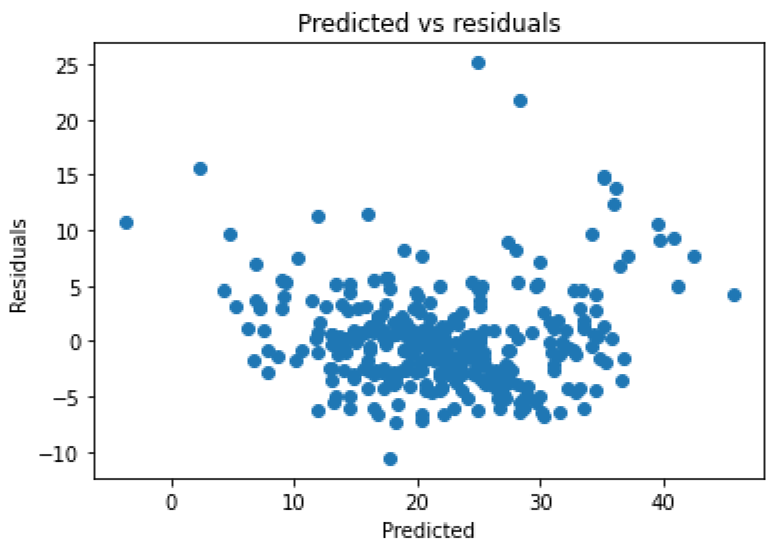
We can see that the dots are concentrated around the value 20 in x-axis and 20 in y-axis.

We can see that spread wider on the y –axis than to x-axis.

CHECKING RESIDUALS :

Residuals are the differences between the residual values and predicted values. If we square these errors and sum them up then we get SSE (Residual Sum Of Squares) .Analysis of Residuals is a method where we evaluate a regression model by analyzing these residuals (error terms) by plotting them on graph. If the residuals appear to behave randomly and show no pattern then it means that the model is **Good** .

```
In [27]: # Checking residuals
         plt.scatter(y_pred,y_train-y_pred)
         plt.title("Predicted vs residuals")
         plt.xlabel("Predicted")
         plt.ylabel("Residuals")
         plt.show()
```

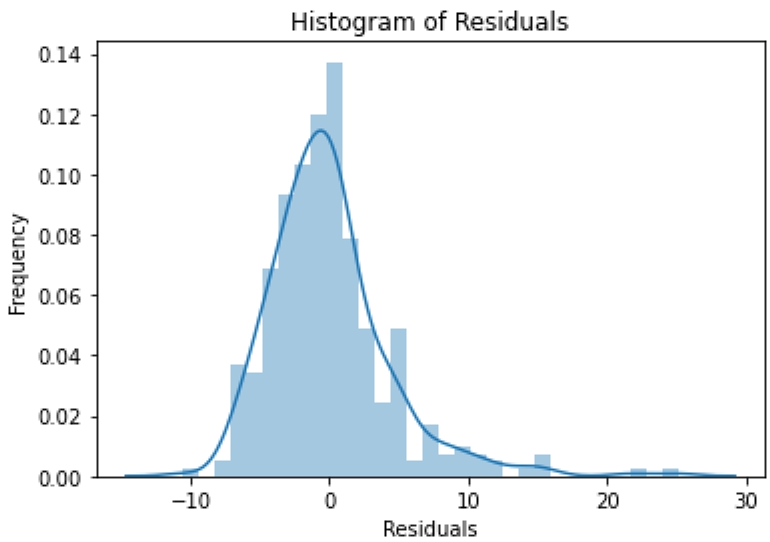


In this plot there is no pattern visible in this plot and values is distributed equally around Zero. So the model is Good . And the Linear assumption is satisfied.

Plotting Histogram :

A Histogram is basically used to represent data provided in a form of same groups .It is accurate method for the graphical representation of numerical data distribution .It is a type of bar plot where x-axis represent the bin ranges while y-axis gives information about frequency.

```
In [28]: # Checking Normality of errors
         sns.distplot(y_train-y_pred)
         plt.title("Histogram of Residuals")
         plt.xlabel("Residuals")
         plt.ylabel("Frequency")
         plt.show()
```



Here the residuals are normally distributed. Normality assumption is satisfied.

Predict Test data.

```
In [30]: # Predicting Test data with the model
         y_test_pred = lm.predict(X_test)
```

```
In [31]: # Model Evaluation
         acc_linreg = metrics.r2_score(y_test, y_test_pred)
         print('R^2:', acc_linreg)
         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))
         print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
         print('MSE:',metrics.mean_squared_error(y_test, y_test_pred))
         print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
R^2: 0.7121818377409193
Adjusted R^2: 0.6850685326005711
MAE: 3.8590055923707482
MSE: 30.053993307124163
RMSE: 5.482152251362978
```

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

Conclusion :

- The data collected in 1978 is not really relevant today due to rising population levels and changing population density of different areas.
- We analyzed the data-type of the features .
- We checked for missing values and also check the y_train and y_test and y_predicted value .
- After performing data-frame we plot scatter plot and histogram to get into more details .
- The data collected in an urban city will not be applicable in a rural city. because the people might value different aspect of a home depending on whether they live in an urban city or a rural areas.

As we observe that the value of R^2 *100 = 71.21818377409193 that means 71% of variability in dependent variable (`PRICE)`**is explained by the independent variable using our model.**

ACKNOWLEDGEMENT :

I would like to express my special thanks of gratitude to my Project guide **"Anindita Ghosal"** who gave me the golden opportunity to do this wonderful project on the topic " **Boston Housing price Prediction** "which also helped me in doing a lot of research and I came to know about so many new things I am really thankful to her.

Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Date : **14/01/2021** <<Name>> **Susanta Dhurua**

Place : **Howrah** <<Roll no>> **1811108010005**

<<Year>> **3rdyear(5th semester)**