BOSTON HOUSE PRICE PREDICTION

Understand the Problem Statement

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.

In [1]:

```
#Importing required libraries

import numpy as np
import pandas as pd

import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

```
from sklearn.datasets import load_boston
load_boston = load_boston()
x = load_boston.data
y = load_boston.target

data = pd.DataFrame(x, columns=load_boston.feature_names)
data["SalesPrice"] = y
data.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	SalesPrice
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	24.0
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	21.6
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	34.7
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	33.4
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	36.2

In [3]:

```
print(load_boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - 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 centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black people by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
```

```
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [5]:

```python
print(data.shape)
```

```
(506, 14)
```

In [6]:

```python
data.info
```

Out[6]:

```
<bound method DataFrame.info of         CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
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
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ...
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786  1.0  273.0
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875  1.0  273.0
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675  1.0  273.0
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889  1.0  273.0
505  0.04741   0.0  11.93   0.0  0.573  6.030  80.8  2.5050  1.0  273.0

     PTRATIO       B  LSTAT  SalesPrice
0       15.3  396.90   4.98        24.0
1       17.8  396.90   9.14        21.6
2       17.8  392.83   4.03        34.7
3       18.7  394.63   2.94        33.4
4       18.7  396.90   5.33        36.2
..       ...     ...    ...         ...
501     21.0  391.99   9.67        22.4
502     21.0  396.90   9.08        20.6
503     21.0  396.90   5.64        23.9
504     21.0  393.45   6.48        22.0
505     21.0  396.90   7.88        11.9

[506 rows x 14 columns]>
```

In [7]:

```python
data.describe()
```

Out[7]:

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

EDA

In [8]:

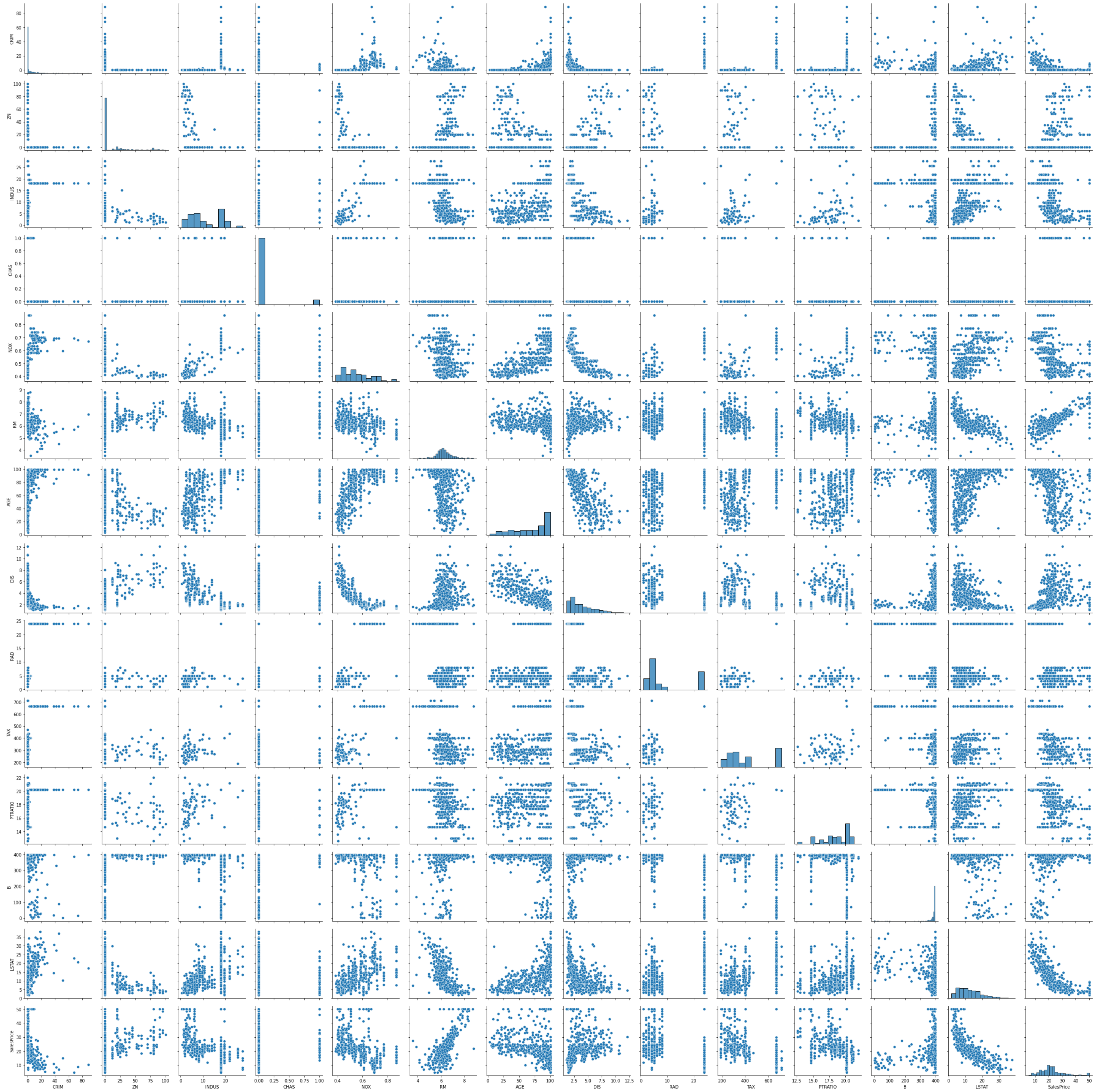
```python
data.isnull().sum()
```

Out[8]:

```
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
SalesPrice    0
dtype: int64
```

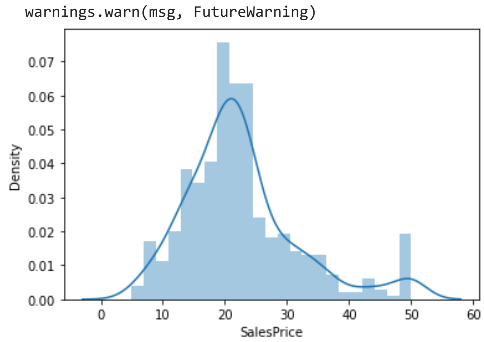
In [11]:

```python
sns.pairplot(data,height=2.5)
plt.tight_layout()
```



In [12]:
```
sns.distplot(data['SalesPrice']);
```

```
C:\Users\gourj\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

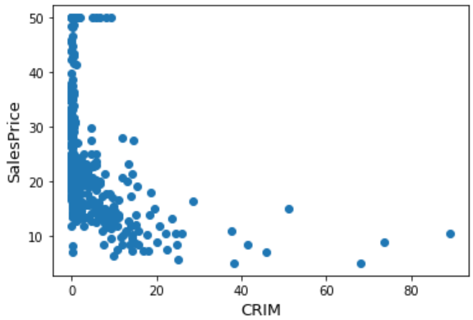


In [10]:
```
print("skewness: %f" % data['SalesPrice'].skew())
print("kurtiosis: %f" % data['SalesPrice'].kurt())
```

```
skewness: 1.108098
kurtiosis: 1.495197
```

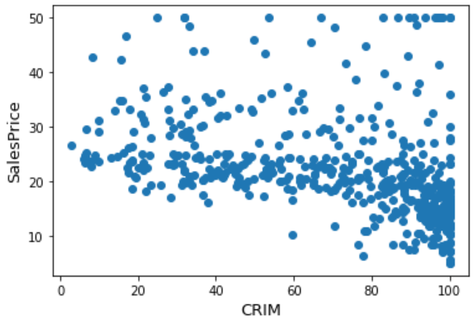
In [14]:
```
fig,ax = plt.subplots()
ax.scatter(x = data['CRIM'], y = data['SalesPrice'])
plt.ylabel('SalesPrice', fontsize=13)
```

```
plt.xlabel('CRIM', fontsize=13)
plt.show()
```



In [15]:

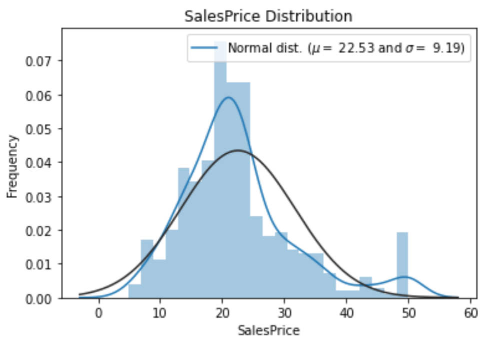
```
fig,ax = plt.subplots()
ax.scatter(x = data['AGE'], y = data['SalesPrice'])
plt.ylabel('SalesPrice', fontsize=13)
plt.xlabel('CRIM', fontsize=13)
plt.show()
```

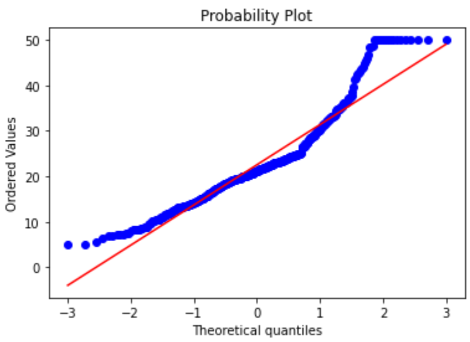


In [18]:

```
from scipy import stats
from scipy.stats import norm, skew

sns.distplot(data['SalesPrice'] , fit=norm);

(mu, sigma) = norm.fit(data['SalesPrice'])
print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu,sigma))

plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f})'.format(mu, sigma)], loc='best')
plt.ylabel('Frequency')
plt.title('SalesPrice Distribution')

# QQ-Plot
fig = plt.figure()
res = stats.probplot(data['SalesPrice'], plot=plt)
plt.show()
```

```
C:\Users\gourj\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
 mu = 22.53 and sigma = 9.19
```



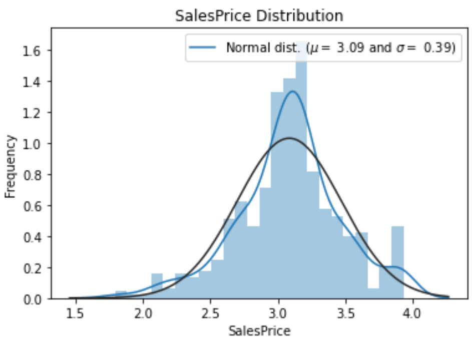


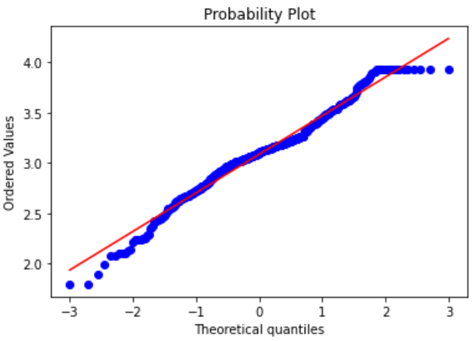
In [19]:

```
data["SalesPrice"] = np.log1p(data["SalesPrice"])

sns.distplot(data['SalesPrice'] , fit=norm);
```

```
(mu, sigma) = norm.fit(data['SalesPrice'])
print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu,sigma))

plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f})'.format(mu, sigma)], loc='best')
plt.ylabel('Frequency')
plt.title('SalesPrice Distribution')

# QQ-PLot
fig = plt.figure()
res = stats.probplot(data['SalesPrice'], plot=plt)
plt.show()
```

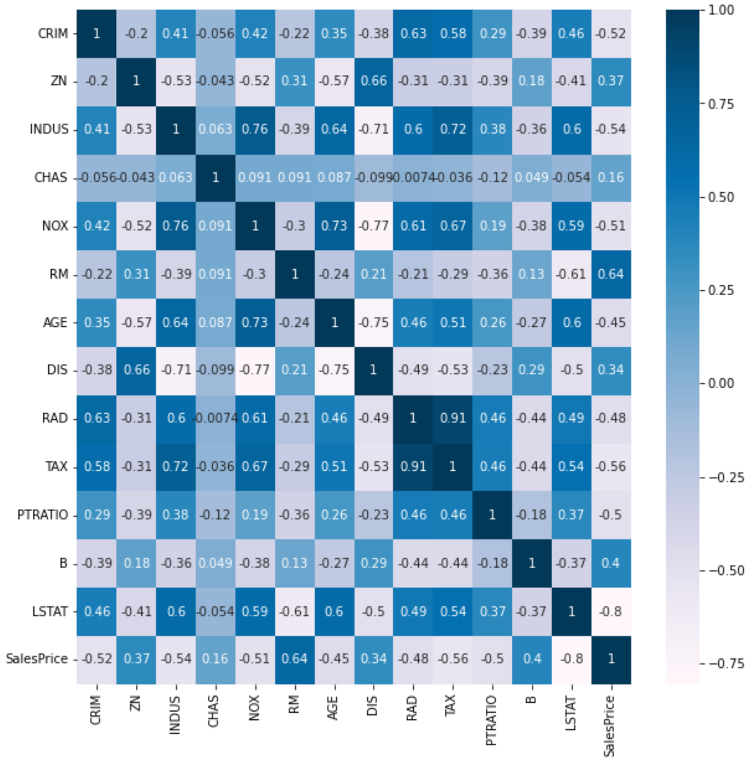
```
C:\Users\gourj\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
 mu = 3.09 and sigma = 0.39
```





Data Correlation

In [20]:
```
plt.figure(figsize=(10,10))
cor = data.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.PuBu)
plt.show()
```



In [21]:
```
cor_target = abs(cor["SalesPrice"])

relevant_features = cor_target[cor_target>0.2]

names = [index for index, value in relevant_features.iteritems()]
```

```
names.remove('SalesPrice')

print(names)
print(len(names))
```

```
['CRIM', 'ZN', 'INDUS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']
12
```

Model Building

In [22]:

```
from sklearn.model_selection import train_test_split

x = data.drop("SalesPrice" , axis=1)
y = data["SalesPrice"]
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [23]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(404, 13)
(102, 13)
(404,)
(102,)
```

In [24]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train, y_train)
```

Out[24]:

```
LinearRegression()
```

In [26]:

```
predictions = lr.predict(x_test)

print("Actual value of the house:- ", y_test[0])
print("Model predicted value of the house:- ", predictions[0])
```

```
Actual value of the house:-  3.2188758248682006
Model predicted value of the house:-  3.36689497999696
```

In [27]:

```
from sklearn.metrics import mean_squared_error

mse=mean_squared_error(y_test, predictions)
rmse = np.sqrt(mse)
print("Mean square error:- ", mse)
print("Root mean square error:- ", rmse)
```

```
Mean square error:-  0.03532837249537253
Root mean square error:-  0.18795843289241515
```