In [200…

```
#!pip install -U scikit-learn
#!pip install numpy
#!pip install pandas
#!pip install matplotlib
#!pip install seaborn
```

In [200…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

In [200…

```
df=pd.read_csv(r"C:\Users\vaibhav kumar\Downloads\Boston_Train_Test.csv", encoding= 'unicode_escape
df
```

Out[200…

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 15 columns

In [201…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  506 non-null    int64  
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64  
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64  
 10  tax         506 non-null    int64  
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

In [201…

```
df.describe()
```

Out[201…

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	252.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	126.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	252.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	378.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	505.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

In [201…

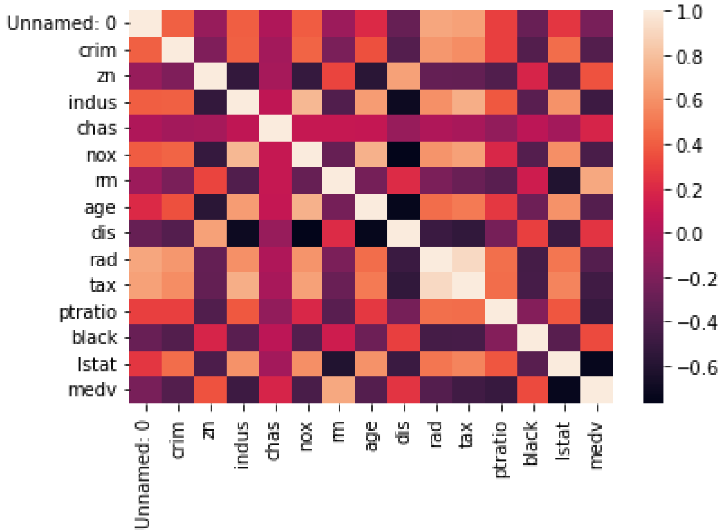
```
df.corr()
```

Out[201…

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rac
Unnamed: 0	1.000000	0.407407	-0.103393	0.399439	-0.003759	0.398736	-0.079971	0.203784	-0.302211	0.686002
crim	0.407407	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505
zn	-0.103393	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948
indus	0.399439	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129
chas	-0.003759	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368
nox	0.398736	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441
rm	-0.079971	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847
age	0.203784	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022
dis	-0.302211	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588
rad	0.686002	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000
tax	0.666626	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228
ptratio	0.291074	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741
black	-0.295041	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413
lstat	0.258465	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676
medv	-0.226604	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626

In [201…

```
sns.heatmap(df.corr())
plt.show()
```



X y Split

In [201…

```
X = df.drop("medv", axis = 1)
y = df["medv"]
```

Test Train Splitting

In [201…

```
from sklearn.model_selection import train_test_split
```

In [201…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [201…

```
X_train
```

Out[201…

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
240	240	0.11329	30.0	4.93	0	0.428	6.897	54.3	6.3361	6	300	16.6	391.25	11.38
380	380	88.97620	0.0	18.10	0	0.671	6.968	91.9	1.4165	24	666	20.2	396.90	17.21
212	212	0.21719	0.0	10.59	1	0.489	5.807	53.8	3.6526	4	277	18.6	390.94	16.03
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
104	104	0.13960	0.0	8.56	0	0.520	6.167	90.0	2.4210	5	384	20.9	392.69	12.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
458	458	7.75223	0.0	18.10	0	0.713	6.301	83.7	2.7831	24	666	20.2	272.21	16.23
91	91	0.03932	0.0	3.41	0	0.489	6.405	73.9	3.0921	2	270	17.8	393.55	8.20
390	390	6.96215	0.0	18.10	0	0.700	5.713	97.0	1.9265	24	666	20.2	394.43	17.11
295	295	0.12932	0.0	13.92	0	0.437	6.678	31.1	5.9604	4	289	16.0	396.90	6.27
23	23	0.98843	0.0	8.14	0	0.538	5.813	100.0	4.0952	4	307	21.0	394.54	19.88

354 rows × 14 columns

In [201…

```
y_train
```

Out[201…

```
240    22.0
380    10.4
212    22.4
2      34.7
104    20.1
       ...
458    14.9
91     22.0
390    15.1
295    28.6
23     14.5
Name: medv, Length: 354, dtype: float64
```

In [201…

```
# Splliting of Train data  is done randomly to avoid potential selection biases arising. This is be
# non-biased resuts, it also means that results can differ from run to run.
```

In [202…

```
X_test
```

Out[202…

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
504	504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48
393	393	8.64476	0.0	18.10	0	0.693	6.193	92.6	1.7912	24	666	20.2	396.90	15.17
248	248	0.16439	22.0	5.86	0	0.431	6.433	49.1	7.8265	7	330	19.1	374.71	9.52
134	134	0.97617	0.0	21.89	0	0.624	5.757	98.4	2.3460	4	437	21.2	262.76	17.31
317	317	0.24522	0.0	9.90	0	0.544	5.782	71.7	4.0317	4	304	18.4	396.90	15.94
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
343	343	0.02543	55.0	3.78	0	0.484	6.696	56.4	5.7321	5	370	17.6	396.90	7.18
61	61	0.17171	25.0	5.13	0	0.453	5.966	93.4	6.8185	8	284	19.7	378.08	14.44
105	105	0.13262	0.0	8.56	0	0.520	5.851	96.7	2.1069	5	384	20.9	394.05	16.47
499	499	0.17783	0.0	9.69	0	0.585	5.569	73.5	2.3999	6	391	19.2	395.77	15.10
450	450	6.71772	0.0	18.10	0	0.713	6.749	92.6	2.3236	24	666	20.2	0.32	17.44

152 rows × 14 columns

In [202…

```
y_test
```

Out[202…

```
504    22.0
393    13.8
248    24.5
134    15.6
317    19.8
       ...
343    23.9
61     16.0
105    19.5
499    17.5
450    13.4
Name: medv, Length: 152, dtype: float64
```

In [202…

```
# Splliting of Test data  is done randomly to avoid potential selection biases arising. This is ber
# non-biased resuts, it also means that results can differ from run to run.
```

Normalization

In [202…

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
```

In [202…

```
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

In [202…

```
X_train.shape
```

Out[202…

```
(354, 14)
```

In [202…

```
X_train
```

Out[202…

```
array([[0.47524752, 0.00120232, 0.3       , ..., 0.42553191, 0.98567372,
        0.27375887],
       [0.75247525, 1.        , 0.        , ..., 0.80851064, 1.        ,
        0.43914894],
       [0.41980198, 0.00237013, 0.        , ..., 0.63829787, 0.98488767,
        0.40567376],
       ...,
       [0.77227723, 0.07818185, 0.        , ..., 0.80851064, 0.993737  ,
        0.43631206],
       [0.58415842, 0.00138249, 0.        , ..., 0.36170213, 1.        ,
        0.12879433],
       [0.04554455, 0.01103868, 0.        , ..., 0.89361702, 0.99401592,
        0.51489362]])
```

In [202…

```
X_test
```

Out[202…

```
array([[ 0.9980198 ,  0.00116073,  0.        , ...,  0.89361702,
         0.99125209,  0.13475177],
       [ 0.77821782,  0.09709398,  0.        , ...,  0.80851064,
         1.        ,  0.3812766 ],
       [ 0.49108911,  0.00177667,  0.22      , ...,  0.69148936,
         0.94373447,  0.22099291],
       ...,
       [ 0.20792079,  0.00141958,  0.        , ...,  0.88297872,
         0.99277347,  0.41815603],
       [ 0.98811881,  0.00192773,  0.        , ...,  0.70212766,
         0.99713474,  0.37929078],
       [ 0.89108911,  0.07543452,  0.        , ...,  0.80851064,
        -0.00557838,  0.44567376]])
```

In [202…

```
X_test.shape
```

Out[202…

```
(152, 14)
```

Linear Regression

In [202…

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

In [203…

```
X = df["nox"]
```

```
y = df["dis"]
```

In [203…

```
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)
```

In [203…

```
X_train = np.array(X_train).reshape(-1, 1)
X_test = np.array(X_test).reshape(-1, 1)
```

In [203…

```
#X_train
```

In [203…

```
model = reg.fit(X_train, y_train)
```

In [203…

```
model.coef_
```

Out[203…

```
array([-13.58585091])
```

In [203…

```
model.intercept_
```

Out[203…

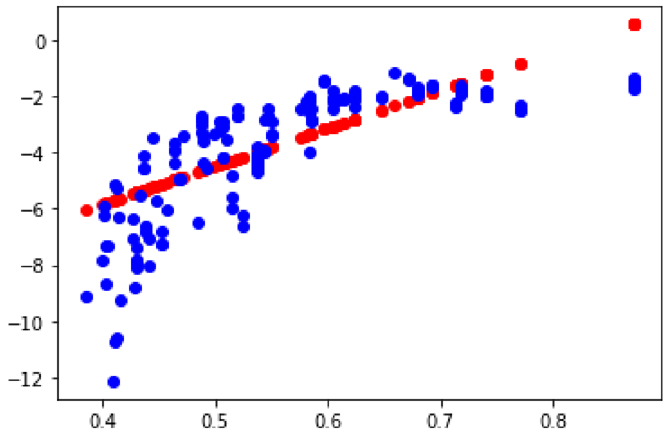
```
11.268763847893577
```

In [203…

```
y_pred = model.predict(X_test)
```

In [203…

```
plt.scatter(X_test, -y_pred, c = "r")
plt.scatter(X_test, -y_test, c = "b")
plt.show()
```



In [203…

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [204…

```
mean_absolute_error(y_test, y_pred)
```

Out[204…

```
1.1609318877854597
```

In [204…

```
mean_squared_error(y_test, y_pred)
```

Out[204…

```
2.281290171525275
```

In [204…

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[204…

```
1.510394045117126
```

In [204…

```
p=r2_score(y_test, y_pred)
```

In [204…

```
p
```

Out[204…

```
0.5644595318232439
```

Multiple Linear Regression

In [204…

```
X = df.drop("medv", axis = 1)
y = df["medv"]
```

In [204…

```
X
```

Out[204…

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67
502	502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08
503	503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64
504	504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48
505	505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88

506 rows × 14 columns

In [204…

```
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=55)
```

In [204…

```
reg = LinearRegression()
model = reg.fit(X_train, y_train)
```

In [204…

```
model.coef_
```

Out[204…

```
array([-7.60758272e-04, -1.33440243e-01,  5.76329038e-02,  3.54041723e-02,
        1.69281625e+00, -1.86212072e+01,  3.56460749e+00, -1.74212324e-03,
       -1.58644494e+00,  3.26202485e-01, -1.27117387e-02, -9.64352356e-01,
        8.58633637e-03, -5.37874887e-01])
```

In [205…

```
model.intercept_
```

Out[205…

```
39.62322051209671
```

In [205…

```
y_pred  = model.predict(X_test)
```

In [205…

```
mean_absolute_error(y_test, y_pred)
```

Out[205…

```
2.9495942985662755
```

In [205…

```
mean_squared_error(y_test, y_pred)
```

Out[205…

```
17.836167010161777
```

In [205…

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[205…

```
4.223288648690944
```

In [205…

```
r2_score(y_test, y_pred)
```

Out[205…

```
0.753625070837884
```

Polynomial Regression

In [205…

```
X = df["nox"]
y = df["dis"]
```

In [205…

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [205…

```
p_test = X_test
```

In [205…

```
from sklearn.preprocessing import PolynomialFeatures
```

In [206…

```
poly = PolynomialFeatures(2)
```

In [206…

```
X_train = np.array(X_train).reshape(-1, 1)
X_test = np.array(X_test).reshape(-1, 1)
```

In [206…

```
poly.fit(X_train)
```

Out[206…

```
▾ PolynomialFeatures
PolynomialFeatures()
```

In [206…

```
X_train = poly.transform(X_train)
X_test = poly.transform(X_test)
```

In [206…

```
#X_test
```

In [206…

```
reg = LinearRegression()
```

In [206…

```
model = reg.fit(X_train, y_train)
```

In [206…

```
model.coef_
```

Out[206…

```
array([  0.        , -73.89605897,  50.02247161])
```

In [206…

```
model.intercept_
```

Out[206…

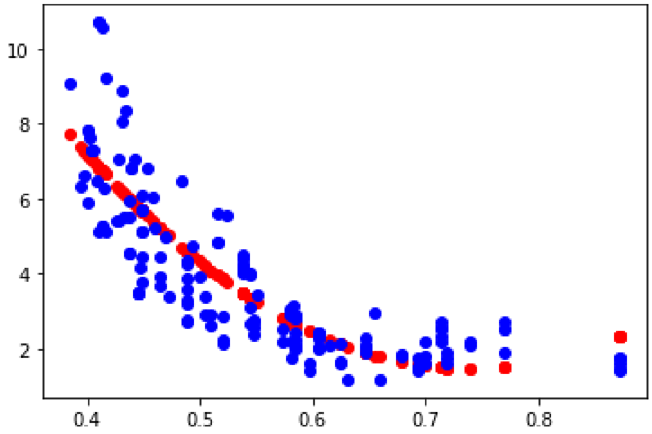
```
28.746142405931856
```

In [206…

```
y_pred = model.predict(X_test)
```

In [207…

```
plt.scatter(X_test[:, 1], y_pred, c = "r")
plt.scatter(X_test[:, 1], y_test, c = "b")
plt.show()
```



In [207…

```
mean_absolute_error(y_test, y_pred)
```

Out[207…

```
0.8763119152067171
```

In [207…

```
mean_squared_error(y_test, y_pred)
```

Out[207…

```
1.3003165078669456
```

In [207…

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[207…

```
1.140314214533409
```

In [207…

```
r2_score(y_test, y_pred)
```

Out[207…

```
0.7193327117492254
```

Observations

In [207…

```
# In case of Multiple Liner Regression the r2 value comes near to 0.754 that
# means 75% data is reliable so house prediction(Medv) in Boston.
# It is nearly a Good data but not perfect.
```

THANK YOU