Assignment no - 4

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

```
df=pd.read_csv("Boston.csv")
df
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	me
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34
3	4	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	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
501	502	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22
502	503	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20
503	504	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23
504	505	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22
505	506	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11

506 rows × 15 columns

```
df.head()
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	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	2	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	3	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	4	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	5	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

```
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
```

```
#null values must be removed before using regression
df.isna().sum()
```

```
Unnamed: 0    0
crim          0
zn            0
indus         0
```

```
chas          0
nox           0
rm            0
age           0
dis           0
rad           0
tax           0
ptratio       0
black         0
lstat         0
medv          0
dtype: int64
```

```
target_features = "medv"
#seperate object from target feature
y = df[target_features]
#sepperate object for input features
x = df.drop(target_features, axis=1)
```

```
x.head()
y.head()
```

```
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: medv, dtype: float64
```

```
#Spliting data for training and testing
#Here, 20% data used for testing and 80% data used for training
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,random_state=2)
```

```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(x_train,y_train)
```

▾ LinearRegression
LinearRegression()

```
train_score=round(regression.score(x_train,y_train)*100,2)
print('Train score of linear regression',train_score)
y_pred = regression.predict(x_test)
```

```
Train score of linear regression 72.91
```

```
from sklearn.metrics import r2_score
score=round(r2_score(y_test,y_pred)*100,2)
print('r_2 score',score)
```

```
r_2 score 78.1
```

```
round(regression.score(x_test,y_test)*100,2)
```

```
78.1
```

```
from sklearn import metrics
print("Mean absolute error on test data of linear regression",metrics.mean_absolute_error(y_test,y_pred))
print("Mean squared error on test data of linear regression",metrics.mean_squared_error(y_test,y_pred))
print("Root mean squared error on test data of linear regression",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Mean absolute error on test data of linear regression 3.0812603233002447
Mean squared error on test data of linear regression 18.321720821929564
Root mean squared error on test data of linear regression 4.280387928906627
```

```
df1=pd.DataFrame({'Actual':y_test,'Predicted':y_pred,'Variance':y_test-y_pred})
df1.head()
```

	Actual	Predicted	Variance
463	20.2	22.935008	-2.735008
152	15.3	21.334270	-6.034270
291	37.3	33.643417	3.656583
183	32.5	31.381211	1.118789
384	8.8	3.218861	5.581139