In [35]:
```
import pandas as pd

from sklearn.datasets import fetch_california_housing
boston = fetch_california_housing()
boston_df=pd.DataFrame(boston.data , columns= boston.feature_names)
boston_df['Price_House'] = boston.target
boston_df
```

Out[35]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitud
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.2
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.2
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.2
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.2
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.2
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.0
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.2
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.2
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.3
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.2

20640 rows × 9 columns

In [36]:
```
x=boston_df.iloc[:,:-1]
```

In [37]:
```
x.shape
```

Out[37]: (20640, 8)

In [38]:
```
y = boston_df.iloc[:,-1]
```

In [39]:
```
y.shape
```

Out[39]: (20640,)

In [40]:
```
#scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_sc = sc.fit_transform(x)
```

In [41]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x_sc, y ,test_size=0.3 , r

```

In [42]:
```
#applying linear regression
from sklearn.metrics import r2_score,mean_absolute_error
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(xtrain,ytrain)
```

Out[42]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [56]:
```
predict = lr.predict(xtest)
print(r2_score(ytest,predict))
```

```
0.592807983357122
```

In [57]:
```
print(mean_absolute_error(ytest,predict))
```

```
0.5355113615407027
```

In [58]:
```
#applying linear regression
from sklearn.linear_model import Ridge
rr = Ridge()
rr.fit(xtrain,ytrain)
```

Out[58]:
```
Ridge()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [59]:
```
predict1 = rr.predict(xtest)
print(r2_score(ytest,predict1))
```

```
0.5929933232418384
```

In [60]:
```
print(mean_absolute_error(ytest,predict1))
```

```
0.5350945553017961
```

In [61]:
```
#now applying repeated k fold
from sklearn.model_selection import RepeatedKFold
cv=RepeatedKFold(n_splits=10,n_repeats=3, random_state=1)
from sklearn.metrics import r2_score
ypred=rr.predict(xtest)
r2_score(ytest,ypred)

from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
x_sc = sc.fit_transform(x)
xtrain,xtest,ytrain,ytest = train_test_split(x_sc,y, test_size=0.25 , r
model1=Ridge()
params={'alpha':[0.00001,0.0001,0.001,0.01,0.1,1,5,10]}

from sklearn.model_selection import GridSearchCV
search = GridSearchCV(model1,params,cv=cv)
result=search.fit(x_sc,y)
result.best_params_

```

Out[61]:
```
{'alpha': 10}
```

In [54]:
```

model2 = Ridge(alpha=10)
model2.fit(xtrain,ytrain)
ypred2=model2.predict(xtest)
r2_score(ytest, ypred2)
```

Out[54]:
```
0.5930434772040074
```

In [62]:
```
#applying lasso
from sklearn.linear_model import Lasso
lr = Lasso()
lr.fit(xtrain,ytrain)
```

Out[62]:
```
Lasso()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [63]:
```
predict2 = lr.predict(xtest)
print(r2_score(ytest,predict2))
```

```
-0.0010175243170345016
```

In [64]:
```
print(mean_absolute_error(ytest,predict2))
```

```
0.9099307895899686
```

In [65]:
```
#now applying repeated k fold
from sklearn.model_selection import RepeatedKFold
cv=RepeatedKFold(n_splits=10,n_repeats=3, random_state=1)
from sklearn.metrics import r2_score
ypred=lr.predict(xtest)
r2_score(ytest,ypred)

from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
x_sc = sc.fit_transform(x)
xtrain,xtest,ytrain,ytest = train_test_split(x_sc,y, test_size=0.25 , r
model1=Ridge()
params={'alpha':[0.00001,0.0001,0.001,0.01,0.1,1,5,10]}

from sklearn.model_selection import GridSearchCV
search = GridSearchCV(model1,params,cv=cv)
result=search.fit(x_sc,y)
result.best_params_

```

Out[65]:
```
{'alpha': 10}
```

In [66]:
```
model2 = Ridge(alpha=10)
model2.fit(xtrain,ytrain)
ypred2=model2.predict(xtest)
r2_score(ytest, ypred2)
```

Out[66]:
```
0.5930434772040074
```

In [ ]:
```

```