Linear Regression, Ridge and Lasso

In [1]:
```python
# House Pricing Dataset
from sklearn.datasets import fetch_california_housing

housing = fetch_california_housing()
```

In [2]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```python
df=fetch_california_housing ()
```

In [4]:
```python
df
```

Out[4]:
```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.55555556,
          37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
          37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
          37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
          39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
          39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
          39.37      , -121.24      ]]),
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing dataset\n--------------------------\n\n**Data Set Characteristics:**\n\n    :Number of Insta
nces: 20640\n\n    :Number of Attributes: 8 numeric, predictive attributes and the target\n\n    :Attribute Information:\n        - MedInc        median incom
e in block group\n        - HouseAge      median house age in block group\n        - AveRooms      average number of rooms per household\n        - AveB
edrms     average number of bedrooms per household\n        - Population    block group population\n        - AveOccup      average number of househol
d members\n        - Latitude      block group latitude\n        - Longitude     block group longitude\n\n    :Missing Attribute Values: None\n\nThis dataset
was obtained from the StatLib repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\nThe target variable is the median house va
lue for California districts,\nexpressed in hundreds of thousands of dollars ($100,000).\n\nThis dataset was derived from the 1990 U.S. census, using o
ne row per census\nblock group. A block group is the smallest geographical unit for which the U.S.\nCensus Bureau publishes sample data (a block gr
oup typically has a population\nof 600 to 3,000 people).\n\nAn household is a group of people residing within a home. Since the average\nnumber of r
ooms and bedrooms in this dataset are provided per household, these\ncolumns may take surpinsingly large values for block groups with few househo
lds\nand many empty houses, such as vacation resorts.\n\nIt can be downloaded/loaded using the\n:func:`sklearn.datasets.fetch_california_housing` f
unction.\n\n.. topic:: References\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,\n      Statistics and Probability Letters, 33 (
1997) 291-297\n'}
```

In [5]:
```python
dataset=pd.DataFrame(df.data)
dataset.columns=df.feature_names
dataset.head()
```

Out[5]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

In [6]:
```python
dataset['price']=df.target
dataset.head()
```

Out[6]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	price
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

In [7]:
```python
## Dividing the dataset into dependent and independent features
```

```
X=dataset.iloc[:,:-1] ##Independent Features
y=dataset.iloc[:,-1] ##Dependent Features
```

In [8]:
```
X.head()
```

Out[8]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

Linear Regression

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score  # Corrected import statement
lin_reg = LinearRegression()
mse = cross_val_score(lin_reg, X, y, scoring='neg_mean_squared_error', cv=5)
mean_mse=np.mean(mse)
print(mean_mse)
```

Ridge Regression

In [11]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

ridge = Ridge()
params = {'alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 1, 4, 10, 20]}

Ridge_regressor = GridSearchCV(Ridge(), params, scoring='neg_mean_squared_error', cv=5)
Ridge_regressor.fit(X, y)

print(Ridge_regressor.best_params_)
print(Ridge_regressor.best_score_)
```

```
{'alpha': 20}
-0.5581020035625643
```

In [12]:
```
#Ridge Regression
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV  # Corrected import statement

lasso = Lasso()
params = {'alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 1, 4, 10, 20]}

Lasso_regressor = GridSearchCV(Lasso(), params, scoring='neg_mean_squared_error', cv=5)
Lasso_regressor.fit(X, y)
```

```
D:\Anaconda\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:631: ConvergenceWarning: Objective did not converge. You might want to incre
ase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 2.959e+03, tolerance: 2.228e+00
  model = cd_fast.enet_coordinate_descent(
D:\Anaconda\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:631: ConvergenceWarning: Objective did not converge. You might want to incre
ase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.570e+03, tolerance: 2.256e+00
  model = cd_fast.enet_coordinate_descent(
D:\Anaconda\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:631: ConvergenceWarning: Objective did not converge. You might want to incre
ase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.960e+03, tolerance: 2.110e+00
  model = cd_fast.enet_coordinate_descent(
D:\Anaconda\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:631: ConvergenceWarning: Objective did not converge. You might want to incre
ase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.206e+03, tolerance: 2.236e+00
  model = cd_fast.enet_coordinate_descent(
D:\Anaconda\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:631: ConvergenceWarning: Objective did not converge. You might want to incre
ase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.816e+03, tolerance: 2.128e+00
  model = cd_fast.enet_coordinate_descent(
```

Out[12]:

```
▸ GridSearchCV
▸ estimator: Lasso
  ▸ Lasso
```

In [13]:
```
print(Lasso_regressor.best_params_)
print(Lasso_regressor.best_score_)
```

```
{'alpha': 0.001}
-0.558275929386898
```

In [21]:
```
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
```

```
from sklearn.model_selection import train_test_split

# Assuming you have a dataset 'X' containing features and 'y' containing target values
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the Lasso regression model
lasso_regressor = Lasso(alpha=0.01)  # You should specify the appropriate alpha value
lasso_regressor.fit(X_train, y_train)

# Now you can make predictions using the trained model
y_pred = lasso_regressor.predict(X_test)

# Calculate the R-squared (R2) score
r2_score1 = r2_score(y_test, y_pred)
```

In [22]:
```
print(r2_score1)
```

0.5845196673976367

In [ ]: