import libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

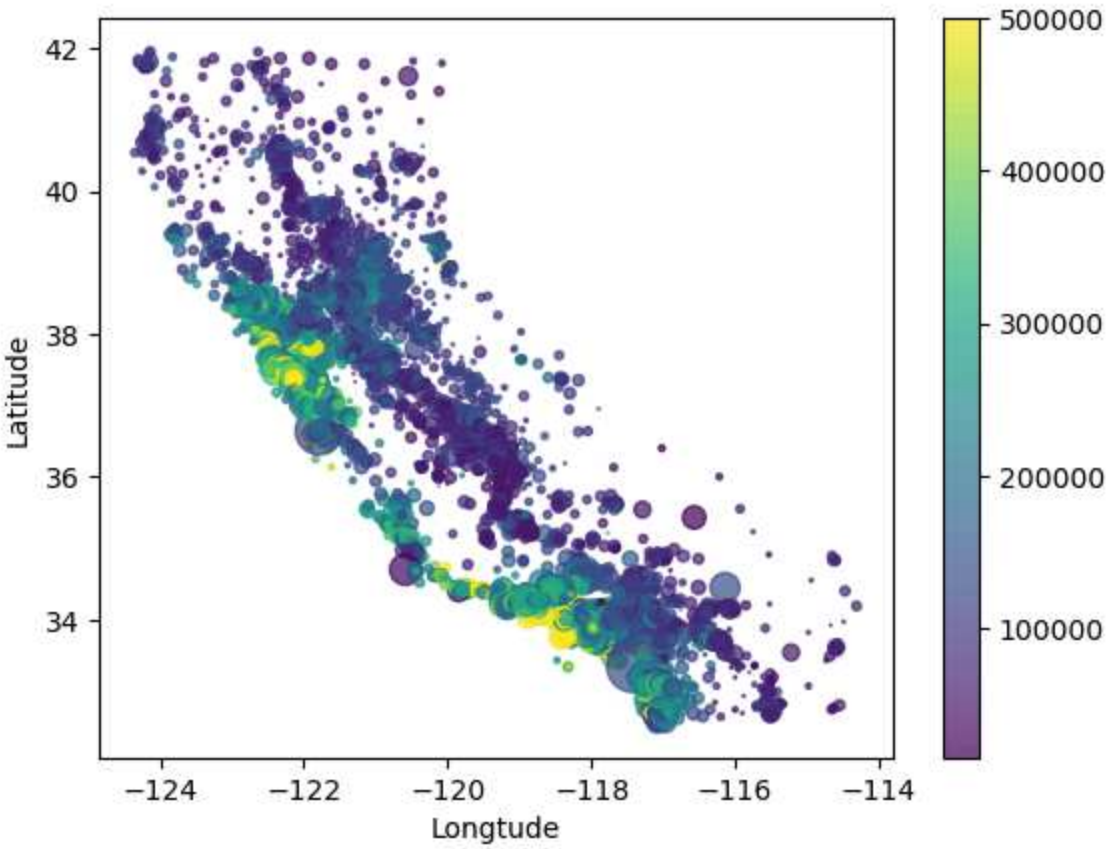
import dataset

In [2]:
```
dataset = pd.read_csv('California_Houses.csv')
features = dataset.iloc[: , 1:]
depending_v = dataset.iloc[: , :1]
print(features)
print("*"*80)
print(depending_v)
```

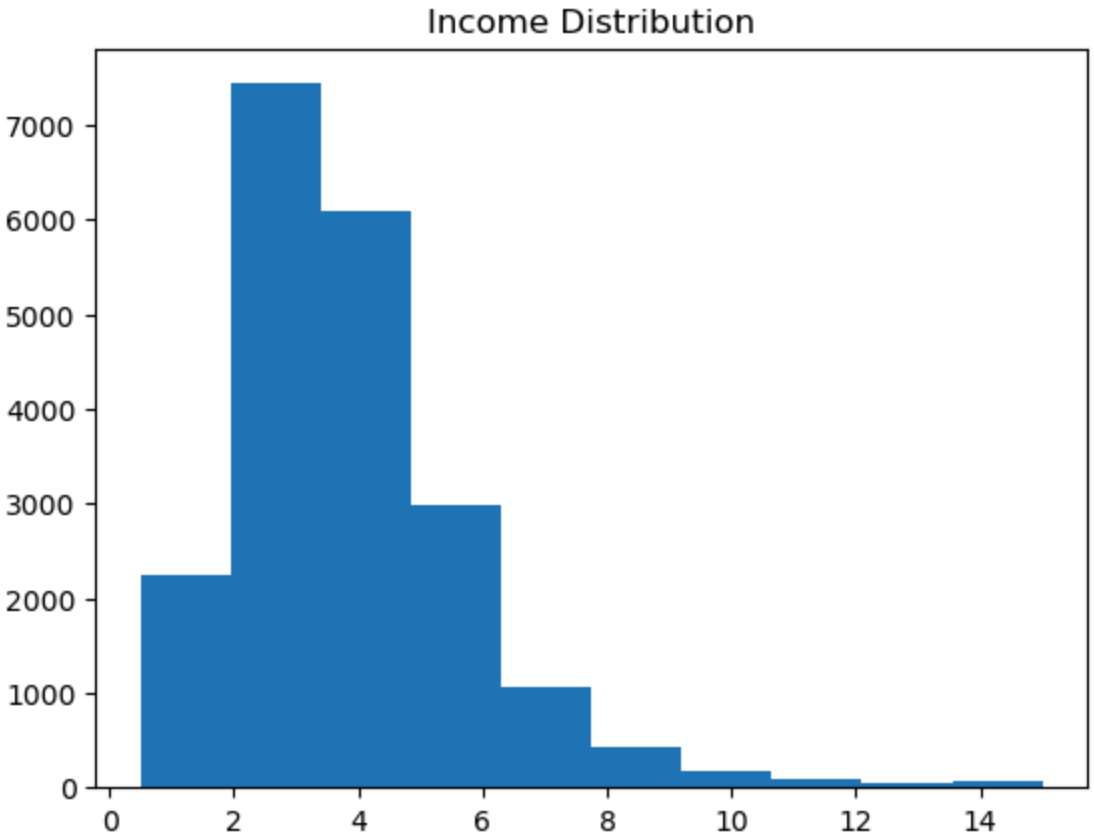
```
       Median_Income  Median_Age  Tot_Rooms  Tot_Bedrooms  Population  \
0             8.3252          41        880           129         322
1             8.3014          21       7099          1106        2401
2             7.2574          52       1467           190         496
3             5.6431          52       1274           235         558
4             3.8462          52       1627           280         565
...              ...         ...        ...           ...         ...
20635         1.5603          25       1665           374         845
20636         2.5568          18        697           150         356
20637         1.7000          17       2254           485        1007
20638         1.8672          18       1860           409         741
20639         2.3886          16       2785           616        1387

       Households  Latitude  Longitude  Distance_to_coast  Distance_to_LA  \
0             126     37.88    -122.23        9263.040773   556529.158342
1            1138     37.86    -122.22       10225.733072   554279.850069
2             177     37.85    -122.24        8259.085109   554610.717069
3             219     37.85    -122.25        7768.086571   555194.266086
4             259     37.85    -122.25        7768.086571   555194.266086
...           ...       ...        ...                ...             ...
20635         330     39.48    -121.09      162031.481121   654530.186299
20636         114     39.49    -121.21      160445.433537   659747.068444
20637         433     39.43    -121.22      153754.341182   654042.214020
20638         349     39.43    -121.32      152005.022239   657698.007703
20639         530     39.37    -121.24      146866.196892   648723.337126

       Distance_to_SanDiego  Distance_to_SanJose  Distance_to_SanFrancisco
0             735501.806984         67432.517001              21250.213767
1             733236.884360         65049.908574              20880.600400
2             733525.682937         64867.289833              18811.487450
3             734095.290744         65287.138412              18031.047568
4             734095.290744         65287.138412              18031.047568
...                     ...                  ...                       ...
20635         830631.543047        248510.058162             222619.890417
20636         836245.915229        246849.888948             218314.424634
20637         830699.573163        240172.220489             212097.936232
20638         834672.461887        238193.865909             207923.199166
20639         825569.179028        233282.769063             205473.376575

[20640 rows x 13 columns]
**********************************************************************************
       Median_House_Value
0                452600.0
1                358500.0
2                352100.0
3                341300.0
4                342200.0
...                   ...
20635             78100.0
20636             77100.0
20637             92300.0
20638             84700.0
20639             89400.0

[20640 rows x 1 columns]
```

Visualize Data

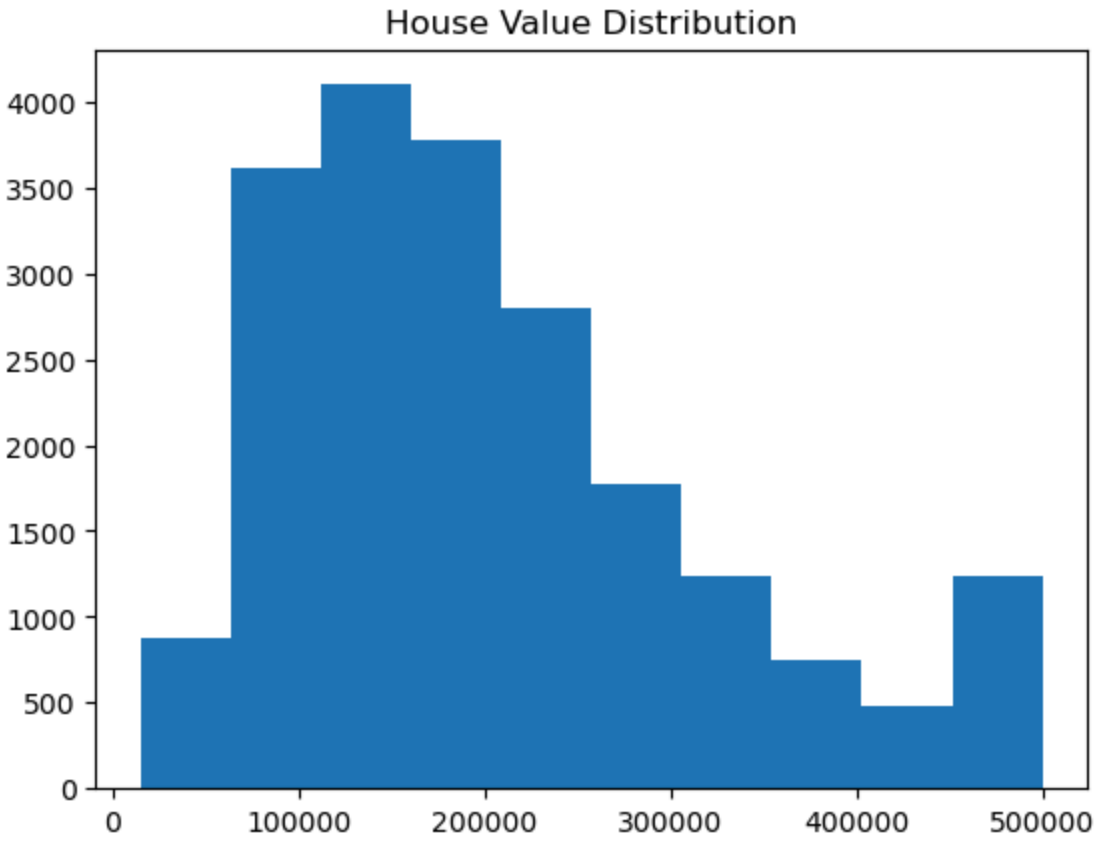
In [3]:
```
plt.scatter(x = dataset['Longitude'],y=dataset['Latitude'],alpha=0.7,s=dataset['Pop
plt.xlabel("Longtude")
plt.ylabel("Latitude")
plt.colorbar()
plt.show()
```



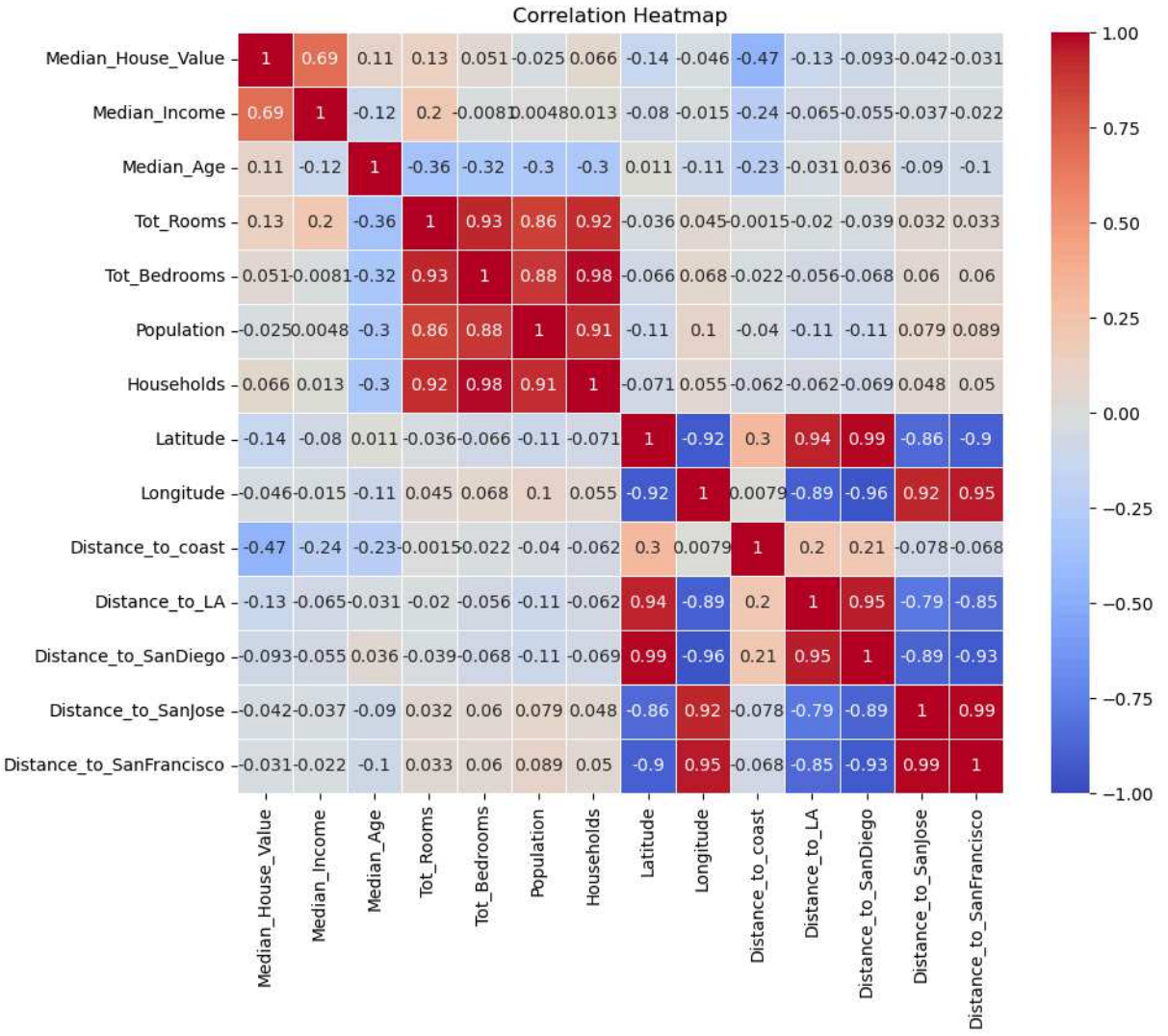
In [4]:
```
plt.hist(dataset['Median_Income'])
plt.title("Income Distribution")
plt.show()
```



In [5]:
```
plt.hist(dataset['Median_House_Value'])
plt.title("House Value Distribution")
plt.show()
```



In [6]:
```
import seaborn as sns
correlation_matrix = dataset.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", vmin=-1, vmax=1, linew

plt.title("Correlation Heatmap")

plt.show()
```



Handling missing data

In [7]:
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=  np.nan ,strategy='mean')
imputer.fit(features.iloc[:,:])
features.iloc[: ,:] = imputer.transform(features.iloc[: ,:])
print(features)
```

```
       Median_Income  Median_Age  Tot_Rooms  Tot_Bedrooms  Population  \
0             8.3252          41        880           129         322
1             8.3014          21       7099          1106        2401
2             7.2574          52       1467           190         496
3             5.6431          52       1274           235         558
4             3.8462          52       1627           280         565
...              ...         ...        ...           ...         ...
20635         1.5603          25       1665           374         845
20636         2.5568          18        697           150         356
20637         1.7000          17       2254           485        1007
20638         1.8672          18       1860           409         741
20639         2.3886          16       2785           616        1387

       Households  Latitude  Longitude  Distance_to_coast  Distance_to_LA  \
0             126     37.88    -122.23        9263.040773   556529.158342
1            1138     37.86    -122.22       10225.733072   554279.850069
2             177     37.85    -122.24        8259.085109   554610.717069
3             219     37.85    -122.25        7768.086571   555194.266086
4             259     37.85    -122.25        7768.086571   555194.266086
...           ...       ...        ...                ...             ...
20635         330     39.48    -121.09      162031.481121   654530.186299
20636         114     39.49    -121.21      160445.433537   659747.068444
20637         433     39.43    -121.22      153754.341182   654042.214020
20638         349     39.43    -121.32      152005.022239   657698.007703
20639         530     39.37    -121.24      146866.196892   648723.337126

       Distance_to_SanDiego  Distance_to_SanJose  Distance_to_SanFrancisco
0              735501.806984         67432.517001              21250.213767
1              733236.884360         65049.908574              20880.600400
2              733525.682937         64867.289833              18811.487450
3              734095.290744         65287.138412              18031.047568
4              734095.290744         65287.138412              18031.047568
...                      ...                  ...                       ...
20635          830631.543047        248510.058162             222619.890417
20636          836245.915229        246849.888948             218314.424634
20637          830699.573163        240172.220489             212097.936232
20638          834672.461887        238193.865909             207923.199166
20639          825569.179028        233282.769063             205473.376575

[20640 rows x 13 columns]
```

Feature scaling

In [8]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
features = sc.fit_transform(features)
# Print column names of the dataset
print(features)
```

```
[[ 2.34476576  0.98214266 -0.8048191  ...  1.16566824 -1.29754558
  -1.46107412]
 [ 2.33223796 -0.60701891  2.0458901  ...  1.1578418  -1.30851803
  -1.46255188]
 [ 1.7826994   1.85618152 -0.53574589 ...  1.15883974 -1.30935903
  -1.47082449]
 ...
 [-1.14259331 -0.92485123 -0.17499526 ...  1.49462431 -0.50204022
  -0.69803768]
 [-1.05458292 -0.84539315 -0.35559977 ...  1.50835264 -0.51115099
  -0.71472888]
 [-0.78012947 -1.00430931  0.06840827 ...  1.47689622 -0.5337677
  -0.72452362]]
```

Split dataset randomly

In [9]:
```
from sklearn.model_selection import train_test_split
X_train, X_temp, y_train, y_temp = train_test_split(features, depending_v, test_siz
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.50, ran
```

Linear Regression

In [10]:
```
linear_model = LinearRegression()
lr = linear_model.fit(X_train, y_train)


# Predictions
y_traind_pred_linear = linear_model.predict(X_train)
y_val_pred_linear = linear_model.predict(X_val)

#Calculate metrics
linear_mae = mean_absolute_error(y_val, y_val_pred_linear)
linear_mse = mean_squared_error(y_val, y_val_pred_linear)

# Print
print(f'Linear Regression MAE={linear_mae:.2f}')
print(f'Linear Regression MSE={linear_mse:.2f}')
print(f"R2 train  = {lr.score(X_train,y_train)}")#R2 score
print(f"R2 Validation  = {lr.score(X_val,y_val)}")#R2 score
```

```
Linear Regression MAE=50790.06
Linear Regression MSE=4907211997.37
R2 train  = 0.6469080431646521
R2 Validation  = 0.6233241175944966
```

Lasso

In [25]:
```
# List for different learning rates
alphas = [0.005,0.01,0.05, 0.1, 1, 10, 100]
```

```python
# Loop over different learning rates
for LR in alphas:
    # Train the model on current alpha
    lasso_model = Lasso(alpha = LR)
    lr = lasso_model.fit(X_train, y_train)

    # Prediction
    y_train_pred_lasso = lasso_model.predict(X_train)
    y_val_pred_lasso = lasso_model.predict(X_val)

    # Calculate metrics
    lasso_mae = mean_absolute_error(y_val, y_val_pred_lasso)
    lasso_mse = mean_squared_error(y_val, y_val_pred_lasso)

    # Print
    print(f'Lasso Regression with alpha={LR}:')
    print(f'  MAE = {lasso_mae:.2f}')
    print(f'  MSE = {lasso_mse:.2f}')
    print(f"R2 train  = {lr.score(X_train,y_train)}")#R2 score
    print(f"R2 Validation  = {lr.score(X_val,y_val)}")#R2 score
    print('-' * 30)
```

```
C:\Users\abdel\anaconda3\envs\torch-2024\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.408e+13, tolerance: 1.936e+10
  model = cd_fast.enet_coordinate_descent(
C:\Users\abdel\anaconda3\envs\torch-2024\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.399e+13, tolerance: 1.936e+10
  model = cd_fast.enet_coordinate_descent(
C:\Users\abdel\anaconda3\envs\torch-2024\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:697: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 3.329e+13, tolerance: 1.936e+10
  model = cd_fast.enet_coordinate_descent(
Lasso Regression with alpha=0.005:
  MAE = 50790.11
  MSE = 4907208093.09
R2 train  = 0.6469078639215751
R2 Validation  = 0.6233244172862968
------------------------------
Lasso Regression with alpha=0.01:
  MAE = 50790.12
  MSE = 4907208186.23
R2 train  = 0.6469078627217622
R2 Validation  = 0.6233244101365578
------------------------------
Lasso Regression with alpha=0.05:
  MAE = 50790.15
  MSE = 4907208934.07
R2 train  = 0.6469078529301964
R2 Validation  = 0.6233243527326693
------------------------------
```

```
C:\Users\abdel\anaconda3\envs\torch-2024\Lib\site-packages\sklearn\linear_model\_coo
rdinate_descent.py:697: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations, check the scale of the features or consider
increasing regularisation. Duality gap: 3.244e+13, tolerance: 1.936e+10
  model = cd_fast.enet_coordinate_descent(
C:\Users\abdel\anaconda3\envs\torch-2024\Lib\site-packages\sklearn\linear_model\_coo
rdinate_descent.py:697: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations, check the scale of the features or consider
increasing regularisation. Duality gap: 2.135e+13, tolerance: 1.936e+10
  model = cd_fast.enet_coordinate_descent(
C:\Users\abdel\anaconda3\envs\torch-2024\Lib\site-packages\sklearn\linear_model\_coo
rdinate_descent.py:697: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations, check the scale of the features or consider
increasing regularisation. Duality gap: 2.416e+12, tolerance: 1.936e+10
  model = cd_fast.enet_coordinate_descent(
Lasso Regression with alpha=0.1:
  MAE = 50790.18
  MSE = 4907209875.58
R2 train  = 0.6469078402080843
R2 Validation  = 0.6233242804628666
------------------------------
Lasso Regression with alpha=1:
  MAE = 50790.83
  MSE = 4907228097.33
R2 train  = 0.6469075195056713
R2 Validation  = 0.6233228817674693
------------------------------
Lasso Regression with alpha=10:
  MAE = 50797.50
  MSE = 4907543143.09
R2 train  = 0.6468947559180014
R2 Validation  = 0.6232986989650667
------------------------------
Lasso Regression with alpha=100:
  MAE = 50976.07
  MSE = 4923965930.53
R2 train  = 0.6458121848949574
R2 Validation  = 0.6220380915253566
------------------------------
C:\Users\abdel\anaconda3\envs\torch-2024\Lib\site-packages\sklearn\linear_model\_coo
rdinate_descent.py:697: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations, check the scale of the features or consider
increasing regularisation. Duality gap: 2.120e+10, tolerance: 1.936e+10
  model = cd_fast.enet_coordinate_descent(
```

Ridge Regression

In [24]:
```python
# List for different learning rates
alphas = [0.005,0.01,0.05, 0.1, 1, 10, 100]

# Loop over different learning rates
for LR in alphas:
    # Train the model on current alpha
```

```
    ridge_model = Ridge(alpha = LR)
    lr = ridge_model.fit(X_train, y_train)

    # Prediction
    y_train_pred_ridge = ridge_model.predict(X_train)
    y_test_pred_ridge = ridge_model.predict(X_test)

    # Calculate metrics
    ridge_mae = mean_absolute_error(y_test, y_test_pred_ridge)
    ridge_mse = mean_squared_error(y_test, y_test_pred_ridge)

    # Print
    print(f'Ridge Regression with alpha={LR}:')
    print(f'  MAE = {ridge_mae:.2f}')
    print(f'  MSE = {ridge_mse:.2f}')
    print(f"R2 train  = {lr.score(X_train,y_train)}")#R2 score
    print(f"R2 Validation  = {lr.score(X_val,y_val)}")#R2 score
    print('-' * 30)
```

```
Ridge Regression with alpha=0.005:
  MAE = 48782.04
  MSE = 4400950872.91
R2 train  = 0.6469080431076897
R2 Validation  = 0.6233241032244756
------------------------------
Ridge Regression with alpha=0.01:
  MAE = 48782.05
  MSE = 4400948597.34
R2 train  = 0.6469080429368658
R2 Validation  = 0.6233240887264477
------------------------------
Ridge Regression with alpha=0.05:
  MAE = 48782.14
  MSE = 4400930469.29
R2 train  = 0.6469080374827213
R2 Validation  = 0.623323968147329
------------------------------
Ridge Regression with alpha=0.1:
  MAE = 48782.25
  MSE = 4400907999.72
R2 train  = 0.6469080205003196
R2 Validation  = 0.6233238059993311
------------------------------
Ridge Regression with alpha=1:
  MAE = 48784.32
  MSE = 4400538182.34
R2 train  = 0.6469058861887672
R2 Validation  = 0.6233188363537969
------------------------------
Ridge Regression with alpha=10:
  MAE = 48818.48
  MSE = 4399297220.15
R2 train  = 0.6467674001304082
R2 Validation  = 0.6231403938020457
------------------------------
Ridge Regression with alpha=100:
  MAE = 49140.47
  MSE = 4423446008.95
R2 train  = 0.6447384288181826
R2 Validation  = 0.6208715623737877
------------------------------
```

In [ ]: