Lab3_Ex3

May 30, 2020

[32]:

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

[33]:

```
#import Boston_Housing.csv
dataset = pd.read_csv('E:/University Works/3rd Year/Semester 6/CO 544 - Machine␣
 ↪Learning and Data Mining/Lab/3/Boston_Housing.csv',sep=',')
print(dataset.head())
```

```
      RM  LSTAT  PTRATIO      MEDV
0  6.575   4.98     15.3  504000.0
1  6.421   9.14     17.8  453600.0
2  7.185   4.03     17.8  728700.0
3  6.998   2.94     18.7  701400.0
4  7.147   5.33     18.7  760200.0
```

[34]:

```
print ("Dataset Length: ", len(dataset))
print ("Dataset Shape: ", dataset.shape)
```

```
Dataset Length:  489
Dataset Shape:  (489, 4)
```

[35]:

```
#split 80% train 20% test
data_train = dataset.values[0:round(len(dataset)*0.8),0:4]
data_test = dataset.values[round(len(dataset)*0.8):,0:4]
df_train = pd.DataFrame(data_train)
df_test = pd.DataFrame(data_test)
```

[36]:

```
print("df_train shape = ",df_train.shape)
print("df_test shape = ",df_test.shape)
```

```
df_train shape =  (391, 4)
df_test shape =  (98, 4)
```

[37]:

```
#creating 50 samples (100 instances in each)
#with replacement (means not removing the row that was selected so that it is␣
 ↪available for future selections)
def subsample(df_train):
    sampleList = list()
    while len(sampleList) < 50:
        sampleList.append(df_train.sample(n=100,replace=True))
    return sampleList
```

[38]:

```
sampleList = subsample(df_train)
```

[39]:

```
#For training samples
def createTrainXY(sample):
    xylist = list()
    X = sample.values[:,0:3]
    X = np.hstack((np.ones((100,1)),X))
    xylist.append(X)

    Y = sample.values[:,3]
    xylist.append(Y)

    return xylist
```

[40]:

```
#For test values
X_test = df_test.values[:,0:3]
Y_test = df_test.values[:,3]
X_test = np.hstack((np.ones((98,1)),X_test))
```

[41]:

```
#Determine linear regression model
def modelLR(sample):
    xylist = createTrainXY(sample)
    b_hat = np.dot(np.dot(np.linalg.inv(np.dot(xylist[0].
 ↪T,xylist[0])),xylist[0].T),xylist[1])
    return b_hat
```

[42]:

```
#Building all the linear regression models of samples
beta_hat = list()
i = 0
while i < 50:
    b_hat = modelLR(sampleList[i])
    beta_hat.append(b_hat)
    i = i + 1
```

[43]:

```
#Find the response variable(y) for test data
#for each model
y_hat_X_test_list = list()
i = 0
while i < 50:
    p = np.dot(X_test,beta_hat[i])
    y_hat_X_test_list.append(p)
    i = i + 1
```

[44]:

```python
print("y_hat_X_test_list = ",y_hat_X_test_list[0].shape)
```

```
y_hat_X_test_list =  (98,)
```

[45]:

```python
y_hat_X_test = np.zeros([98,1],dtype=float)
```

[46]:

```python
#Simple voting
i = 0
while i < 50:
    y_hat_X_test = y_hat_X_test + y_hat_X_test_list[i]
    i = i + 1
```

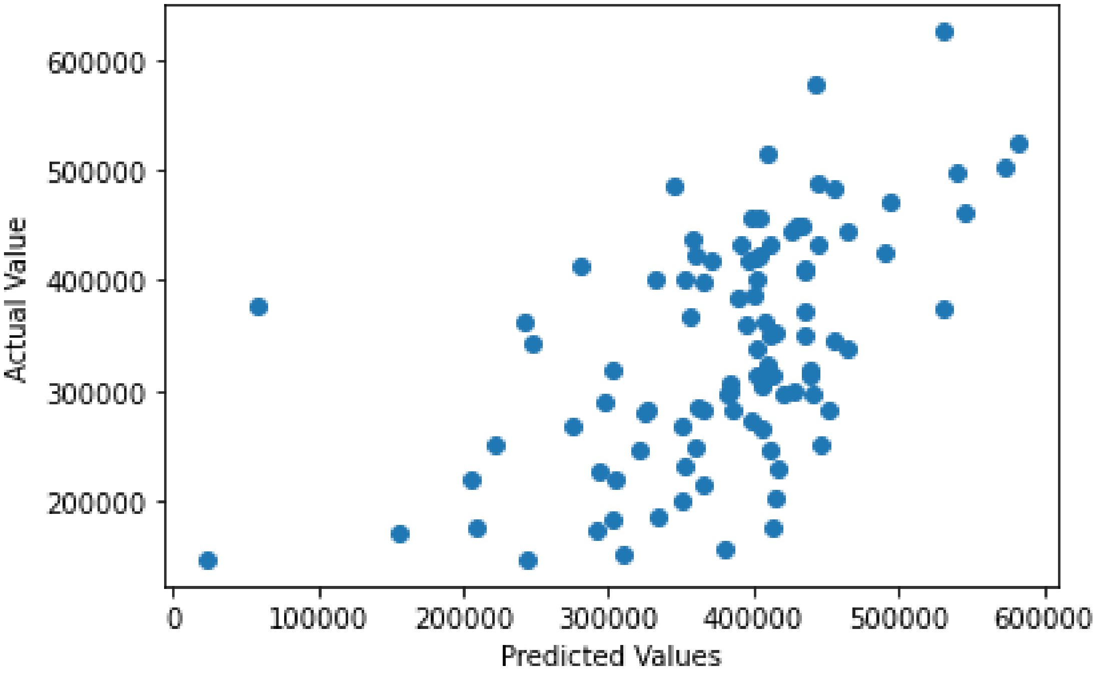
[47]:

```python
y_hat_X_test_avg = y_hat_X_test[0]/50
print("Predictions for test set = \n",y_hat_X_test_avg)
```

```
Predictions for test set =
 [243020.33741483 442451.16045007 407066.02168212 407369.26167437
  59024.73363714 248479.28009813  22985.37671426 310971.34349787
 379860.69195582 205562.75041927 333944.70783792 412354.36427321
 436077.84201921 384247.62941616 358466.16095485 326096.1546597
 321261.02572205 292756.54746425 365293.39959746 417075.75023552
 352348.80580417 350121.89697645 405131.38059934 441129.14201809
 464869.97977766 427973.00917111 411835.73355692 385555.11655693
 414134.03328188 302856.41628181 209501.13248612 276185.66167962
 304718.56974044 394556.20856507 400150.78870266 409557.23239926
 293484.49752803 359934.88663057 402977.25202222 405122.26165289
 382302.40625468 397707.22110294 452141.94721421 438909.20620953
 402751.66938807 531088.0083112  438216.07371502 420618.28972896
 350739.81066661 365072.81537621 412668.33141281 401801.47688649
 456016.49665587 435501.23120673 434859.49595786 490624.18894175
 429696.9653163  370702.11554092 364964.53476129 332757.35143874
 353114.71854927 359784.05885327 397021.25671189 435136.47583764
 445015.07763891 530782.69617055 296789.50671851 325372.61775592
 410991.84706401 221308.88924651 383212.58768978 432952.0403934
 455988.3594793  540542.24019388 582234.32867159 404807.92810793
 390514.25166975 465333.21402016 402971.86344072 411548.26914057
 302080.30364471 243780.82983329 155634.76813834 361604.29243618
 404564.85617293 399337.08422062 409636.15425446 344288.2723263
 280513.59254492 390118.65064543 426214.8909376  355660.73693645
 414282.88290208 493123.45699202 445189.37191989 572965.4846226
 544809.69353152 445969.10939791]
```

[48]:

```python
#visualize residual error for test data
plt.scatter(y_hat_X_test_avg,Y_test)
plt.xlabel("Predicted Values")
plt.ylabel("Actual Value")
plt.show()
```


600000
500000
400000
300000
200000
Actual Value
0
100000
200000
300000
400000
500000
600000
Predicted Values

[ ]: