Name : Mithun Muralidhar

ASU ID: 1211309824

Problem 1: Ridge Regression, implement your own Gradient Optimization

Part3 : Analytical solution

1) The least sqaure analytical solution obtained is

[[1.10034069] [0.89299444]]

2)Ridge Regression analytical solution is

[[1.0053259] [0.95681108]]

Part 3: Gradient descent

Gradient descent with 100 iterations. alpha=0.00010 and initial beta=[-1.00000,-1.50000]

Iteration 0 --- beta:[-1.00000, -1.50000] --- Cost: 19409.36672

Iteration 10 --- beta:[1.19608, 0.73077] --- Cost: 315.42673

Iteration 20 --- beta:[1.19783, 0.76408] --- Cost: 313.73328

Iteration 30 --- beta:[1.18338, 0.77880] --- Cost: 313.30934

Iteration 40 --- beta:[1.16990, 0.79228] --- Cost: 312.94722

Iteration 50 --- beta:[1.15744, 0.80473] --- Cost: 312.63786

Iteration 60 --- beta:[1.14593, 0.81625] --- Cost: 312.37357

Iteration 70 --- beta:[1.13528, 0.82689] --- Cost: 312.14778

Iteration 80 --- beta:[1.12544, 0.83672] --- Cost: 311.95489

Iteration 90 --- beta:[1.11635, 0.84582] --- Cost: 311.79010

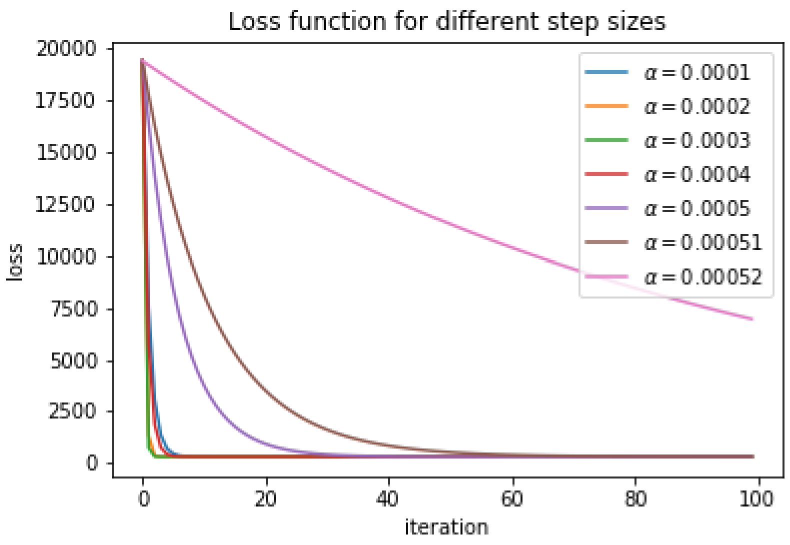
final beta: [1.10794, 0.85422]

Gradient descent with 100 iterations. alpha=0.00020 and initial beta=[-1.00000,-1.50000]

Iteration 0 --- beta:[-1.00000, -1.50000] --- Cost: 19409.36672

```
Iteration 10 --- beta:[ 1.19785, 0.76434] --- Cost: 313.72957

Iteration 20 --- beta:[ 1.16970, 0.79248] --- Cost: 312.94192

Iteration 30 --- beta:[ 1.14566, 0.81651] --- Cost: 312.36777

Iteration 40 --- beta:[ 1.12514, 0.83702] --- Cost: 311.94925

Iteration 50 --- beta:[ 1.10762, 0.85454] --- Cost: 311.64417

Iteration 60 --- beta:[ 1.09267, 0.86949] --- Cost: 311.42179

Iteration 70 --- beta:[ 1.07989, 0.88226] --- Cost: 311.25968

Iteration 80 --- beta:[ 1.06899, 0.89316] --- Cost: 311.14152

Iteration 90 --- beta:[ 1.05968, 0.90247] --- Cost: 311.05538

final beta: [ 1.05173, 0.91041

Gradient descent with 100 iterations. alpha=0.00050 and initial beta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672

Iteration 10 --- beta:[ 0.20856, -0.14246] --- Cost: 3760.25597

Iteration 20 --- beta:[ 0.70386, 0.45262] --- Cost: 934.12165

Iteration 30 --- beta:[ 0.90209, 0.71771] --- Cost: 423.58039

Iteration 40 --- beta:[ 0.97813, 0.83855] --- Cost: 331.27986

Iteration 50 --- beta:[ 1.00494, 0.89539] --- Cost: 314.56090

Iteration 60 --- beta:[ 1.01265, 0.92322] --- Cost: 311.51814

Iteration 70 --- beta:[ 1.01346, 0.93752] --- Cost: 310.95794

Iteration 80 --- beta:[ 1.01215, 0.94525] --- Cost: 310.85193

Iteration 90 --- beta:[ 1.01048, 0.94964] --- Cost: 310.83061

final beta: [ 1.00903, 0.95225]

Gradient descent with 100 iterations. alpha=0.00051 and initial beta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672

Iteration 10 --- beta:[ -0.27220, -0.62088] --- Cost: 8130.69140

Iteration 20 --- beta:[ 0.19162, -0.05646] --- Cost: 3512.66020

Iteration 30 --- beta:[ 0.48715, 0.30595] --- Cost: 1621.81413

Iteration 40 --- beta:[ 0.67541, 0.53868] --- Cost: 847.60895
Iteration 50 --- beta:[ 0.79532, 0.68816] --- Cost: 530.61089

Iteration 60 --- beta:[ 0.87168, 0.78418] --- Cost: 400.81593

Iteration 70 --- beta:[ 0.92030, 0.84586] --- Cost: 347.67126

Iteration 80 --- beta:[ 0.95124, 0.88549] --- Cost: 325.91108

Iteration 90 --- beta:[ 0.97093, 0.91096] --- Cost: 317.00131

final beta: [ 0.98346, 0.92733]

Gradient descent with 100 iterations. alpha=0.00052 and initial beta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672

Iteration 10 --- beta:[ -0.96120, -1.30761] --- Cost: 17479.58523

Iteration 20 --- beta:[ -0.90223, -1.14736] --- Cost: 15745.65406

Iteration 30 --- beta:[ -0.83171, -1.01002] --- Cost: 14187.21191

Iteration 40 --- beta:[ -0.75519, -0.88944] --- Cost: 12786.28682

Iteration 50 --- beta:[ -0.67628, -0.78147] --- Cost: 11526.86622

Iteration 60 --- beta:[ -0.59727, -0.68328] --- Cost: 10394.61714

Iteration 70 --- beta:[ -0.51958, -0.59294] --- Cost: 9376.68091

Iteration 80 --- beta:[ -0.44410, -0.50911] --- Cost: 8461.50886

Iteration 90 --- beta:[ -0.37133, -0.43082] --- Cost: 7638.72331

final beta: [ -0.30153, -0.35739]
```

Part4



Here is the plot of loss function over different iterations. We can infer that at lower values of alpha, there is a sharp peak heading down at 0 and remains the same over iterations. Increase in alpha bends the curve, allowing loss function values to shrink to zero after more iterations are performed. Values greater than 0.0005 yielded huge loss function values and thus the value was restriced at 0.00052 after testing it with further values.

Problem 2: Ridge Regression, implement your own BIC and Cross-validation

Part 1: Use BIC for tuning parameter selection

RSS and degree of freedom (df) over different tuning parameter lambda

0 RSS: 0.970792847637 df: 9.82308656395

1 RSS: 0.975895498409 df: 9.30920279796

2 RSS: 0.980540837598 df: 8.96050627987

3 RSS: 0.984653817296 df: 8.52024853314

4 RSS: 0.997057821527 df: 7.54431135548

5 RSS: 1.93155095369 df: 5.20387301475

6 RSS: 7.02934916923 df: 4.56413471694

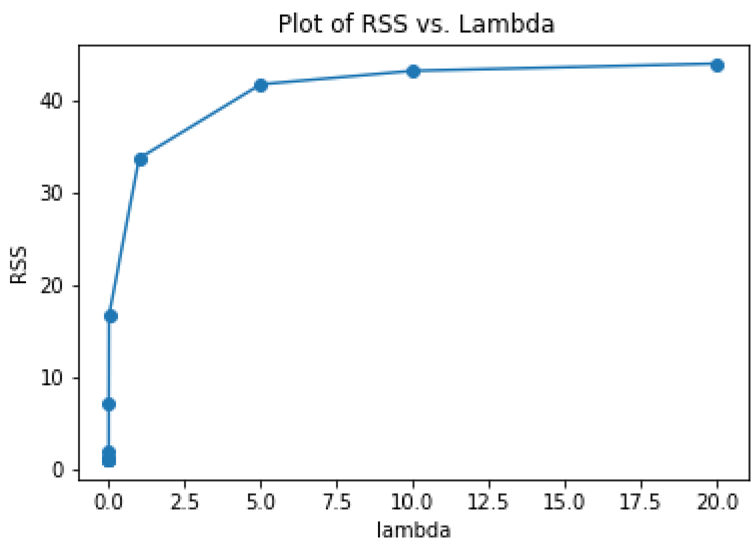
7 RSS: 16.6285292603 df: 3.86486735182

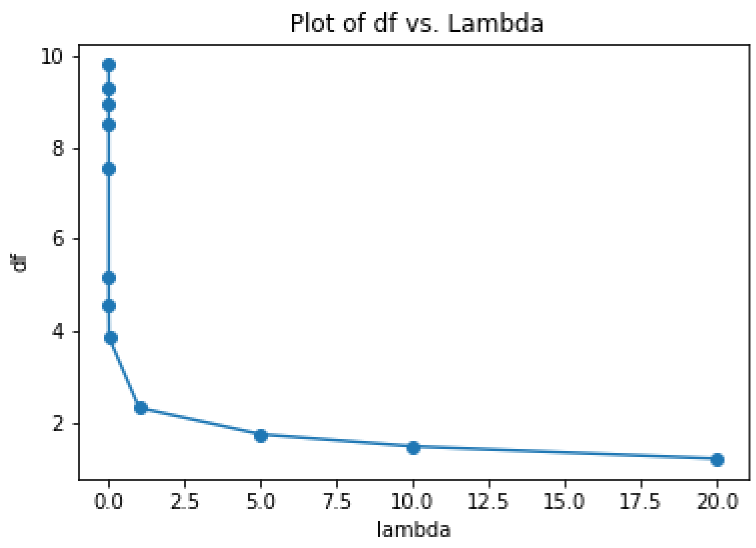
8 RSS: 33.6372072148 df: 2.3290027998

9 RSS: 41.6838828757 df: 1.75074530968

10 RSS: 43.1482596684 df: 1.48786666756

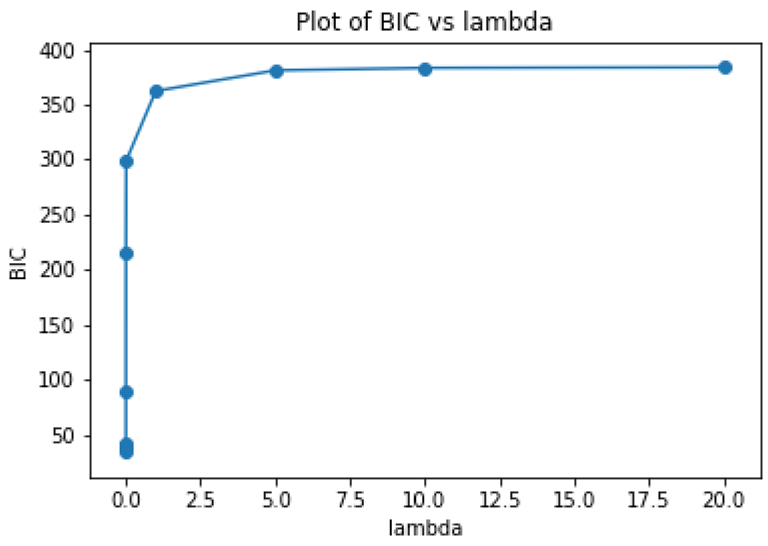
11 RSS: 43.9350989812 df: 1.21930631764





BIC criterion function over different tuning parameter lambda

BIC[i]=383.886465286



Part 3 Implement your own Cross-validation

RMSE for alpha: 1e-13 is 0.00565201029726

RMSE for alpha: 1e-12 is 0.0118391912101

RMSE for alpha: 1e-11 is 0.0271396621793

RMSE for alpha: 1e-10 is 0.0273320784586

RMSE for alpha: 1e-08 is 0.0289176812408

RMSE for alpha: 0.0001 is 0.00646876931264

RMSE for alpha: 0.001 is 0.212653259267

RMSE for alpha: 0.01 is 0.0192600353424

RMSE for alpha: 1 is 0.0577741548376

RMSE for alpha: 5 is 0.611654065478

RMSE for alpha: 10 is 0.317193087649

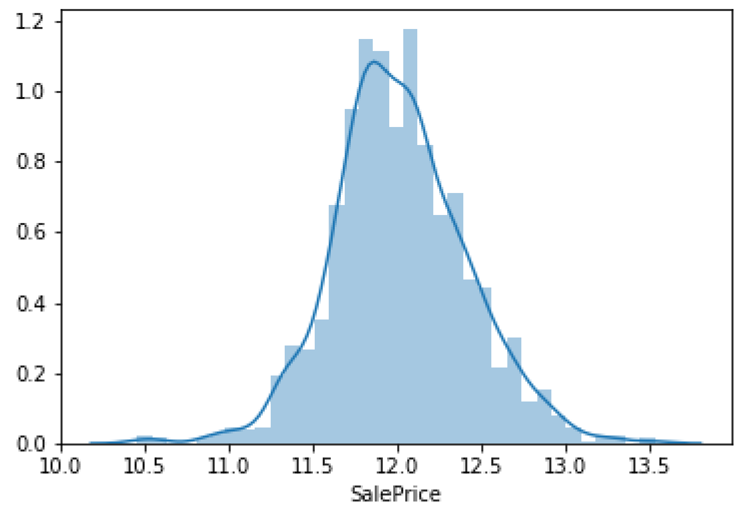
RMSE for alpha: 20 is 0.864374966122

Best lambda from the set of model

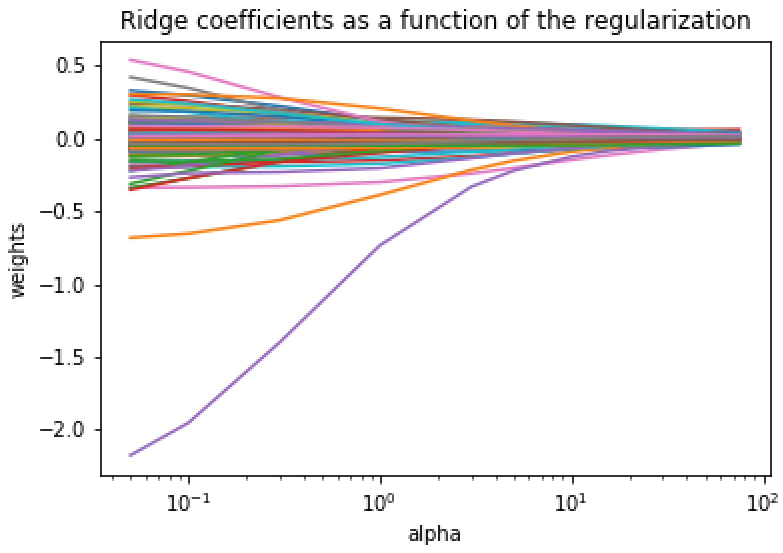
Best alpha/lambda for the model is alpha 1e-08 with RMSE: 0.00472711123718

Problem 3: Implementation for House Price Prediction

Transformation on the respones variable SalePrice

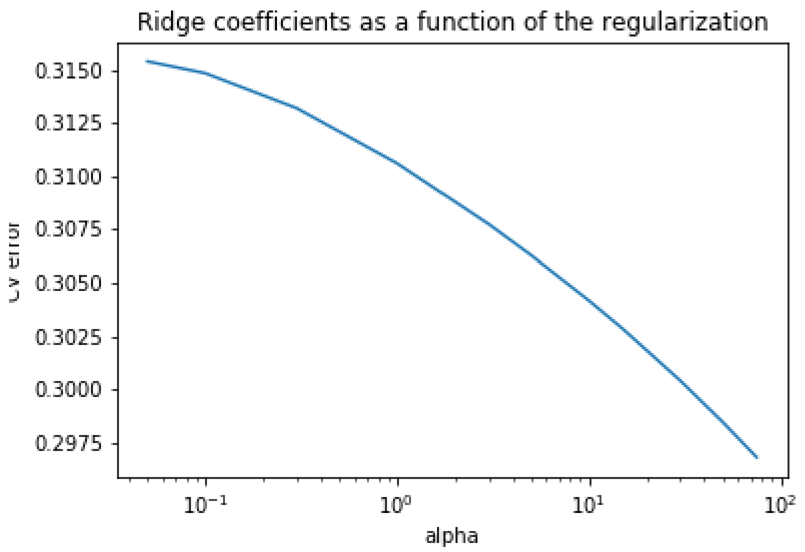


Model 1: Use Ridge regression and select the best tuning parameter



As alpha value increases, the ridge coefficients tend to be close to zero

Trying Ridge with different tuning parameter alpha



As alpha value increases, the mean squared error decreases, this confirms with the fact that flexibility of the model is decreasing too and hence the variability too sharply with little increase in bias.

Compute Regularization path using RidgeCV

ridgecv.alpha_ = 10.0

Model with alpha chosen by CV

The mean square error of model with alpha determined by CV is RMSE is: 0.304173883998

Applying Ridge model on testing dataset to compare performance

The ridge model on applying to the test data yielded the follwing predictions which matches closely with the train dataset responses.

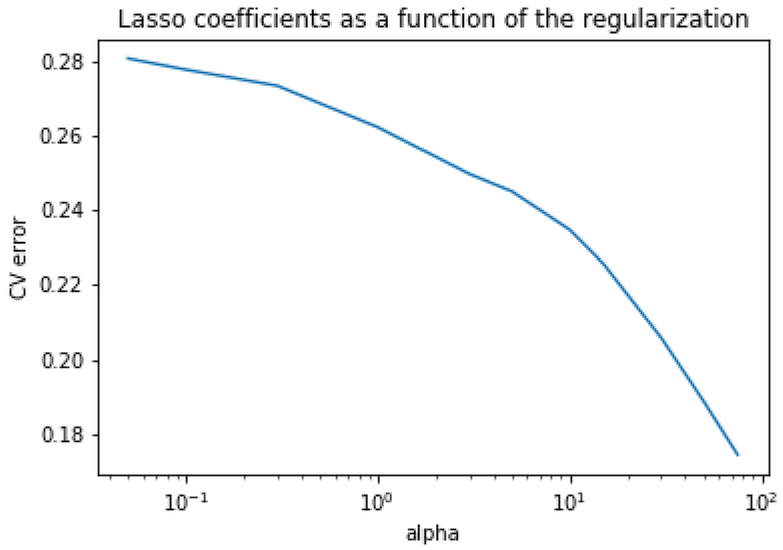
```
array([ 113732.81647612, 149449.48482876, 171345.75377362, ...,
        159712.01334373,  111954.83515282,  226064.39594212])
```

Model 2: Use Lasso regression and select the best tuning parameter

Cross validation error of lasso model is as follows

[0.28087988728729008, 0.27787193440383989, 0.27354742353378603, 0.26240073591282215, 0.24979018842010167, 0.24509220719862918

, 0.23478318558847838, 0.22565938404046113, 0.2057898449053783, 0.18865420470396066, 0.17436315781022818]



Compute Regularization using LassoCV

lassocv.alpha_ = 0.050000000000000003

Cross validation chose the best alpha value for lasso model as 0.05 and the residual mean sqaure error of this model is RMSE is: 0.262113167522

Performance of lasso model on test data

The lasso model yeilded good predictions close to train dataset responses.

```
array([ 139363.78163446, 145156.36685373, 192310.17166384, ...,
        178638.59714107,  124175.17586695,  233422.83339617])
```