IEE 598: DATA SCIENCE FOR SYSTEM INFORMATICS:

ASSIGNMENT 2

HETUL VARAIYA

1211306106

Problem 1: Ridge Regression, implement your own Gradient Optimization

For Ridge regression problem, please answer the following questions

$$\min_{\beta} l(\beta) = \|y - X\beta\|^2 + \lambda\|\beta\|_2^2$$

1. Please derive the gradient descent algorithm for the ridge regression problem for each iteration and write down for each iteration the way to update the coefficient β.
2. Please compute the Hessian Matrix of the loss function $\nabla^2 l(\beta) = \nabla\nabla l(\beta)$ and demonstrate that the problem indeed has unique global optimum by showing it is positive definite matrix.
3. Please complete the example in the jupyter notebook and implement the ridge regression algorithm in Python by yourself via both analytical solutions and gradient descent (Use only numpy library).
4. Please plot the loss function over different iterations.

Problem 1:

1) $Y \rightarrow$ Actual Value.
$f(x) \rightarrow$ Predicted value by model
$f(x) \Rightarrow E[Y/X = x_0]$

$$\frac{\partial}{\partial \beta}(y - X\beta)(y - X\beta)^T$$

$$(y - X\beta)\frac{\partial}{\partial \beta}(y - X\beta)^T + (y - X\beta)\frac{\partial}{\partial \beta}(y - X\beta)^T$$

$$2(y - X\beta)\frac{\partial}{\partial \beta}(y - X\beta)^T = -2X^T(y - X\beta)$$

$$\frac{\partial}{\partial \beta}\lambda \beta^T\beta = \lambda\beta\left(\frac{\partial \beta}{\partial \beta}\right)^T + \lambda\beta\left(\frac{\partial \beta^T}{\partial \beta}\right) = 2\lambda\beta$$

$$\frac{\partial}{\partial \beta}\min_{\beta} l(\beta) = -2X^T(y - X\beta) + 2\lambda\beta = 2(\lambda\beta - X^T(y - X\beta))$$

2) ~~$\frac{\partial^2}{\partial \beta^2}[(y - X\beta)(y - X\beta)^T + \lambda\beta\beta^T]$~~

$$\beta_{k+1} = \beta_k - \alpha_k \hat{\xi} \nabla l(\beta_k)$$
$$= \beta_k - 2(\lambda\beta - X^T(y - X\beta)$$
$$= \beta_k + 2(X^T(y - X\beta) - \lambda\beta)$$

2) $\frac{\partial^2}{\partial \beta^2}\left[(y-X\beta)(y-X\beta)^T + \lambda \beta^T \beta\right]$

$= \frac{\partial}{\partial \beta}\left[2\lambda\beta - X^T(y-X\beta)\right]$

$= 2\lambda - \frac{\partial}{\partial \beta}(X^T)(y-X\beta) \Rightarrow 2\lambda - X^T \frac{\partial}{\partial \beta}(y-X\beta)$

$= 2\lambda - X^T(-X)$

$= 2\lambda + XX^T = 2\lambda + X^2$

Part 1: Analytical solution

(1) Let's compute the Least Square analytical solution directly by solving $X^T X\theta = X^T y$

In [22]:
```
#IdentityMatrix = np.identity(p)
IdentityMatrix = 0
lambda1 = 20
beta_OLS_est = np.dot(np.linalg.inv(np.dot(X.T,X)),np.dot(X.T,y))
print(beta_OLS_est)
```

```
[[1.10034069]
 [0.89299444]]
```

(2) Let's compute the Ridge Regression analytical solution directly by solving $(X^T X + \lambda_1 I)\theta = X^T y$

In [27]:
```
IdentityMatrix1 = np.identity(p)
lambda1 = 30
beta_ridge_est = np.dot(np.linalg.inv(X.T.dot(X) + (lambda1 * IdentityMatrix1)), np.dot(X.T,y))
print(beta_ridge_est)
```

```
[[1.0053259 ]
 [0.95681108]]
```

(b) Let's use Gradient Descent to update parameter until convergence, please try different stepsize to see the Convergence performance

```
Gradient descent with 1000 iterations. step_size=0.00000 and initial be
ta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672
Iteration 100 --- beta:[ -0.28992, -0.78622] --- Cost: 9167.58771
Iteration 200 --- beta:[ 0.19303, -0.29965] --- Cost: 4419.14312
Iteration 300 --- beta:[ 0.52133, 0.03220] --- Cost: 2217.58240
Iteration 400 --- beta:[ 0.74433, 0.25869] --- Cost: 1196.83799
Iteration 500 --- beta:[ 0.89564, 0.41345] --- Cost: 723.55781
Iteration 600 --- beta:[ 0.99813, 0.51936] --- Cost: 504.09982
Iteration 700 --- beta:[ 1.06739, 0.59200] --- Cost: 402.32230
Iteration 800 --- beta:[ 1.11403, 0.64198] --- Cost: 355.10561
Iteration 900 --- beta:[ 1.14527, 0.67654] --- Cost: 333.18552

 final beta: [ 1.16602, 0.70058]

Gradient descent with 1000 iterations. step_size=0.00000 and initial be
ta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672
Iteration 100 --- beta:[ 0.52446, 0.03533] --- Cost: 2200.68375
Iteration 200 --- beta:[ 1.00011, 0.52133] --- Cost: 500.74653
Iteration 300 --- beta:[ 1.14620, 0.67747] --- Cost: 332.68642
Iteration 400 --- beta:[ 1.18880, 0.72985] --- Cost: 315.94372
```

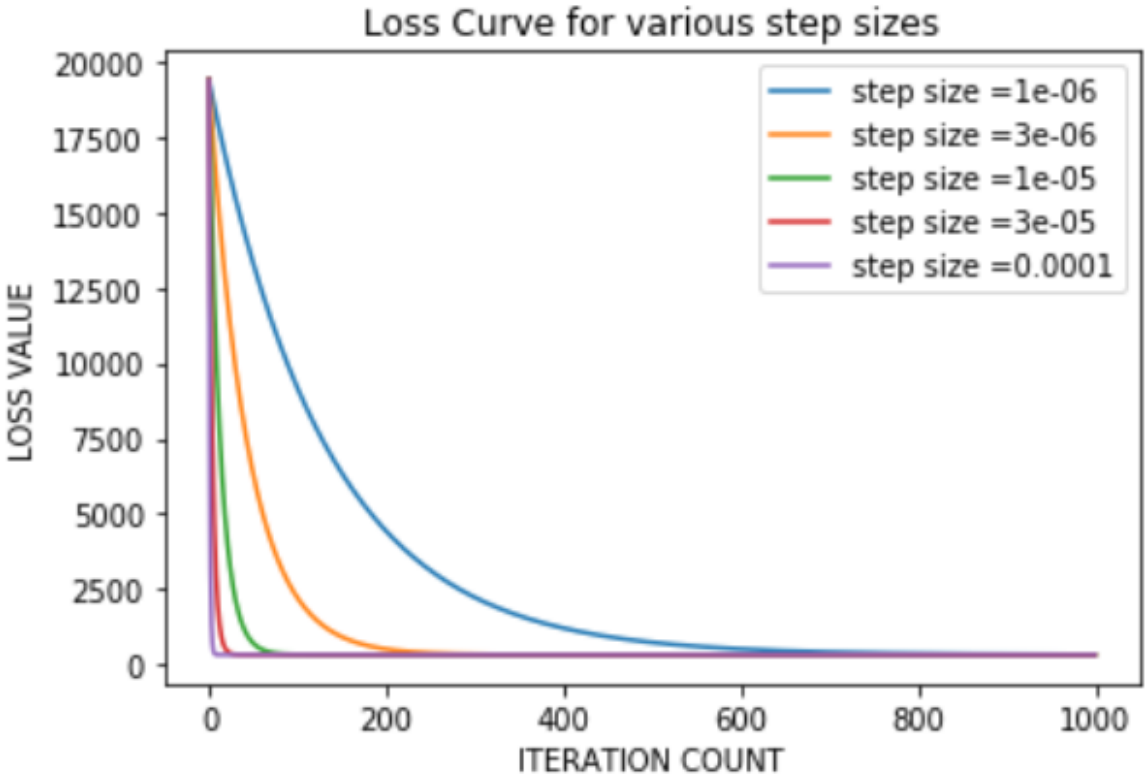
```
Iteration 500 --- beta:[ 1.19895, 0.74955] --- Cost: 314.15388
Iteration 600 --- beta:[ 1.19898, 0.75890] --- Cost: 313.84726
Iteration 700 --- beta:[ 1.19590, 0.76493] --- Cost: 313.69311
Iteration 800 --- beta:[ 1.19192, 0.76984] --- Cost: 313.55972
Iteration 900 --- beta:[ 1.18772, 0.77433] --- Cost: 313.43380

 final beta: [ 1.18352, 0.77862]

Gradient descent with 1000 iterations. step_size=0.00001 and initial be
ta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672
Iteration 100 --- beta:[ 1.16916, 0.70373] --- Cost: 321.87769
Iteration 200 --- beta:[ 1.19718, 0.76322] --- Cost: 313.73950
Iteration 300 --- beta:[ 1.18351, 0.77863] --- Cost: 313.31349
Iteration 400 --- beta:[ 1.17009, 0.79209] --- Cost: 312.95195
Iteration 500 --- beta:[ 1.15765, 0.80452] --- Cost: 312.64292
Iteration 600 --- beta:[ 1.14616, 0.81601] --- Cost: 312.37875
Iteration 700 --- beta:[ 1.13553, 0.82663] --- Cost: 312.15295
Iteration 800 --- beta:[ 1.12571, 0.83646] --- Cost: 311.95994
Iteration 900 --- beta:[ 1.11663, 0.84554] --- Cost: 311.79495

 final beta: [ 1.10823, 0.85393]

Gradient descent with 1000 iterations. step_size=0.00003 and initial be
ta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672
Iteration 100 --- beta:[ 1.18349, 0.77867] --- Cost: 313.31257
Iteration 200 --- beta:[ 1.14611, 0.81606] --- Cost: 312.37760
Iteration 300 --- beta:[ 1.11657, 0.84560] --- Cost: 311.79387
Iteration 400 --- beta:[ 1.09322, 0.86894] --- Cost: 311.42943
Iteration 500 --- beta:[ 1.07478, 0.88738] --- Cost: 311.20189
Iteration 600 --- beta:[ 1.06020, 0.90195] --- Cost: 311.05984
Iteration 700 --- beta:[ 1.04869, 0.91346] --- Cost: 310.97114
Iteration 800 --- beta:[ 1.03959, 0.92256] --- Cost: 310.91577
Iteration 900 --- beta:[ 1.03240, 0.92975] --- Cost: 310.88120

 final beta: [ 1.02672, 0.93543]

Gradient descent with 1000 iterations. step_size=0.00010 and initial be
ta=[-1.00000,-1.50000]

Iteration 0 --- beta:[ -1.00000, -1.50000] --- Cost: 19409.36672
Iteration 100 --- beta:[ 1.10794, 0.85422] --- Cost: 311.64931
Iteration 200 --- beta:[ 1.05203, 0.91012] --- Cost: 310.99472
Iteration 300 --- beta:[ 1.02658, 0.93556] --- Cost: 310.85915
Iteration 400 --- beta:[ 1.01500, 0.94714] --- Cost: 310.83107
Iteration 500 --- beta:[ 1.00973, 0.95241] --- Cost: 310.82526
Iteration 600 --- beta:[ 1.00733, 0.95481] --- Cost: 310.82406
Iteration 700 --- beta:[ 1.00624, 0.95590] --- Cost: 310.82381
Iteration 800 --- beta:[ 1.00574, 0.95640] --- Cost: 310.82375
Iteration 900 --- beta:[ 1.00551, 0.95662] --- Cost: 310.82374

 final beta: [ 1.00541, 0.95673]
```

Conclusion:

From the above result we can comment about the performance of the convergence that as the step size is increased the number of iterations required to converge to the constant cost decreases.

Hence mathematically, we can say step size is directly proportional to the Performance of the convergence.

(c) Normally, it is good to visualize the loss function over time, please plot the loss (cost) function for theta in each iteration.



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

Ridge regression can be combined with the polynomial regression for more robust fitting. Here we will explore an alternative method for model complexity control for HW1 Problem 3 to fit the $f(x) = sin(x)$ curve. We sample 100 points from 0 to 10 as our training samples,

the goal is to use the polynomial regression to fit this sin function. However, here we use a 10^{th} order polynomial

$$\min_{\beta} \|y - X\beta\|^2 + \lambda\|\beta\|_2^2$$

Please answer the following questions.

1. Please compare the computational complexity of analytical solution of ridge regression and the gradient descent.

2. Here, please feel free to use the sklearn library "sklearn.linear_model.Ridge" to solve the Ridge regression in this section to find the tuning parameter selection.

   (a) Please compute and plot on how do the $RSS = \|y - \hat{y}\|^2$ and degree of freedom $df = trace(X(X^TX + \lambda I)^{-1}X^T)$ change according to the tuning parameter λ.

   (b) Please compute and plot the following BIC criterion function for each tuning parameter λ.

   $$BIC = n\log(RSS) + df \cdot \log(n)$$

3. Please use the K-fold function in sklearn as "sklearn.model_selection.KFold" to perform cross-validation. Please report the mean sum of square error for the cross-validation. Please select the best λ from the set for model.

Problem 2:-

1. Analytical solution of Ridge Regression:

$$Bridge = (X^T X + \lambda I)^{-1} X^T y$$

Time complexity :

$X^T X \rightarrow O(np^2)$

$\lambda I \rightarrow O(p^2)$

$(X^T X + \lambda I) \rightarrow O(p^2)$

$(X^T X + \lambda I)^{-1} \rightarrow O(p^3)$

$(X^T) \rightarrow O(np)$

$(X^T y) \rightarrow O(np)$

$(X^T X + \lambda I)^{-1} X^T y \rightarrow O(p^2)$

Time Complexity $\rightarrow$ i and $n > p$

$\Rightarrow O(np^2)$

i and $p > n$

$\Rightarrow O(p^3)$

Space complexity:

$X \rightarrow n \times p = O(np)$

$X^T \rightarrow p \times n = O(np)$

$X^T X \rightarrow p \times p = O(p^2)$

$\lambda I \rightarrow p \times p = O(p^2)$

$(X^T X + \lambda I) \rightarrow p \times p = O(p^2)$

$X^T y \rightarrow p \times 1 = O(p)$

$(X^T X + \lambda I)(X^T y) \rightarrow p \times 1 = O(p)$

Space complexity: $\Rightarrow$ if $n > p$ $\rightarrow O(np)$

if $p > n$ $\Rightarrow O(p^2)$

$\Rightarrow$ Gradient: $-2X^T(y - X\beta) + 2\lambda\beta$.

Time complexity

$X\beta \rightarrow O(np)$

$y - X\beta \rightarrow O(n)$

$X^T \rightarrow O(n)$

$-2X^T(y - X\beta) \rightarrow O(np)$

$2\lambda\beta \rightarrow O(p)$

$(-2X^T(y - X\beta) + 2\lambda\beta) \rightarrow O(p)$

Time complexity $= O(np)$

Space complexity

$X \rightarrow O(np)$

$X\beta \rightarrow O(n)$

$(y - X\beta) \rightarrow O(n)$

$X^T(y - X\beta) \rightarrow O(p)$

$\lambda\beta \rightarrow O(p)$

$-2X^T(y - X\beta) + 2\lambda\beta \rightarrow O(p)$

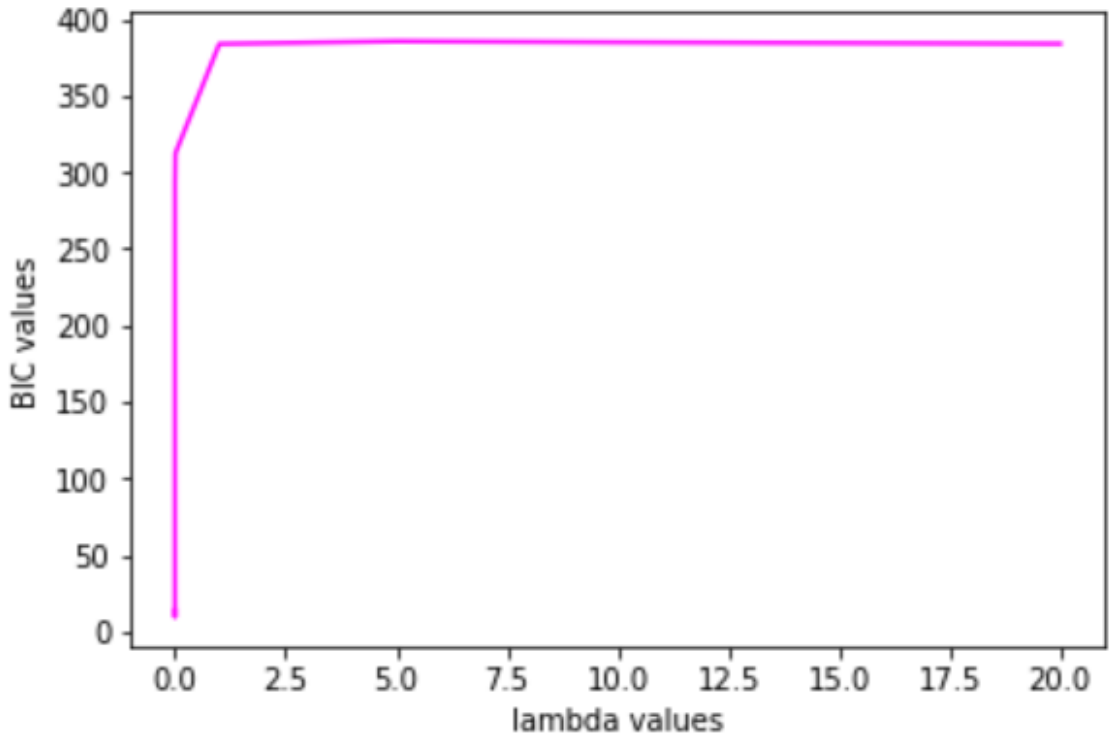
Space complexity $= O(np)$

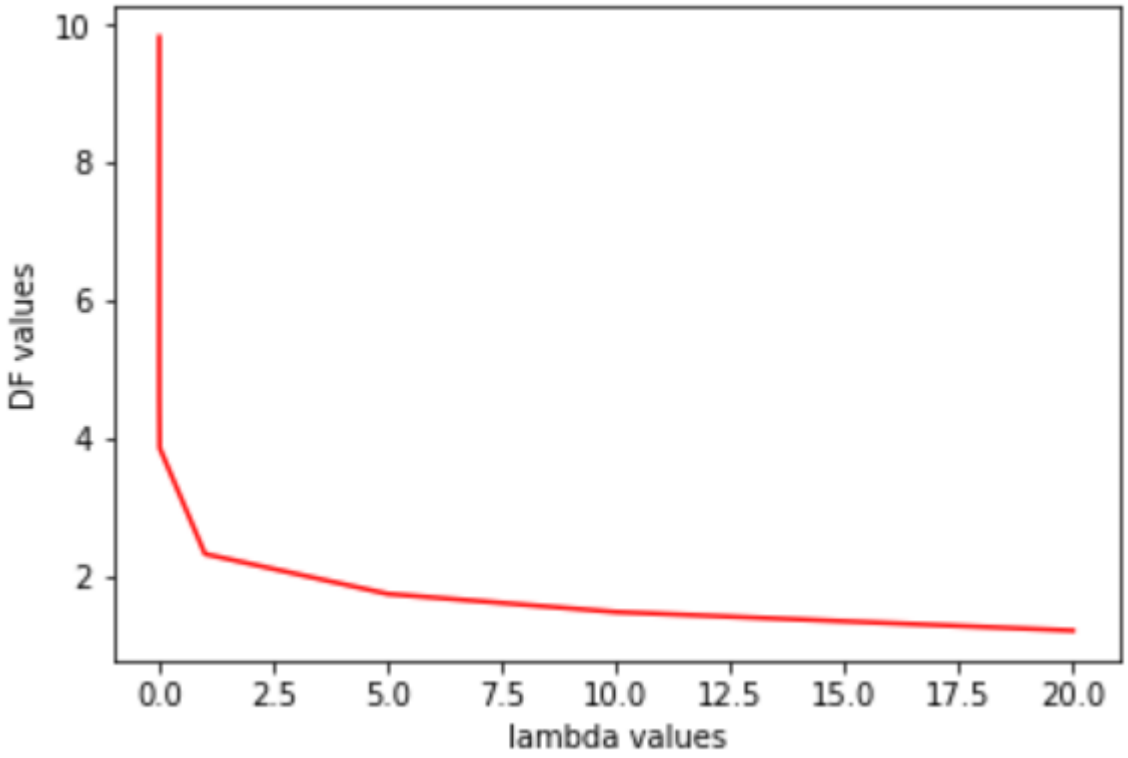
```
BIC values are:
 [ 13.88919634  12.40444272  11.07247853   9.4133197    9.73088628
 188.34691967 293.06075256 312.45922949 383.80634709 385.48820754
 384.85548041 383.91423961]
df values are:
 [9.82326058 9.3090507  8.96051229 8.52024861 7.54431132 5.20387301
 4.56413472 3.86486735 2.3290028  1.75074531 1.48786667 1.21930632]
RSS values are
 [ 0.73089468  0.73737905  0.73939569  0.74212414  0.77870771  5.17491021
 15.18671989 19.04126709 41.71283722 43.56513935 43.81761966 43.94730335]
```

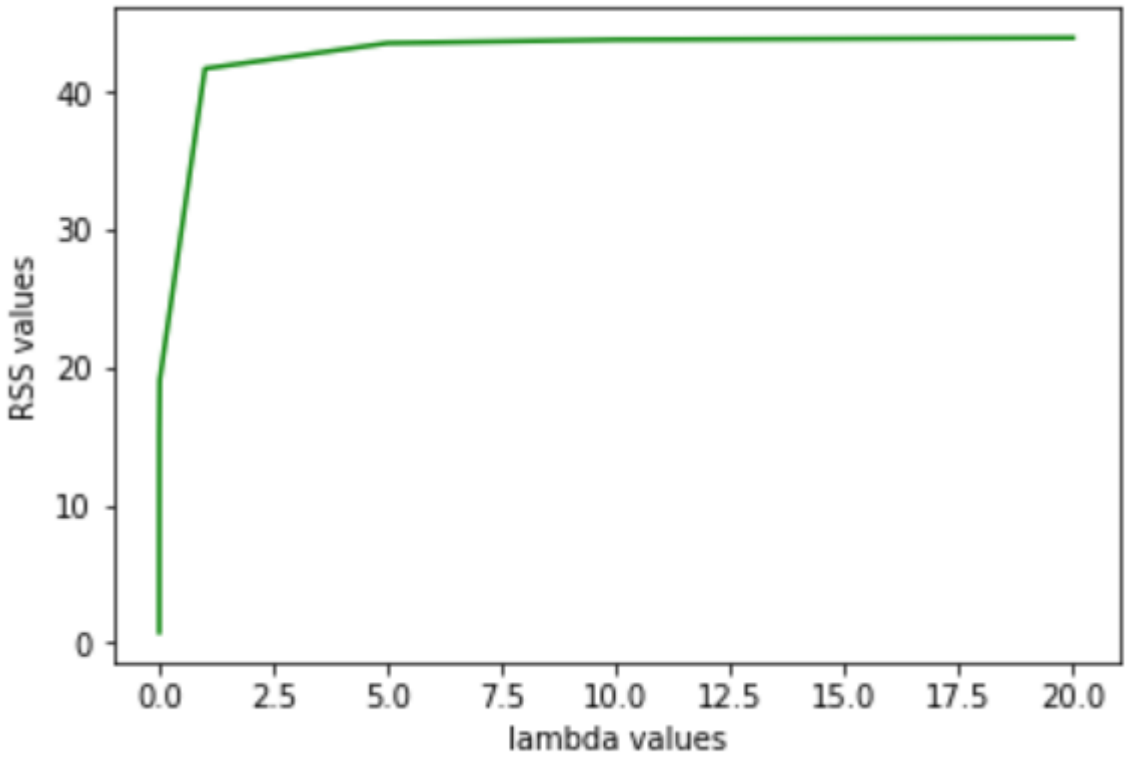
The values displayed above are the corresponding values that increase or decrease as the value of Lambda value increases.

When Lambda values are increased the BIC used as a tuning parameter for selection increases and shoots up drastically when the value of lambda is changed from 5 to 6. The BIC indicates how the predictors will affect the response. Therefore, the increase in BIC values results in the decrease of degrees of freedom as the number of regressors are reduced due to their less significant effect on the response variable. The BIC values stabilize and is highest when the lambda value is 10.

The above explanation is explained by the plot shown on the next page.

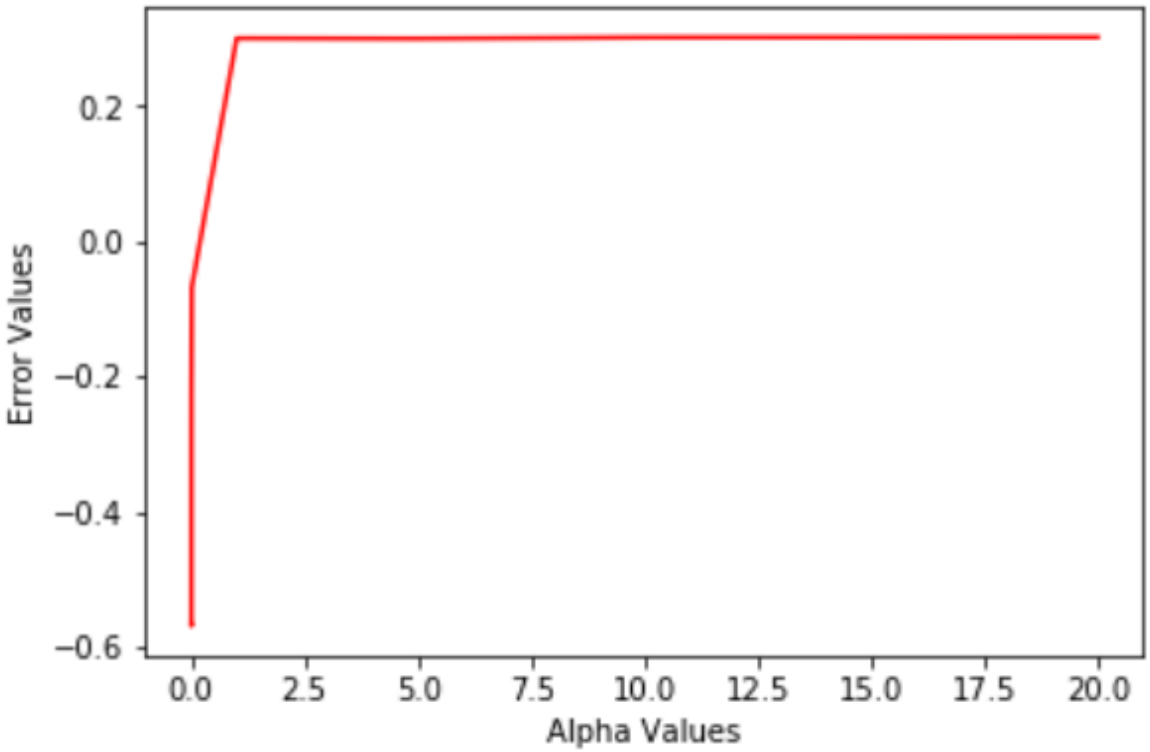






```
The minimum MSE value is: -0.5681743544671313
```

The below plot shows the effect alpha values on the error.

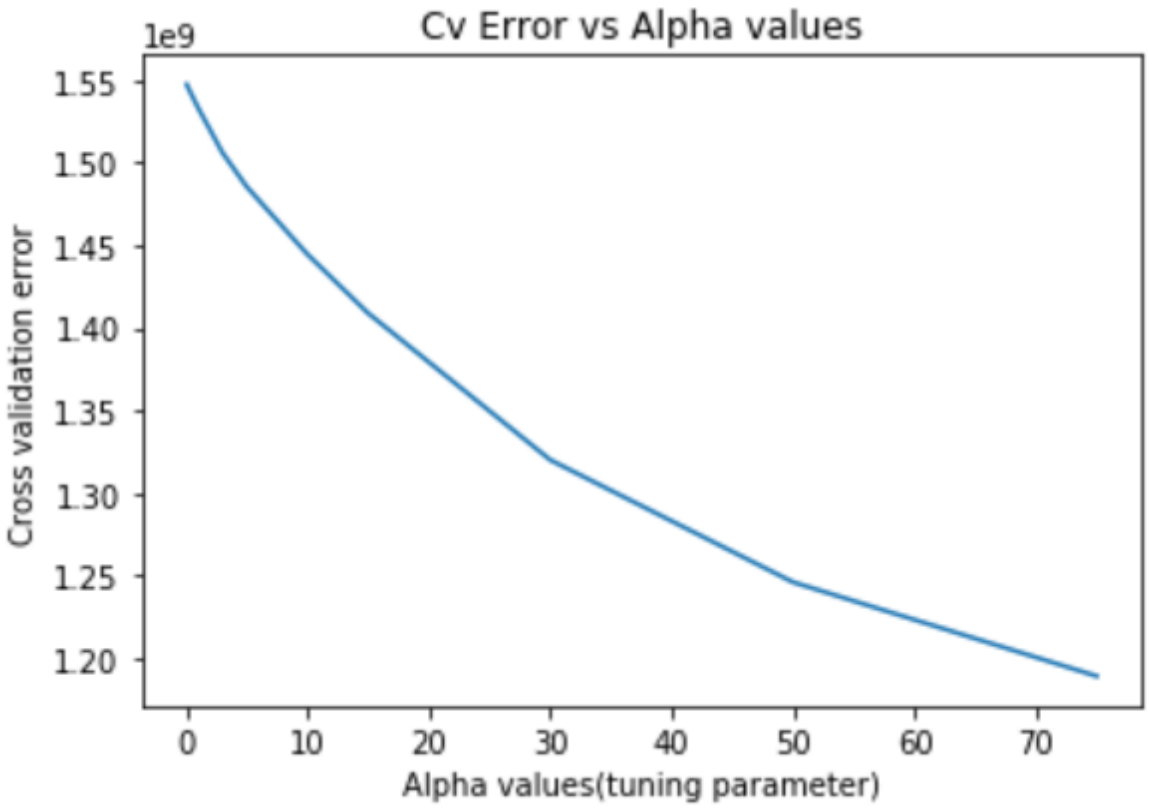

0.2
0.0
−0.2
−0.4
−0.6
Error Values
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Alpha Values

Problem 3: Implementation for House Price Prediction

Explore the use of Lasso and Ridge regression in the housing price prediction. Please feel free to use all functions provided by sklearn. With 79 explanatory variables describing (almost) every aspect of residential homes, the goal is for you to predict the final price of each home. The training data is given in 'train.csv' and testing data is given in 'test.csv'.

1. Please first read the training and testing data and perform the following pre-processing
   (a) Data normalization
   (b) Create dummy variables
2. Please apply the Lasso regression with different tuning parameters. Please plot the cross-validation error vs different tuning parameters.
3. Please apply the ridge regression with different tuning parameters. Please plot the cross-validation error vs different tuning parameters.
4. Please use cross-validation to choose the best tuning parameter for both methods. Please compare the cross-validation error (mean of the Residual Mean Square Error for different replications). Please apply the model on the testing dataset. Since the label is not provided, you don't need to compute the testing performance, only the prediction is good enough.

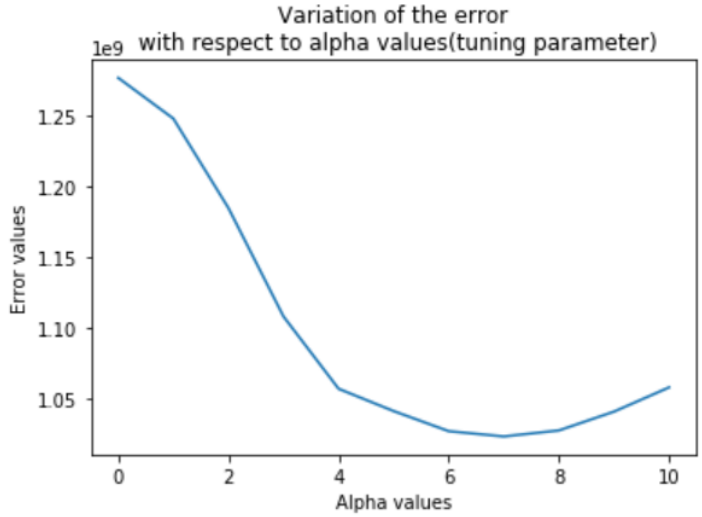
2. Model 2. Cross validation error vs. Alpha values (tuning parameters) for the model using LASSO regression



3. Model 1. Cross validation error vs tuning parameters for the model with Ridge regression:

```
the Minimin value of the error value is: 1022858854.5353096
```

```
Text(0.5,1,'Variation of the error\n with respect to alpha values(tuning parameter)')
```



---

4. Applying the fitted models to predict the response variable values:

Predicted values for Ridge regression:

```
array([[103135.73652034],
       [145213.25579652],
       [174828.74203761],
       ...,
       [154981.79289185],
       [104259.89024762],
       [226282.35987252]])
```

Predicted Values for LASSO regression:

```
array([115224.73575392, 148466.9960682 , 182143.32199918, ...,
       165492.84492149, 108701.65762168, 221164.1988029 ])
```
