Aim : Implementation of L2 regularization

In [1]:
```
from sklearn.datasets import make_regression
```

In [2]:
```
x, y = make_regression(n_samples=100, n_features=1,n_targets=1, noise=20,random_state=2)
```

In [3]:
```
x
```

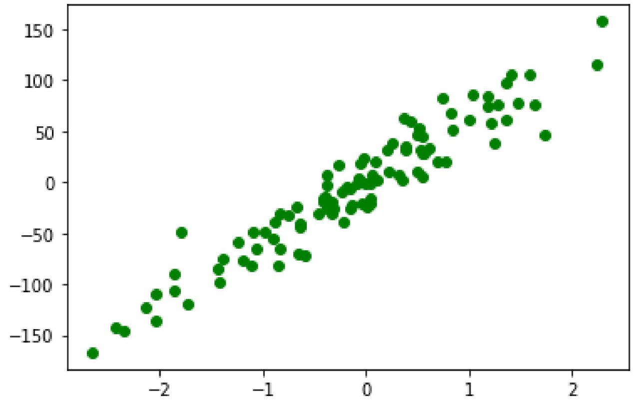
Out[3]:
```
array([[-8.78107893e-01],
       [ 1.35963386e+00],
       [ 1.64027081e+00],
       [ 5.42352572e-01],
       [ 8.24703005e-01],
       [-5.96159700e-01],
       [-5.62668272e-02],
       [-1.32328898e-01],
       [-2.43476758e+00],
       [-4.03892269e-01],
       [ 3.81866234e-01],
       [ 1.36723542e+00],
       [-7.47870949e-01],
       [-3.81516482e-01],
       [ 8.42456282e-01],
       [-7.72186654e-02],
       [ 4.33496330e-01],
       [-2.17135269e-01],
       [ 1.46767801e+00],
       [ 1.12726505e-01],
       [ 1.73118467e+00],
       [-3.35677339e-01],
       [-1.09873895e+00],
       [-8.41747366e-01],
       [ 5.66275441e-01],
       [-1.05795222e+00],
       [ 6.95119605e-01],
       [ 5.39058321e-01],
       [ 5.01857207e-01],
       [-1.43943903e+00],
       [-1.89469265e-01],
       [-7.44707629e-02],
       [-1.79343559e+00],
       [ 2.23136679e+00],
       [ 4.15393930e-02],
       [ 1.17353150e+00],
       [-1.38451867e+00],
       [ 2.16116006e-01],
       [-9.14526229e-02],
       [ 1.27837923e+00],
       [-2.13619610e+00],
       [-2.36184031e-01],
       [-1.85861239e+00],
       [ 3.26003433e-01],
       [ 9.76147160e-06],
       [-3.75669423e-01],
       [ 1.04082395e+00],
       [-3.38821966e-01],
       [-6.77675577e-01],
       [ 1.21788563e+00],
       [-1.24528809e+00],
       [-8.29135289e-01],
       [-4.19316482e-01],
       [ 3.80471970e-01],
       [-1.86809065e+00],
       [ 1.58448706e+00],
       [ 6.11340780e-01],
       [ 1.00036589e+00],
       [-9.09007615e-01],
       [-3.95702397e-02],
       [-1.91304965e-02],
       [-1.42121723e+00],
       [ 3.50888494e-01],
       [-9.88779049e-01],
       [ 4.62555231e-02],
       [-3.81092518e-01],
       [ 7.35279576e-01],
       [-6.37655012e-01],
```

```
       [-2.65944946e+00],
       [-4.16757847e-01],
       [ 2.56570452e-01],
       [ 3.70444537e-01],
       [-6.34679305e-01],
       [ 4.79705919e-02],
       [-2.34360319e+00],
       [ 5.08396243e-01],
       [-6.53250268e-01],
       [-8.44213704e-01],
       [-4.62005348e-01],
       [ 2.04207979e-01],
       [ 5.02881417e-01],
       [-2.69056960e-01],
       [-1.11792545e+00],
       [ 5.24296430e-01],
       [ 8.77102184e-02],
       [ 5.51454045e-01],
       [-2.03346655e+00],
       [-3.13508197e-01],
       [ 1.17500122e+00],
       [ 6.64890091e-02],
       [ 1.24821292e+00],
       [-1.18761229e+00],
       [ 1.40669624e+00],
       [-1.56434170e-01],
       [ 2.29220801e+00],
       [-1.73795950e+00],
       [-2.04032305e+00],
       [ 7.71011738e-01],
       [ 9.02525097e-03],
       [-1.53495196e-01]])
```

In [4]:
```
y
```

Out[4]:
```
array([ -39.28111368,   60.36088967,   75.52329534,   44.96973636,
         67.66126148,  -72.56783539,   19.01828591,  -22.48725552,
       -142.24576438,  -15.18039423,   30.97837048,   96.49616637,
        -32.18524905,   -2.73343352,   51.91525767,    3.15477977,
         59.67514565,  -39.18356672,   77.51342439,    2.12711018,
         46.68645147,  -19.761177  ,  -48.75839881,  -64.53825927,
         28.77572075,  -64.91364323,   19.42416802,   27.95857927,
         47.08740928,  -84.85367831,   -5.27930487,    1.63049652,
        -49.65719062,  115.44179673,  -15.90690036,   84.79331953,
        -75.6361891 ,    9.44506047,   -1.10256915,   75.29608508,
       -123.12209367,   -9.2643982 ,  -89.78307298,    6.92428027,
         -1.37114106,  -25.12536695,   86.08123884,  -30.90683686,
        -25.0568712 ,   57.12871401,  -59.45098934,  -31.08914872,
        -19.63623245,   35.24070197, -105.60242742,  105.79285848,
         33.5416947 ,   61.17460846,  -55.74511813,  -20.22825731,
         22.61124007,  -98.27810976,    2.51352804,  -48.6592866 ,
         -0.80311042,    7.54015095,   82.78973256,  -40.39608706,
       -167.01636233,  -16.85774471,   38.07736278,   62.11552806,
        -43.76716725,  -21.72322566, -145.21657026,   52.47950288,
        -70.08095861,  -81.46341231,  -31.22319839,   31.93634207,
          9.80885417,   16.27732067,  -82.0026651 ,   32.12925769,
         19.84053667,    4.7638933 , -135.75753122,  -26.07994395,
         74.9019179 ,    7.4922353 ,   37.76513716,  -77.51141039,
        105.71270776,  -26.05475138,  157.64933729, -120.01528779,
       -110.34447888,   19.41625394,  -24.34594172,   -6.97226993])
```

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.scatter(x, y,c='g')
plt.show()
```



In [6]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

In [7]:
```
lr.fit(x,y)
```

Out[7]:
```
LinearRegression()
```

In [8]:
```
lr.coef_ #Magnitude,coefficient,slope
```

Out[8]:
```
array([58.11865999])
```

In [9]:
```
lr.intercept_ #Bias, y_intercept
```

Out[9]:
```
0.5334392393890015
```

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

Ridge Regression

In [12]:
```
from sklearn.linear_model import Ridge
```

In [13]:
```
rr = Ridge(alpha=10)
```

In [14]:
```
rr.fit(x,y)
```

Out[14]:
```
Ridge(alpha=10)
```

In [15]:
```
rr.coef_
```

Out[15]:
```
array([53.17425635])
```

In [16]:
```
rr.intercept_
```

Out[16]:
```
0.020501198988966074
```

In [17]:
```
rrr = Ridge(alpha=10)
rrr.fit(x,y)
print(rrr.coef_)
print(rrr.intercept_)
```

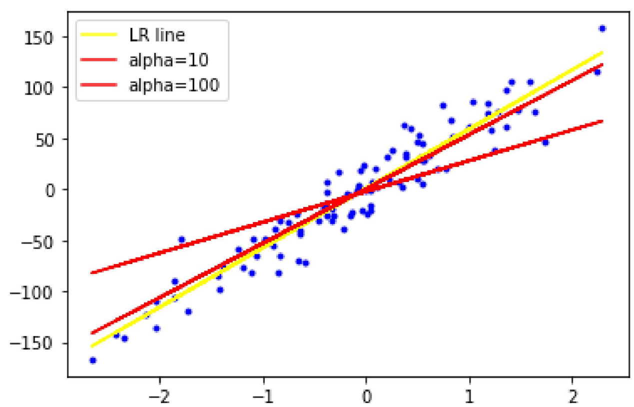
```
[53.17425635]
0.020501198988966074
```

In [18]:
```
rrr = Ridge(alpha=100)
rrr.fit(x,y)
print(rrr.coef_)
print(rrr.intercept_)
```

```
[30.11564938]
-2.3716248342245936
```

In [19]:
```
plt.plot(x,y,'b.')
plt.plot(x,lr.predict(x),color='yellow',label="LR line")
plt.plot(x,rr.predict(x),color='red',label="alpha=10")
plt.plot(x,rrr.predict(x),color='red',label="alpha=100")
plt.legend()
```

Out[19]: <matplotlib.legend.Legend at 0x2973b8e9730>



Aim : Demonstrate how co-efficient affected values of the lambda(alpha)

In [20]:
```
from sklearn.datasets import load_diabetes
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [21]:
```
data = load_diabetes()
```

In [22]:
```
# array to dataframe
data
```

Out[22]:
```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
          0.01990842, -0.01764613],
        [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
         -0.06832974, -0.09220405],
        [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
          0.00286377, -0.02593034],
        ...,
        [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
         -0.04687948,  0.01549073],
        [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
          0.04452837, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
         -0.00421986,  0.00306441]]),
 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310., 101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
        197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
         59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
        237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
        143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
        142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
         77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
         78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
        154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
         71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
        150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
        145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
         94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
         60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
         31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
        114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
        244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
        263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
         77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
        140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
        219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
         43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
         84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
         94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
        220.,  57.]),
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen baseline variables,
age, sex, body mass index, average blood\npressure, and six blood serum measurements were obtained
for each of n =\n442 diabetes patients, as well as the response of interest, a\nquantitative measu
re of disease progression one year after baseline.\n\n**Data Set Characteristics:**\n\n  :Number o
f Instances: 442\n\n  :Number of Attributes: First 10 columns are numeric predictive values\n\n  :
Target: Column 11 is a quantitative measure of disease progression one year after baseline\n\n  :A
ttribute Information:\n      - age     age in years\n      - sex\n      - bmi     body mass index
\n      - bp      average blood pressure\n      - s1      tc, total serum cholesterol\n      - s2
ldl, low-density lipoproteins\n      - s3      hdl, high-density lipoproteins\n      - s4      tc
h, total cholesterol / HDL\n      - s5      ltg, possibly log of serum triglycerides level\n
- s6      glu, blood sugar level\n\nNote: Each of these 10 feature variables have been mean center
ed and scaled by the standard deviation times `n_samples` (i.e. the sum of squares of each column
totals 1).\n\nSource URL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more i
```

```
nformation see:\nBradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least
Angle Regression," Annals of Statistics (with discussion), 407-499.\n(https://web.stanford.edu/~ha
stie/Papers/LARS/LeastAngle_2002.pdf)',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'data_filename': 'diabetes_data.csv.gz',
 'target_filename': 'diabetes_target.csv.gz',
 'data_module': 'sklearn.datasets.data'}
```

In [23]:
```
df = pd.DataFrame(data.data,columns=data.feature_names)
```

In [24]:
```
df
```

Out[24]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641
...	...	...	...	...	...	...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.031193	0.007207
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.018118	0.044485
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.046879	0.015491
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.044528	-0.025930
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.004220	0.003064

442 rows × 10 columns

In [25]:
```
df['Target'] = data.target
```

In [26]:
```
df
```

Out[26]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	Target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	135.0
...	...	...	...	...	...	...	...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.031193	0.007207	178.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.018118	0.044485	104.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.046879	0.015491	132.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.044528	-0.025930	220.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.004220	0.003064	57.0

442 rows × 11 columns

In [27]:
```
df.shape
```

Out[27]:
```
(442, 11)
```

In [28]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(data.data,data.target,test_size=0.2,random_state=2
```

In [29]:
```
len(x_train)
```

Out[29]:
```
353
```

In [30]:
```
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
```

In [31]:
```
coefs = []
r2_scores = []
for i in [0,10,100,1000]:
    reg = Ridge(alpha=i)
    reg.fit(x_train,y_train)

    coefs.append(reg.coef_.tolist())
    y_pred = reg.predict(x_test)
    r2_scores.append(r2_score(y_test,y_pred))
```

In [32]:
```
coefs
```

Out[32]:
```
[[-9.160884832463061,
  -205.46225987708985,
  516.684623831389,
  340.6273410788926,
  -895.5436086743563,
  561.2145330558951,
  153.88478595250416,
  126.73431596154839,
  861.1213995461823,
  52.41982835857488],
 [21.17400371774996,
  1.659796134738543,
  63.659771901799736,
  48.4932400316976,
  18.421491990472827,
  12.875448426495623,
  -38.9154350572375,
  38.842463722063044,
  61.61240510619145,
  35.50535526561315],
 [2.8589794382553477,
  0.6294520371235355,
  7.540604496094514,
  5.8499966438735935,
  2.7108785152669643,
  2.142134389296116,
  -4.83404696857779,
  5.108223239548698,
  7.4484662433551705,
  4.576128672131118],
 [0.2957255603009537,
  0.06929028636932703,
  0.7690038061994643,
  0.5978292887031441,
  0.2828995133533437,
  0.22593550596063292,
  -0.4956069088303586,
  0.5270313419211985,
  0.7614974792951518,
  0.4710290658232608]]
```

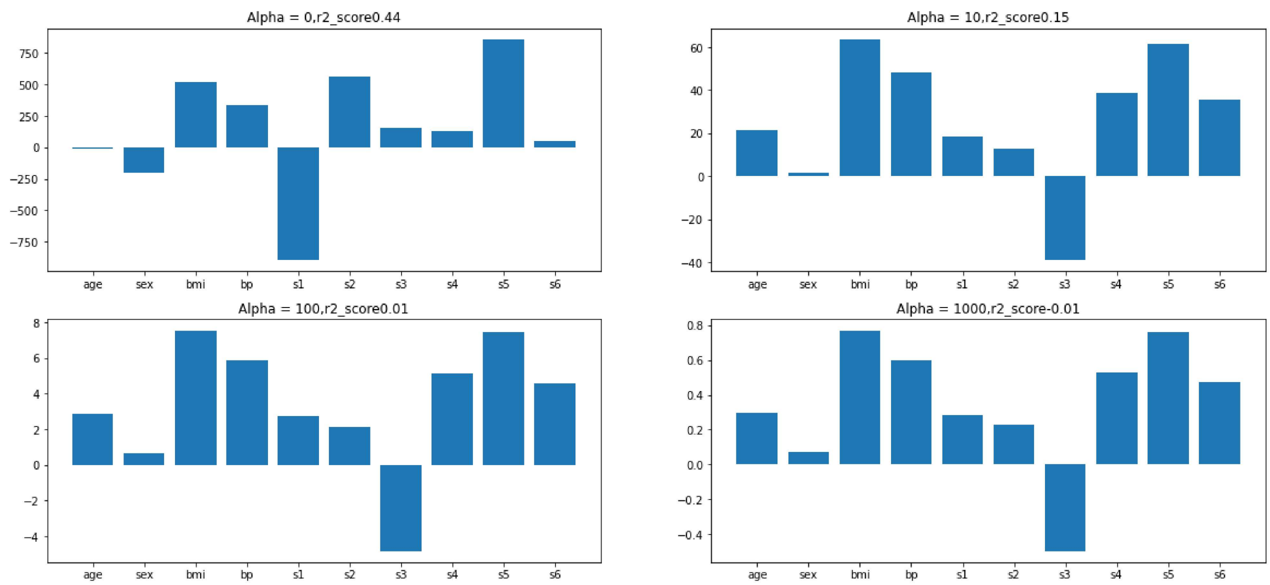
In [33]:
```
plt.figure(figsize=(20,9))
plt.subplot(221)
plt.bar(data.feature_names,coefs[0])
plt.title("Alpha = 0,r2_score{}".format(round(r2_scores[0],2)))

plt.subplot(222)
plt.bar(data.feature_names,coefs[1])
plt.title("Alpha = 10,r2_score{}".format(round(r2_scores[1],2)))

plt.subplot(223)
plt.bar(data.feature_names,coefs[2])
plt.title("Alpha = 100,r2_score{}".format(round(r2_scores[2],2)))

plt.subplot(224)
plt.bar(data.feature_names,coefs[3])
plt.title("Alpha = 1000,r2_score{}".format(round(r2_scores[3],2)))
```

Out[33]: Text(0.5, 1.0, 'Alpha = 1000,r2_score-0.01')



Aim : In ridge regression prove that 'The more higher co-efficient are affected more'

In [34]:
```
coefs = []
alphas = [0,0.0001,0.001,0.01,0.1,1,10,100,1000,10000]
for i in alphas:
    reg = Ridge(alpha=i)
    reg.fit(x_train,y_train)
    coefs.append(reg.coef_.tolist())
```

In [35]:
```
np_arr = np.array(coefs)
```

In [37]:
```
coef_df = pd.DataFrame(np_arr,columns=data.feature_names)
coef_df['alpha'] = alphas
coef_df.set_index('alpha')
```

Out[37]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5
alpha									
0.0000	-9.160885	-205.462260	516.684624	340.627341	-895.543609	561.214533	153.884786	126.734316	861.121400
0.0001	-9.118336	-205.337133	516.880570	340.556792	-883.415291	551.553259	148.578680	125.355917	856.480254
0.0010	-8.763583	-204.321125	518.371729	339.975385	-787.690766	475.274718	106.786540	114.632063	819.739542
0.0100	-6.401088	-198.669767	522.048548	336.348363	-383.709187	152.663678	-66.060583	75.611090	659.869402
0.1000	6.642753	-172.242166	485.523872	314.682122	-72.939323	-80.590053	-174.466515	83.616653	484.363285
1.0000	42.242217	-57.305508	282.170831	198.061386	14.363544	-22.551274	-136.930053	102.023193	260.104308
10.0000	21.174004	1.659796	63.659772	48.493240	18.421492	12.875448	-38.915435	38.842464	61.612405
100.0000	2.858979	0.629452	7.540604	5.849997	2.710879	2.142134	-4.834047	5.108223	7.448466
1000.0000	0.295726	0.069290	0.769004	0.597829	0.282900	0.225936	-0.495607	0.527031	0.761497
10000.0000	0.029674	0.006995	0.077054	0.059915	0.028412	0.022715	-0.049686	0.052870	0.076321

In [ ]: