Ridge Regularization

In [1]:

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

In [2]:

```
from sklearn.datasets import load_diabetes
data = load_diabetes()
```

In [3]:

```
data.DESCR
```

Out[3]:

```
'.. _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 measure of disease progres
sion one year after baseline.\n\n**Data Set Characteristics:**\n\n  :Number
of Instances: 442\n\n  :Number of Attributes: First 10 columns are numeric p
redictive values\n\n  :Target: Column 11 is a quantitative measure of diseas
e progression one year after baseline\n\n  :Attribute Information:\n      -
age     age in years\n      - sex\n      - bmi     body mass index\n      -
bp      average blood pressure\n      - s1      tc, T-Cells (a type of white
blood cells)\n      - s2      ldl, low-density lipoproteins\n      - s3
hdl, high-density lipoproteins\n      - s4      tch, thyroid stimulating hor
mone\n      - s5      ltg, lamotrigine\n      - s6      glu, blood sugar lev
el\n\nNote: Each of these 10 feature variables have been mean centered and s
caled by the standard deviation times `n_samples` (i.e. the sum of squares o
f each column totals 1).\n\nSource URL:\nhttps://www4.stat.ncsu.edu/~boos/va
r.select/diabetes.html\n\nFor more information see:\nBradley Efron, Trevor H
astie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression,"
Annals of Statistics (with discussion), 407-499.\n(https://web.stanford.edu/
~hastie/Papers/LARS/LeastAngle_2002.pdf)'
```

In [4]:

```
x = data.data
y = data.target
```

In [5]:

```
x
```

Out[5]:

```
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]])
```

In [6]:

```
y
```

Out[6]:

```
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.])
```

In [7]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=45)
```

In [8]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

In [9]:

```
lr.fit(X_train,y_train)
```

Out[9]:

```
LinearRegression()
```

In [10]:

```
lr.coef_
```

Out[10]:

```
array([  23.45388514, -247.43107084,  492.10188174,  329.36498638,
       -970.79784704,  573.54460121,  182.41386124,  255.9162021 ,
        794.21654496,   89.32074078])
```

In [11]:

```
lr.intercept_
```

Out[11]:

```
152.13619339070766
```

In [12]:

```
y_pred = lr.predict(X_test)
y_pred
```

Out[12]:

```
array([226.51666145, 157.46174876,  89.85952394, 207.92164744,
       175.26801312, 146.48388955, 131.11418872,  97.37619407,
       102.94176683, 181.69421036, 237.97133408, 134.74602577,
       189.44001676,  59.93090877, 179.98603311, 117.78640765,
       120.30704482, 126.66365457, 165.19760646, 147.7794185 ,
       145.43611319, 124.41603451,  51.95784538, 227.75026698,
       218.09974354, 129.82788151, 160.13813249, 201.1733737 ,
       184.84256427,  68.91311254, 237.00908624,  58.17038384,
       154.40273359, 119.16002486, 234.03288057, 172.80124283,
       139.94748943, 169.99141456, 214.59266744, 220.47790767,
       128.98001836, 186.20573984, 162.87312596, 179.77626906,
       107.76819766, 249.53008184, 140.92296676,  32.74662537,
       177.96075481, 145.76735049, 291.96466984, 125.71393834,
       107.27437771, 156.10370698, 115.93097942, 160.35431782,
       229.47682139, 173.38591961, 159.88140556, 123.04674096,
        88.5518427 , 122.87575187, 169.12903634, 101.97650772,
       282.75280809, 148.00612928, 164.73111978, 157.31651016,
       232.69943194, 121.12075222,  95.66048311, 186.24657654,
        87.52947663, 160.95896049, 242.3763008 , 149.13877387,
       164.93049903, 209.27728194, 112.08505539, 130.02848231,
        98.40717571,  43.04758443, 104.01721785, 227.29449724,
       144.03768453, 111.74963631, 154.06279876, 174.11068281,
        77.82721919])
```

In [13]:

```
y_test
```

Out[13]:

```
array([155., 168., 115., 233., 190., 202.,  59., 101., 118., 244., 252.,
       148., 232.,  72., 107.,  71., 191.,  65., 245.,  85., 185.,  84.,
        78., 268., 248., 178., 196., 248., 144.,  83., 275.,  39., 113.,
        64., 232., 200., 200., 122., 163., 180., 135., 164., 156., 126.,
        68., 306.,  83.,  45.,  91.,  25., 270., 178., 108.,  86., 160.,
       196., 248., 139., 155., 150.,  74.,  89., 216.,  65., 242., 136.,
       185.,  91., 246.,  68., 101., 164., 113., 131., 215., 246., 265.,
       220., 107., 131.,  94., 116.,  63., 217., 302.,  72., 252., 111.,
        72.])
```

In [14]:

```
from sklearn.linear_model import Ridge
```

In [16]:

```
R = Ridge(alpha=100000)
```

In [17]:

```
R.fit(X_train,y_train)
```

Out[17]:

```
Ridge(alpha=100000)
```

In [18]:

```
print("Coefficient of Ridge:")
R.coef_
```

```
Coefficient of Ridge:
```

Out[18]:

```
array([ 0.00260126,  0.00057066,  0.00776597,  0.00609765,  0.00233864,
        0.00184724, -0.00513942,  0.0052716 ,  0.00734601,  0.00528629])
```

In [19]:

```
print("Intercept of Ridge:")
R.intercept_
```

```
Intercept of Ridge:
```

Out[19]:

```
151.8328793076644
```

In [21]:

```
y_predi = R.predict(X_test)
y_predi
```

Out[21]:

```
array([151.83477275, 151.83386921, 151.83108257, 151.83398787,
       151.83402542, 151.83280281, 151.83242964, 151.83122134,
       151.83147431, 151.83410883, 151.83553309, 151.83198245,
       151.83405538, 151.83101673, 151.83358329, 151.83210612,
       151.83216681, 151.83301965, 151.83334727, 151.83265975,
       151.83148647, 151.8330864 , 151.83108105, 151.83567117,
       151.83556836, 151.83195992, 151.8343562 , 151.83385907,
       151.8335132 , 151.83092697, 151.83474929, 151.8306436 ,
       151.83273662, 151.83195511, 151.83431226, 151.83318655,
       151.83261999, 151.83398334, 151.83454502, 151.8351703 ,
       151.83221958, 151.8339252 , 151.83281469, 151.83334572,
       151.83170553, 151.83557763, 151.83260111, 151.83061765,
       151.83371664, 151.8319765 , 151.83526051, 151.83180247,
       151.83227793, 151.83219084, 151.83172006, 151.83356454,
       151.83549471, 151.83481515, 151.83249354, 151.83110293,
       151.83081263, 151.83358508, 151.8327506 , 151.83075787,
       151.83651784, 151.83173562, 151.83312968, 151.83383698,
       151.83419181, 151.83161468, 151.83061866, 151.83392834,
       151.83146232, 151.83349685, 151.83460543, 151.83252464,
       151.83328949, 151.83359012, 151.83291121, 151.83173837,
       151.83171502, 151.83080283, 151.83171494, 151.83605216,
       151.83232788, 151.83175297, 151.83270664, 151.83355905,
       151.83104514])
```

In [22]:

```
y_test
```

Out[22]:

```
array([155., 168., 115., 233., 190., 202.,  59., 101., 118., 244., 252.,
       148., 232.,  72., 107.,  71., 191.,  65., 245.,  85., 185.,  84.,
        78., 268., 248., 178., 196., 248., 144.,  83., 275.,  39., 113.,
        64., 232., 200., 200., 122., 163., 180., 135., 164., 156., 126.,
        68., 306.,  83.,  45.,  91.,  25., 270., 178., 108.,  86., 160.,
       196., 248., 139., 155., 150.,  74.,  89., 216.,  65., 242., 136.,
       185.,  91., 246.,  68., 101., 164., 113., 131., 215., 246., 265.,
       220., 107., 131.,  94., 116.,  63., 217., 302.,  72., 252., 111.,
        72.])
```

In [25]:

```
from sklearn.metrics import r2_score,mean_squared_error
print("R2 Score of Ridge L2 Regularisation:")
r2_score(y_test,y_predi)
```

```
R2 Score of Ridge L2 Regularisation:
```

Out[25]:

```
-0.0004249020401270176
```

In [29]:

```
print("Mean_Squared_Error of Ridge L2 Regularisation:")
mean_squared_error(y_test,y_predi)
```

```
Mean_Squared_Error of Ridge L2 Regularisation:
```

Out[29]:

```
4936.406155071465
```

In [30]:

```
m = 100
x1 = 5 * np.random.rand(m,1)-2
x2 = 0.7 * x1 ** 2-2 * x1 + np.random.randn(m,1)
```

In [31]:

```
x1
```

Out[31]:

```
array([[-1.18863277],
       [ 1.63330431],
       [ 2.44192555],
       [ 2.88659968],
       [-1.46394848],
       [ 1.62355957],
       [-1.98841678],
       [-0.61991712],
       [ 0.8288363 ],
       [ 2.16473134],
       [-1.79961759],
       [ 0.26007743],
       [ 1.61195521],
       [-1.24989114],
       [ 1.7114329 ],
       [-0.01296004],
       [ 1.98081106],
       [-0.1945169 ],
       [-0.5098491 ],
       [-1.26166746],
       [ 0.19488166],
       [ 1.76227641],
       [-1.87314935],
       [ 0.96750241],
       [-0.14418211],
       [ 1.72808006],
       [ 1.46883135],
       [-0.58097817],
       [-0.64227655],
       [ 0.39257723],
       [ 2.26045385],
       [ 2.25696045],
       [ 2.8210457 ],
       [ 2.73537919],
       [ 1.63037843],
       [-0.9091248 ],
       [-0.46213025],
       [-1.48784751],
       [ 1.9843541 ],
       [ 1.09615042],
       [-0.248069  ],
       [ 2.37349768],
       [-0.0793638 ],
       [-0.97608284],
       [ 2.95224159],
       [-1.64695138],
       [ 1.2826268 ],
       [-0.81536014],
       [-1.04039545],
       [-1.47027172],
       [ 0.40157483],
       [ 0.31897942],
       [ 1.4521466 ],
       [-1.55375725],
       [ 0.0631179 ],
```

```
       [-1.18528648],
       [ 1.89505391],
       [-0.32061357],
       [ 0.6849201 ],
       [-1.11141772],
       [-1.83894055],
       [ 1.53075577],
       [ 2.81243207],
       [-0.20375009],
       [-0.66654755],
       [-0.49627575],
       [-0.10200213],
       [ 0.71967209],
       [ 0.09034431],
       [ 1.63138809],
       [ 1.61407328],
       [-1.36317565],
       [ 2.95914605],
       [-0.4808781 ],
       [-1.54198151],
       [ 2.45264191],
       [-0.10617538],
       [ 0.99440202],
       [ 2.1860346 ],
       [-0.39739395],
       [-1.01498628],
       [ 0.23931524],
       [ 1.40757762],
       [ 2.39645289],
       [ 0.12292703],
       [ 1.94489123],
       [ 1.46166638],
       [ 0.57871831],
       [ 2.01111179],
       [-0.62266555],
       [ 1.99283492],
       [-1.63286832],
       [ 0.54407358],
       [-0.26108496],
       [ 2.75175334],
       [ 1.46744192],
       [ 0.09493854],
       [-1.32555539],
       [-0.11606588],
       [-0.28614129]])
```

In [32]:

```
x2
```

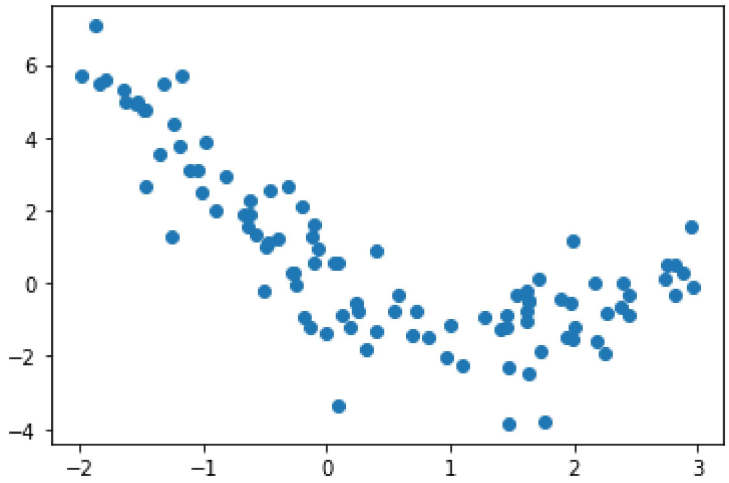
Out[32]:

```
array([[ 3.76317322],
       [-0.46210263],
       [-0.89721686],
       [ 0.28152903],
       [ 2.68275697],
       [-1.03275595],
       [ 5.69904617],
       [ 1.91465696],
       [-1.473192  ],
       [ 0.01469022],
       [ 5.56654243],
       [-0.7662882 ],
       [-0.19210501],
       [ 4.36771669],
       [ 0.1129644 ],
       [-1.38449706],
       [-0.56027851],
       [-0.90344308],
       [-0.23266473],
       [ 1.26665599],
       [-1.22854188],
       [-3.79851572],
       [ 7.06558262],
       [-2.02416998],
       [-1.20594268],
       [-1.8680826 ],
       [-3.84617688],
       [ 1.32230131],
       [ 1.55255916],
       [-1.31945153],
       [-0.79406346],
       [-1.9110256 ],
       [ 0.53448157],
       [ 0.13472234],
       [-2.45268938],
       [ 2.01909418],
       [ 2.56887224],
       [ 4.76273659],
       [-1.5458571 ],
       [-2.22269511],
       [-0.04654073],
       [-0.66492667],
       [ 0.95574109],
       [ 3.86280335],
       [ 1.56680123],
       [ 5.33867964],
       [-0.92093068],
       [ 2.92598043],
       [ 3.08167584],
       [ 4.77504695],
       [ 0.92409372],
       [-1.80994684],
       [-1.18641881],
       [ 4.93303592],
       [ 0.56514961],
```

```
       [ 5.7011503 ],
       [-0.4116621 ],
       [ 2.68564556],
       [-1.41907042],
       [ 3.10642   ],
       [ 5.4998089 ],
       [-0.33950934],
       [-0.34433995],
       [ 2.10400642],
       [ 1.88416493],
       [ 0.98379867],
       [ 0.5848366 ],
       [-0.7747257 ],
       [ 0.58669225],
       [-0.49988521],
       [-0.74349302],
       [ 3.54042013],
       [-0.09939267],
       [ 1.14301634],
       [ 4.99540512],
       [-0.29634107],
       [ 1.58887418],
       [-1.1238194 ],
       [-1.59149449],
       [ 1.2109695 ],
       [ 2.4874801 ],
       [-0.53116386],
       [-1.24116593],
       [ 0.02410421],
       [-0.89277949],
       [-1.50014036],
       [-0.87239678],
       [-0.3275309 ],
       [-1.1930289 ],
       [ 2.27392834],
       [ 1.1938397 ],
       [ 4.99867697],
       [-0.76815621],
       [ 0.31338983],
       [ 0.53180426],
       [-2.31634932],
       [-3.37664155],
       [ 5.49209309],
       [ 1.27687676],
       [ 0.31244893]])
```

In [33]:

```
plt.scatter(x1,x2)
plt.show()
```



In [34]:

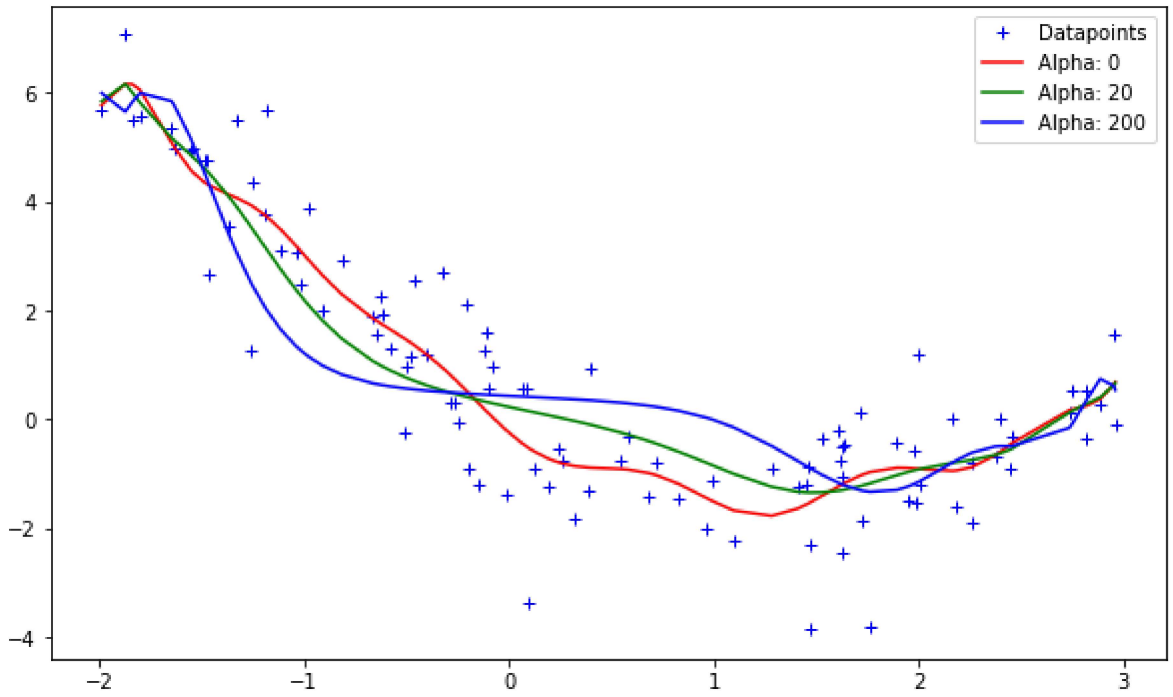
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
```

In [39]:

```
def get_preds_ridge(x1,x2,alpha):
    model = Pipeline([
        ('Poly_features',PolynomialFeatures(degree=16)),
        ('ridge',Ridge(alpha=alpha))
    ])
    model.fit(x1,x2)
    return model.predict(x1)

alphas = [0, 20, 200]
cs = ['r', 'g', 'b']

plt.figure(figsize=(10,6))
plt.plot(x1, x2, 'b+', label='Datapoints')

for alpha, c in zip(alphas, cs):
    preds = get_preds_ridge(x1,x2,alpha)
    plt.plot (sorted(x1[:,0]),preds[np.argsort(x1[:,0])],c, label = 'Alpha: {}'.format(alph
plt.legend()
plt.show()
```

Out[39]:

```
<matplotlib.legend.Legend at 0x1a311b1f850>
```



Lasso Regularization

In [2]:

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

In [3]:

```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso


from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
```

In [5]:

```
x,y = make_regression(n_samples=100,n_features=1,n_targets=1,noise=20,random_state=13)
```

In [6]:

```
x
```

Out[6]:

```
array([[-0.71239066],
       [-0.43714566],
       [-0.45375238],
       [ 0.95283061],
       [ 0.23785784],
       [ 0.86121137],
       [-0.24332625],
       [ 0.18494595],
       [-0.72099967],
       [-0.42989708],
       [ 2.01522083],
       [ 0.39724133],
       [ 0.20780005],
       [-0.23242587],
       [-0.76862702],
       [ 0.45315861],
       [ 0.63988397],
       [ 0.3595323 ],
       [-1.61510796],
       [ 1.74924179],
       [-0.78898902],
       [-0.51196509],
       [-0.92833523],
       [ 2.15038297],
       [-0.2073497 ],
       [-1.63909341],
       [-0.33861825],
       [-0.32212366],
       [-0.48137142],
       [-0.52316421],
       [ 0.72196506],
       [ 0.76591105],
       [ 0.45348104],
       [-1.26160595],
       [-2.18711527],
       [-1.18541881],
       [ 0.21745166],
       [ 1.33031692],
       [-1.08718159],
       [ 0.56226171],
       [-1.51284512],
       [-0.00238903],
       [-0.27813452],
       [ 0.45181234],
       [ 1.19070527],
       [ 0.92234415],
       [ 0.81499544],
       [-0.6209797 ],
       [ 0.9137407 ],
       [ 1.13833305],
       [ 1.47868574],
       [-0.65105648],
       [-0.37591996],
       [-0.77466003],
       [ 0.50113729],
```

```
       [ 1.3501879 ],
       [ 0.72916547],
       [-0.08165156],
       [-0.85414295],
       [ 0.46565797],
       [-0.04450308],
       [-0.05753239],
       [ 1.89274222],
       [-1.04537713],
       [ 0.56465429],
       [-1.92415945],
       [-0.76403397],
       [ 0.12730328],
       [-0.02677165],
       [-0.14521133],
       [ 0.56284679],
       [ 0.31735092],
       [ 0.71097479],
       [ 0.75376638],
       [-0.37011608],
       [ 1.34510171],
       [ 0.53233789],
       [-0.98416078],
       [ 1.350306  ],
       [-0.34660679],
       [ 0.51432886],
       [ 0.10126979],
       [-0.65751727],
       [ 0.83090566],
       [-0.31726597],
       [-0.98027432],
       [ 1.39923842],
       [ 0.54791831],
       [-0.53032741],
       [ 0.49087183],
       [ 0.34875059],
       [ 2.05369324],
       [ 0.60628866],
       [-0.38445769],
       [-1.94539068],
       [-0.31485808],
       [ 1.84961257],
       [-1.12050687],
       [-0.33267578],
       [-0.75745323]])
```

In [7]:

```
y
```

Out[7]:

```
array([-3.43198806e+01, -9.42120961e+00, -1.90881877e+01,  2.04372122e+01,
        2.77559659e+01, -2.90750046e+00, -1.41987828e+01,  5.40025891e+00,
       -2.64264302e+01, -3.49067872e+01,  3.73362043e+01,  1.28532816e+01,
        2.50289888e+01, -1.89608736e+01, -2.34655852e+01,  3.77839324e+01,
        6.69670792e+00, -5.57201352e+00, -4.92158778e+01,  1.59474399e+01,
       -4.29667324e+01,  6.09015466e+00, -2.53194769e+01,  6.28216706e+01,
        1.24870400e+01, -3.27136530e+01, -1.88255476e+01, -2.93912926e+01,
       -2.86886731e+01,  4.38924069e+00,  4.63542396e+01,  2.43919519e+01,
        3.79848517e+01, -3.45767718e+01, -6.18736296e+01, -4.64421597e+01,
       -6.88808416e+00,  3.96988084e+01, -3.52373298e+01,  8.36850884e+00,
       -3.96814412e+01,  8.27318308e+00, -4.40722161e+00, -3.01350607e+00,
        5.78213629e+01,  2.46525603e+01,  1.81131707e+01, -5.22849035e+01,
        3.59187182e+01,  1.58411788e+01,  2.40080546e+01, -2.51245994e+01,
       -4.39284313e+01,  1.73925049e+01,  1.50322799e+01,  3.78339073e+01,
        6.33015059e+00,  5.07266140e+00, -5.57047189e+00,  1.98411137e+01,
        7.04149700e+00, -1.89353310e+01,  3.48047372e+01, -5.71391242e+01,
        3.10524837e+01, -7.57291461e+01, -4.43062883e+01,  2.12404103e+01,
       -1.86288622e-01, -2.85657165e+01, -7.39169323e+00,  2.52239775e+01,
        3.25432136e+01,  2.93379129e-01, -1.86381423e+01,  3.93501682e+01,
        2.54005098e+00, -4.86320699e+01,  5.42818051e+01, -1.02170910e+01,
       -5.66921620e+00,  2.34628620e+01, -6.92299670e-02,  5.63079459e+00,
       -9.68771368e+00, -4.46638656e+01,  3.27083097e+01,  1.81174790e+01,
       -1.61432262e+01,  4.50414413e+01,  2.27466324e+01,  3.71085471e+01,
        1.24760264e+01, -3.58452002e+01, -2.86386661e+01, -2.79790662e+01,
        6.41794705e+01, -6.07154394e+01,  2.22333646e+01, -2.40222151e+00])
```

In [8]:

```
X_train,X_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=13)
```

In [9]:

```
from sklearn.linear_model import LinearRegression
```

In [10]:

```
lr = LinearRegression()
```

In [12]:

```
lr.fit(X_train,y_train)
```

Out[12]:

```
LinearRegression()
```

In [39]:

```
print("Coefficient of Regression")
m = lr.coef_
m
```

```
Coefficient of Regression
```

Out[39]:

```
array([28.67684216])
```

In [40]:

```
print("Intercept of Regression")
b= lr.intercept_
b
```

```
Intercept of Regression
```

Out[40]:

```
-2.088699766134101
```

In [18]:

```
Ri = Ridge(alpha=5)
```

In [19]:

```
Ri.fit(X_train,y_train)
```

Out[19]:

```
Ridge(alpha=5)
```

In [41]:

```
print("Coefficient of Regression")
m1= Ri.coef_
m1
```

```
Coefficient of Regression
```

Out[41]:

```
array([26.61000356])
```

In [42]:

```
print("Intercept of Regression")
b1=Ri.intercept_
b1
```

```
Intercept of Regression
```

Out[42]:

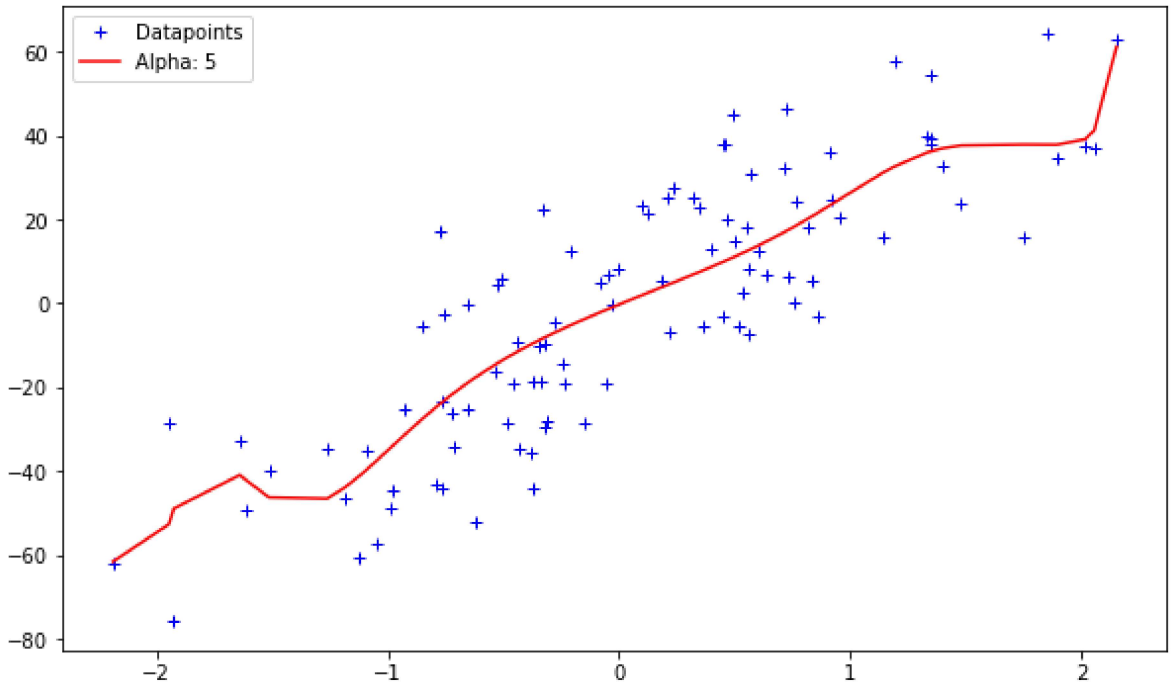
```
-2.201165618510922
```

In [23]:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
```

In [29]:

```
def get_preds_ridge(x,y,alpha):
    model = Pipeline([
        ('Poly_features',PolynomialFeatures(degree=16)),
        ('ridge',Ridge(alpha=alpha))
    ])
    model.fit(x,y)
    return model.predict(x)

alphas = [5]
cs = ['r']

plt.figure(figsize=(10,6))
plt.plot(x, y, 'b+', label='Datapoints')

for alpha, c in zip(alphas, cs):
    preds = get_preds_ridge(x,y,alpha)
    plt.plot (sorted(x[:,0]),preds[np.argsort(x[:,0])],c, label = 'Alpha: {}'.format(alpha)
plt.legend()
plt.show()
```

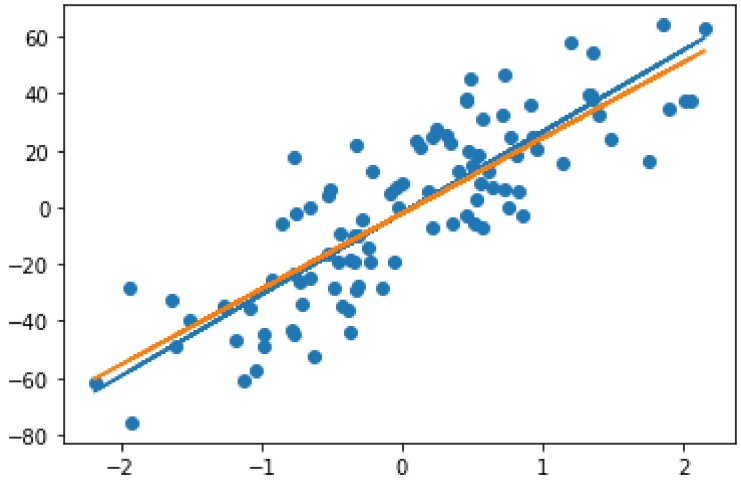


In [35]:

```
plt.scatter(x,y)
plt.plot(x, m*x + b)
plt.plot(x, m1*x + b1)
```

Out[35]:

```
[<matplotlib.lines.Line2D at 0x2af0f28e8b0>]
```

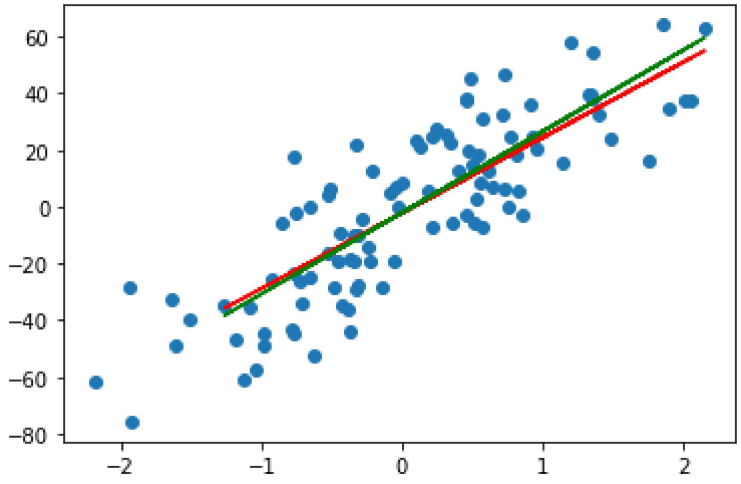


In [43]:

```
plt.scatter(x,y)
plt.plot(X_test,Ri.predict(X_test),label='Ridge',c='r')
plt.plot(X_test,lr.predict(X_test),label='Linear',c='g')
```

Out[43]:

```
[<matplotlib.lines.Line2D at 0x2af103e0400>]
```



In [46]:

```
la = Lasso(alpha=10)
```

In [47]:

```
la.fit(X_train,y_train)
```

Out[47]:

```
Lasso(alpha=10)
```

In [48]:

```
print("Coefficient of Regression")
la.coef_
```

```
Coefficient of Regression
```

Out[48]:

```
array([16.24940391])
```

In [51]:

```
print("Intercept of Regression")
la.intercept_
```

```
Intercept of Regression
```

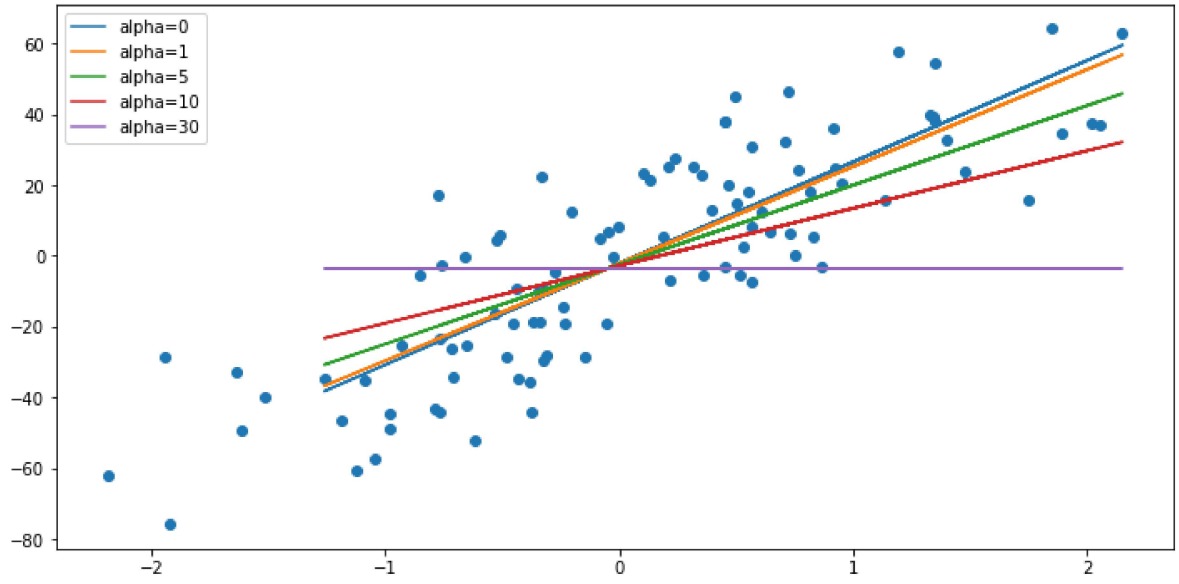
Out[51]:

```
-2.7649317831463702
```

In [52]:

```
alphas = [0,1,5,10,30]
plt.figure(figsize=(12,6))
plt.scatter(x,y)
for i in alphas:
    L = Lasso(alpha=i)
    L.fit(X_train,y_train)
    plt.plot(X_test,L.predict(X_test),label='alpha={}'.format(i))
plt.legend()
plt.show()
```

```
<ipython-input-52-d0acee6e9477>:6: UserWarning: With alpha=0, this algorithm
does not converge well. You are advised to use the LinearRegression estimato
r
  L.fit(X_train,y_train)
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_
descent.py:530: UserWarning: Coordinate descent with no regularization may l
ead to unexpected results and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_
descent.py:530: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations. Duality gap: 12057.35658838604, tol
erance: 7.705313925235513
  model = cd_fast.enet_coordinate_descent(
```



In [53]:

```
from sklearn.datasets import load_diabetes
```

In [54]:

```
data = load_diabetes()
```

In [66]:

```
x1 = data.data
y1 = data.target
```

In [67]:

```
x1
```

Out[67]:

```
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]])
```

In [68]:

```
y1
```

Out[68]:

```
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.])
```

In [69]:

```
df = pd.DataFrame(x1,y1,columns=data.feature_names)
```

In [70]:

```
df
```

Out[70]:

	age	sex	bmi	bp	s1	s2	s3	s4	
151.0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0
75.0	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0
141.0	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0
206.0	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0
135.0	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0
...	...	...	...	...	...	...	...	...	
178.0	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0
104.0	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0
132.0	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0
220.0	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0
57.0	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0

442 rows × 10 columns

In [71]:

```
df.shape
```

Out[71]:

```
(442, 10)
```

In [77]:

```
x1_train,x1_test,y1_train,y1_test = train_test_split(x1,y1,test_size=0.2,random_state=2)
```

In [78]:

```
from sklearn.metrics import r2_score
coefs = []
r2_scores = []
for i in [0,0.1,1,10]:
    reg = Lasso(alpha=i)
    reg.fit(x1_train,y1_train)
    coefs.append(reg.coef_.tolist())
    y_pred = reg.predict(x1_test)
    r2_scores.append(r2_score(y1_test,y_pred))
```

```
<ipython-input-78-5349f46ad3fe>:6: UserWarning: With alpha=0, this algorithm
does not converge well. You are advised to use the LinearRegression estimato
r
  reg.fit(x1_train,y1_train)
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_
descent.py:530: UserWarning: Coordinate descent with no regularization may l
ead to unexpected results and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_
descent.py:530: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations. Duality gap: 496708.2205472759, tol
erance: 212.4353654390935
  model = cd_fast.enet_coordinate_descent(
```

In [79]:

```
r2_scores
```

Out[79]:

```
[0.4399387661037827,
 0.4334654091230754,
 0.32568169677801695,
 -0.012517603619692785]
```

In [80]:

```
coefs
```

Out[80]:

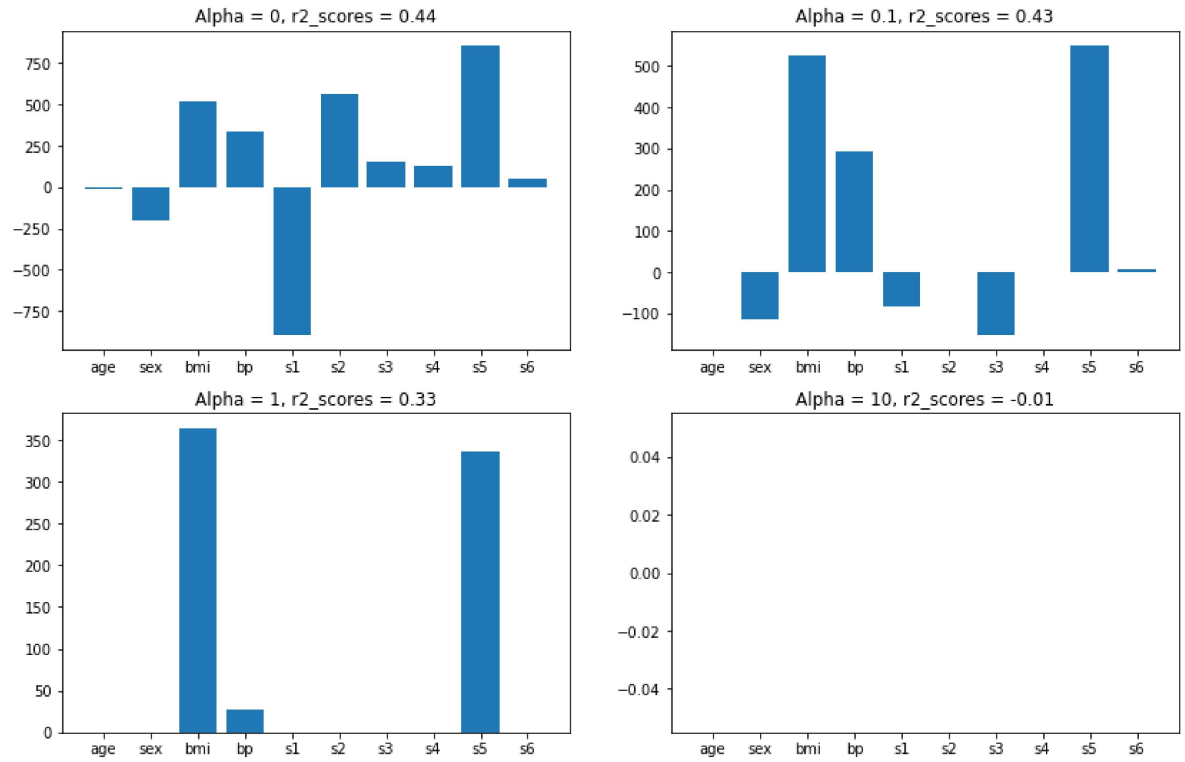
```
[[-9.160884830102175,
  -205.46225976623592,
  516.6846240655403,
  340.62734101990014,
  -895.5435958123823,
  561.2145229912762,
  153.88478023466485,
  126.73431430271964,
  861.1213947849758,
  52.41982835037267],
 [0.0,
  -113.97604628790536,
  526.7371120851084,
  292.6354225287151,
  -82.69192843359384,
  -0.0,
  -152.6913315736965,
  0.0,
  551.0771996122245,
  7.16985202642276],
 [0.0,
  0.0,
  363.88263603837737,
  27.27842001502803,
  0.0,
  0.0,
  -0.0,
  0.0,
  336.13597084247954,
  0.0],
 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0, 0.0]]
```

In [84]:

```
plt.figure(figsize=(14,9))
plt.subplot(221)
plt.bar(data.feature_names,coefs[0])
plt.title('Alpha = 0, r2_scores = {}'.format(round(r2_scores[0],2)))

plt.subplot(222)
plt.bar(data.feature_names,coefs[1])
plt.title('Alpha = 0.1, r2_scores = {}'.format(round(r2_scores[1],2)))

plt.subplot(223)
plt.bar(data.feature_names,coefs[2])
plt.title('Alpha = 1, r2_scores = {}'.format(round(r2_scores[2],2)))

plt.subplot(224)
plt.bar(data.feature_names,coefs[3])
plt.title('Alpha = 10, r2_scores = {}'.format(round(r2_scores[3],2)))
```

Out[84]:

```
Text(0.5, 1.0, 'Alpha = 10, r2_scores = -0.01')
```



In [89]:

In [90]:

```
alphas = [0,0.0001,0.001,0.01,0.1,1,10,100,1000,10000]
coefs = []

for i in alphas:
    reg = Lasso(alpha=i)
    reg.fit(x1_train,y1_train)
    coefs.append(reg.coef_.tolist())
```

```
<ipython-input-90-71c94ca80c83>:6: UserWarning: With alpha=0, this algorithm
does not converge well. You are advised to use the LinearRegression estimato
r
  reg.fit(x1_train,y1_train)
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_
descent.py:530: UserWarning: Coordinate descent with no regularization may l
ead to unexpected results and is discouraged.
  model = cd_fast.enet_coordinate_descent(
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_
descent.py:530: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations. Duality gap: 496708.2205472759, tol
erance: 212.4353654390935
  model = cd_fast.enet_coordinate_descent(
```

In [92]:

```
input_array = np.array(coefs)
coef_df = pd.DataFrame(input_array,columns=data.feature_names)
coef_df['alpha'] = alphas
coef_df.set_index('alpha')
```

Out[92]:

alpha	age	sex	bmi	bp	s1	s2	s
0.0000	-9.160885	-205.462260	516.684624	340.627341	-895.543596	561.214523	153.88478
0.0001	-9.071288	-205.337332	516.780313	340.539730	-888.652320	555.952271	150.58526
0.0010	-8.264924	-204.213177	517.641106	339.751339	-826.653342	508.609613	120.89958
0.0100	-1.361404	-192.944226	526.348511	332.649058	-430.205495	191.277876	-44.04811
0.1000	0.000000	-113.976046	526.737112	292.635423	-82.691928	-0.000000	-152.69133
1.0000	0.000000	0.000000	363.882636	27.278420	0.000000	0.000000	-0.00000
10.0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.00000
100.0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.00000
1000.0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.00000
10000.0000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.00000

In [102]:

```
coef_df
```

Out[102]:

	age	sex	bmi	bp	s1	s2	s3	
0	-9.160885	-205.462260	516.684624	340.627341	-895.543596	561.214523	153.884780	126.73
1	-9.071288	-205.337332	516.780313	340.539730	-888.652320	555.952271	150.585260	125.45
2	-8.264924	-204.213177	517.641106	339.751339	-826.653342	508.609613	120.899583	113.92
3	-1.361404	-192.944226	526.348511	332.649058	-430.205495	191.277876	-44.048113	68.99
4	0.000000	-113.976046	526.737112	292.635423	-82.691928	-0.000000	-152.691332	0.00
5	0.000000	0.000000	363.882636	27.278420	0.000000	0.000000	-0.000000	0.00
6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.00
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.00
8	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.00
9	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.00

In [106]:

```
reg = Ridge(alpha=0.1)
reg.fit(x1_train,y1_train)
y_pred = reg.predict(x1_test)
r2_score(y1_test,y_pred)
```

Out[106]:

```
0.4519973816947852
```

In [107]:

```
reg = Lasso(alpha=0.1)
reg.fit(x1_train,y1_train)
y_pred = reg.predict(x1_test)
r2_score(y1_test,y_pred)
```

Out[107]:

```
0.4334654091230754
```

ElasticNet

In [105]:

```
from sklearn.linear_model import ElasticNet
```

In [95]:

```
reg = ElasticNet(alpha=0.005,l1_ratio=0.9)
reg.fit(x1_train,y1_train)
y_pred = reg.predict(x1_test)
r2_score(y1_test,y_pred)
```

Out[95]:

```
0.4531493801165679
```

In [ ]:

```

```