L1 nd L2

February 16, 2026

[3]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression,Lasso,Ridge
```

[4]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
```

[5]:
```
import numpy as np
```

[ ]:
```

```

[8]:
```
df = pd.read_csv("dataset.csv")
df.head()
```

[8]:

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	…	f142	f143	f144	f145	f146	f147	f148	f149	f150	target
0	-0.924357	-0.326536	-1.875007	-1.780626	-0.630143	0.788204	2.792209	-0.772192	-0.450994	0.400000	…	0.968645	-0.702053	-0.327662	-0.392108	-1.463515	0.296120	0.261055	0.005113	-0.234587	10.681366
1	0.001795	-1.285599	-0.726774	0.385711	0.891863	0.599451	-0.140553	-0.761760	0.117707	0.333231	…	0.856399	0.214094	-1.245739	0.173181	0.385317	-0.883857	0.153725	0.058209	-1.142970	-60.163343
2	0.956702	2.319330	-0.705012	0.081829	0.330880	0.838491	2.493000	1.227669	-0.785989	-0.920674	…	-0.493001	-0.589365	0.849602	0.357015	-0.692910	0.899600	0.307300	0.812862	0.629629	131.226545
3	-0.982294	0.190424	1.082691	0.714610	-1.907808	0.224216	0.156973	0.556553	0.058984	-1.397118	…	0.491919	-1.320233	1.831459	1.179440	-0.469176	-1.713135	1.353872	-0.114540	1.237816	-131.889020
4	-0.057061	-0.592465	-2.850030	-1.935430	2.002427	-1.012644	1.059950	-1.000629	0.141596	-0.642776	…	1.479944	0.077368	-0.861284	1.523124	0.538910	-1.037246	-0.190339	-0.875618	-1.382800	-138.566956

[5 rows x 151 columns]

[9]:
```
# split features and target columns
X=df.drop(columns=["target"],axis=1)
y=df[["target"]]
```

[10]:
```
# apply train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪2,random_state=42)
```

[13]:
```
# now fit the model
lm= LinearRegression()
lm.fit(X_train,y_train)
```

[13]:
```
LinearRegression()
```

[14]:
```
lm.score(X_test,y_test)
```

[14]:
```
0.8656382496943877
```

[20]:
```
lm.coef_
```

[20]:
```
array([[ 16.09991474, -22.63376235,  28.96109738,  -5.38609505,
         -6.59142675, -17.14586485,  26.38138158,  72.18567052,
        -22.61915958, -30.22835704, -11.55296337,  -4.4899415 ,
         12.19009326,  -3.58275018, -11.51687844,   9.66448518,
         20.14591539, -10.45313204,  14.6661052 ,  82.64905762,
        -12.2080552 ,  16.75579843,  -9.05049485,  39.93369362,
          4.79024103,  -7.16223267,   0.70102237,  -0.58181591,
         -1.77336497,   4.12975986,   4.62395143,  -8.36419857,
         78.42444413,   3.6307766 ,  -1.09459776,   5.11089404,
          1.85035023,   2.49751151,   3.05190887,   6.50431128,
         -3.62701413,   0.46826063,   1.46777465,   2.45875141,
         -1.43324418,   2.95444345,  -2.51507289,  -0.39269459,
         -0.62514819,  -4.46637275, -17.87892607,  23.47855275,
         -2.26073964,   3.88326112,  34.85163853,  14.3241288 ,
        -28.79051415,  -8.14129702,  16.09691136,  33.19117032,
          9.5565303 ,   5.23151557, -16.49424669,  -1.80621724,
         14.37332998, -11.59412477,  17.51875371,  11.53637794,
         -6.44652541,  -9.88827935,   8.97839157,  -9.51888418,
         10.04622596, -29.8871577 ,  -5.92242987,   1.45819983,
         -0.62153818,   2.87751412,   0.35401519,   0.31587749,
          1.41927317,   0.35222994,  -3.684496  ,   7.40321398,
          0.38100724,   4.48635766,   6.06302095,   1.80353362,
          7.54250458,   1.49098302,   6.38865687,  -1.5713567 ,
          0.5151771 ,   0.51171514,  -2.16215451,   0.58561864,
          6.65569182,   3.88520043,  -7.43158235,  -2.91848647,
          0.47978074,   2.62370224,  -0.37412035,   0.14837353,
          1.17390546,   3.67464834,   2.38395572,  -2.16544621,
          5.78650167,   4.7775463 ,   1.81818406,  -0.85874745,
```

```
       -2.63903169,  -6.6825241 ,  -2.85999844,  -4.41028438,
       -1.20270594,  -0.91467825,  -3.21771016,  -2.46120788,
       -1.15191548,   3.65264178,   3.92358807,   2.70146867,
       -1.96429675,   2.23660027,  -1.76699083,  -0.86051013,
        2.75241126,  -0.86538069,  -0.18182279,  -7.99403672,
       -0.29619632,  -2.11185252,   1.50097536,   1.7426837 ,
       -5.4708664 ,  -3.01376959,  -3.64984677,  -4.13697723,
        1.3109445 ,   0.13412157,  -2.93040205,  -3.55684315,
        1.76530163,  -3.20693131,  -5.38989471,   3.93471188,
        1.89006553,  -7.53269528]])
```

[19]:
```
# applaying lasso and checking score
lasso_model=Lasso(alpha=1.0)
lasso_model.fit(X_train,y_train)
lasso_model.score(X_test,y_test)
```

[19]:
```
0.9905099255716072
```

[21]:
```
lasso_model.coef_
```

[21]:
```
array([-0.00000000e+00, -5.34619854e-01,  2.25509740e+01,  0.00000000e+00,
        3.05493434e+01,  0.00000000e+00, -0.00000000e+00,  5.74343267e+01,
       -0.00000000e+00,  0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
        3.23225581e+01, -1.09232307e-01,  6.35091548e+00,  7.57167276e+01,
       -0.00000000e+00,  9.05547571e+00,  0.00000000e+00,  7.14020137e+00,
        0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  2.35366264e-01,
       -0.00000000e+00, -0.00000000e+00,  0.00000000e+00, -4.00439224e-01,
        7.41718158e+01, -0.00000000e+00,  0.00000000e+00,  3.45811190e+00,
       -0.00000000e+00,  0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
       -3.53466609e-01, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
        5.98903162e+00, -0.00000000e+00,  0.00000000e+00,  4.89773505e-01,
       -0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00, -7.65395699e-01, -1.10250462e+00,
        1.71246844e-01, -0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
        0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00, -5.17127604e-01, -0.00000000e+00,  0.00000000e+00,
        4.40128604e-01,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
       -8.86086508e-01,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
        0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
```

```
 1.83761643e-01, -0.00000000e+00,  1.31185828e-01, -0.00000000e+00,
-0.00000000e+00, -0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
-3.92334464e-01, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
 0.00000000e+00, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
 0.00000000e+00, -0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
-0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
-0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
-1.78027677e-01, -0.00000000e+00,  0.00000000e+00, -5.61230501e-02,
-0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00, -0.00000000e+00])
```

[22]:
```
# we know higher number of irrelevent features contributes to overfitting
```

[23]:
```
# By applying Regularization we have make the value of Beta close to zero that␣
↪do not contributes much in predecting the target value.
```

[ ]:
```

```