ModelSElandREG

November 3, 2021

SL Lab with Python 5: Model Selection and Regularization

Statistical Learning (Sejong University)

Date: 2021.11.03 (By: S. M. Riazul Islam)

```
In [60]: import numpy as np
         import pandas as pd

         import matplotlib.pyplot as plt
         import seaborn as sns

         %matplotlib inline
```

Principal Component Analysis (PCA)

Breast Cancer Data from scikit-learn

```
In [61]: from sklearn.datasets import load_breast_cancer
         bc = load_breast_cancer()
         bc.keys()
```

```
Out[61]: dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

```
In [62]: print(bc['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
```

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

- class:
  - WDBC-Malignant
  - WDBC-Benign

:Summary Statistics:

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2

```
    area (worst):                          185.2  4254.0
    smoothness (worst):                    0.071  0.223
    compactness (worst):                   0.027  1.058
    concavity (worst):                     0.0    1.252
    concave points (worst):                0.0    0.291
    symmetry (worst):                      0.156  0.664
    fractal dimension (worst):             0.055  0.208
    ====================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References
```

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) 163-171.

```
In [67]: print(bc['data'])
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

```
In [68]: print(bc['feature_names'])
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

```
In [69]: bcdata = pd.DataFrame(bc['data'],columns=bc['feature_names'])
```

```
In [70]: bcdata.head()
```

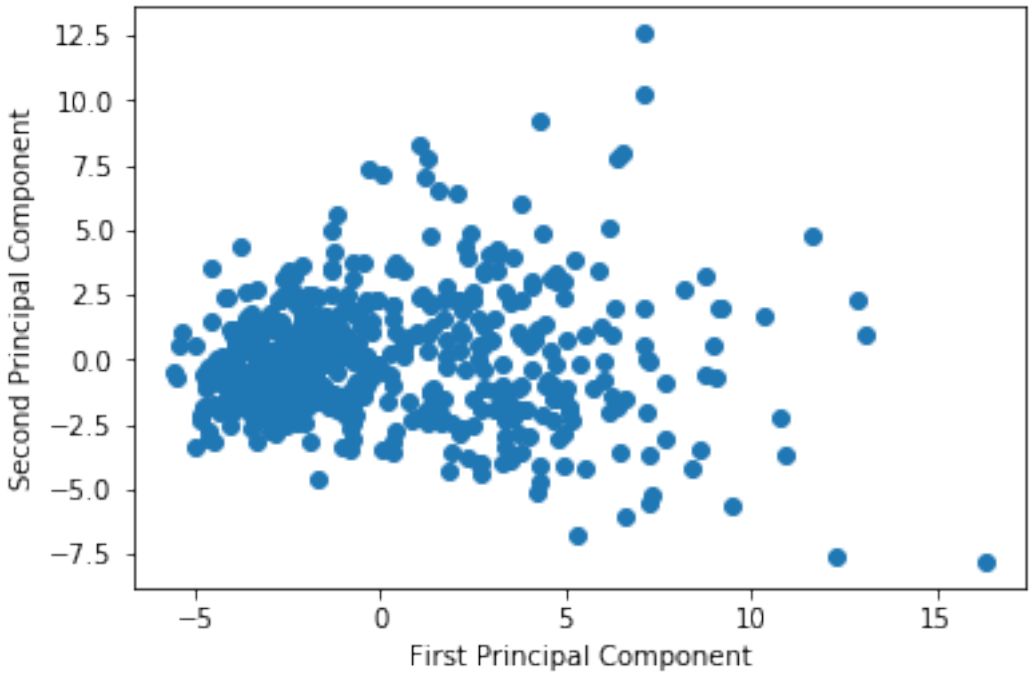
```
Out[70]:    mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
         0        17.99         10.38          122.80     1001.0          0.11840
         1        20.57         17.77          132.90     1326.0          0.08474
         2        19.69         21.25          130.00     1203.0          0.10960
         3        11.42         20.38           77.58      386.1          0.14250
         4        20.29         14.34          135.10     1297.0          0.10030

            mean compactness  mean concavity  mean concave points  mean symmetry  \
         0           0.27760          0.3001              0.14710         0.2419
```

```
1           0.07864          0.0869            0.07017         0.1812
2           0.15990          0.1974            0.12790         0.2069
3           0.28390          0.2414            0.10520         0.2597
4           0.13280          0.1980            0.10430         0.1809

   mean fractal dimension  ...  worst radius  worst texture  worst perimeter  \
0                 0.07871  ...         25.38          17.33           184.60
1                 0.05667  ...         24.99          23.41           158.80
2                 0.05999  ...         23.57          25.53           152.50
3                 0.09744  ...         14.91          26.50            98.87
4                 0.05883  ...         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

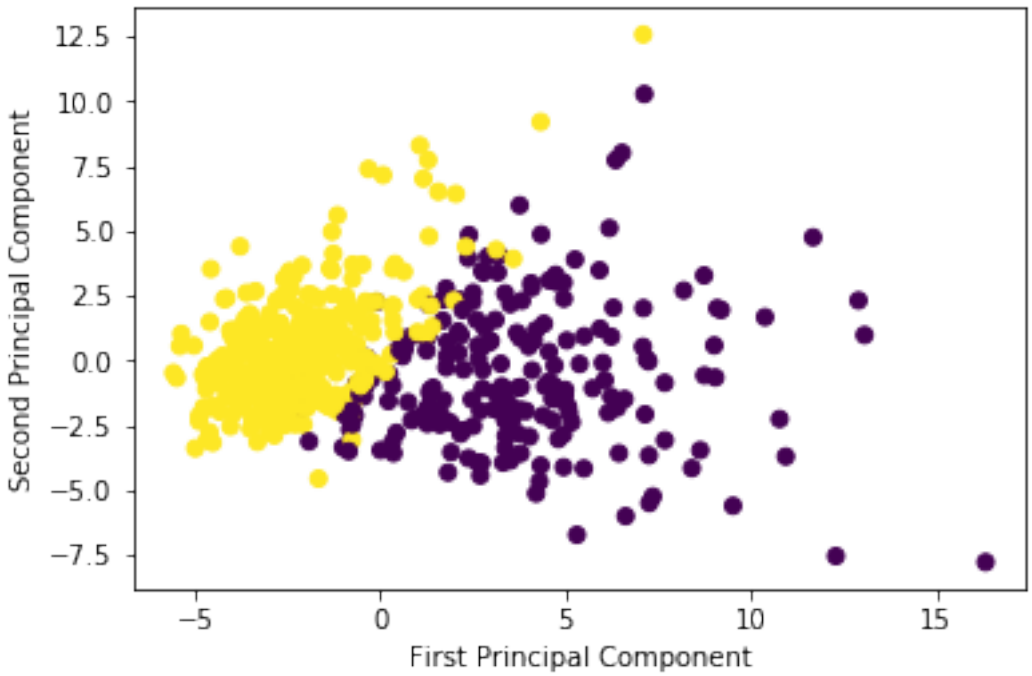
Standard Scaling

```
In [71]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         scaler.fit(bcdata)

         scaled_bcdata = scaler.transform(bcdata)
         # numpy array, not pandas dataframe

In [72]: type(scaled_bcdata)

Out[72]: numpy.ndarray

In [73]: from sklearn.decomposition import PCA
         pca = PCA(n_components=2)
         pca.fit(scaled_bcdata)

         pca_data = pca.transform(scaled_bcdata)
         # PCA scores

In [74]: scaled_bcdata.shape
```

```
Out[74]: (569, 30)

In [75]: pca_data.shape

Out[75]: (569, 2)

In [76]: # Plotting PCA Data
         plt.scatter(pca_data[:,0],pca_data[:,1])
         plt.xlabel('First Principal Component')
         plt.ylabel('Second Principal Component')

Out[76]: Text(0, 0.5, 'Second Principal Component')
```




```
In [77]: # Plotting PCA Data
         plt.scatter(pca_data[:,0],pca_data[:,1], c=bc['target'])
         plt.xlabel('First Principal Component')
         plt.ylabel('Second Principal Component')

Out[77]: Text(0, 0.5, 'Second Principal Component')
```



Ridge Regression and Lasso

```
In [78]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
```

Working with Boston Data Set

```
In [79]: from sklearn.datasets import load_boston
         boston = load_boston()
         boston.keys()
```

```
Out[79]: dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

```
In [80]: print(boston['DESCR'])
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506
```

```
    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is us

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon Universi

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regressi
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources o
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on th
```

```
In [81]: print(boston['data'])
```

```
[[6.3200e-03 1.8000e+01 2.3100e+00 ... 1.5300e+01 3.9690e+02 4.9800e+00]
 [2.7310e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9690e+02 9.1400e+00]
```

```
 [2.7290e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9283e+02 4.0300e+00]
 ...
 [6.0760e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 5.6400e+00]
 [1.0959e-01 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9345e+02 6.4800e+00]
 [4.7410e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 7.8800e+00]]
```

```
In [82]: print(boston['feature_names'])
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

```
In [83]: housedata = pd.DataFrame(boston['data'],columns=boston['feature_names'])

In [84]: housedata.head()
```

Out[84]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0

	PTRATIO	B	LSTAT
0	15.3	396.90	4.98
1	17.8	396.90	9.14
2	17.8	392.83	4.03
3	18.7	394.63	2.94
4	18.7	396.90	5.33

```
In [85]: housedata["Price"]=boston.target

In [86]: housedata.head()
```

Out[86]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0

	PTRATIO	B	LSTAT	Price
0	15.3	396.90	4.98	24.0
1	17.8	396.90	9.14	21.6
2	17.8	392.83	4.03	34.7
3	18.7	394.63	2.94	33.4
4	18.7	396.90	5.33	36.2

```
In [87]: X=housedata.iloc[:,:-1]
         # Predictors

         y=housedata.iloc[:,-1]
         # Response
```

Linear Regression

```
In [88]: from sklearn.linear_model import LinearRegression
         from sklearn.model_selection import cross_val_score

         lin_regressor=LinearRegression()
         mse=cross_val_score(lin_regressor,X,y,scoring='neg_mean_squared_error',cv=5)
         print(mse)
```

```
[-12.46030057 -26.04862111 -33.07413798 -80.76237112 -33.31360656]
```

```
In [89]: mean_mse=np.mean(mse)
         print(mean_mse)
```

```
-37.13180746769922
```

Ridge Regression

```
In [90]: from sklearn.linear_model import Ridge
         from sklearn.model_selection import GridSearchCV

         alphas = 10**np.linspace(10,-2,100)*0.5

         ridge=Ridge()
         parameters={'alpha':alphas}
         ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
         ridge_regressor.fit(X,y)

         print(ridge_regressor.best_params_)
         print(ridge_regressor.best_score_)
```

```
{'alpha': 152.69277544167062}
-29.721340277199285
```

```
C:\Users\Preload\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:841: Deprecati
  DeprecationWarning)
```

LASSO

```
In [91]: from sklearn.linear_model import Lasso
         from sklearn.model_selection import GridSearchCV

         alphas = 10**np.linspace(10,-2,100)*0.5

         lasso=Lasso()
         parameters={'alpha':alphas}
         lasso_regressor=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error',cv=5)
         lasso_regressor.fit(X,y)

         print(lasso_regressor.best_params_)
         print(lasso_regressor.best_score_)
```

```
{'alpha': 0.3289666123287841}
-34.39670631558307
```

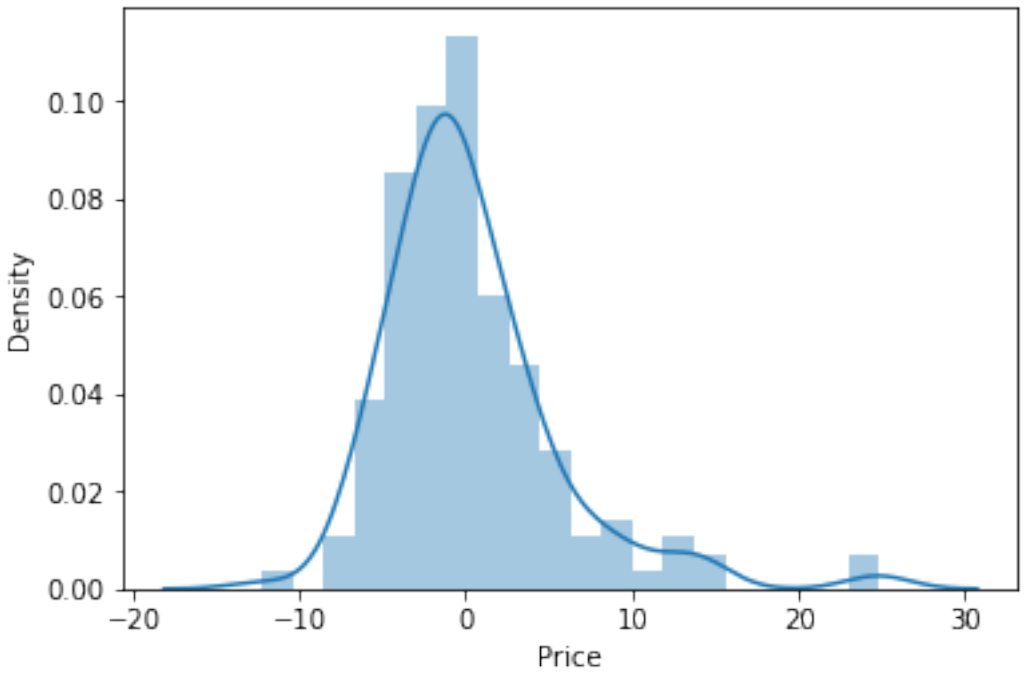
```
C:\Users\Preload\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:841: Deprecatio
  DeprecationWarning)
```

Test Error: Redge Regression and LASSO

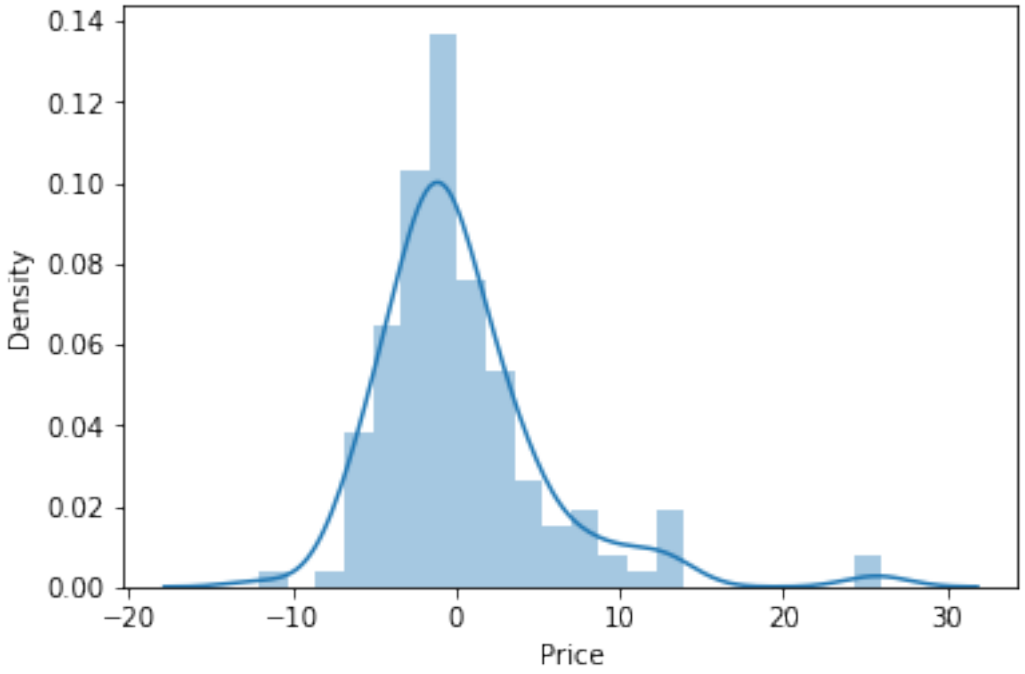
```
In [92]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=

In [93]: # Prediction
         prediction_ridge=ridge_regressor.predict(X_test)
         prediction_lasso=lasso_regressor.predict(X_test)

In [94]: import seaborn as sns
         sns.distplot(y_test-prediction_ridge)
         #sns.histplot(y_test-prediction_ridge)
```

```
C:\Users\Preload\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `dis
  warnings.warn(msg, FutureWarning)
```

```
Out[94]: <matplotlib.axes._subplots.AxesSubplot at 0x2851c5e6908>
```




```
In [95]: sns.distplot(y_test-prediction_lasso)
         #sns.histplot(y_test-prediction_lasso)

C:\Users\Preload\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `dis
  warnings.warn(msg, FutureWarning)


Out[95]: <matplotlib.axes._subplots.AxesSubplot at 0x2851c68b668>
```


0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
−20
−10
0
10
20
30
Price