Regularization

- It is one of the most important concepts of machine learning. This technique prevents the model from overfitting by adding extra information to it.
- It is a form of regression that shrinks the coefficient estimates towards zero. In other words, this technique forces us not to learn a more complex or flexible model, to avoid the problem of overfitting.
- Now, let's understand the "How flexibility of a model is represented?" For regression problems, the increase in flexibility of a model is represented by an increase in its coefficients, which are calculated from the regression line.
- In simple words, "In the Regularization technique, we reduce the magnitude of the independent variables by keeping the same number of variables". It maintains accuracy as well as a generalization of the model.

Why Regularization?

Sometimes what happens is that our Machine learning model performs well on the training data but does not perform well on the unseen or test data. It means the model is not able to predict the output or target column for the unseen data by introducing noise in the output, and hence the model is called an overfitted model.

Let's understand the meaning of "Noise" in a brief manner:

By noise we mean those data points in the dataset which don't really represent the true properties of your data, but only due to a random chance.

How does Regularization Work?

Regularization works by adding a penalty or complexity term or shrinkage term with Residual Sum of Squares (RSS) to the complex model.

Let's consider the Simple linear regression equation:

Here Y represents the dependent feature or response which is the learned relation. Then,

Y is approximated to $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$

Here, $X_1, X_2, \ldots X_p$ are the independent features or predictors for Y, and

$\beta_0, \beta_1, \ldots \beta_n$ represents the coefficients estimates for different variables or predictors(X), which describes the weights or magnitude attached to the features, respectively.

In simple linear regression, our optimization function or loss function is known as the residual sum of squares (RSS).

We choose those set of coefficients, such that the following loss function is minimized:

$$\mathrm{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 .$$

Now, this will adjust the coefficient estimates based on the training data. If there is noise present in the training data, then the estimated coefficients won't generalize well and are not able to predict the future data.

This is where regularization comes into the picture, which shrinks or regularizes these learned estimates towards zero, by adding a loss function with optimizing parameters to make a model that can predict the accurate value of Y.

Techniques of Regularization

Mainly, there are two types of regularization techniques, which are given below:

- Ridge Regression
- Lasso Regression

Ridge Regression

Ridge Regression is another type of regression algorithm in data science and is usually considered when there is a high correlation between the independent variables or model parameters. As the value of correlation increases the least square estimates evaluates unbiased values. But if the collinearity in the dataset is very high, there can be some bias value. Therefore, we create a bias matrix in the equation of Ridge Regression algorithm. It is a useful regression method in which the model is less susceptible to overfitting and hence the model works well even if the dataset is very small.

In Statistics, it is known as the L-2 norm.

In this technique, the cost function is altered by adding the penalty term (shrinkage term), which multiplies the lambda with the squared weight of each individual feature. Therefore, the optimization function(cost function) becomes:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \mathrm{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$$

Mathematical Formulation

For ridge regression, the total sum of squares of coefficients is less than or equal to s

Here, s is a constant which exists for each value of the shrinkage factor λ.

These equations are also known as constraint functions.

👉 Let's take an example to understand the mathematical formulation clearly,

For Example, Consider there are 2 parameters for a given problem

According to the above mathematical formulation, the ridge regression is described by $\beta_1^2 + \beta_2^2 \le s$.

This implies that ridge regression coefficients have the smallest RSS (loss function) for all points that lie within the circle given by $\beta_1^2 + \beta_2^2 \le s$.

Usage of Ridge Regression:

- When we have the independent variables which are having high collinearity (problem of multicolliniearity) between them, at that time general linear or polynomial regression will fail so to solve such problems, Ridge regression can be used.
- If we have more parameters than the samples, then Ridge regression helps to solve the problems.

Limitation of Ridge Regression:

- Not helps in Feature Selection: It decreases the complexity of a model but does not reduce the number of independent variables since it never leads to a coefficient being zero rather only minimizes it. Hence, this technique is not good for feature selection.
- Model Interpretability: Its disadvantage is model interpretability since it will shrink the coefficients for least important predictors, very close to zero but it will never make them exactly zero. In other words, the final model will include all the independent variables, also known as predictors.

Lasso Regression

The word "LASSO" denotes Least Absolute Shrinkage and Selection Operator. Lasso regression follows the regularization technique to create prediction. It is given more priority over the other regression methods because it gives an accurate prediction. Lasso regression model uses shrinkage technique. In this technique, the data values are shrunk towards a central point similar to the concept of mean. The lasso regression algorithm suggests a simple, sparse models (i.e. models with fewer parameters), which is well-suited for models or data showing high levels of multicollinearity or when we would like to automate certain parts of model selection, like variable selection or parameter elimination using feature engineering.

It is similar to the Ridge Regression except that the penalty term includes the absolute weights instead of a square of weights. Therefore, the optimization function becomes:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

Mathematical Formulation

Lasso regression, the total sum of modulus of coefficients is less than or equal to s.

Here, s is a constant which exists for each value of the shrinkage factor λ.

These equations are also known as constraint functions.

Let's take an example to understand the mathematical formulation clearly,

For Example, Consider there are 2 parameters for a given problem

According to the above mathematical formulation, the equation becomes,$|\beta_1|+|\beta_2| \le s$.

This implies that the coefficients for lasso regression have the smallest RSS (loss function) for all points that lie within the diamond given by $|\beta_1|+|\beta_2| \le s$.

In statistics, it is known as the L-1 norm.

In this technique, the L1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero which means there is a complete removal of some of the features for model evaluation when the tuning parameter λ is sufficiently large. Therefore, the lasso method also performs Feature selection and is said to yield sparse models.

Limitation of Lasso Regression:

- Problems with some types of Dataset: If the number of predictors is greater than the number of data points, Lasso will pick at most n predictors as non-zero, even if all predictors are relevant.
- Multicollinearity Problem: If there are two or more highly collinear variables then LASSO regression selects one of them randomly which is not good for the interpretation of our model.

Key Differences between Ridge and Lasso Regression

- Ridge regression helps us to reduce only the overfitting in the model while keeping all the features present in the model. It reduces the complexity of the model by shrinking the coefficients whereas Lasso regression helps in reducing the problem of overfitting in the model as well as automatic feature selection.
- Lasso Regression tends to make coefficients to absolute zero whereas Ridge regression never sets the value of coefficient to absolute zero.

What does Regularization achieve?

- In simple linear regression, the standard least-squares model tends to have some variance in it, i.e. this model won't generalize well for a future data set that is different from its training data.
- Regularization tries to reduce the variance of the model, without a substantial increase in the bias.
- How λ relates to the principle of "Curse of Dimensionality"?

As the value of λ rises, it significantly reduces the value of coefficient estimates and thus reduces the variance. Till a point, this increase in λ is beneficial for our model as it is only reducing the variance (hence avoiding overfitting), without losing any important properties in the data. But after a certain value of λ, the model starts losing some important properties, giving rise to bias in the model and thus underfitting. Therefore, we have to select the value of λ carefully. To select the good value of λ, cross-validation comes in handy.

Important points about λ:

- λ is the tuning parameter used in regularization that decides how much we want to penalize the flexibility of our model i.e, controls the impact on bias and variance.
- When λ = 0, the penalty term has no effect, the equation becomes the cost function of the linear regression model. Hence, for the minimum value of λ i.e, λ=0, the model will resemble the linear regression model. So, the estimates produced by ridge regression will be equal to least squares.
- However, as λ→∞ (tends to infinity), the impact of the shrinkage penalty increases, and the ridge regression coefficient estimates will approach zero.

Ridge and LAsso Regression implementation

In [15]:
```
#Importing libraries
from sklearn.datasets import load_boston #loading inbuild datset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV  #For tuning the model
```

In [4]:
```
df=load_boston()
```

In [6]:
```
df
```

Out[6]:
```
{'data': array([[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]]),
 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
        18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
        15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
        13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
        21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
        35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
        19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
        20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
        23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
        33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
        21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
        20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
        23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
        15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
        17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
        25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
        23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
        32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
        34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
        20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
        26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
        31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
        22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
        42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
        36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
        32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
        20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
        20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
        22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
        21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
        19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
        32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
        18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
        16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
        13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
         7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
        12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
        27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
         8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
         9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
        10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
        15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
        19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
        29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
        20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
        23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]),
 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
        'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'),
 'DESCR': ".. _boston_dataset:\n\nBoston house prices dataset\n---------------------------\n\n**Data Set Characteristics:**  \n\n    :Number of Instances: 506 
\n\n    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.\n\n    :Attribute Information (in orde
r):\n        - CRIM     per capita crime rate by town\n        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.\n        - INDUS    
proportion of non-retail business acres per town\n        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\n        - NOX      
nitric oxides concentration (parts per 10 million)\n        - RM       average number of rooms per dwelling\n        - AGE      proportion of owner-occupied u
nits built prior to 1940\n        - DIS      weighted distances to five Boston employment centres\n        - RAD      index of accessibility to radial highway
s\n        - TAX      full-value property-tax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by town\n        - B        1000(Bk - 0.63)^2 where Bk 
is the proportion of blacks by town\n        - LSTAT    % lower status of the population\n        - MEDV     Median value of owner-occupied homes in $1000's\n
\n    :Missing Attribute Values: None\n\n    :Creator: Harrison, D. and Rubinfeld, D.L.\n\nThis is a copy of UCI ML housing dataset.\nhttps://archive.ics.uci.
edu/ml/machine-learning-databases/housing/\n\n\nThis dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.\n\nThe Bost
on house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic\nprices and the demand for clean air', J. Environ. Economics & Management,\nvol.5, 81-102, 19
78.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics\n...', Wiley, 1980.   N.B. Various transformations are used in the table on\npages 244-261 of the
latter.\n\nThe Boston house-price data has been used in many machine learning papers that address regression\nproblems.   \n     \n.. topic:: References\n\n 
- Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.\n   - Quinlan,R. (1993). Co
mbining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusett
s, Amherst. Morgan Kaufmann.\n",
 'filename': 'C:\\Users\\Lenovvo\\anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\boston_house_prices.csv'}
```

In [7]:
```
dataset = pd.DataFrame(df.data)  #Converting to dataframe
print(dataset.head())
```

```
         0     1     2    3      4      5     6       7    8      9    10  \
0  0.00632  18.0  2.31  0.0  0.538  6.575  65.2  4.0900  1.0  296.0  15.3   
1  0.02731   0.0  7.07  0.0  0.469  6.421  78.9  4.9671  2.0  242.0  17.8   
2  0.02729   0.0  7.07  0.0  0.469  7.185  61.1  4.9671  2.0  242.0  17.8   
3  0.03237   0.0  2.18  0.0  0.458  6.998  45.8  6.0622  3.0  222.0  18.7   
4  0.06905   0.0  2.18  0.0  0.458  7.147  54.2  6.0622  3.0  222.0  18.7   

       11    12  
0  396.90  4.98  
1  396.90  9.14  
2  392.83  4.03  
3  394.63  2.94  
4  396.90  5.33  
```

In [8]:
```
dataset.columns=df.feature_names
```

In [9]:
```
dataset.head()
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [10]:
```
df.target.shape
```

Out[10]:
```
(506,)
```

In [13]:
```
X=dataset.iloc[:,:-1] ## independent features
y=dataset.iloc[:,-1] ## dependent features
```

Ridge Regression

In [16]:
```
ridge=Ridge()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
ridge_regressor.fit(X,y)
```

Out[16]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10,
                                   20, 30, 35, 40, 45, 50, 55, 100]},
             scoring='neg_mean_squared_error')
```

In [17]:
```
print(ridge_regressor.best_params_)
print(ridge_regressor.best_score_)
```

```
{'alpha': 100}
-22.96774759693227
```

Lasso Regression

In [26]:
```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
lasso=Lasso()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
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_)
```

```
C:\Users\Lenovvo\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:529: ConvergenceWarning: Objective did not converge. You might want t
o increase the number of iterations. Duality gap: 2747.5880421571305, tolerance: 2.2051708305693074
  model = cd_fast.enet_coordinate_descent(
C:\Users\Lenovvo\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:529: ConvergenceWarning: Objective did not converge. You might want t
o increase the number of iterations. Duality gap: 2859.6149240116897, tolerance: 2.0776240319999997
  model = cd_fast.enet_coordinate_descent(
C:\Users\Lenovvo\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:529: ConvergenceWarning: Objective did not converge. You might want t
o increase the number of iterations. Duality gap: 3453.528021408897, tolerance: 2.1125855173827164
  model = cd_fast.enet_coordinate_descent(
C:\Users\Lenovvo\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:529: ConvergenceWarning: Objective did not converge. You might want t
o increase the number of iterations. Duality gap: 2552.6947579629013, tolerance: 1.8864144117530866
  model = cd_fast.enet_coordinate_descent(
C:\Users\Lenovvo\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:529: ConvergenceWarning: Objective did not converge. You might want t
o increase the number of iterations. Duality gap: 2682.2632276018417, tolerance: 1.901930263111111
  model = cd_fast.enet_coordinate_descent(
{'alpha': 1}
-22.841784268980348
```

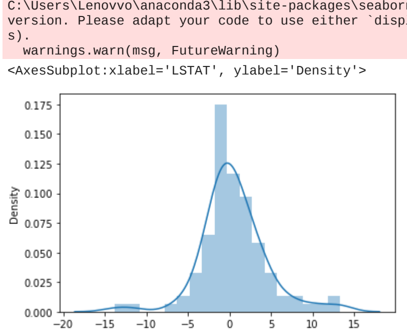
In [28]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
prediction_lasso=lasso_regressor.predict(X_test)
prediction_ridge=ridge_regressor.predict(X_test)
```

In [29]:
```
import seaborn as sns

sns.distplot(y_test-prediction_lasso)
```

```
C:\Users\Lenovvo\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[29]:
```
<AxesSubplot:xlabel='LSTAT', ylabel='Density'>
```



In [30]:
```
import seaborn as sns

sns.distplot(y_test-prediction_ridge)
```

```
C:\Users\Lenovvo\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[30]:
```
<AxesSubplot:xlabel='LSTAT', ylabel='Density'>
```


In [ ]: