Ridge regression:

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values.

Ridge Regression is a type of regularized linear regression that helps to prevent overfitting. It is similar to ordinary least squares regression, but with an additional penalty term added to the cost function. The penalty term is a regularization parameter, denoted by 'alpha', that controls the degree of shrinkage of the regression coefficients towards zero. The higher the value of alpha, the greater the degree of shrinkage and the more robust the model becomes. Ridge Regression, also known as Tikhonov regularization or L2 regularization, is a linear regression technique that introduces a regularization term to the cost function. This regularization term penalizes large coefficients, preventing overfitting and improving the model's generalization ability. The Ridge Regression cost function is defined as:

$$J(\theta) = \text{Least Squares Cost Function} + \lambda \sum_{j=1}^{n} \theta_j^2$$

Here, θ represents the regression coefficients, n is the number of features, and λ is the regularization parameter (also known as the Ridge parameter).

Another Method:

The L2 regularization term is added to the standard linear regression or logistic regression cost function, resulting in the following regularized cost function:

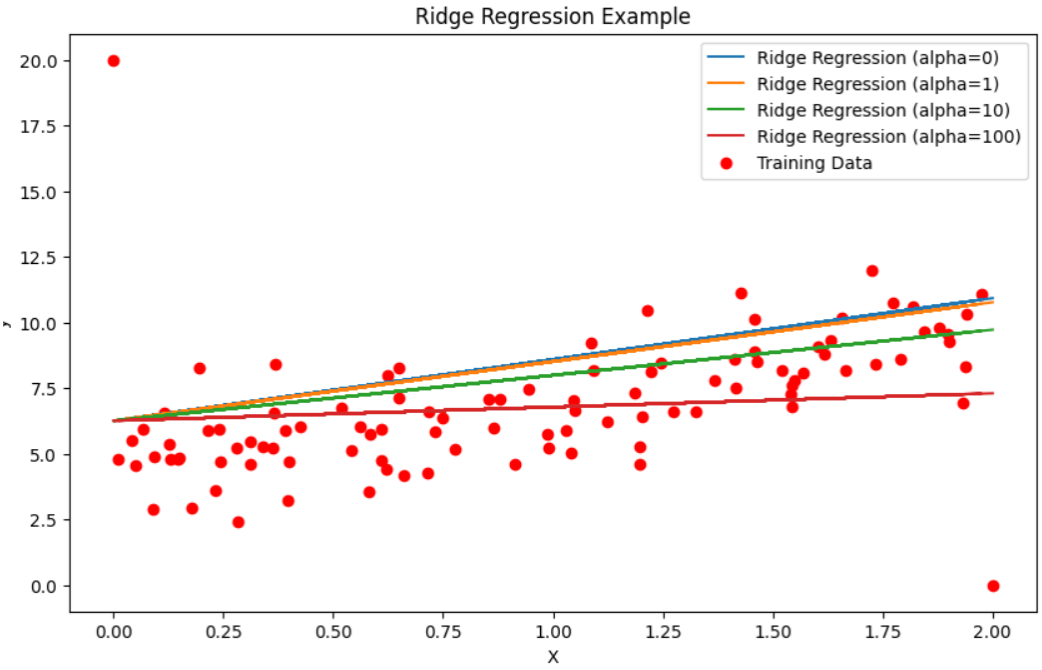
$$J(\theta) = \text{Original Cost Function} + \lambda \sum_{j=1}^{n} \theta_j^2$$

Here:

- $J(\theta)$ is the regularized cost function.
- The "Original Cost Function" represents the standard cost function used for linear regression or logistic regression.
- λ is the regularization parameter (also known as the Ridge parameter).
- θ_j are the model coefficients (weights) associated with each feature.
- The summation term $\sum_{j=1}^{n} \theta_j^2$ computes the sum of the squared coefficients.

The regularization parameter λ controls the strength of the regularization. A higher λ leads to a stronger regularization effect, penalizing larger coefficients more heavily. The regularization term discourages the model from fitting the training data too closely, helping to prevent overfitting.

Let's go through a simple example using Python and the scikit-learn library:



Note: In the above graph (alpha) is used In place of (Lambda)

In this example:

- We generate synthetic data with a linear relationship and add a couple of outliers.
- We split the data into training and test sets.
- We apply Ridge Regression with different alpha values (regularization strengths).
- We plot the Ridge Regression lines for each alpha along with the training data.

The regularization strength (alpha) controls the trade-off between fitting the training data and keeping the model coefficients small. Higher alpha values result in more regularization.

Ridge regression performs '**L2 regularization**', i.e., it adds a factor of the sum of squares of coefficients in the optimization objective. Thus, ridge regression optimizes the following:

Objective = RSS + α * (sum of the square of coefficients)

Here, α (alpha) is the parameter that balances the amount of emphasis given to minimizing RSS vs minimizing the sum of squares of coefficients. α can take various values:

1. **α = 0:**
    - The objective becomes the same as simple linear regression.
    - We'll get the same coefficients as simple linear regression.
2. **α = ∞:**
    - The coefficients will be zero. Why? Because of infinite weightage on the square of coefficients, anything less than zero will make the objective infinite.

3. **$0 < \alpha < \infty$:**
    - The magnitude of α will decide the weightage given to different parts of the objective.
    - The coefficients will be somewhere between 0 and ones for simple linear regression.

Grid Search:

Grid Search is a technique used to search for the optimal hyperparameters of a machine learning model by evaluating the model's performance for every combination of hyperparameter values specified in a grid. It's an exhaustive search that helps identify the best set of hyperparameters that result in the highest model performance.

Here's an example of how to perform Grid Search using Python and scikit-learn. In this example, we'll use the popular Iris dataset and perform Grid Search to find the optimal parameters for a Support Vector Machine (SVM) classifier.

In this example:

- We load the Iris dataset and split it into training and testing sets.
- We define a parameter grid containing different values for the regularization parameter *C*, kernel type, and kernel coefficient.
- We create a Support Vector Machine (SVM) classifier.
- We use GridSearchCV to perform an exhaustive search over the specified parameter grid, using 5-fold cross-validation and accuracy as the scoring metric.
- We fit the GridSearchCV object to the training data.
- We print the best parameters found by Grid Search.
- We get the best SVM model and use it to make predictions on the test set.
- Finally, we evaluate the accuracy of the best model on the test set.

Hyperparameter tuning using grid search is a technique to systematically search through a predefined set of hyperparameter values for a machine learning model to find the combination that produces the best performance. Grid search is an exhaustive search approach where all possible combinations of hyperparameter values are evaluated using cross-validation.

- We load the Iris dataset and split it into training and testing sets.
- We define a parameter grid containing different values for hyperparameters such as the number of trees in the forest (**n_estimators**), maximum depth of the trees (**max_depth**), minimum number of samples required to split an internal node (**min_samples_split**), and minimum number of samples required to be at a leaf node (**min_samples_leaf**).
- We create a Random Forest classifier.
- We use GridSearchCV to perform an exhaustive search over the specified parameter grid, using 5-fold cross-validation and accuracy as the scoring metric.

- We fit the GridSearchCV object to the training data.
- We print the best parameters found by Grid Search.
- We get the best Random Forest model and use it to make predictions on the test set.
- Finally, we evaluate the accuracy of the best model on the test set.

Randomized Search is an alternative approach to hyperparameter tuning that, unlike grid search, does not exhaustively search through all possible combinations of hyperparameter values. Instead, it randomly samples a specified number of combinations from the hyperparameter space. This method is particularly useful when the hyperparameter search space is large and an exhaustive search becomes computationally expensive. In Randomized Searched technique:

- We load the Iris dataset and split it into training and testing sets.
- We define a parameter distribution containing different values for hyperparameters.
- We create a Random Forest classifier.
- We use RandomizedSearchCV to randomly sample 10 combinations from the specified parameter distribution, using 5-fold cross-validation and accuracy as the scoring metric.
- We fit the RandomizedSearchCV object to the training data.
- We print the best parameters found by Randomized Search.
- We get the best Random Forest model and use it to make predictions on the test set.
- Finally, we evaluate the accuracy of the best model on the test set.

Why Grid Search?

Grid search is a hyperparameter tuning technique in machine learning where a predefined set of hyperparameter values is systematically searched to find the combination that produces the best model performance. The term "grid search" comes from the fact that it involves creating a grid of all possible combinations of hyperparameter values and evaluating the model's performance for each combination.