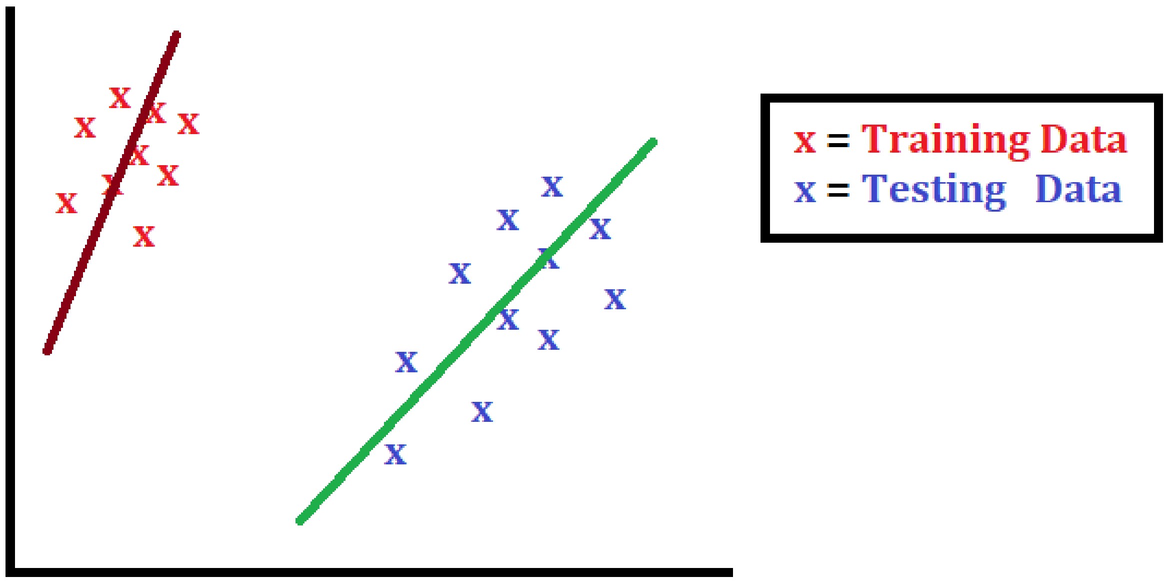
Ridge Regression(L2), in terms of Linear Regression



Linear Regression

[We've the **Best Fit Line** whose **slope is higher** but we want the **Best For Line** whose **slope is lower**. To go from **Best Fit Line** to **Best For Line,** we need to lower our slope. This is Overfitting(to get an in-depth clear details on Overfitting, watch **campusx** till 6:06) issue]

We train our Linear Regression model using a training dataset which will return us the Maroon color Regression Line because it's the **Best Fit Line** for our training dataset. But now if we want to perform prediction using our Testing Dataset, we will have very bad results since the testing dataset is far away from the training dataset i.e. from **Best Fit Line**.

Ultimately what we want to do? Getting good predictions on the Testing dataset i.e. we want the green color **Best Fit Line** since that's where our Testing Dataset is. So for the Training dataset, want the **Best Fit Line**(i.e. Slopes and Intercepts) on the Testing Dataset as much as possible.

How can we do that? Adding a penalty with the Loss Function because ultimately we minimize our Loss Function. So if we want to bring changes, we need to bring it on the Loss Function.

Loss function of Linear Regression : (this loss function is for **Best For Line**)

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

New loss function with **penalty** : (this loss function is for **Best For Line**)

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \boldsymbol{\lambda(m^2)}$$

λ = Hyper parameter which is as ‘alpha’ in sklearn. Its value is from 0 to infinity.

m = Slope of **Best For Line**.

The reason Ridge Regression is called L2 is because we do square(**m^2**) in penalty.

This complete explanation is from Ridge Regression Part 1.

From google => sklearn’s **`ridge_regression`** is a type of linear regression model. It's specifically a regularized linear regression model that uses L2 regularization to **prevent overfitting and handle multicollinearity**. While it's still a linear model because it aims to find a linear relationship between the input features and the output, it incorporates a penalty term (L2 regularization) to minimize the magnitude of the coefficients.

Lasso Regression(L1), in terms of Linear Regression

New loss function with **penalty** : (this loss function is for **Best For Line**)

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \boldsymbol{\lambda(|m|)}$$

Instead of taking square, here we take absolute(| |).

When to use L1 and L2?

The key difference between L1 and L2 regularization techniques is that lasso regression shrinks the less important feature's coefficient to zero, removing some features altogether. In other words, L1 regularization works well for **feature selection in case we have a huge number of features**. Also L1 can cause underfitting when its **λ** i.e. alpha value is very big.

L2 regularization is effective for handling multicollinearity and prioritizing model accuracy and stability.

Campusx :

1) You use L1 when you know that your data has both important and unimportant features/columns.
2) You use L2 when you know that your data has only important features/columns and when there are columns whose values depend on each other i.e. multicollinearity.

Elastic, in terms of Linear Regression

You use Elastic, when you don’t know if your data has both important and unimportant features/columns or not. So it's a combination of both Ridge(L2) and Lasso(L1).

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \boldsymbol{\lambda_1 (m^2) + \lambda_2 (|m|)}$$

Since we’ve two **λ**s (**λ_1**, **λ_2)** here, so

$$\boldsymbol{\lambda = \lambda_1 + \lambda_2}$$

Sklearn’s **Elasticnet** has 2 parameters to set, alpha(λ) and l1_ratio :

alpha, $\lambda = a + b$ $[a = \lambda_1, b = \lambda_2]$

$$\text{l1\_ratio} = \frac{a}{\lambda} = \frac{a}{a + b}$$

From alpha and l1_ratio, the model automatically finds out a and b,

a = l1_ratio * λ [from l1_ratio equation]
$b = \lambda - a$ [from $\lambda = a + b$]