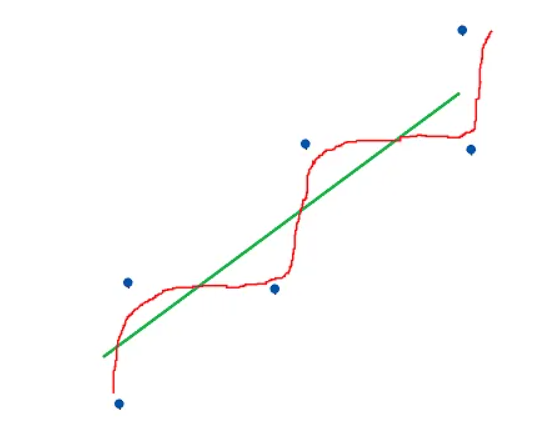


**Techniques to avoid Overfitting in Machine Learning Regression**

**Introduction**

Building a model that is too complex for the amount of information that is provided to the model is called overfitting. This occurs when a model is fit too closely to the particularities of the training set and obtain a model that works well on the training set but is not able to generalize to new data. The overfitted model works very well on the training dataset, but it doesn’t work accurately on the test or new dataset.



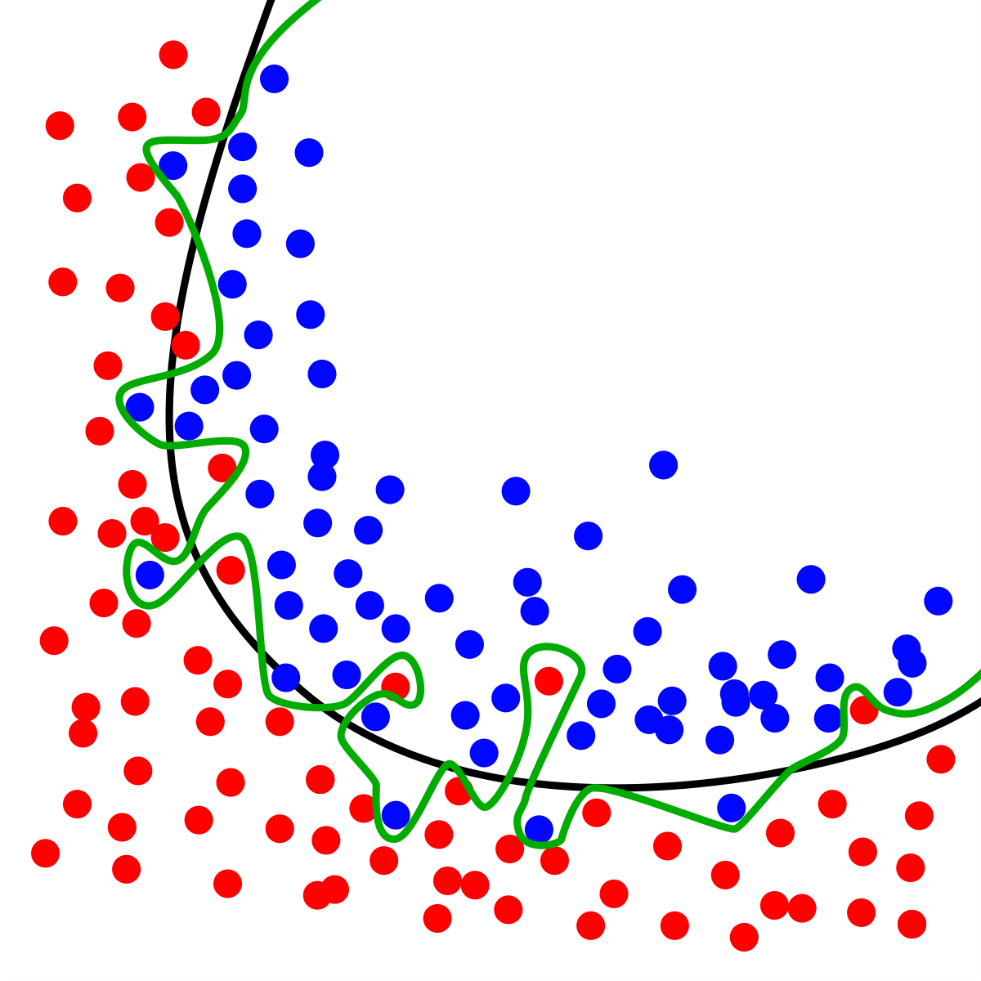
**Why Overfitting Pops up in Regression?**

Building a simple model looks easy enough but this brings the distress of model underfitting. The simple model fails due to fewer features. So, Data Scientists try to add more features or explaining variables to overcome from the affliction of model underfitting. After that model becomes more engrossing and complex. The augmentation of more features and variables makes the result of the model quite good but not as exemplary as much required for the production. Due to the enlargement of the model by adjoining more variable makes it more complex and fiddly.

There are several reasons for overfitting:

* Too complex model
* Too many features
* Not enough data

The Overfitted model will probably have poor prediction and generalization power. Because it will cling too much to the data and will learn the background noise.



Overfitting with outliers

Suppose

Image for post

The score from the above prediction shows very high accuracy for the Training data-set. There is a big gap between training and test score accuracy. This discrepancy clearly shows the case of overfitting as the model’s accuracy is very good on training data-set but accuracy on test data-set is very low.

**Regularization solves the Problem of Overfitting**

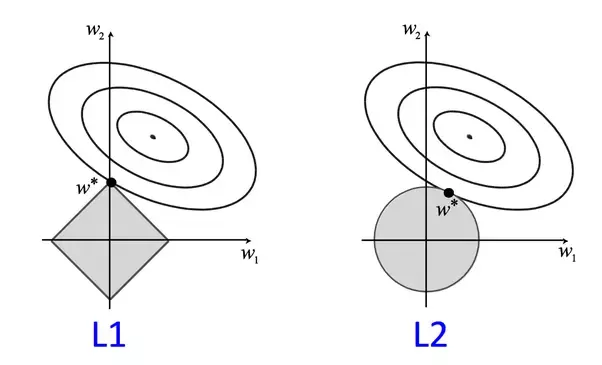
Regularization can be used to solve overfitting problems. Regularization means explicitly restricting models to avoid overfitting. In the context of machine learning, regularization is the process which regularizes or shrinks the coefficients towards zero.

In a simple sentence, regularization discourages learning a more complex or flexible model, to prevent overfitting. It does fitting a function appropriately on the given training set and avoids overfitting.

There are two types of Regularization:

* Lasso Regularization or L1 Regularization
* Ridge Regularization or L2 Regularization

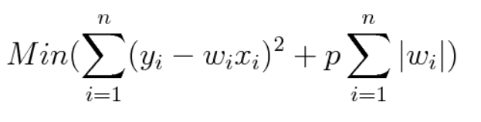
Regularization in Regression would be = Cost function + Penalty Term



Regularization

**Lasso Regularization or L1 Regularization**

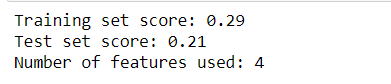
L1 Regularization adds a penalty to the error function. The penalty is the sum of the absolute values of weights.



Here, p is the tuning parameter which decides how much we want to penalize the model.

Let’s apply Lasso:

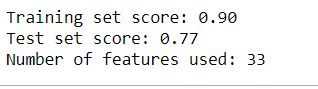
from sklearn.linear\_model import Lassolasso = Lasso().fit(X\_train, y\_train)print("Training set score: {:.2f}".format(lasso.score(X\_train, y\_train)))print("Test set score: {:.2f}".format(lasso.score(X\_test, y\_test)))print("Number of features used: {}".format(np.sum(lasso.coef\_ != 0)))



From the above output, the performance of the algorithm is very bad and is underfitting the algorithm it can be due to very low features used by Lasso.

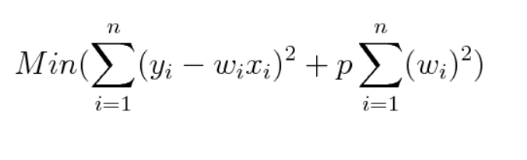
To reduce underfitting, let’s try decreasing alpha.

lasso001 = Lasso(alpha=0.01, max\_iter=100000).fit(X\_train, y\_train)print("Training set score: {:.2f}".format(lasso001.score(X\_train, y\_train)))print("Test set score: {:.2f}".format(lasso001.score(X\_test, y\_test)))print("Number of features used: {}".format(np.sum(lasso001.coef\_ != 0)))



**Ridge Regularization or L2 Regularization**

L2 regularization also adds a penalty to the error function. But the penalty here is the sum of the squared values of weights.



Here, p is the tuning parameter which decides how much we want to penalize the model.

Let’s apply Ridge:

from sklearn.linear\_model import Ridgeridge = Ridge().fit(X\_train, y\_train)print("Training set score: {:.2f}".format(ridge.score(X\_train, y\_train)))print("Test set score: {:.2f}".format(ridge.score(X\_test, y\_test)))

Image for post

The above score is consistent as per expectation and it is avoiding the overfitting.