Contents

[1. Cross- Validation 2](#_Toc102178660)

[2. Linear Regression/ OLS – Mathematical Understanding 4](#_Toc102178661)

[Multiple Linear Regressio**n** 8](#_Toc102178662)

[3. Bias-Variance & Overfitting-Underfitting 9](#_Toc102178663)

[5. R-Squared and Adjusted R Squared 13](#_Toc102178664)

[6. Error Metrics for Regression Models 16](#_Toc102178665)

[7. Linear Regression Model using Python – First ML model 18](#_Toc102178666)

[8. Ridge and Lasso Regression ( Regularization ) 21](#_Toc102178667)

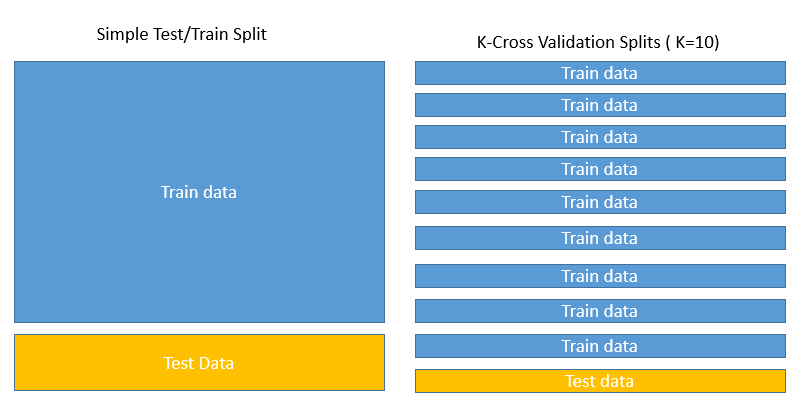
[9. MCQ 26](#_Toc102178668)

# Cross- Validation

Train/Test split is a general step before training any ML model. Let’s discuss K-Cross validation now.

The fundamental drawback with a simple Train/Test split was that entire model and performance measurement was done on a single random split of the data into Train/Test dataset. This random split may change with every iteration and model might be trained differently each time, which is not the desirable situation.

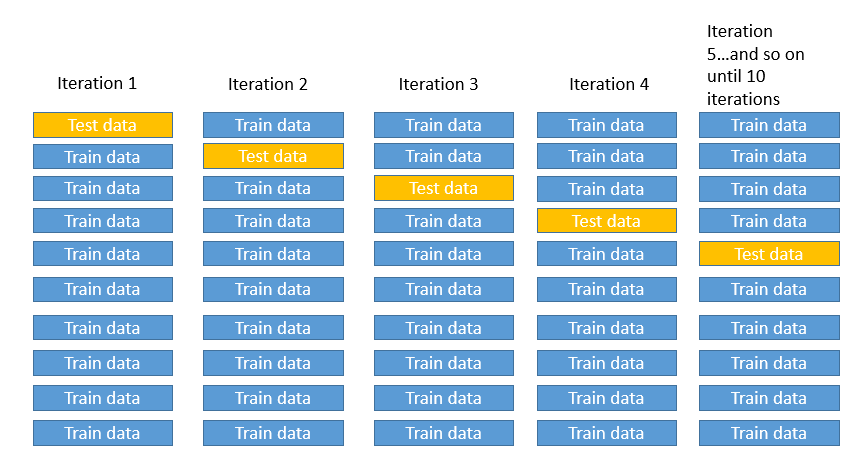
K-Cross Validation resolves this issue by multiple iterations of Train/Test splits and training model multiple times on these splits. In the end, average performance of each iteration is accepted to be the overall performance of the model.



The difference in how the Train/Test splits happen between simple Train/Test split vs K-Cross Validation is shown with the help of above diagram

K in K-Cross validation represents the number of chunks entire data is divided into. For example, if the value of K = 10, then entire data will be divided into 10 chunks, and one of these 10 chunks will act as Test data while the rest 9 will be treated as train data. This will be repeated 10 times until each chunk has been treated as Test data. Again, in this way the model will be trained 10 times with 10 different splits, and the mean performance will be accepted as the final performance of the model.

Below chart shows how in different iterations, test data will change in case of K=10 ( only 5 of 10 the 10 iterations are shown )–



**What is the ideal K value ?**

Just like every other concept in ML, this too is too much case and data dependent and there is no standard rule for best value of K.

As the K value is increases, the chunks get smaller and smaller, iterations increase and we move towards a model which has low bias and low variance. But it comes at the cost of higher computational power and more time, which can be a huge factor in real life projects where data is in TBs and time is really valuable.

Generally, it has been observed that K=5 or K=10 tend to give a good tradeoff between performance, but this is no standard though.

Before we move on, it is also worthwhile to touchbase upon variations in K-Cross validation–

* LOOCV ( Leave One Out Cross Validation ) : This is the extreme case of K-Cross validation where K = number of data points (n). Here, each single data point is treated as test data one by one via n iterations
* Train/Test Split : This is an extreme case in opposite sense of LOOCV. Here, K=2 which is the minimum value K can take. In this case we’ll simply have just 2 chunks, train and test data .
* Stratified Cross Validation : Of all the variations, this is the most important and most used. It makes sure that the chunks are not entirely random, but they are homogeneous as well in some sense. It will make sure that each chunk of data will contain equal amount of data points pertaining to specified feature(s). Thus, with stratified variation, model will be trained and tested on similar data K times, which would remove bias further.
* Nested : Here, K-Cross validation is performed within each chunk of the K-Folds. Although very useful in removing bias, but this variation is complex. and might take a computational toll on larger data.

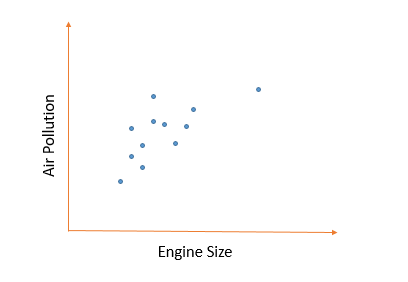
# Linear Regression/ OLS – Mathematical Understanding

Linear Regression or Ordinary Least Squares is method to “fit” a line to a given data. Let’s understand in detail what “fitting” means –

Suppose we have some data points for 2 variables –

* Air Pollution of a Vehicle ( in ppm )
* Engine Capacity of the Vehicle ( in cc)

We can represent this data on a scatter plot as below-



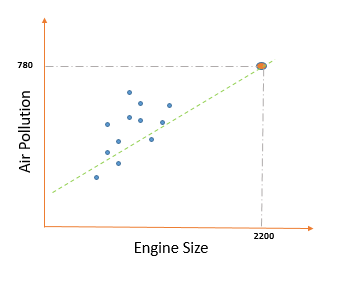
In above chart, we can observe how increase in engine size can cause more pollution level. We also have precise data points to suggest the same. But what if , we want to see the air pollution level for a particular engine size that is not on the chart ?

Alternatively, what if we want to see what engine size can cause XYZ amount of air pollution and we don’t have that data ?

For this purpose, we try to represent the data via a “best fit line”, and the process or mechanism behind achieving that “best fit line” is called regression.

Simple Linear Regression or OLS is simplest regression algorithm which is based on Least Squares method, which we’ll discuss now.

In the above example, let’s assume that the best fit line can be drawn as below –



Now that we have the best fit line, we can predict that an engine of 2200cc will produce 780 units of air pollution as per our regression line. Without this line, we did not have this data point, and this is how a regression line can be used for prediction.

Now let’s move on to a more fundamental question – how this best fit line is drawn ?

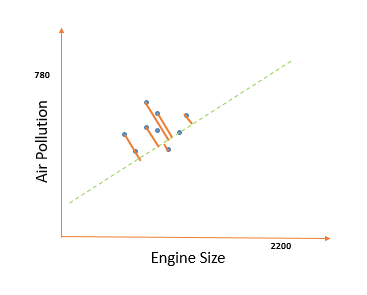
There’s lot of mathematics involved a the backed on any Machine algorithm and same is the case here as well.

Let us assume that we have 2 variables, one independent variable “x” and one Target variable “y”.

As per our above example , Engine Size ~ x and Air Pollution ~ y.

Now, in Simple Linear Regression or Ordinary Least Squared Regression, best fit line would have least squared residuals sum of all the possible lines. Residuals – The distance of a “Regression or fitted line” from a data point is called Residual. In other words, In OLS/SLR , if we square all of these residuals and sum those together, the resulting sum would be minimum possible.

Let’s understand what Least Square means via diagram –



The orange lines to the fitted lines from each data point represent Residuals in the above diagram. All of these residuals are squared and summed up, and the best fitted line will have lowest sum of squared residuals. This function , “ **Sum of residuals squared**” is also the “**Cost function** “ for SLR/OLS. Cost function is a very important concept, and is used in multiple occasions in ML and Deep learning. For now, we need to understand that this Cost Function is **Minimized** in order to achieve the best fit line in SLR.

Now let’s try to draw this best fit line from scratch again .

Let’s assume that a best fit line exists such that –

Here, y = Target variable, x = independent variable, m = slope of best fit line and c = y-intercept of best fit line

Thus, in order to find the best fit line, we just need to estimate appropriate values of m and c and our job will be done.

We have already learnt, that the **best fit line must have Minimum sum of squared residuals.**

This is also represented by the **Cost Function** mathematically as –

Here,

N = Number of data points

= Predicted y value by best fit line

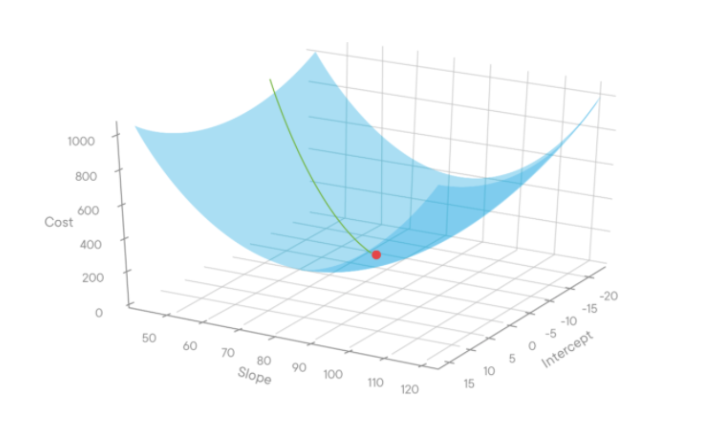
y = actual y value as per data point available

As we can see, this cost function is nothing but residuals squared divided by twice the number of data points.

One may ask why the residuals are squared ? Answer – So as to nullify the –ve and +ve differences while taking distance from best fit line. Or in other words, the distance or error must always be added and should never cancel each other. Over the course of time, mathematicians have figured out that squaring the distances is much easier for advanced calculation rather than taking absolute values or mods.

Back to our journey of plotting the best fit line, so fat we know that Cost function must be minimized, and whichever combination of m and c would do that, that will help us find our best fit line. Although no one has ever done this calculation by hand, and it is done with the help of computing power only, but still it is crucial to understand the concept and understand how computer performs this operation.

The Cost Function can be plotted against m and c in a 3-D plane. This diagram is known as Gradient Descent Diagram –



The computer will make use of **Convergence Theorem** in order to arrive at the minima ( the point where Cost function is minimum ).

The Convergence Theorem is given by formula –

Here,

= New slope value to be tested ( of best fit line)

m = slope value of best fit line

M = slope of a point on Gradient descent curve. At minima, M would be zero.

= Learning Rate

Learning Rate – A multiplier to adjust the amount by which slope (M) would get increased/decreased in order to find the minima of Gradient Descent. If the alpha is too high, the minima might get skipped. If the alpha is too low, the computation will require more power and time. Hence, alpha value can be a trade off factor between accuracy and bandwidth.

**High Level Mechanism** :

1. Start with a point on Gradient Descent Curve for a particular c
2. If the slope (M) is +ve, decrease some small amount from M such that M tends to zero
3. If M is –ve, add some small number to M such that M tends to zero .
4. When M=0, the corresponding m is the minima of the cost function, and is the slope of the best fit line.

Please note that for the sake of simplicity, we have taken an example of a single input variable so that everything can be explained via diagrams. But in real life problems, one might never work with 1 input variable. There will be tens of variables if not hundreds in most scenarios. Since that sort of setting is impossible to visualize, we have discussed a simple problem only.

But the mechanism and logic remains same. Even in case of N-input variables, the best fit regression plane would estimate for N different slopes corresponding to each variable. We will see the practical implementation of the same in today’s session only.

## **Multiple Linear Regression**

So far we have discussed the example of Air Pollution vs Engine Size for understanding Simple Linear Regression. But considering a more real scenario, Engine size might not be the only feature impacting Air Pollution. Other factors such as Age of the vehicle, Type of Fuel used, Engine Manufacturer, KMs driven etc. might come into play.

Thus, instead of just 1 independent feature x, we are going to have multiple features every time, let’s say x1, x2, x3, x4 and so on….

But how does our regression line or best fit line equation changes now ?

Each of the feature will have a slope and intercept value associated with it. The final equation of the best fit line would look like –

Overfitting is a common problem when more features are added in a Simple Linear Regression model. Specially in cases when the data is not huge, SLR is known to have overfitting issues. We will not get into the mathematical reasoning behind this due to time constraints, but if we have to explain the reasoning in few words – It is because linear regression does not penalizes the addition of more features. We will understand this more in details as we’ll discuss Ridge and Lasso Regression later in today’s session.

# Bias-Variance & Overfitting-Underfitting

So far we have learnt all about Train/Test split and how a model ( be it regression or classification ) is first trained on Train dataset and then tested on Test dataset for performance check.

Bias and Variance are 2 extremely important concepts in ML which we’ll discuss now. Please note that this topic is not just related to Regression algorithms, but with ML as a whole. Bias or Variances exist for all sorts of ML algorithms, not just regression. Bias and Variance have very simple definitions as given below -

**Bias :** Error in Training data is called Bias.

**Variance :** Error in Test data is known as variance.

Explanation –

After a model is trained, it can be used to predict target variable for both Train and Test datasets. The error that the model gives with Train dataset is known as Bias and the error associated with Test dataset is known as Variance. Thus, a good model which can predict target accurately with both datasets should have low Bias an low variance.

Please note that this Variance should not be confused with the statistical Variance of a sample which is square of Standard deviation.

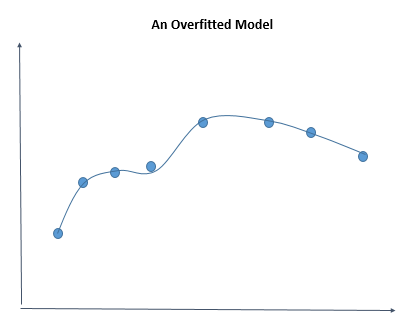
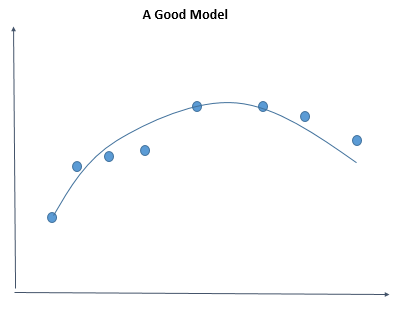
There are specific cases associated with the continuation of this concepts which is called Overfitting and Underfitting in a model. Let’s understand these concepts as well –

**Overfitting :**

When a model performs too well with respect to the train data but doesn’t perform well otherwise ( with test or new data ), the model is called to be Overfitted. As per this definition , an overfitted model will have –

* Low Bias - Means the model performs well with train data
* High Variance - Model doesn’t perform well with test data, error is high

Let’s take an example of some random points and an overfitted model for visual understanding –

As can be seen in the diagrams above, an Overfitted model is passing through almost all the points in the data, it is closer to the data points and hence will have low bias. But the issue is that it is biased so much by the data, when the new data points are given to the model to predict the target, its prediction will be way off.

On the other hand, a good model will preserve the balance, it might not perform as good as an overfitted model for the train dataset, but it is more open to new data and will give more accurate predictions.

Overfitting can be caused by multiple reasons, for example –

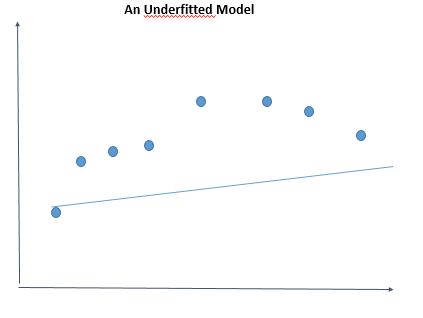
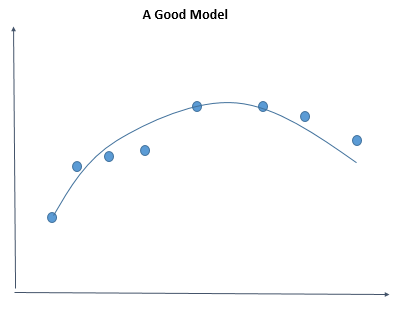
* Data used for modelling is not cleaned and contains garbage values
* Size of training data used is not enough
* Model is too complex
* Excessive use of insignificant features
* Using a single test/train split instead of K-Cross validation ( Will be discussed later today )

**Underfitting :**

When a model does not perform well either with the train data or test data, it is said to be underfitted. The idea is that model is not trained well enough and needs more/better data to train effectively. An underfitted model will have –

* High Bias - Model doesn’t perform well with train data and exhibits high errors
* High Variance - Model doesn’t perform well with test data also and exhibits high errors

Let’s consider visual example of an underfitted model below in comparison to a good model –

As can be seen from above charts, and underfitted model is no good in predicting target for the existing data points itself.

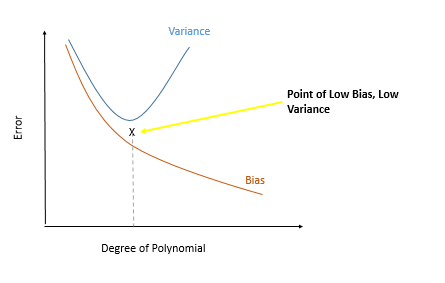
Some of the factors which can cause underfitting are -

* Model trained on insufficient data
* Model trained on insignificant features
* Bad model selection

**Impact of additional Features –**

As we add on more features to our model training dataset, the degree of the polynomial increases. With this increase in degree of polynomial, the Train error will always decrease because any addition of a feature to the model will explain the target variable by chance up to some extent. This is the main reason why models are Overfitted, because data scientists use a lot of insignificant variables for training the model. In other words, we can say that the model goes towards low bias as we add more features.

But the behavior with Variance is not same. The below chart explains how Bias and Variance in a model is impacted as we add more and more features –



To summarize, as per the chart above, as we increase the number of features in training a model ( increasing degree of Polynomial ), the Bias keeps on decreasing but Variance first decreases then increases. Hence, there will be always a point or a threshold value at which Bias and Variance both will be considerably minimum and that is the point at which we’ll have a good fit model which is neither Overfitted not Underfitted.

High Variance is a common problem specially in Decision trees, as we’ll discuss in Classification session later. High variance in Decision Trees is resolved by Random Forest algorithm, tree pruning and hyper-parameter tuning.

# R-Squared and Adjusted R Squared

Now that we have learnt what is Simple Linear Regression and how to do a regression fit in Python, one question stills remains – How do we measure how good is our best fit line ?

Since the target variable is a continuous one, unlike classification, we can not just measure accuracy by counting how many of the data points are accurately predicted. Instead we need a better measure which takes into account the “Residuals”.

R-Squared and R-Squared Adjusted are the most used performance measurement metrics used for all kinds of regression. Let’s study both in detail –

**R-Squared :**

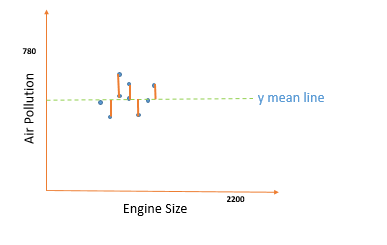
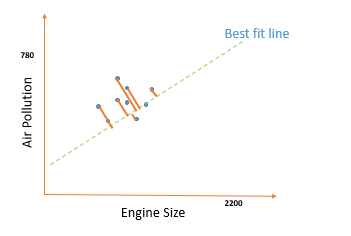
R-Squared is calculated as per following mathematical formula –

Here;

Here;

Thus, in a way R squared measures how worse the fitted line is performing in comparison to the average line. The average line is supposed to have fairly low distances from each of the data points, and hence, if a line is able to perform better or provide lower residuals that average line, that can be considered a good fit. However this is of no significance, because in general, almost all the time a fitted line will perform better than the mean line.

This also means that logically, R squared basically means, by what % , a particular variable is responsible for variation in an anther variable. A 60% R-squared for our air pollution example would mean that the 60% of the variation in air pollution can be explained by engine sizes.



**Characteristics of R squared :**

* Generally R squared value lies between 0 to 1. The closer it is to 1, the better. It means that the line is better fitted and residuals are low.
* Can R Squared be negative ? Answer – Yes. If the fitted line is worse than the average line, then SS ( tot) < SS (res) and equation will become –ve. It basically means that the model is an awfully waste model.
* R squared is used to determine Goodness of fit of a regression line.
* **Drawback -** As more independent features are added, R Squared value will always increase because any feature would explain variation in the target variable up to some extent by chance. R Squared will never decrease upon addition of new variables.

This is a problem, because R squared is increasing with addition of more features even if the features are not at all related to the target variable.

In order to overcome this issue, **R Squared adjusted** is used.

**R-Squared Adjusted :**

The main difference here is that in order to overcome drawback of R Squared increasing upon addition of more features, additional feature are punished here.

This is more evident by the mathematical formula which is given as –

p = Number of predictors

N = Total Sample size

**How it works ?**

Case 1 : Additional feature are related to the target variable

In this case R Square will improve in the Numerator. Although N-P-1 in denominator will also increase but the R Squared increase will be dominant, and thus R Squared Adjusted will overall increase which is the desired result in this case.

Case 2 : Additional feature are not related to the target variable

R Squared will decrease in numerator.

N-p-1 will increase in denominator.

Overall, R Squared adjusted will decrease as per the mathematical formula.

**Differences between and , :**

1. Every time new features are added to a regression model, increases regardless of whether the added features are related to the target variable or not. It never declines. In contrast, only increases when additional independent variable is significant and is related to the dependent variable.
2. is always less than or equal to .

# Error Metrics for Regression Models

In yesterday’s session we learnt that following are the measurement metrics for regression models –

* ( Discussed )
* ( Discussed )
* Mean Absolute Error (MAE)
* Mean Squared Error ( MSE)
* Root Mean Squared Error ( RMSE)
* Root Mean Squared Log Error ( RMSLE)
* Mean Absolute Percentage Error ( MAPE )

and are the most superior metrics to determine the performance of a regression model as they take into consideration the relative performance with respect to Mean prediction line, and also punishes additional features.

While we have already discussed and which are the most used performance metrics for regression, let’s also discuss some other error metrics now which can aid in further determining a better model in cases where 2 models have similar and .

It is also important to have knowledge about these because many inbuilt Python functions would report these Errors by default, and a good data scientist should have an understanding of what these reported errors mean.

By now, we must be familiar with the concept of Predicted Values and Observed Values.

Predicted values are the ones predicted by the best fit line, and observed values are the original data which is predicted. It is obvious that there will always be a difference between Predicted Value and Observed value. This difference is known as Residuals, and all of these errors try to capture residuals in different mathematical ways as below -

1. **Mean Absolute Error (MAE)**

The formula for MAE is given as –

Where ;

oints

Thus, it can be observed that MAE is nothing but Mean of absolute residuals for all the data points by a regression model.

1. **Mean Squared Error (MAE)**

The formula for MSE is given as –

Where ;

oints

Thus, it can be observed that MSE is nothing but Mean of squared residuals for all the data points by a regression model.

The only difference in MSE is that here residuals are squared instead of absolute values. For this reason, MSE is a more sensitive metric, and even with small residuals, it will magnify the the difference more than MAE.

1. **Mean Absolute Percentage Error (MAPE )**

The formula for MAPE is given as –

` Where ;

oints

MAPE is in general a better metric than MSE and MAE because it is in normalized in percentage terms.

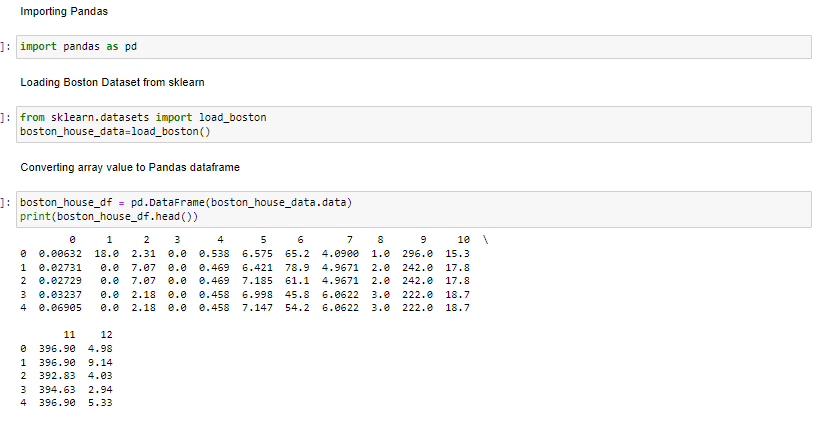
# Linear Regression Model using Python – First ML model

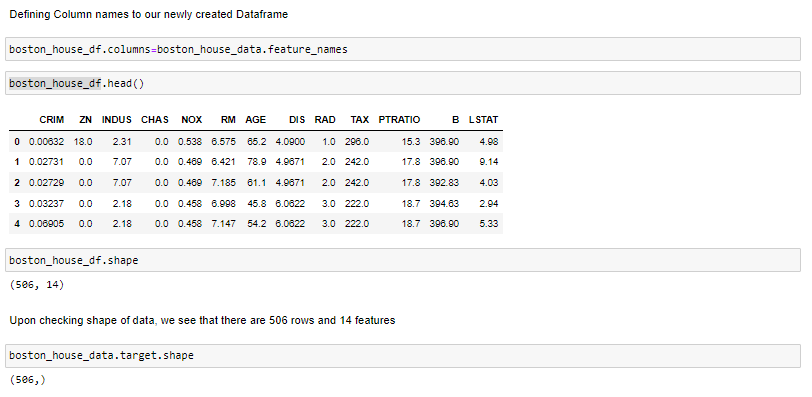
Now let’s proceed towards our first ML model using Python. We are going to train a Linear Regression model using an inbuilt dataset “Boston House Price” data. This dataset is included in sklearn library and is a very common dataset to learn about regression.

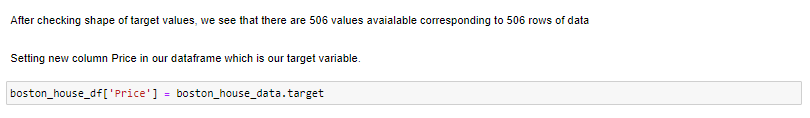
The objective here will be to predict house prices in Boston as per various other features included in the dataset. We will need to first import the data and then convert it into a pandas dataframe for our processing as sklearn doesn’t provide data in Dataframe format.

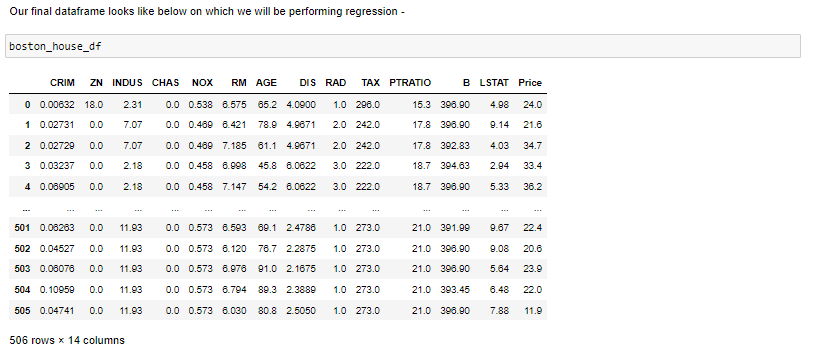
Here we go –

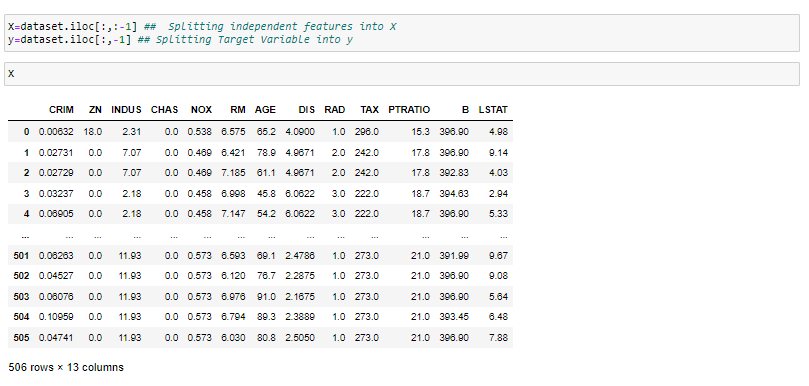
1. **Preparing the data for Regression Model**

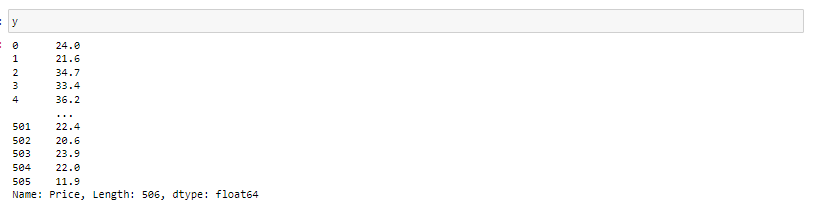




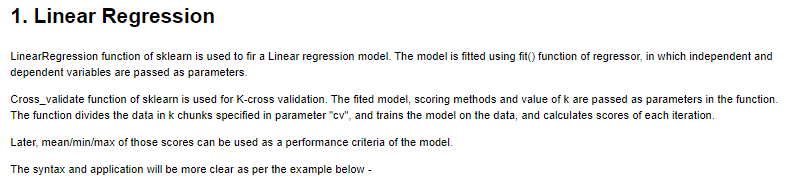


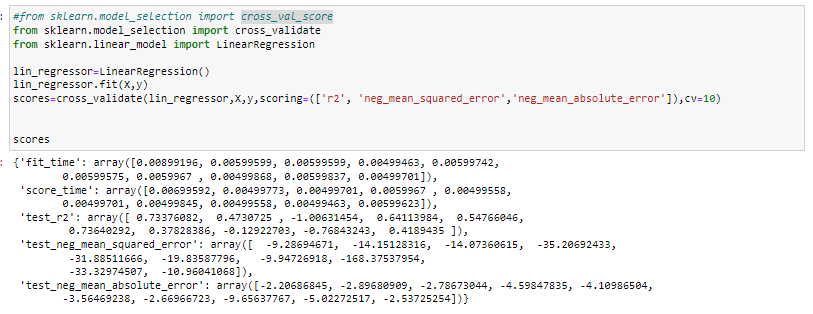


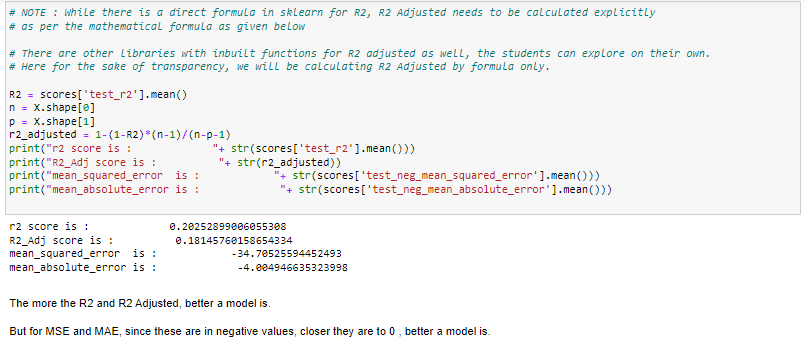




1. **Training the Model and measuring performance using Cross Validation**







# Ridge and Lasso Regression ( Regularization )

Ridge and Lasso Regression are advanced regression techniques which try to overcome the overfitting drawbacks of SLR. These algorithms do so by “Penalizing” the features with high slope values.

These also called Regularization techniques. Regularization, by definition are type of regression techniques which tends to push or make the coefficients ( or slopes) of features towards Zero. This is exactly what Ridge and Lasso regression do – Make higher slopes tend to Zero or exactly zero.

**How do Ridge and Lasso Regression achieve this?**

**Ridge Regression :**

The magic lies in the cost function. The cost function for Ridge Regression is given as –

In case of multiple features, Cost function becomes –

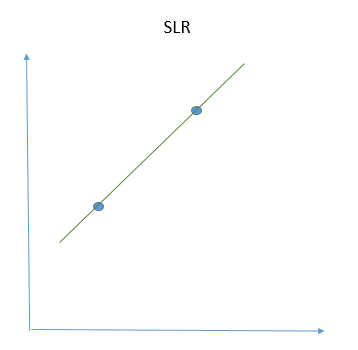
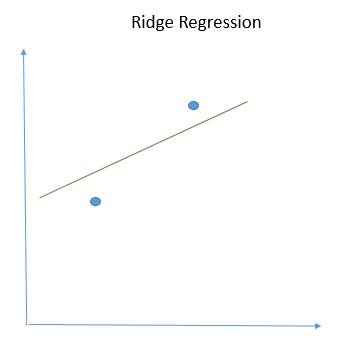
Thus, we can see that while SLR tries to minimize the Sum of squared residuals, in case of Ridge regression there is an additional term .

The purpose of this term is to penalize high slopes ( or features having high slopes).

here is the penalizing factor, it can take any value between 0 to Infinity. Higher the value of lambda, more the penalizing factor and higher slopes will tend more towards zero with increasing .

This is as far the mathematics is concerned. Now let’s see how Ridge Regression is different from SLR practically via charts –

Let’s take an example of 2 data point for simplicity. The SLR and Ridge Regression lines are plotted for this scenario –

From above 2 poit example, we can see the SLR line is overfitted. It has 0 Bias, but it is bound to perfrom worse for new data.

But Ridge Regression on the other hand, has introduced some Bias in the model, in order to trade off for better Variance for new data. While the model doesn’t perform better than SLR for train data, but it is a more generalised model which is bound to perfrom better for a test data.

Hence, this way Overfitting is reduced by Ridge Regression.

**Lasso Regression :**

Lasso Regression is very similar to Ridge Regression with a small difference in cost function.

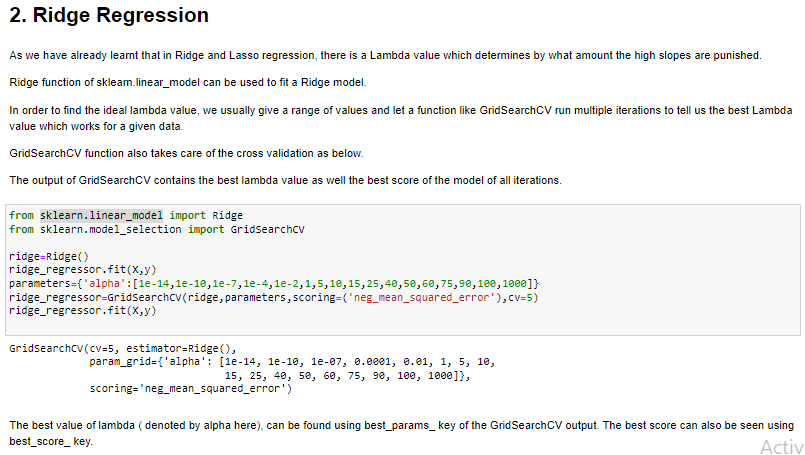
In Lasso Regression, instead of Slope Squared, Absolute Value of slope is used in Penalizing factor.

While this may sound like a normal difference, but it’s effect is huge. Due to this change, Lasso regression is fundamentally different as it can make Slope of Coefficients 0, which Ridge Regression can not do. Ridge Regression can make slope very close to 0 but not 0, meaning it never eliminates features. On the other hand, Lasso regression penalizes higher slopes so aggressively that it can altogether remove features by making the slope 0 at higher values of .

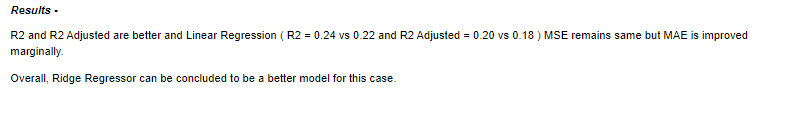
The Cost function for Lasso Regression is given as –

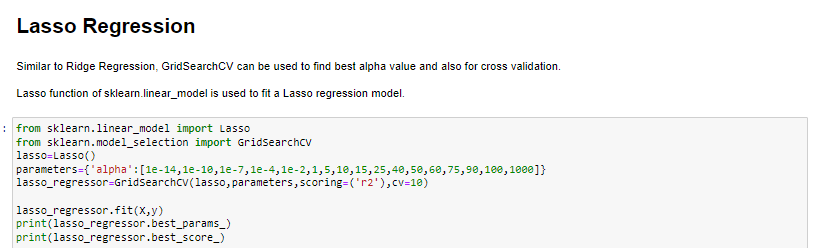
In case of multiple features, Cost function becomes –

Since Lasso Regression elliniates insignificant features, it is also used for Feature Selection for training other models.

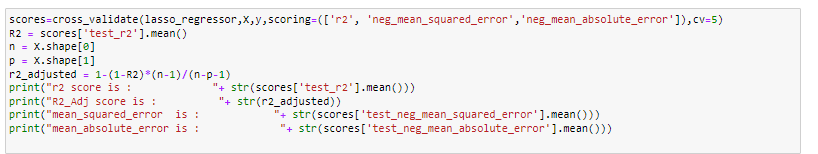


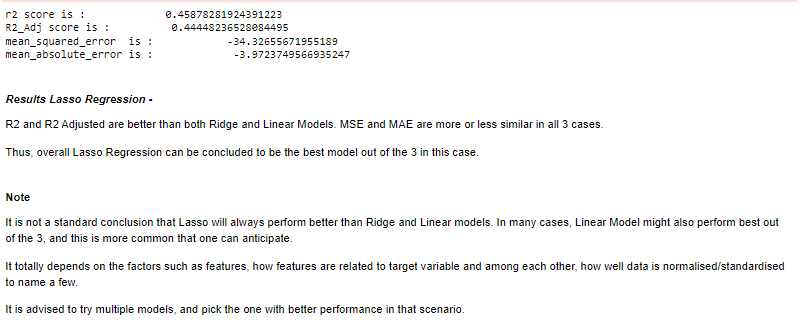


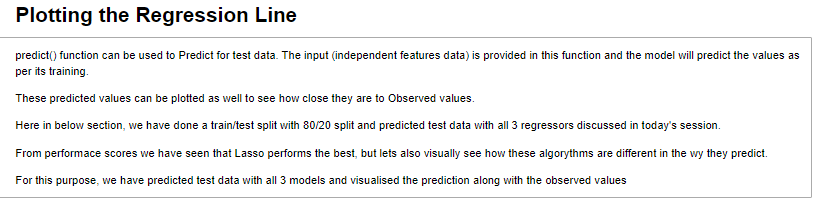


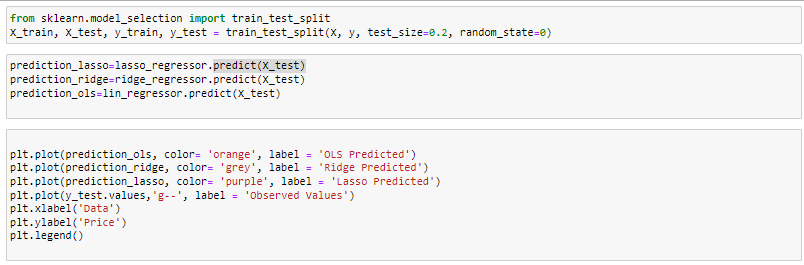


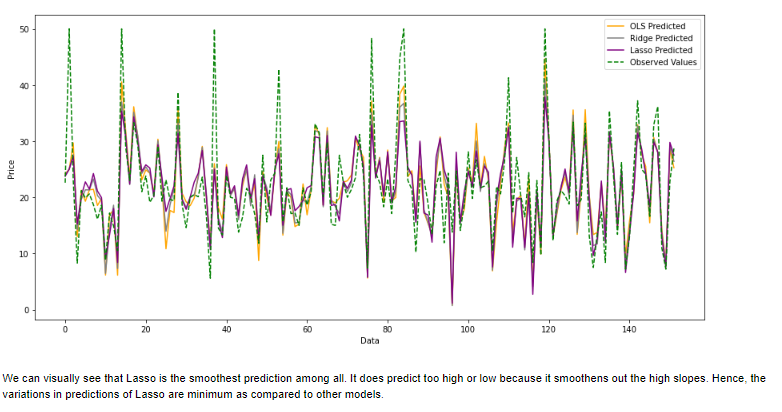




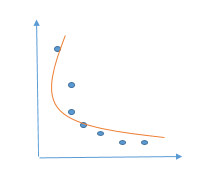
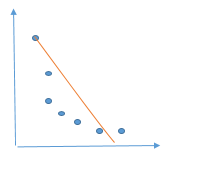
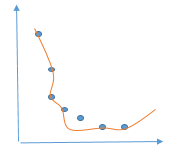
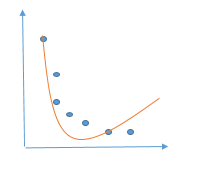
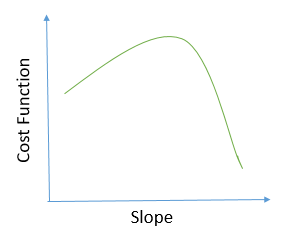
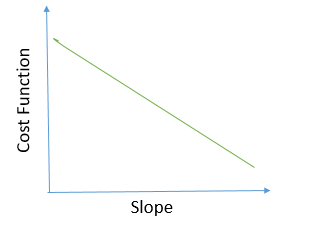
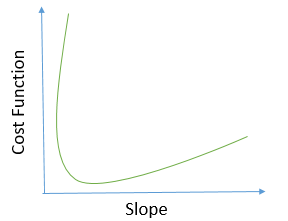
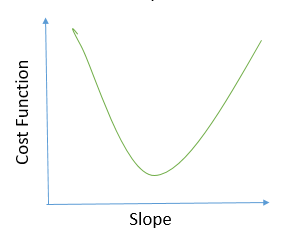








# MCQ

1. What is Overfitting ?
2. Good performance in both Training set and Test set
3. Good Performance in Test set but not in Training set
4. Good performance in Training set but not in Test set
5. Average performance in both Train and Test dataset
6. In Underfitting there is –
7. Low Bias and Low Variance
8. Low Bias and High Variance
9. High Bias and High Variance
10. High Bias and Low Variance
11. The difference between Lasso and Ridge Regression is –
    1. Lasso regression uses absolute values of slopes in penalizing factor instead of squares like in Ridge regression
    2. Lasso regression can make slopes 0 while Ridge regression can only minimize slopes closer to zero at max
    3. Lasso Regression can remove features and can be used for feature selection while Ridge Regression do not eliminate features
    4. All of the above
12. Which is true for R2 and Adjusted R2 –
    1. R2 is always lesser than Adjusted R2
    2. Adjusted R2 penalizes additional features while R2 does not
    3. Closer the values of R2 and Adjusted R2 to Zero, better a model is
    4. R2 =1 represents a very good model
13. The Value of R2 can be Negative.
    1. True
    2. False
14. If for a Regression model, Train error = 0, then–
    1. Test error must also be Zero
    2. Test error will be non Zero but smaller
    3. Test Error will be huge
    4. Cannot comment anything about test error
15. Which of the following can not be used to measure the performance of a regression model –
    1. R2
    2. MAPE
    3. MSE
    4. Accuracy Score
16. Which of the following represents an overfitted model –
    1. 
    2. 
    3. 
    4. 
17. Which of the following represents the relationship between Cost Function and Slope ( Gradient Descent Curve ) –
    1. 
    2. 
    3. 
    4. 
18. The ideal value for Ridge and Lasso regression is –
    1. 5
    2. 10
    3. Half of the number of features
    4. Must be found out using hit and trial