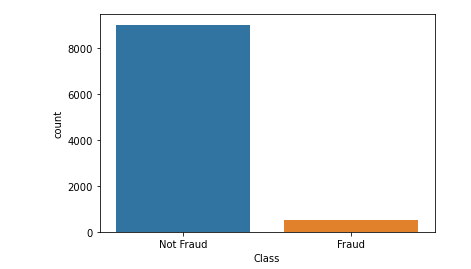
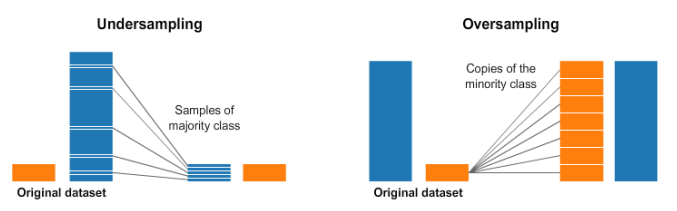
**Imbalance data:**

When observation in one class is higher than in other classes, there exists a class imbalance.



**Resampling Technique:**

A widely adopted technique for dealing with highly unbalanced datasets is called resampling. It consists of **removing samples from the majority class** (under-sampling) and/or **adding more examples from the minority class** (over-sampling).



**Random Oversampling:**

The simplest implementation of over-sampling is to duplicate random records from the minority class, which can cause overfishing.

Oversampling can be defined as adding more copies to the minority class. Oversampling can be a good choice when you don’t have a ton of data to work with.

A con to consider when undersampling is that it can cause overfitting and poor generalization to your test set.

**Random Undersampling:**

In under-sampling, the simplest technique involves removing random records from the majority class, which can cause a loss of information.

Undersampling can be defined as removing some observations of the majority class. This is done until the majority and minority class is balanced out.

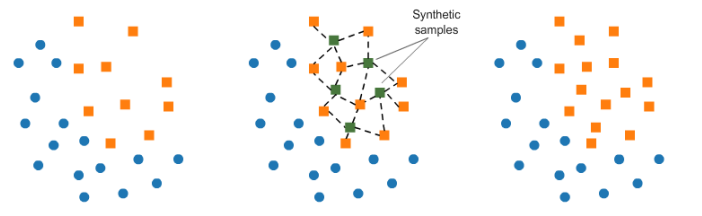
Undersampling can be a good choice when you have a ton of data -think millions of rows. But a drawback to undersampling is that we are removing information that may be valuable.

**Synthetic Minority Oversampling Technique (SMOTE):**

This technique generates synthetic data for the minority class.

SMOTE (Synthetic Minority Oversampling Technique) works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point.

The synthetic points are added between the chosen point and its neighbors.



**SMOTE algorithm works in 4 simple steps:**

1. Choose a minority class as the input vector

2. Find its k nearest neighbors (k\_neighbors is specified as an argument in the SMOTE() function)

3. Choose one of these neighbors and place a synthetic point anywhere on the line joining the point under consideration and its chosen neighbor

4. Repeat the steps until the data is balanced.

**Cross Validation:**

Cross-Validation is a data resampling technique in which we keep some percentage of the data for testing and the remaining data is used for training. Cross-validation is used to protect a model from overfitting.

**What is the need of Cross-validation?**

Used to check the performance of the model by checking how well it performs on the test or unseen data.It allows us to compare different machine learning models and get a sense of how well they will perform in practice.

Steps:

1. separate data into train and test

2. Train the model on the training set.

3. Test the model on the testing set

**Types Cross-Validation:**

1] Hold-out Validation

2] Leave-One-Out CrossValidation (LOOCV)

3] K-Fold Cross-Validation

4] Stratified K-Fold CrossValidation

5] Cross-validation for Time series 1

**1] Hold-out Validation**

Simplest kind of CrossValidation. Dataset is separated into a training set and testing set. Then, the training set is again separated into a training set and a validation set.

**Pros**: Needs only one iteration hence the computation costs are much lower.

**Cons:** Provides a **high variance** estimate since changing which observations happen to be in the testing set can significantly change testing accuracy.

**2] Leave-One-Out CrossValidation (LOOCV)**

Only one data point is kept as a testing set and the model is trained on the rest of the data. This process is iterated for each data point. For example: If we have 1000 data it will repeat 1000 times, at each iteration one data point for testing and the remaining for training.

**Pros:** Covers all the data points and learns everything, hence the bias will be very low.

**Cons:** Since we repeat the process n time where n is the total number of data points, it results in high execution time.

**3] K-Fold Cross-Validation:**

Simply splitting data into train and test doesn’t help due to the variance it provides. Creating a bunch of train/test splits, calculating the testing accuracy for each, and averaging the results together? This is k-Folds Cross-Validation.

1. Split the dataset into k groups randomly. Take one group as a testing set and the rest as a training set.

2. Fit the model on the training set and evaluate it on the test set.

3. Set aside this evaluation score.

4. Take another group as a test set and rest as a training set.

5. Repeat the train test model fit setup until each group has become a test set

**Pros:** Validates your model’s performance on multiple folds of your data.

**Cons:** Doesn't work well with sequential data like time series

**4] Stratified K-Fold CrossValidation**

Due to random sampling, there could be an improper division of data between train and test sets. For example, in the case of binary classification problems, there could be more negative samples in one test set, and in another test set, it could be very less. To rectify this problem, stratification is required which means rearranging in such a way that each fold is a good representative of the whole

**In the case of categorical variables**: In each set, there are equal or close to results from each category.

**In the case of continuous variables:** The means of all outcomes are comparable

**Multicollinearity:**

Multicollinearity occurs when two or more independent variables are highly correlated with one another in a regression model.

This means that an independent variable can be predicted from another independent variable in a regression model.

**The Problem with having Multicollinearity:**

Multicollinearity can be a problem in a regression model because we would not be able to distinguish between the individual effects of the independent variables on the dependent variable.

For example, let’s assume that in the following linear equation:

Y = W0+W1\*X1+W2\*X2

Coefficient W1 is the increase in Y for a unit increase in X1 while keeping X2 constant. But since X1 and X2 are highly correlated, changes in X1 would also cause changes in X2 and we would not be able to see their individual effect on Y.

“ This makes the effects of X1 on Y difficult to distinguish from the effects of X2 on Y. ”

Multicollinearity may not affect the accuracy of the model as much. **But we might lose reliability in determining the effects of individual features in your model – and that can be a problem when it comes to**

**Variance inflation Factor(VIF):**

Multicollinearity can be detected via various methods. here we will focus on the most common one – VIF (Variable Inflation Factors)

VIF score of an independent variable represents how well the variable is explained by other independent variables.

VIF tells us how much variance of an independent variable is influenced by other independent variables.

Variance inflation factors allow a quick measure of how much a variable is contributing to the [standard error](https://www.investopedia.com/terms/s/standard-error.asp) in the regression

VIF determines the strength of the correlation between the independent variables. It is predicted by taking a variable and fitting a regression model against every other variable.

The VIF is variance inflation factor the ratio of the variance of βˆj when fitting the full model divided by the variance of βˆj if fit on its own.

Ex- suppose you have four features x1,x2,x3,x4 and Y as target variable.Now to find VIF for the x1 we fit the model on x2,x3,x4 and predict x1 and calculate R-squared score then using r-square we calculate VIF value.we repeat this process for all the features.

R^2 value is determined to find out how well an independent variable is described by the other independent variables. A high value of R^2 means that the variable is highly correlated with the other variables.

VIF = 1 / 1 - R-Square

The closer the R^2 value to 1, the higher the value of VIF and the higher the multicollinearity with the particular independent variable.

Now we have VIF values for all the features we remove the features which have VIF>5.

VIF starts at 1 and has no upper limit

VIF = 1, no correlation between the independent variable and the other variables

VIF exceeding 5 or 10 indicates high multicollinearity between this independent variable and the others

Dropping variables should be an iterative process starting with the variable having the largest VIF value because its trend is highly captured by other variables. If you do this, you will notice that VIF values for other variables would have reduced too,

**Feature Selection:**

**The top reasons to use feature selection are:**

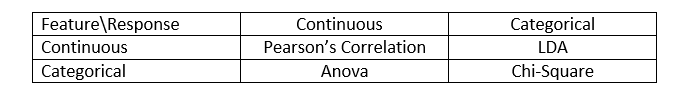
It enables the machine learning algorithm to train faster.

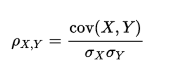
It reduces the complexity of a model and makes it easier to interpret.

It improves the accuracy of a model if the right subset is chosen.

It reduces overfitting.

**Different Methods for Feature Selection:**

1] **Filter Method**

**Pearson’s Correlation:** A Pearson's correlation is used when you want to find a linear relationship between two variables. Its value varies from -1 to +1. Pearson’s correlation is given as: 

**LDA:** Linear discriminant analysis is used to find a linear combination of features that separates two or more classes (or levels) of a categorical variable.

LDA is like PCA but it focuses on maximizing the separability among known categories.

**ANOVA:** ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using two or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not. When we have two cate IV and one cat DV then we can use T test.

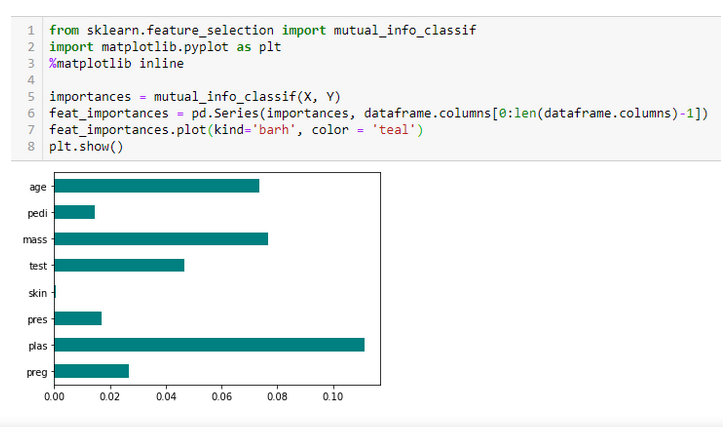
**Chi-Square:** It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

One thing that should be kept in mind is that the filter method do not remove multicollinearity. So, you must deal with the multicollinearity of features as well before training models for your data.

**Some Additional Filter Methods:**

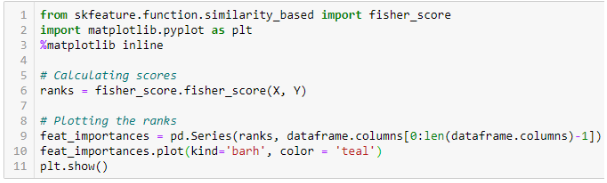
**1] Information Gain**

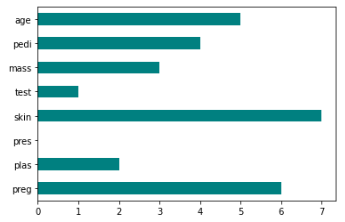
Information gain calculates the reduction in entropy from the transformation of a dataset. It can be used for feature selection by evaluating the Information gained from each variable in the context of the target variable.



2] Fisher’s Score

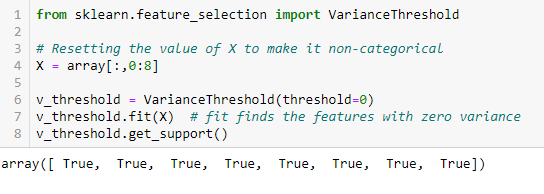
Fisher score is one of the most widely used supervised feature selection methods. The algorithm which we will use returns the ranks of the variables based on the fisher’s score in descending order. We can then select the variables as per the case.





3] Variance Threshold

The variance threshold is a simple baseline approach to feature selection. It removes all features whose variance doesn’t meet some threshold. By default, it removes all zero-variance features, i.e., features that have the same value in all samples. We assume that features with a higher variance may contain more useful information, but note that we are not taking the relationship between feature variables or feature and target variables into account, which is one of the drawbacks of filter methods.



**2] Wrapper Method:**

In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset. The problem is essentially reduced to a search problem. These methods are usually computationally very expensive.

Some common examples of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination, etc.

**Forward Selection:** Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.

**Backward Elimination:** In backward elimination, we start with all the features and remove the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed in the removal of features.

**Recursive Feature elimination:** It is a greedy optimization algorithm that aims to find the best-performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

One of the best ways for implementing feature selection with wrapper methods is to use the Boruta package that finds the importance of a feature by creating shadow features.

It works in the following steps:

1. Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).
2. Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.
3. At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the feature has a higher Z-score than the maximum Z-score of its shadow features) and constantly removes features which are deemed highly unimportant.
4. Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs.

5. Difference between Filter and Wrapper methods

1. The main differences between the filter and wrapper methods for feature selection are:
2. Filter methods measure the relevance of features by their correlation with dependent variables while wrapper methods measure the usefulness of a subset of features by actually training a model on it.
3. Filter methods are much faster compared to wrapper methods as they do not involve training the models. On the other hand, wrapper methods are computationally very expensive as well.
4. Filter methods use statistical methods for evaluation of a subset of features while wrapper methods use cross validation.

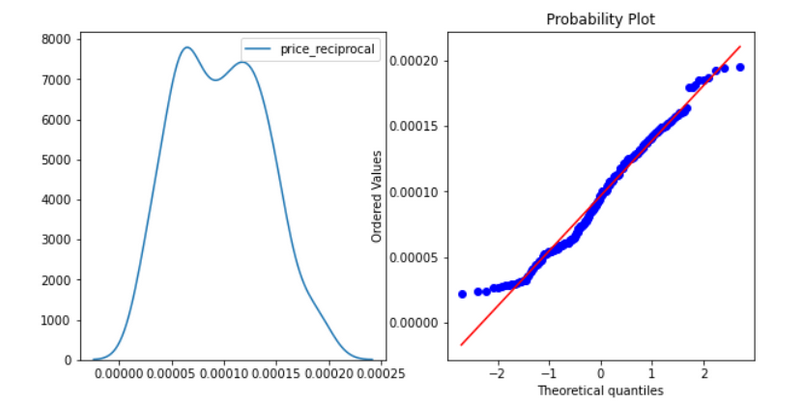
**Feature Transformation:**

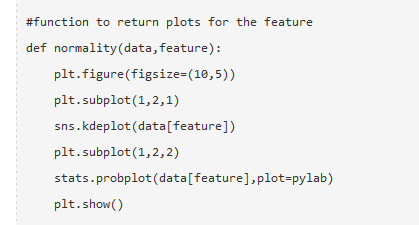
We Use Feature Transformation when our data is not normally distributed or data is positively or negative skew.

We can check data distribution using a Histogram or kde plot or Q-Q plot AKA probability plot.

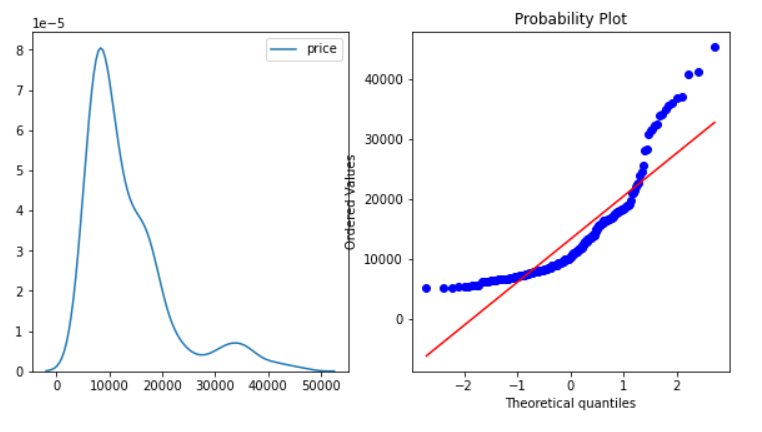
**Check Normally DIstributed Data using kde and QQ plot:**

A Q-Q plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a roughly straight line.

****

****

**Check Not Normally DIstributed Data using kde and QQ plot:**

****

**Feature Transformation Methods:**

**1]** **Logarithmic Transformation:**

This will convert the Price value to its log value i.e log(Price)

**ex.-** cp['price\_log']=np.log(cp['price'])

**2] Reciprocal Transformation :**

This will inverse values of Price i.e1/Price

**3] Square Root Transformation :**

This transformation will take the square root of the Price column i.e sqrt(Price).

**4] Exponential Transformation:**

The exponential value of the Price variable will be taken.

**5]** **Box-Cox Transformation –:**

The Box-Cox transformation is defined as:

**Ex-**  cp['price\_Boxcox'],parameters=stats.boxcox(cp['price'])

**6]QuantileTransformer:**

This method transforms the features to follow a uniform or a normal distribution.

Detecting and Treating Outliers:

**What is Outlier:**

Outliers means something unusual in comparison to the others in a group.

Similarly, an Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set.

Why do they occur?

An outlier may occur due to the variability in the data, or due to experimental error/human error.

They may indicate an experimental error or heavy skewness in the data (heavy-tailed distribution)

What do they affect?

In statistics, we have three measures of **central tendency** namely Mean, Median, and Mode. They help us describe the data.

Mean is the accurate measure to describe the data when we do not have any outliers present.

Median is used if there is an outlier in the dataset.

Mode is used if there is an outlier AND about ½ or more of the data is the same.

‘Mean’ is the only measure of central tendency that is affected by the outliers which in turn impacts Standard deviation.

Detecting Outliers:

If our dataset is small, we can detect the outlier by just looking at the dataset. But what if we have a huge dataset, how do we identify the outliers then? We need to use visualization and mathematical techniques.

Below are some of the techniques of detecting outliers

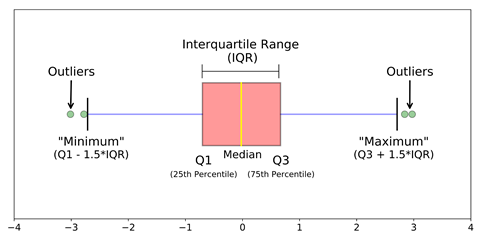
* Boxplots
* Z-score
* Inter Quantile Range(IQR)

Detecting outliers using the Z-scores

**Criteria:** any data point whose Z-score falls out of 3rd standard deviation is an outlier.

define a threshold value of 3 and mark the data points whose absolute value of Z-score is greater than the threshold as outliers.

Detecting outliers using the Interquartile Range(IQR)

 IQR to detect outliers

**Criteria:** data points that lie 1.5 times of IQR above Q3 and below Q1 are outliers.

* Sort the dataset in ascending order
* calculate the 1st and 3rd quartiles (Q1, Q3)
* compute IQR=Q3-Q1
* compute lower bound = (Q1–1.5\*IQR), upper bound = (Q3+1.5\*IQR)
* loop through the values of the dataset and check for those who fall below the lower bound and above the upper bound and mark them as outliers

5. Handling Outliers

Till now we learned about detecting outliers. The main question is WHAT do we do with the outliers?

Below are some of the methods of treating the outliers

* Trimming/removing the outlier
* Quantile-based flooring and capping
* Mean/Median imputation

1] Trimming/Remove the outliers

In this technique, we remove the outliers from the dataset. Although it is not a good practice to follow.

2] Quantile based flooring and capping(Winsorization)

In this technique, the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value.

**New array:** [15, 20.7, 18, 7.2, 13, 16, 11, 20.7, 7.2, 15, 10, 9]

The data points that are lesser than the 10th percentile are replaced with the 10th percentile value and the data points that are greater than the 90th percentile are replaced with 90th percentile value.

3] Median imputation:

As the mean value is highly influenced by the outliers, it is advised to replace the outliers with the median value

**4] QuantileTransformer:**This method transforms the features to follow a uniform or a normal distribution. When data is transformed normally the outlier automatically comes in that distribution.

**5] Discretization:** In this technique, by making the groups we include the outliers in a particular group and force them to behave in the same manner as those of other points in that group. This technique is also known as **Binning**.

**There are also some algorithms to detect and impute Anomaly also called as Anomaly Detection Algorithms:**

1] Isolation Forest

2] OneClassSVM

3] Gaussian Mixture Model

**Hyperparameter of Decision Tree & Random Forest & XGBoost:** [**(https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/)**](https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/))

**Max\_depth:**

The max\_depth of a tree in Random Forest is defined as the longest path between the root node and the leaf node:

In this graph, we can clearly see that as the max depth of the decision tree increases, the performance of the model over the training set increases continuously. On the other hand as the max\_depth value increases, the performance over the test set increases initially but after a certain point, it starts to decrease rapidly.

**Min\_sample\_split:**

The minimum number of samples we need at each node for split.

min\_sample\_split – a parameter that tells the decision tree in a random forest the minimum required number of observations in any given node in order to split it.

The default value of the minimum\_sample\_split is assigned to 2.

This means that if any terminal node has more than two observations and is not a pure node, we can split it further into subnodes.

Having a default value as 2 poses the issue that a tree often keeps on splitting until the nodes are completely pure. As a result, the tree grows in size and therefore overfits the data.

By increasing the value of the min\_sample\_split, we can reduce the number of splits that happen in the decision tree and therefore prevent the model from overfitting.

**Max\_leaf\_node:**

This hyperparameter sets a condition on the splitting of the nodes in the tree and hence restricts the growth of the tree. If after splitting we have more terminal nodes than the specified number of terminal nodes, it will stop the splitting and the tree will not grow further.

Note that if the value of the max\_leaf\_nodes is very small, the random forest is likely to underfit

**Min\_samples\_leaf:**

This Random Forest hyperparameter specifies the minimum number of samples that should be present in the leaf node after splitting a node.

**n\_estimators:**

This parameter specifies the number of trees in a Random forest algorithm.

**max\_samples:**

The *max\_samples* hyperparameter determines what fraction of the original dataset is given to any individual tree. You might be thinking that more data is always better.

**max\_features:**

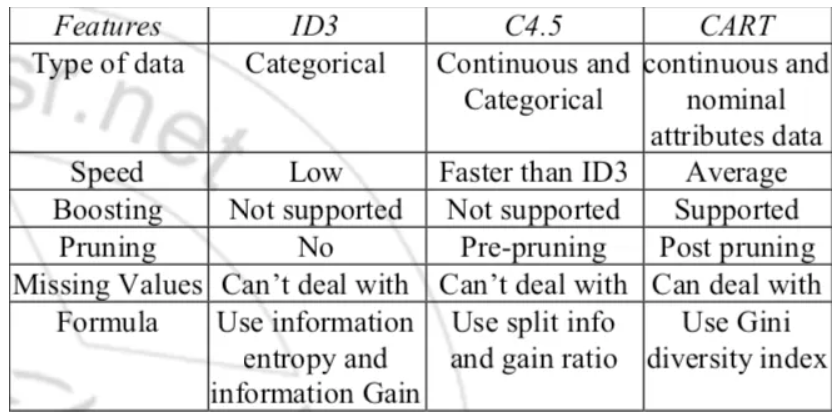
Finally, we will observe the effect of the *max\_features* hyperparameter. This resembles the number of maximum features provided to each tree in a random forest.

**XGBoost specific parameters:**

1. **learning\_rate**
   * This determines the impact of each tree on the final outcome (step 2.4). GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates.
   * Lower values are generally preferred as they make the model robust to the specific characteristics of the tree and thus allow it to generalize well.
   * Lower values would require a higher number of trees to model all the relations and will be computationally expensive.
2. **n\_estimators**
   * The number of sequential trees to be modeled (step 2)
   * Though GBM is fairly robust at higher numbers of trees, but it can still overfit at a point. Hence, this should be tuned using CV for a particular learning rate.

**Note:**

1. Choose a relatively high learning rate. Generally the default value of 0.1 works but somewhere between 0.05 to 0.2 should work for different problems.
2. Determine the optimum number of trees for this learning rate. This should range around 40-70. Remember to choose a value on which your system can work fairly fast. This is because it will be used for testing various scenarios and determining the tree parameters.
3. Tune tree-specific parameters for decided learning rate and number of trees. Note that we can choose different parameters to define a tree and I’ll take up an example here.
4. Lower the learning rate and increase the estimators proportionally to get more robust models.

****