**Machine Learning**

### **Introduction:**

Machine learning is a subset of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

## **Machine Learning Definitions**

**Algorithm:** A Machine Learning algorithm is a set of rules and statistical techniques used to learn patterns from data and draw significant information from it. It is the logic behind a Machine Learning model. An example of a Machine Learning algorithm is the Linear Regression algorithm.

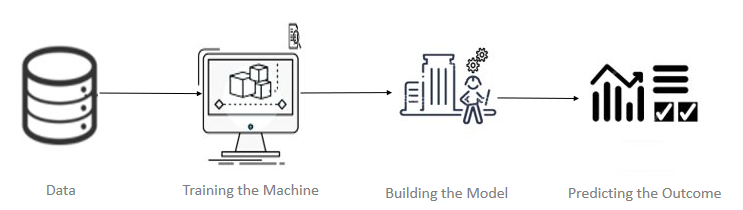
**Model:** A model is the main component of Machine Learning. A model is trained by using a Machine Learning Algorithm. An algorithm maps all the decisions that a model is supposed to take based on the given input, in order to get the correct output.

**Predictor Variable:** It is a feature(s) of the data that can be used to predict the output.

**Response Variable:** It is the feature or the output variable that needs to be predicted by using the predictor variable(s).

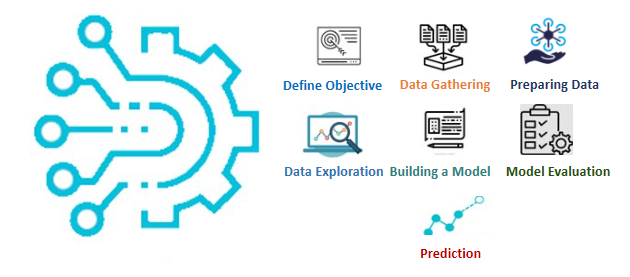
**Training Data:** The Machine Learning model is built using the training data. The training data helps the model to identify key trends and patterns essential to predict the output.

**Testing Data:** After the model is trained, it must be tested to evaluate how accurately it can predict an outcome. This is done by the testing data set.



## **Machine Learning Process**

The Machine Learning process involves building a Predictive model that can be used to find a solution for a Problem Statement. To understand the Machine Learning process let’s assume that you have been given a problem that needs to be solved by using Machine Learning.



**Step 1:** **Define the objective of the Problem Statement**

At this step, we must understand what exactly needs to be predicted. In our case, the objective is to predict the possibility of rain by studying weather conditions. At this stage, it is also essential to take mental notes on what kind of data can be used to solve this problem or the type of approach you must follow to get to the solution.

**Step 2:** **Data Gathering**

The data needed for weather forecasting includes measures such as humidity level, temperature, pressure, locality, whether or not you live in a hill station, etc. Such data must be collected and stored for analysis.

**Step 3**: **Data Preparation**

The data you collected is almost never in the right format. You will encounter a lot of inconsistencies in the data set such as missing values, redundant variables, duplicate values, etc. Removing such inconsistencies is very essential because they might lead to wrongful computations and predictions. Therefore, at this stage, you scan the data set for any inconsistencies and you fix them then and there.

**Step 4: Exploratory Data Analysis**

This stage is all about diving deep into data and finding all the hidden data mysteries. EDA or Exploratory Data Analysis is the brainstorming stage of Machine Learning. Data Exploration involves understanding the patterns and trends in the data. At this stage, all the useful insights are drawn and correlations between the variables are understood.

For example, in the case of predicting rainfall, we know that there is a strong possibility of rain if the temperature has fallen low. Such correlations must be understood and mapped at this stage.

**Step 5:** **Building a Machine Learning Model**

All the insights and patterns derived during Data Exploration are used to build the Machine Learning Model. This stage always begins by splitting the data set into two parts, training data, and testing data. The training data will be used to build and analyze the model. The logic of the model is based on the Machine Learning Algorithm that is being implemented.

In the case of predicting rainfall, since the output will be in the form of True (if it will rain tomorrow) or False (no rain tomorrow), we can use a [**Classification Algorithm**](https://www.edureka.co/blog/classification-algorithms/)such as [**Logistic Regression.**](https://www.edureka.co/blog/logistic-regression-in-r/)

Choosing the right algorithm depends on the type of problem you’re trying to solve, the data set and the level of complexity of the problem.

**Step 6:** **Model Evaluation & Optimization**

After building a model by using the training data set, it is finally time to put the model to a test. The testing data set is used to check the efficiency of the model and how accurately it can predict the outcome. Once the accuracy is calculated, any further improvements in the model can be implemented at this stage. Methods like parameter tuning and cross-validation can be used to improve the performance of the model.

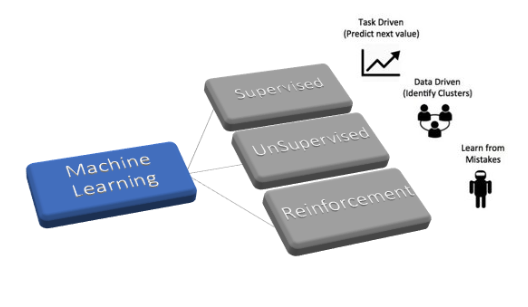
**Step 7: Predictions**

Once the model is evaluated and improved, it is finally used to make predictions. The final output can be a Categorical variable (e.g. True or False) or it can be a Continuous Quantity

**Types of Machine Learning**

A machine can learn to solve a problem by following any one of the following three approaches.

* Supervised Learning
* Unsupervised Learning
* Reinforcement Learning



### **Supervised Learning**

Supervised learning is an approach to creating artificial intelligence ([AI](https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence)), where the program is given labeled input data and the expected output results.

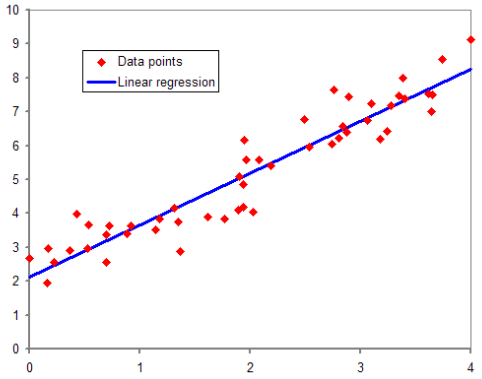
Supervised Learning further divided into two types:

* **Regression:** In this type of problem the output is a continuous quantity. So, for example, if you want to predict the speed of a car given the distance, it is a Regression problem. Regression problems can be solved by using Supervised Learning algorithms like Linear Regression.
* **Classification:** In this type, the output is a categorical value. Classifying emails into two classes, spam and non-spam is a classification problem that can be solved by using Supervised Learning classification algorithms such as Support Vector Machines, Naive Bayes, Logistic Regression, K Nearest Neighbor, etc.

**Linear Regression:**

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It's used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories

Simple linear regression is a regression technique in which the independent variable has a linear relationship with the dependent variable. The straight line in the diagram is the best fit line. The main goal of the simple linear regression is to consider the given data points and plot the best fit line to fit the model in the best way possible.

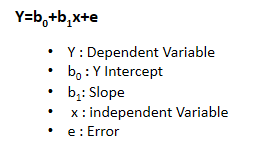


## **Linear Regression Terminologies**

The following terminologies are important to be familiar with before moving on to the linear regression algorithm.

### **Cost Function**

The best fit line can be based on the linear equation given below.



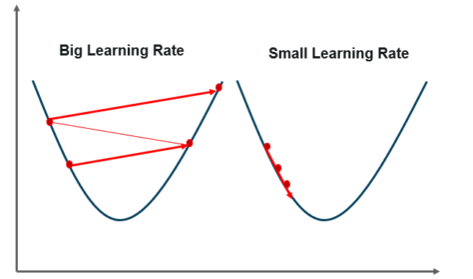
* The dependent variable that is to be predicted is denoted by Y.
* A line that touches the y-axis is denoted by the intercept b0.
* b1 is the slope of the line, x represents the independent variables that determine the prediction of Y.
* The error in the resultant prediction is denoted by e.

The cost function provides the best possible values for b0 and b1 to make the best fit line for the data points. We do it by converting this problem into a minimization problem to get the best values for b0 and b1. The error is minimized in this problem between the actual value and the predicted value.

### **Gradient Descent**

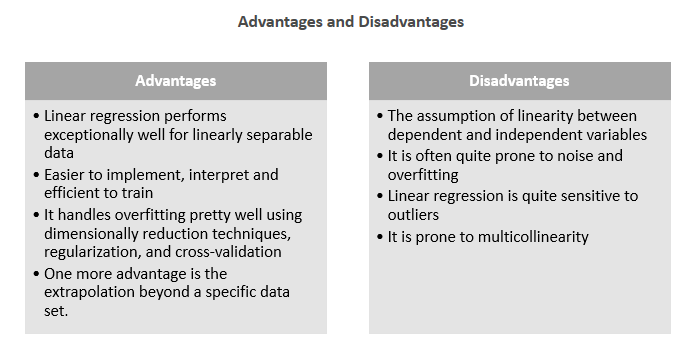
It is a method of updating b0 and b1 values to reduce the MSE. The idea behind this is to keep iterating the b0 and b1 values until we reduce the MSE to the minimum.

To update b0 and b1, we take gradients from the cost function. To find these gradients, we take partial derivatives with respect to b0 and b1. These partial derivatives are the gradients and are used to update the values of b0 and b1.



A smaller learning rate takes closer to the minimum time, but it takes more time in case of a larger learning rate The time taken is sooner, but there is a chance to overshoot the minimum value.

**A few advantages and disadvantages of linear regression for machine learning:**

**Implementing Linear Regression**

The process takes place in the following steps:

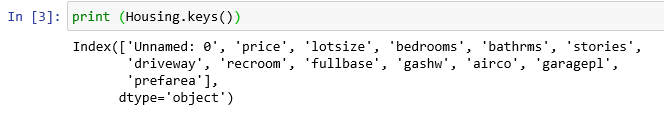
1. Loading the Data
2. Exploring the Data
3. Slicing the Data
4. Train and Split Data
5. Generate the Model
6. Evaluate The accuracy

Let us get into the details of each of the steps to implement linear regression with [Housing data set](https://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Housing.csv) and we want to predict the price of the house.

1. **Loading the Data**

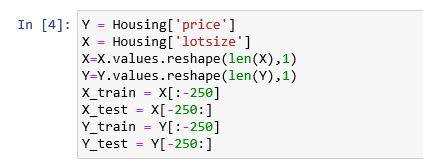


1. **Exploring the Data**

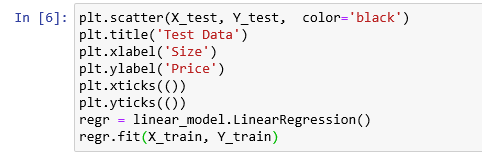


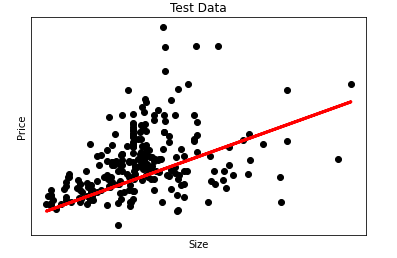
1. **Splitting the Data**

we will split the data into train and test set.

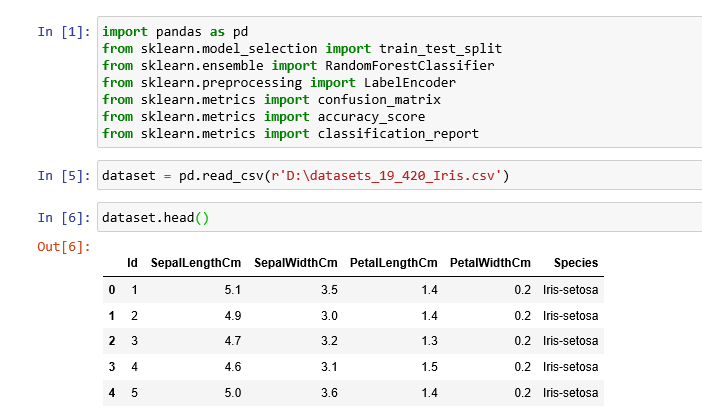


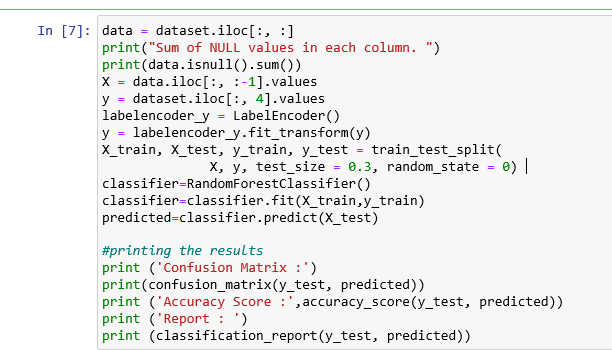
1. **Generating the model**



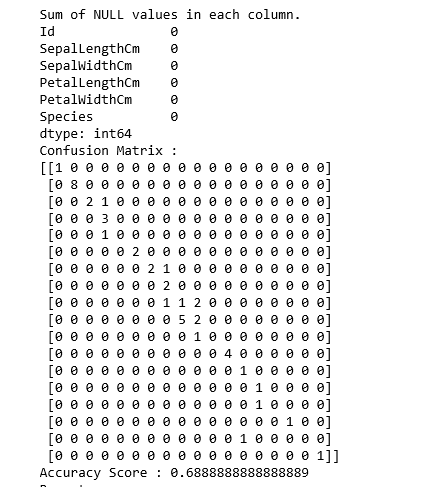
1. **Evaluation**

**Classification Example**





Output:



### **Unsupervised Learning**

Unsupervised learning involves training by using unlabeled data and allowing the model to act on that information without guidance.

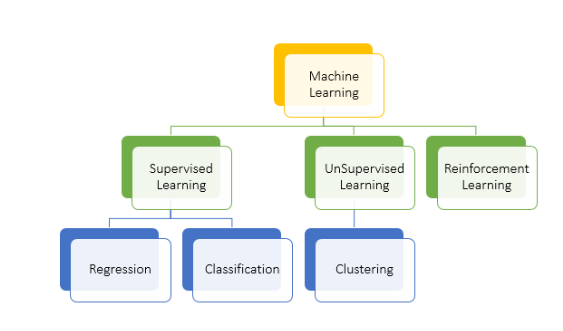
Unsupervised Learning is further divided in to:

**Clustering:** This type of problem involves assigning the input into two or more clusters based on feature similarity. For example, clustering viewers into similar groups based on their interests, age, geography etc. can be done by using Unsupervised Learning algorithms like K-Means Clustering.

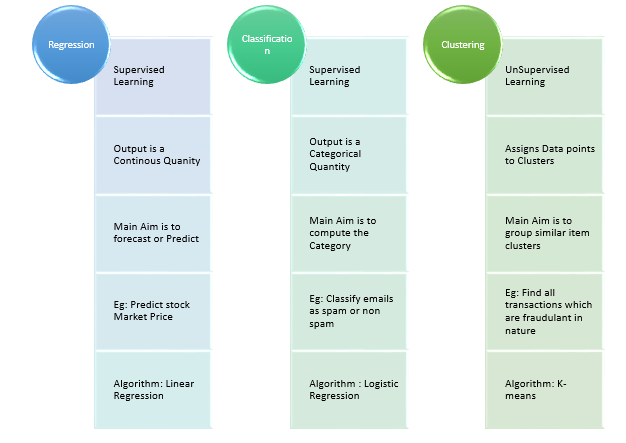
### **Reinforcement Learning**

Reinforcement Learning is a part of Machine learning where an agent is put in an environment and he learns to behave in this environment by performing certain actions and observing the rewards which it gets from those actions.

## **Type of Problems in Machine Learning**



**The difference between Regression, Classification, and Clustering:**



## **Exploratory Data Analysis**

Exploratory data analysis is an approach to [analyzing data sets](https://www.edureka.co/blog/football-world-cup-best-xi-analysis-using-python/) to summarize their main characteristics, often with visual methods.

For example, if you are planning to go on a trip to particular location, things you would do before taking a decision:

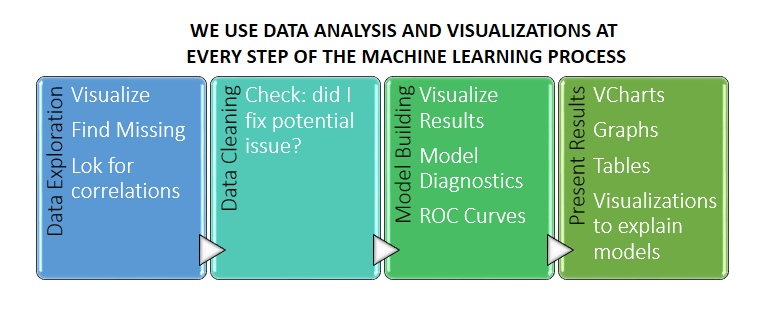
* You will explore the particulars of the location such as waterfalls, trekking, beaches and restaurants.
* You will try to collect information through Google, Instagram, Facebook, and other social Websites.
* Calculate whether it falls under your budget or not.
* Assess the time it takes to cover all the places.
* Type of Travel method.

Similarly, when you are trying to build a **M**[**achine Learning Model**](https://www.edureka.co/blog/logistic-regression-in-python/)**,** you need to be pretty sure whether your data is making sense or not. The main aim of exploratory [data analysis](https://www.edureka.co/blog/python-pandas-tutorial/) is to gain confidence in your data to an extent where you’re ready to engage a machine learning algorithm.

## **Need for Exploratory Data Analysis**

Exploratory Data Analysis is a crucial step before you jump to machine learning or modeling of your data. By doing this, we can get to know whether the selected features are good enough to model. Are all the features required? Are there any correlations detected in the data based on which we can either go back to the Data Pre-processing step or move on with modeling?

Once Exploratory Data Analysisis complete and insights are drawn, its feature can be used for supervised and unsupervised machine learning modeling.InExploratory Data Analysis, you will have many plots, heat-maps, frequency distribution, graphs and correlation matrix along with the hypothesis by which any individual can understand what your data is all about and the insights you got from exploring your data set.



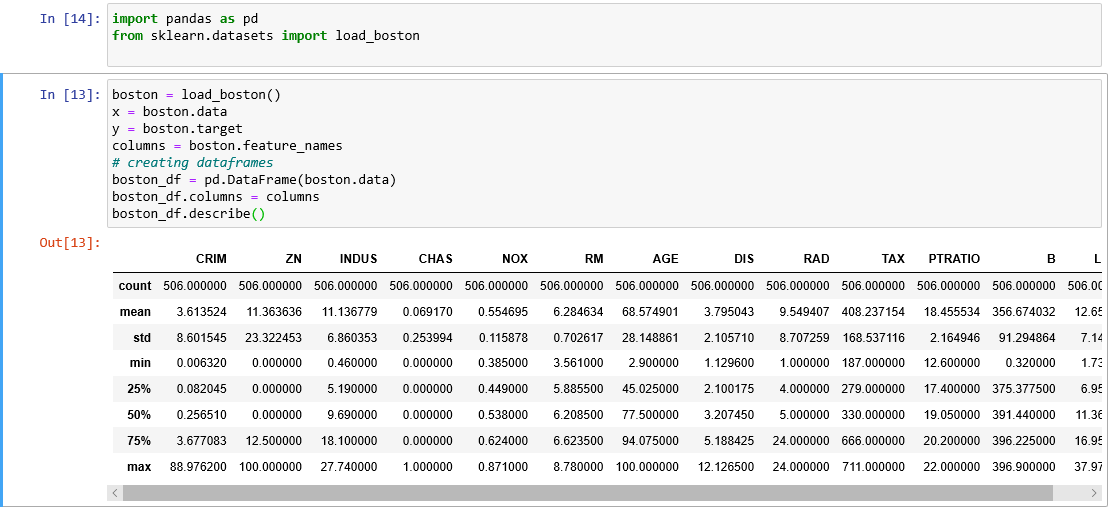
**Steps in Exploratory Data Analysis**

* Description of data
* Handling missing data
* Handling outliers
* Understanding relationships and new insights through plots

Below few steps using the Boston Data Set which can be imported from **sklearn.datasets import load\_boston**

1. **Description of data:**

We need to know the different kinds of data and other statistics of our data before we can move on to the other steps. A good one is to start with **describe ()** function in python. In pandas, we can apply describe () on a Data Frame which helps in generating descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset’s distribution, excluding NaN values. The result’s index will include count, mean, std, min, max as well as lower, 50 and upper percentiles. By default, the lower percentile is 25 and the upper percentile is 75. The 50 percentile is the same as the median.

1. **Handling missing data:**

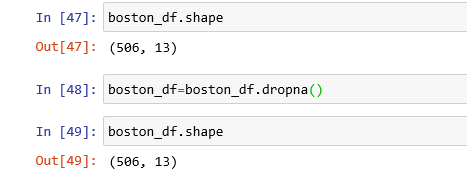
Data in the real-world are rarely clean and homogeneous. The required data can either be missing during data extraction or collection due to several reasons. Missing values need to be handled carefully because they reduce the quality of any of our performance matrix. It can also lead to wrong prediction or classification and can also cause a high bias for any given model being used. There are several options for handling missing values. However, the choice of what should be done is largely dependent on the nature of our data and the missing values.

Below are some of the techniques:

* Drop NULL or missing values
* Fill Missing Values
* Predict Missing values with an ML Algorithm

### **Drop NULL or missing values:**

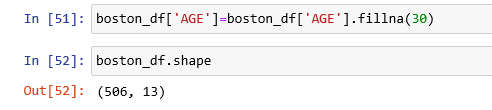
This is the fastest and easiest step to handle missing values. However, it is not generally advised. This method reduces the quality of our model as it reduces sample size because it works by deleting all other observations where any of the variables is missing.



The above code indicates that there are no null values in our data set.

### **Fill Missing Values**:

This is the most common method of handling missing values. This is a process whereby missing values are replaced with a test statistic like mean, median or mode of the particular feature the missing value belongs to. Let’s suppose we have a missing value of age in the Boston data set. Then the below code will fill the missing value with the 30.



### **Predict Missing values with an ML Algorithm:**

This is by far one of the best and most efficient methods for handling missing data. Depending on the class of data that is missing, one can either use a regression or classification model to predict missing data.

1. **Handling outliers:**

An outlier is something which is separate or different from the crowd. Outliers can be a result of a mistake during data collection or it can be just an indication of variance in your data. Some of the methods for detecting and handling outliers:

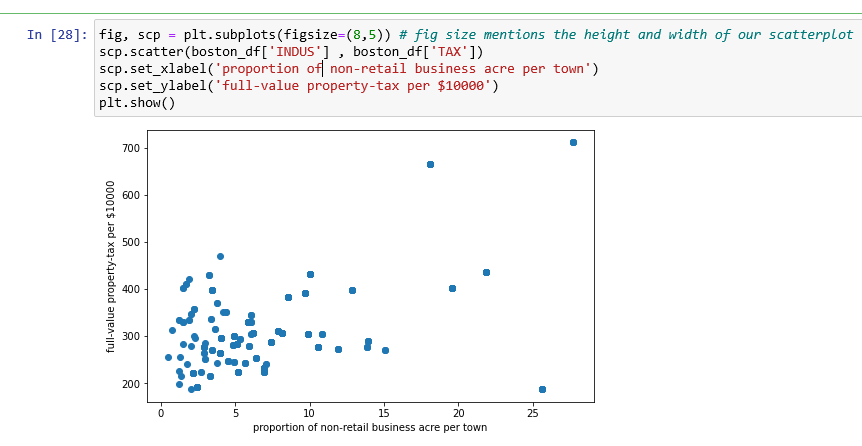
* Boxplot
* [Scatterplot](https://www.edureka.co/blog/python-matplotlib-tutorial/#Scatter)
* Z-score
* IQR (Inter-Quartile Range)

### **Boxplot:**

A box plot is a method for graphically depicting groups of numerical data through their quartiles. The box extends from the Q1 to Q3 quartile values of the data, with a line at the median (Q2). The whiskers extend from the edges of the box to show the range of the data. Outlier points are those past the end of the whiskers. Boxplots show robust measures of location and spread as well as providing information about symmetry and outliers.

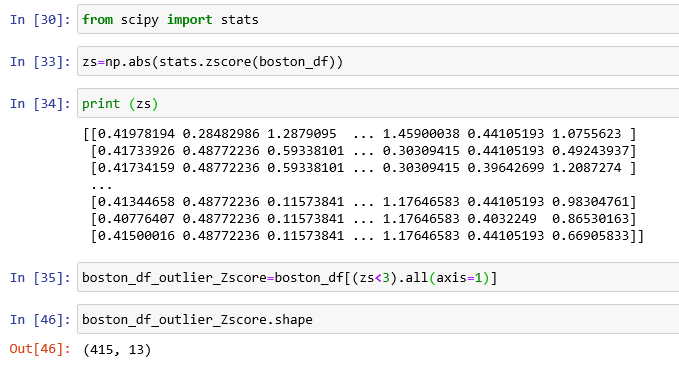
### **Scatterplot**:

A scatter plot is a mathematical diagram using Cartesian coordinates to display values for two variables for a set of data. The data are displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis. The points that are far from the population can be termed as an outlier.



### **Z-score:**

The Z-score is the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured. While calculating the Z-score we re-scale and center the data and look for data points that are too far from zero. These data points which are way too far from zero will be treated as the outliers. In most of the cases a threshold of 3 or -3 is used i.e. if the Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers.

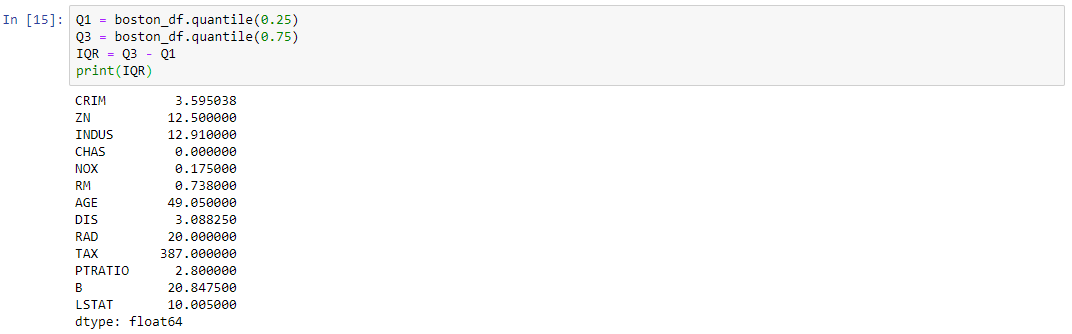


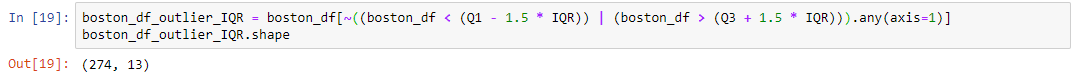
We can see from the above code that the shape changes, **(Earlier we have (506,13) and now changed to (415,13))** which indicates that our dataset has some outliers.

### **IQR:**

The interquartile range (IQR) is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles.

**IQR = Q3 − Q1.**

Once we have IQR scores below code will remove all the outliers in our dataset.



1. **Understanding relationships and new insights through plots:**

We can get many relations in our data by visualizing our dataset. Let’s go through some techniques in order to see the insights.

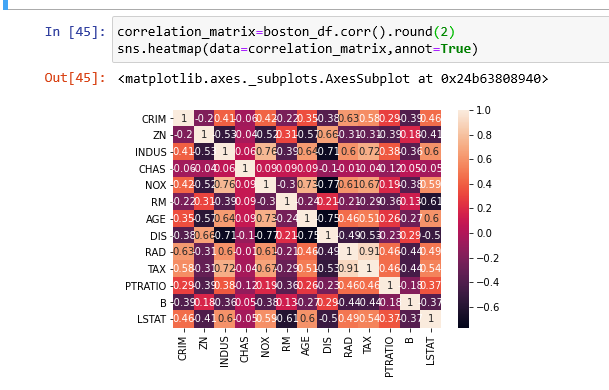
* Histogram
* Heat maps

### **Histogram:**

A histogram is a great tool for quickly assessing a probability distribution

### **Heat maps:**

The Heat Map procedure shows the distribution of a quantitative variable over all combinations of 2 categorical factors. If one of the 2 factors represents time, then the evolution of the variable can be easily viewed using the map. A gradient color scale is used to represent the values of the quantitative variable. The correlation between two random variables is a number that runs from -1 through 0 to +1 and indicates a strong inverse relationship, no relationship, and a strong direct relationship, respectively.



**Performance Tuning Metrics in Machine Learning**

When creating a machine learning model, you'll be presented with design choices as to how to define your model architecture. Often times, we don't immediately know what the optimal model architecture should be for a given model, and thus we'd like to be able to explore a range of possibilities. In true machine learning fashion, we'll ideally ask the machine to perform this exploration and select the optimal model architecture automatically. Parameters which define the model architecture are referred to ashyperparameters and thus this process of searching for the ideal model architecture is referred to as hyperparameter tuning.

## **Optimization for Classification Problems**

### **1. True Positive (Recall)**

The **True Positive Rate,** also called **Recall, in** binary/non-binary classification problems. Most of the time we are only interested in correctly predicting one class. For example, if you were predicting diabetes, you will care more about predicting whether this person has diabetes than predicting that this person does not have diabetes. In this situation, the positive class is, "This person has diabetes," and the negative class is, "This person does not have diabetes." It is merely the accuracy of predicting the positive class

**This is not the Accuracy performance metric. See number 4 below for more details.**

### **2. ROC Curve (Receiver Operating Characteristic Curve)**

An **ROC Curve** shows the performance of your classification model at different thresholds (probability of classification into a certain class). It plots the True Positive Rate and False Positive Rate against each other. Lowering the threshold will increase your True Positive Rate but sacrifice your False Positive Rate and vice versa.

### **3. AUC (Area Under the Curve)**

AUC is also known as ‘Area Under the ROC Curve’. Simply put, the AUC will tell you the probability of correctly classifying your classes. A higher AUC represents a better model.

### **4. Accuracy**

Accuracy is, by default, the first thing to look at. A better way to call it is the **Average Accuracy** of predicting all classes. Like I mentioned with True Positive Rate, it is the most ideal metric to optimize. Accuracy will take the average of the sum of **True Positive** and **True Negative**. Most times, in unbalanced classification problems, the **Negative class** is more represented than the **Positive class** so you are more likely to have a very high **True Negative Rate**. The **Accuracy** will then be biased to the accurate predictions of the **Negative class,** which might not interest anyone.

**Confusion Matrix**  
A much better way to evaluate the performance of a classifier is to look at the confusion matrix. The general idea is to count the number of times instances of class A are classified as class B. For example, to know the number of times the classifier confused images of 5s with 3s, you would look in the 5th row and 3rd column of the confusion matrix.

## **Regression Optimization in Machine Learning**

**5. Error**

Often overlooked next to R2, the error tells us more about the precision of the fitted values to the regression line (i.e. the average distance between the fitted values and the line of best fit). This is more important when calculating confidence and prediction intervals for your model. It is more interpretable due to the use of the natural units of the response variable, whereas the R2 has no units and is only between 0 and 1.

There are different types of errors such as **Mean Absolute Error** and **Root Mean Squared Error**.

**6. R2**

Now, although Standard Error is important, the R2 has become the de-facto measure of a good regression model. It tells us how much the variation between the dependent variable and the independent variables are explained by the model. A higher R2 gives a better model, however, if too high at close to 99%, it can sometimes cause the risk of overfitting. R2 can be misleading due to the **correlation vs causation** debate that can give an illogically high R2.

## **Cross Validation**

Cross Validation is a technique which involves reserving a particular sample of a dataset on which you do not train the model. Later, you test your model on this sample before finalizing it.

Here are the steps involved in cross validation:

* You reserve a sample data set
* Train the model using the remaining part of the dataset
* Use the reserve sample of the test (validation) set. This will help you in gauging the effectiveness of your model’s performance. If your model delivers a positive result on validation data, go ahead with the current model

## **Common methods used for Cross Validation**

There are various methods available for performing cross validation.

**The validation set approach**

In this approach, we reserve 50% of the dataset for validation and the remaining 50% for model training. However, a major disadvantage of this approach is that since we are training a model on only 50% of the dataset, there is a huge possibility that we might miss out on some interesting information about the data which will lead to a higher bias.

### **Leave one out cross validation (LOOCV)**

In this approach, we reserve only one data point from the available dataset, and train the model on the rest of the data. This process iterates for each data point. This also has its own advantages and disadvantages. Let’s look at them:

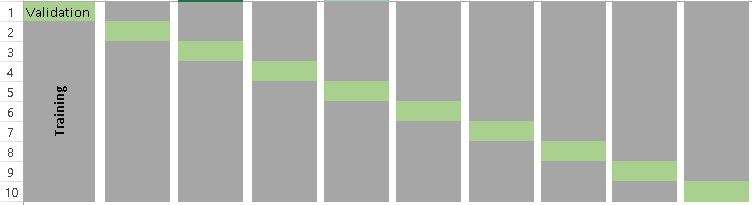
* We make use of all data points, hence the bias will be low
* We repeat the cross-validation process n times (where n is number of data points) which results in a higher execution time
* This approach leads to higher variation in testing model effectiveness because we test against one data point. So, our estimation gets highly influenced by the data point. If the data point turns out to be an outlier, it can lead to a higher variation

### **k-fold cross validation**

It’s easy to follow and implement. Below are the steps for it:

* Randomly split your entire dataset into k-folds
* For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for *kth* fold
* Record the error you see on each of the predictions
* Repeat this until each of the k-folds has served as the test set
* The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model

Below is the visualization of a k-fold validation when k=10.

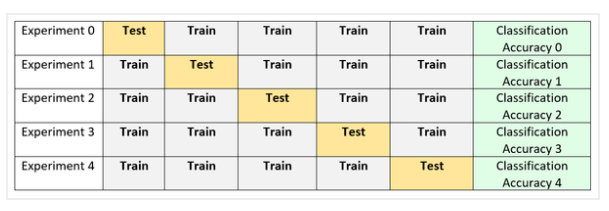


**How to choose the right value of k?**

Always remember, a lower value of **k** is more biased, and hence undesirable. On the other hand, a higher value of K is less biased, but can suffer from large variability. It is important to know that a smaller value of *k* always takes us towards validation set approach, whereas a higher value of *k* leads to LOOCV approach.

### **Stratified k-fold cross validation**

Stratification is the process of rearranging the data so as to ensure that each fold is a good representative of the whole. For example, in a binary classification problem where each class comprises of 50% of the data, it is best to arrange the data such that in every fold, each class comprises of about half the instances.



It is generally a better approach when dealing with both bias and variance. A randomly selected fold might not adequately represent the minor class, particularly in cases where there is a huge class imbalance.

### **Adversarial Validation**

When dealing with real datasets, there are often cases where the test and train sets are very different. As a result, the internal cross-validation techniques might give scores that are not even in the ballpark of the test score. In such cases, adversarial validation offers an interesting solution.

The general idea is to check the degree of similarity between training and tests in terms of feature distribution. If It does not seem to be the case, we can suspect they are quite different. This intuition can be quantified by combining train and test sets, assigning 0/1 labels (0 – train, 1-test) and evaluating a binary classification task.

However, you must be careful while using this type of validation technique. Once the distribution of the test set changes, the validation set might no longer be a good subset to evaluate your model on.

**Cross Validation for time series**

Splitting a time-series dataset randomly does not work because the time section of your data will be messed up. For a time series forecasting problem, we perform cross validation in the following manner.

* Folds for time series cross validation are created in a forward chaining fashion
* Suppose we have a time series for yearly consumer demand for a product during a period of n years. The folds would be created like:

**Custom Cross Validation Techniques**

Unfortunately, there is no single method that works best for all kinds of problem statements. Often, a custom cross validation technique based on a feature, or combination of features, could be created if that gives the user stable cross validation scores while making submissions in hackathons.

**Approaches to Hyper tuning Parameters in Machine Learning**

When creating a machine learning model, you'll be presented with design choices as to how to define your model architecture. Often times, we don't immediately know what the optimal model architecture should be for a given model, and thus we'd like to be able to explore a range of possibilities. In true machine learning fashion, we'll ideally ask the machine to perform this exploration and select the optimal model architecture automatically. Parameters which define the model architecture are referred to as **hyperparameters** and thus this process of searching for the ideal model architecture is referred to as hyperparameter tuning.

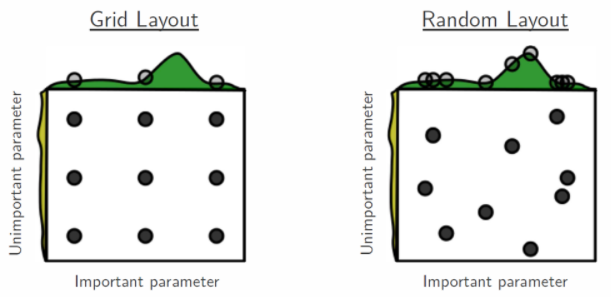
###### **Grid search**

Grid search is arguably the most basic hyperparameter tuning method. With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model, and selecting the architecture which produces the best results.

For example, we would define a list of values to try for both n\_estimators and max\_depth and a grid search would build a model for each possible combination.

**Randomized Search**

RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyperparameter settings. It moves within the grid in random fashion to find the best set hyperparameters. This approach reduces unnecessary computation.



**Examples:**

1. **KNN Algorithm. (**Performing grid search over the defined hyperparameter space**)**

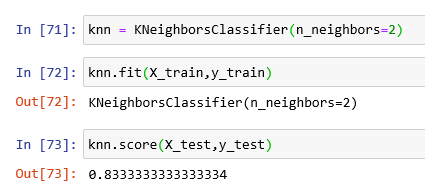
Parameters Used in KNN Algorithm, to improve accuracy was:

leaf size, n neighbors, p

* + **Leaf size** – This can affect the speed of the construction query. **construction time**
    - A larger leaf size leads to a faster tree construction time, because fewer nodes need to be created
    - **query time:** Both a large or small leaf size can lead to suboptimal query cost. For leaf size approaching 1, the overhead involved in traversing nodes can significantly slow query times. For leaf size approaching the size of the training set, queries become essentially brute force. A good compromise between these is leaf size = 30, the default value of the parameter.
    - **Memory:** As leaf size increases, the memory required to store a tree structure decreases. This is especially important in the case of ball tree, which stores a -dimensional centroid for each node.
  + **n neighbors** – Number of Neighbors used to find the accuracy.
  + **P** -- Power parameter for the Minkowski metric. When p = 1, this is equivalent to using Manhattan distance (11), and Euclidean distance (12) for p = 2.

**Example:**

Leaf size range from 1 to 50 and n\_neighbors from 1 to 30 and p value and found best leaf size, p and neighbors.



Using Best Parameters, predicted with accuracy of 83%.

1. **SVM Algorithm.**

### **Gamma:** gamma is a parameter of the RBF kernel and can be thought of as the ‘spread’ of the kernel and therefore the decision region. When gamma is low, the ‘curve’ of the decision boundary is very low and thus the decision region is very broad. When gamma is high, the ‘curve’ of the decision boundary is high, which creates islands of decision-boundaries around data points.

**C:** C is a parameter of the SVC learner and is the penalty for misclassifying a data point. When C is small, the classifier is okay with misclassified data points (high bias, low variance). When C is large, the classifier is heavily penalized for misclassified data and therefore bends over backwards avoid any misclassified data points (low bias, high variance).

**Parameters used in SVM Algorithm, param\_grid, refit, verbose**

**param\_grid:** In which we give c and gamma values to find the best accuracy.

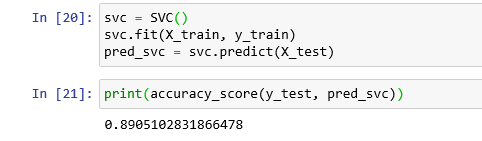
**Refit:** Boolean **Refit** the best estimator with the entire dataset. If “False”, it is impossible to make predictions using this **GridSearchCV** instance after fitting. It looks like it is **True** by default.

**cv:** integer or cross validation generator, optional

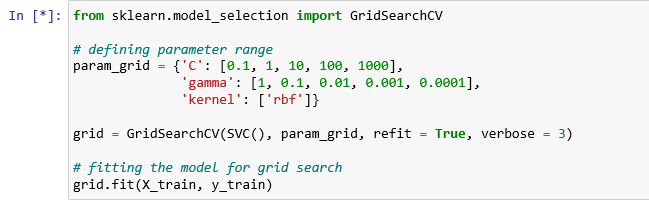
If an integer is passed, it is the number of fold (default 3). Specific cross validation objects can be passed, see sklearn.cross\_validation module for the list of possible objects

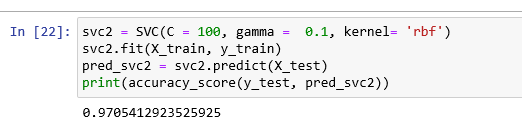
**verbose: integer:**

Controls the verbosity: the higher, the more messages.



After applying Hyper tuning Parameters,

Found best params and finding accuracy.



Increased accuracy from 89 to 97%.

1. **Decision Tree Algorithm.**

Parameters Used in Decision Tree Algorithm.

* **criterion {“gini”, “entropy”}, default=”gini”**

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

* **splitter {“best”, “random”}, default=” best”**

The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

* **max\_depth int, default=None**

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

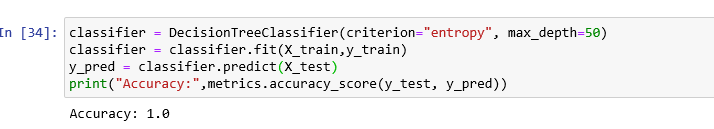
* **min\_samples\_split int or float, default=2**

The minimum number of samples required to split an internal node:

* + If int, then consider min\_samples\_split as the minimum number.
  + If float, then min\_samples\_split is a fraction and ceil(min\_samples\_split \* n\_samples) are the minimum number of samples for each split.

Predicted algorithm with best accuracy by applying gini and entropy parameters randomly. Out of them which is best, we can find using hyper tuning parameters.

Found best estimators and fit the train data.

Implemented using best estimators in Decision tree algorithm with best accuracy.

1. **Random Forest Algorithm.**

# **N\_estimators:** n\_estimators represents the number of trees in the forest. Usually the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably, therefore we do a parameter search to find the sweet spot.

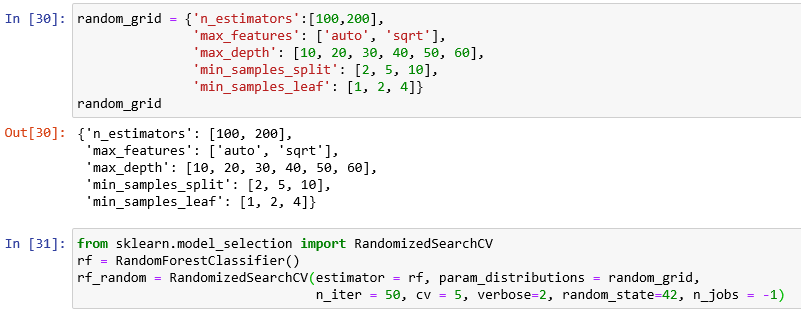
# **max\_depth:** max\_depth represents the depth of each tree in the forest. The deeper the tree, the more splits it has and it captures more information about the data. We fit each decision tree with depths ranging from 1 to 32 and plot the training and test errors.

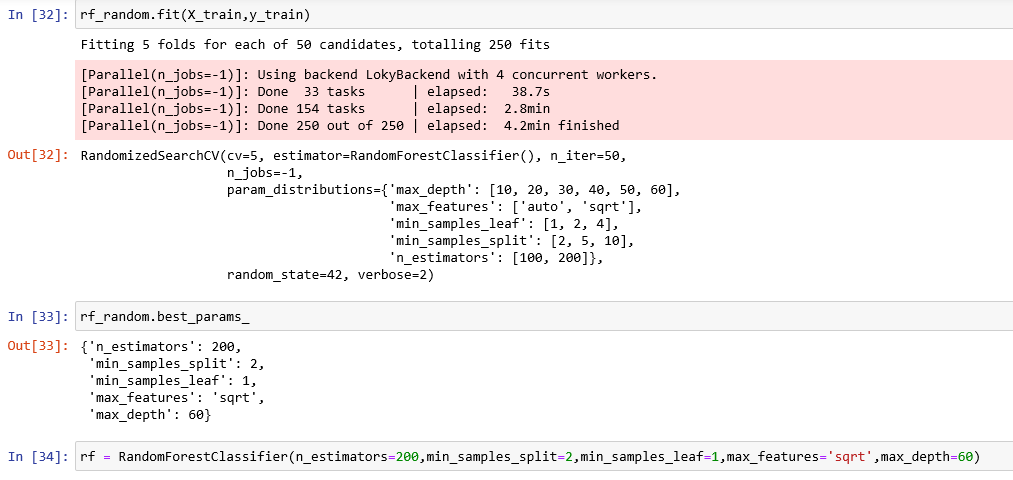
# **min\_samples\_split:** min\_samples\_split represents the minimum number of samples required to split an internal node. This can vary between considering at least one sample at each node to considering all of the samples at each node. When we increase this parameter, each tree in the forest becomes more constrained as it has to consider more samples at each node. Here we will vary the parameter from 10% to 100% of the samples.

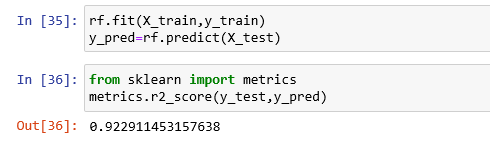
# **min\_samples\_leaf:** min\_samples\_leaf is the minimum number of samples required to be at a leaf node. This parameter is similar to min\_samples\_splits, however, this describe the minimum number of samples of samples at the leafs, the base of the tree.

* **max\_features:** max\_features represent the number of features to consider when looking for the best split**.**

**Example:**

After fitting train and test data, finding best parameters.

Implementing these best params and finding accuracy;



**Random Forest GridsearchCV**

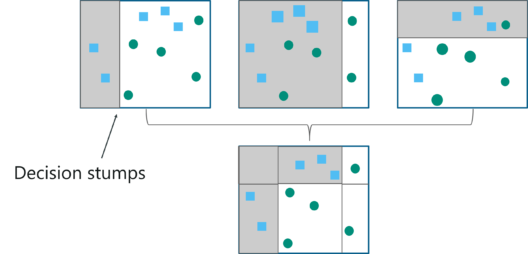
Using Same Parameters for GridsearchCV and finding the best parameters out of them.



Finding best parameters and estimators then implementing with random forest and finding the accuracy.

1. **Boosting**

The basic principle behind the working of the boosting algorithm is to generate multiple weak learners and combine their predictions to form one strong rule. These weak rules are generated by applying base Machine Learning algorithms on different distributions of the data set. These algorithms generate weak rules for each iteration. After multiple iterations, the weak learners are combined to form a strong learner that will predict a more accurate outcome.



how the algorithm works:

**Step 1:** The base algorithm reads the data and assigns equal weight to each sample observation.

**Step 2:** False predictions made by the base learner are identified. In the next iteration, these false predictions are assigned to the next base learner with a higher weightage on these incorrect predictions.

**Step 3:** Repeat step 2 until the algorithm can correctly classify the output.

Therefore, the main aim of Boosting is to focus more on miss-classified predictions.

Now that we know how the boosting algorithm works, let’s understand the different types of boosting techniques.

## **Adaptive Boosting (Adaboost)**

* AdaBoost is implemented by combining several weak learners into a single strong learner.
* The weak learners in AdaBoost take into account a single input feature and draw out a single split decision tree called the decision stump. Each observation is weighed equally while drawing out the first decision stump.
* The results from the first decision stump are analyzed and if any observations are wrongfully classified, they are assigned higher weights.
* Post this, a new decision stump is drawn by considering the observations with higher weights as more significant.
* Again, if any observations are misclassified, they’re given higher weight and this process continues until all the observations fall into the right class.
* Adaboost can be used for both classification and regression-based problems, however, it is more commonly used for classification purpose.

**Parameters used in Adaboost Regressor.**

1. **base\_estimator object, default=None**

The base estimator from which the boosted ensemble is built. If None, then the base estimator is DecisionTreeRegressor (max\_depth=3).

1. **n\_estimators int, default=50**

The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.

1. **learning\_rate float, default=1.**

Learning rate shrinks the contribution of each regressor by learning\_rate. There is a trade-off between learning\_rate and n\_estimators.

1. **loss {‘linear’, ‘square’, ‘exponential’}, default=’linear’**

The loss function to use when updating the weights after each boosting iteration.

1. **random\_state int or Random State, default=None**

Controls the random seed given at each base\_estimator at each boosting iteration. Thus, it is only used when base\_estimator exposes a random\_state. In addition, it controls the bootstrap of the weights used to train the base\_estimator at each boosting iteration. Pass an int for reproducible output across multiple function calls.

**Parameters Used for Adaboost Classifier:**

1. **base\_estimator object, default=None**

The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper classes\_ and n\_classes\_ attributes. If None, then the base estimator is DecisionTreeClassifier (max\_depth=1).

1. **n\_estimators int, default=50**

The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.

1. **learning\_rate float, default=1.**

Learning rate shrinks the contribution of each classifier by learning\_rate. There is a trade-off between learning\_rate and n\_estimators.

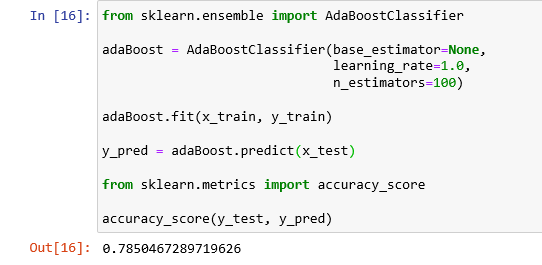
1. **Algorithm {‘SAMME’, ‘SAMME.R’}, default=’SAMME.R’**

If ‘SAMME.R’ then use the SAMME.R real boosting algorithm. base\_estimator must support calculation of class probabilities. If ‘SAMME’ then use the SAMME discrete boosting algorithm. The SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.

1. **random\_state int or Random State, default=None**

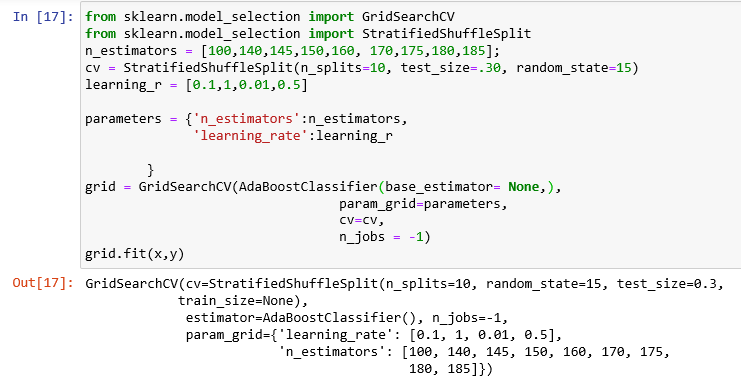
Controls the random seed given at each base\_estimator at each boosting iteration. Thus, it is only used when base\_estimator exposes a random\_state. Pass an int for reproducible output across multiple function

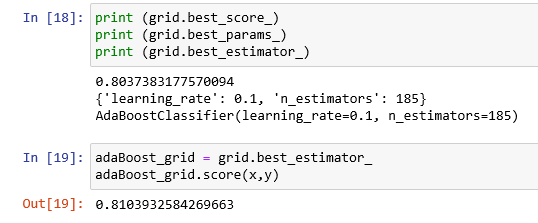
**Example on Decision Tree Classifier: (Default with None as decision tree)**



Predicted accuracy with 78% by default with base\_estimator as Decision Tree.

Using GridSearchCV, tuning the performance of Decision Tree algorithm by giving parameter values.

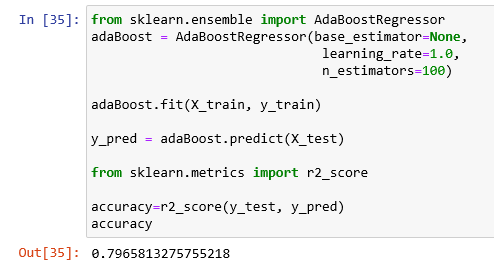




By this we found Best Score, Params and estimators.

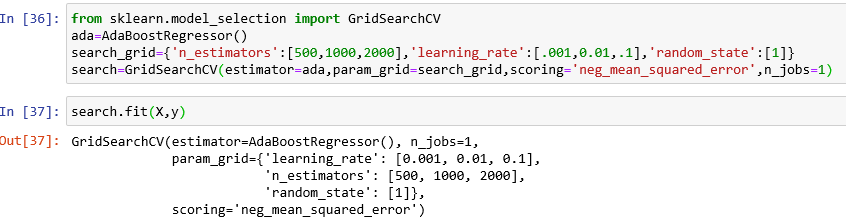
And predicting the accuracy for adaboost with best\_estimator. Increased accuracy after applying Tuning Parameters.

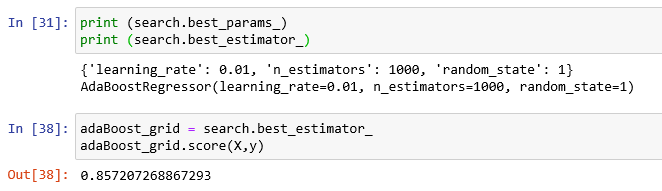
**Example on Decision Tree Regresor: (Default with None as decision tree)**



Predicted accuracy with 83% by default with base\_estimator as Decision Tree.

Using GridSearchCV, tuning the performance of Decision Tree algorithm by giving parameter values.



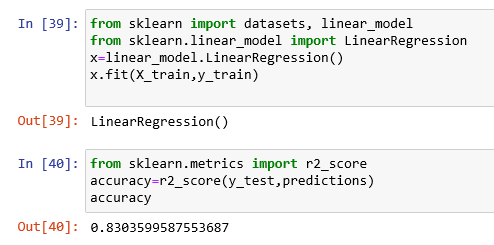


By this we found Best Score, Params and estimators.

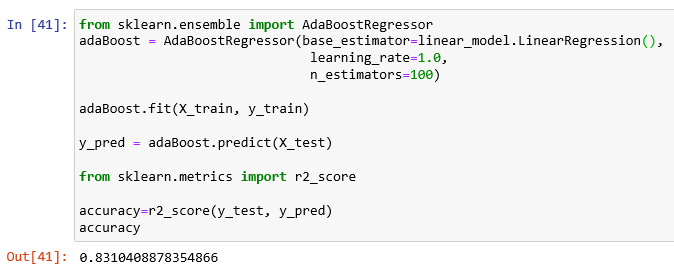
And predicting the accuracy for adaboost with best\_estimator. Increased accuracy after applying Tuning Parameters.

**Example on Linear Regression:**

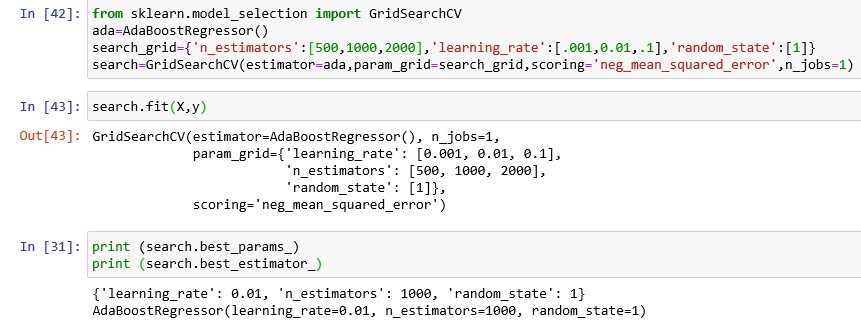
**Adaboost** Example with Linear Regression by considering base\_estimator as linear\_model.

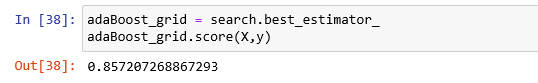


Using Linear Regression Predicted accuracy with 83%. Now, applying ababoost and Cross Validation technique.

Given bae\_estimator as Linear\_model.LinearRegression(). Also, can mention variable which was assigned earlier as **x** in place of model name.

Now, Applying GridSearchCV to improve accuracy with adaboost which is as below.

Found best params and estimators, and checking the accuracy.



Improved accuracy percentage with GridSearchCV.