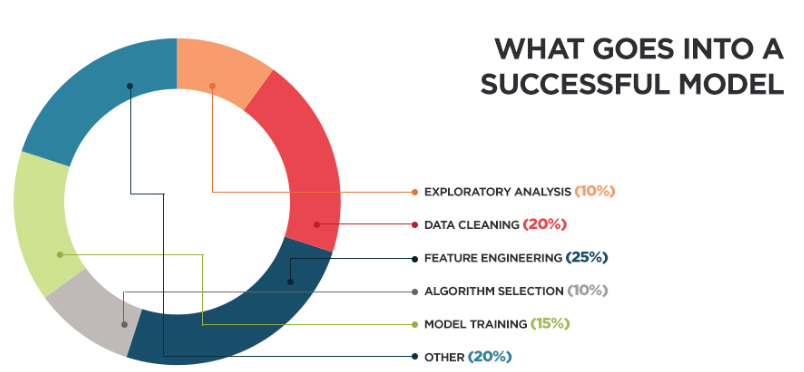
# INTRODUCTION

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to gradually improve their performance on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

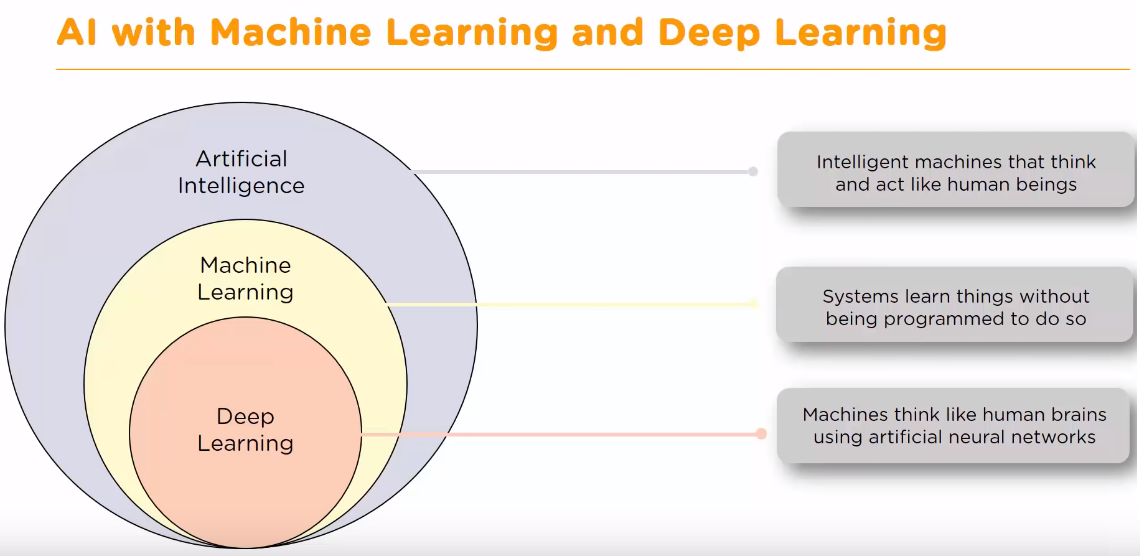


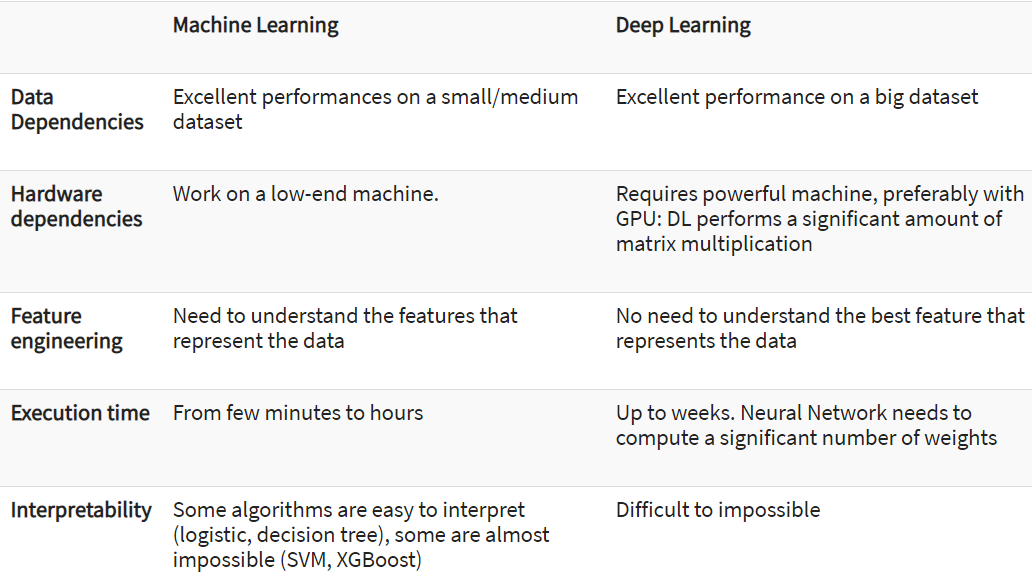
## AI vs ML vs DEEP LEARNING

Artificial Intelligence: Machines exhibiting human like behaviour.

Machine Learning: Machine learns from data / experiences for making predictions and decision making.

Deep Learning: It’s a subset of machine learning with algorithms inspired by structure of the human brain called Artificial Neural Networks.





## DATA SCIENCE COMPETENCIES

A data scient is assumed to be good at below skill.

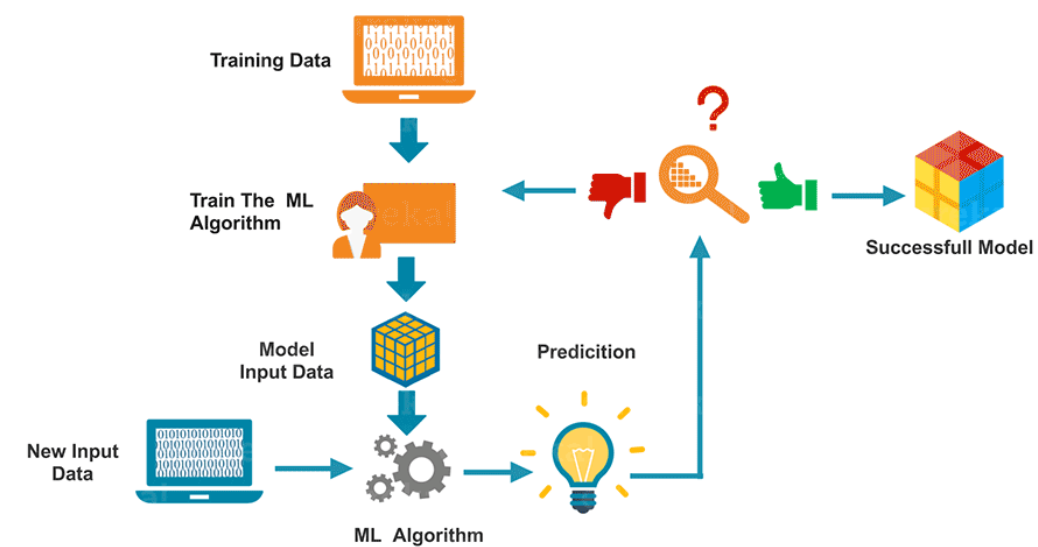
<https://datasciencedegree.wisconsin.edu/data-science-program/data-science-courses/>

## HOW DOES MACHINE LEARNING WORK?

Machine Learning algorithm is trained using a training data set to create a model. When new input data is introduced to the ML algorithm, it makes a prediction on the basis of the model.

The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed. If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with an augmented training data set.

This is just a very high-level example as there are many factors and other steps involved.



## MACHINE LEARNING PROJECT WORKFLOW

In a machine learning projects, we usually have following phases:

1. Understand the business problem or formulate the business problem you are trying to solve.
2. Collect data that is relevant for solving the business problem.
3. Understand data and get insights.
4. Create model for the business problem.
5. Test the model and check its performance.
6. Deploy the model to production.
7. Check business metrics on the model performance after deployment.
8. Keep monitoring the model and make and changes as and when required.

Rather than creating a fancy and complex model, its always desired to create a simple model which can be maintained and tuned after deployment to production.

## MACHINE LEARNING ALGORITHMS

An algorithm is a set of instructions or code stack which is used to learn from data so as to create a model / equation which can later be used for predication, classification, segmentation etc.

Hence algorithm in machine learning is used to produce an output deployable executable Model, which can be used in future to predict values.

There are different ways an algorithm can model a problem based on its interaction with the experience or environment or whatever we want to call the input data.

A learning algorithm is a method used to process data to extract patterns appropriate for application in a new situation. In particular, the goal is to adapt a system to a specific input-output transformation task.

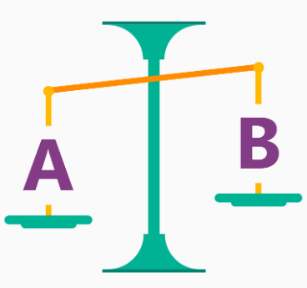
Data Science uses numbers and names (also known as categories or labels) to predict answers to questions. It might surprise you, but there are only five questions that data science answers:

1. Is this A or B?
2. Is this weird?
3. How much – or – How many?
4. How is this organized?
5. What should I do next?

Each one of these questions is answered by a separate family of machine learning methods, called algorithms.

### IS THIS A OR B?

Let's start with the question: Is this A or B? This family of algorithms is called two-class classification. It's useful for any question that has just two possible answers.



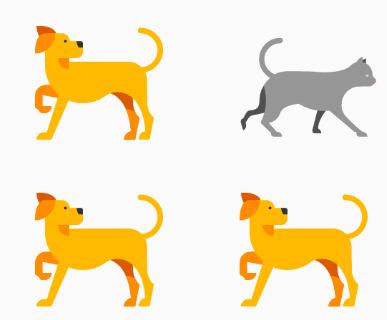
For example:

* Will this tire fail in the next 1,000 miles: Yes or no?
* Which brings in more customers: a $5 coupon or a 25% discount?

This question can also be rephrased to include more than two options: Is this A or B or C or D, etc.? This is called multiclass classification and it's useful when you have several — or several thousand — possible answers. Multiclass classification chooses the most likely one.

### IS THIS WEIRD?

The next question data science can answer is: Is this weird? This question is answered by a family of algorithms called anomaly detection.



If you have a credit card, you’ve already benefited from anomaly detection. Your credit card company analyzes your purchase patterns, so that they can alert you to possible fraud. Charges that are "weird" might be a purchase at a store where you don't normally shop or buying an unusually pricey item.

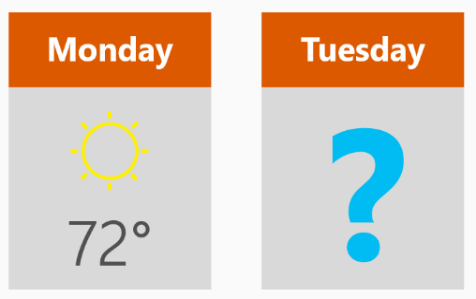
This question can be useful in lots of ways. For instance:

* If you have a car with pressure gauges, you might want to know: Is this pressure gauge reading normal?
* If you're monitoring the internet, you’d want to know: Is this message from the internet typical?

Anomaly detection flags unexpected or unusual events or behaviors. It gives clues where to look for problems.

### HOW MUCH? OR HOW MANY?

Machine learning can also predict the answer to How much? or How many? The algorithm family that answers this question is called regression.



Regression algorithms make numerical predictions, such as:

* What will the temperature be next Tuesday?
* What will my fourth quarter sales be?

They help answer any question that asks for a number.

### HOW IS THIS ORGANIZED?

Sometimes you want to understand the structure of a data set - How is this organized? For this question, you don’t have examples that you already know outcomes for.

There are a lot of ways to tease out the structure of data. One approach is clustering. It separates data into natural "clumps," for easier interpretation. With clustering, there is no one right answer.



Common examples of clustering questions are:

* Which viewers like the same types of movies?
* Which printer models fail the same way?

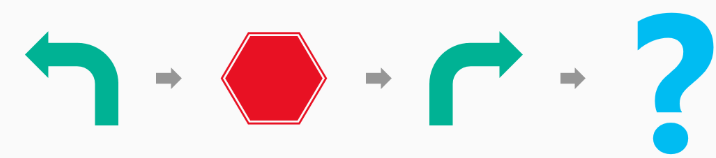
By understanding how data is organized, you can better understand - and predict - behaviors and events.

### WHAT SHOULD I DO NOW?

What should I do now? – uses a family of algorithms called reinforcement learning.

Reinforcement learning was inspired by how the brains of rats and humans respond to punishment and rewards. These algorithms learn from outcomes, and decide on the next action.

Typically, reinforcement learning is a good fit for automated systems that have to make lots of small decisions without human guidance.



Questions it answers are always about what action should be taken - usually by a machine or a robot. Examples are:

* If I'm a temperature control system for a house: Adjust the temperature or leave it where it is?
* If I'm a self-driving car: At a yellow light, brake or accelerate?
* For a robot vacuum: Keep vacuuming, or go back to the charging station?

Reinforcement learning algorithms gather data as they go, learning from trial and error.

## LAZY LEARNER ALGORITHMS

In machine learning, lazy learning is a learning method in which generalization / classification of the training data is delayed until a query is made to the system. Lazy algorithms are also known as instance-based learning algorithms.

Lazy classifiers are most useful for large, continuously changing datasets with few attributes that are commonly queried. Specifically, even if a large set of attributes exist - for example, books have a year of publication, author/s, publisher, title, edition, ISBN, selling price, etc. - recommendation queries rely on far fewer attributes - e.g., purchase or viewing co-occurrence data, and user ratings of items purchased/viewed.

## EAGER LEARNER ALGORITHMS

In machine learning, eager learning is a learning method in which system carry out generalization / classification of the training data before any query is made to the system.

## MACHINE LEARNING MODELS

The term ML model refers to the model that is created by the machine learning algorithm using the training process.

The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

### CONTINUOUS MODELS

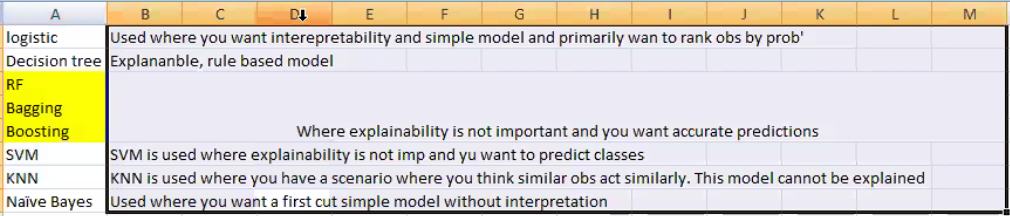
Regression algorithms which uses a continuous line / curve of fit are known as continuous models. For example, linear regression, polynomial regression.

### NON-CONTINUOUS MODELS

Regression algorithms which uses a non-continuous line / curve of fit are known as non-continuous models. For example, Decision Tree, Randomforest.

## WHEN TO USE WHICH MODEL

We can use different models based on business scenario:



## MODEL COMPLEXITY

The complexity of a relation, f(X), between input and response variables, is an important factor to consider while learning from a dataset. A simple relation is easy to interpret. For example, a linear model would look like this



It is easy to infer information from this relation and also it clearly tells how a particular feature impacts the response variable. Such models come under the category of restrictive models as they can take only a particular form, linear in this case.

But a relation may be more complex than this, for example it may be polynomial, circular, etc. These models are more flexible as they fit data points more closely can take different forms. Generally, such methods result in a higher accuracy. But this flexibility comes at the cost of interpretability, as a complex relation is harder to interpret.

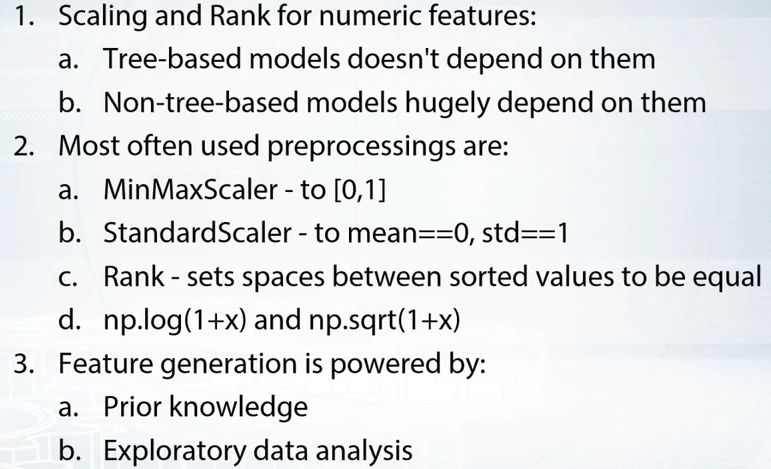
Choosing a flexible model, does not always guarantee high accuracy. It happens because our flexible statistical learning procedure is working too hard to ﬁnd patterns in the training data, and may be picking up some patterns that are just caused by random chance rather than by true properties of the unknown function.

When inference is the goal, there are clear advantages to using simple and relatively inﬂexible statistical learning methods. In some settings, however, we are only interested in prediction, and the interpretability of the predictive model is simply not of interest. This is when we use more flexible methods.

## ML PROJECT FLOW

Below is a typical project flow:

1. Importing Data
2. Cleaning Data
   1. Missing Value treatment (check for None, nan, -1, -999, mean, median)
   2. Outlier Treatment
   3. Checking for class imbalance
   4. Label Encoding
3. Feature pre-processing
   1. Standard scaling
   2. Min max scaling
   3. Rank
   4. Transformation
4. Feature generation (should be done before missing value treatment)
5. EDA
6. Feature Selection
7. Validation for test train split
8. Model Creation
9. Data Leaks
10. Metrices
11. Mean Encoding
12. Hyper parameter optimization



# IMPORTANT DEFINITIONS

Below are some important definitions one should know with respect to machine:

## BAYESIAN STATISTICS

Bayesian statistics is a theory in the field of statistics based on the Bayesian interpretation of probability where probability expresses a degree of belief in an event, which can change as new information is gathered, rather than a fixed value based upon frequency or propensity. The degree of belief may be based on prior knowledge about the event, such as the results of previous experiments, or on personal beliefs about the event.

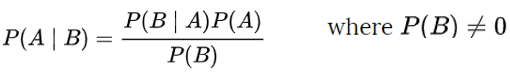
This differs from a number of other interpretations of probability, such as the frequentist interpretation that views probability as the limit of the relative frequency of an event after a large number of trials.

Bayesian statistical methods use Bayes' theorem to compute and update probabilities after obtaining new data.

In Bayesian inference, Bayes' theorem can be used to estimate the parameters of a probability distribution or statistical model. Since Bayesian statistics treats probability as a degree of belief, Bayes' theorem can directly assign a probability distribution that quantifies the belief to the parameter or set of parameters.

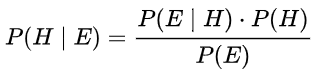
## BAYES' THEOREM

Bayes' theorem is a fundamental theorem in Bayesian statistics, as it is used by Bayesian methods to update probabilities, which are degrees of belief, after obtaining new data. Given two events A and B, the conditional probability of A given that B is true is expressed as follows:



* In the above equation, A usually represents a proposition (such as the statement that a coin lands on heads fifty percent of the time)
* B represents the evidence, or new data that is to be taken into account (such as the result of a series of coin flips).
* P(A) is the prior probability of A which expresses one's beliefs about A before evidence is taken into account. The prior probability may also quantify prior knowledge or information about A.
* P(B|A) is the likelihood function, which can be interpreted as the probability of the evidence B given that A is true. The likelihood quantifies the extent to which the evidence B supports the proposition A.
* P(A|B) is the posterior probability, the probability of the proposition A after taking the evidence B into account.
* Essentially, Bayes' theorem updates one's prior beliefs P(A) after considering the new evidence B.

Other form is



where

* H stands for any hypothesis whose probability may be affected by data (called evidence below). Often there are competing hypotheses, and the task is to determine which is the most probable.
* P(H), the prior probability, is the estimate of the probability of the hypothesis H before the data E, the current evidence, is observed.
* The evidence E corresponds to new data that were not used in computing the prior probability.
* P(H|E), the posterior probability, is the probability of H given E, i.e., after E is observed. This is what we want to know: the probability of a hypothesis given the observed evidence.
* P(E|H) is the probability of observing E given H, and is called the likelihood. As a function of E with H fixed, it indicates the compatibility of the evidence with the given hypothesis. The likelihood function is a function of the evidence, E, while the posterior probability is a function of the hypothesis, H.
* P(E) is sometimes termed the marginal likelihood or "model evidence". This factor is the same for all possible hypotheses being considered (as is evident from the fact that the hypothesis H does not appear anywhere in the symbol, unlike for all the other factors), so this factor does not enter into determining the relative probabilities of different hypotheses.

## BAYESIAN INFERENCE

Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available.

* The prior distribution is the distribution of the parameter(s) before any data is observed. The prior distribution might not be easily determined. In this case, we can use the Jeffreys prior to obtain the posterior distribution before updating them with newer observations.
* The sampling distribution is the distribution of the observed data conditional on its parameters. This is also termed the likelihood.
* The posterior distribution is the distribution of the parameter(s) after taking into account the observed data.

## BAYESIAN PREDICTION

The posterior predictive distribution is the distribution of a new data point, marginalized over the posterior.

## STATISTICAL MODEL

A statistical model is a mathematical model that embodies a set of statistical assumptions concerning the generation of sample data (and similar data from a larger population). A statistical model represents, often in considerably idealized form, the data-generating process.[1]

A statistical model is usually specified as a mathematical relationship between one or more random variables and other non-random variables.

All statistical hypothesis tests and all statistical estimators are derived via statistical models. More generally, statistical models are part of the foundation of statistical inference.

## STATISTICAL INFERENCE

Statistical inference is the process of using data analysis to deduce properties of an underlying probability distribution.[1] Inferential statistical analysis infers properties of a population, for example by testing hypotheses and deriving estimates. It is assumed that the observed data set is sampled from a larger population.

Inferential statistics can be contrasted with descriptive statistics. Descriptive statistics is solely concerned with properties of the observed data, and it does not rest on the assumption that the data come from a larger population.

## STOCHASTIC MODELLING

The word stochastic is an adjective in English that describes something that was randomly determined.

"Stochastic" means being or having a random variable. A stochastic model is a tool for estimating probability distributions of potential outcomes by allowing for random variation in one or more inputs over time.

The random variation is usually based on fluctuations observed in historical data for a selected period using standard time-series techniques. Distributions of potential outcomes are derived from a large number of simulations (stochastic projections) which reflect the random variation in the input(s).

Its application initially started in physics. It is now being applied in engineering, life sciences, social sciences, and finance.

## MEAN SQUARED ERROR (QUALITY OF FIT)

To quantify the extent to which the predicted response value for a given observation is close to the true response value for that observation, the most commonly-used measure in regression setting is the mean squared error (MSE).

As the name goes, it is the mean of square of the errors or differences in predictions and observed values for all inputs. It is known as training MSE if calculated using training data, and test MSE if calculated using testing data.

The expected test MSE, for a given value x0, can always be decomposed into the sum of three fundamental quantities:

* The variance for f(x0)
* The squared bias for f(x0)
* Irreducible error term

## BIAS

Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. So, if the true relation is complex and you try to use linear regression, then it will undoubtedly result in some bias in the estimation of f(X). No matter how many observations you have, it is impossible to produce an accurate prediction if you are using a restrictive/ simple algorithm, when the true relation is highly complex.

## CORRELATION

Correlation between two variables signifies the strength of linear relationship between them. It shows how change in one variable cause change in other.

## COVARIANCE

Covariance between two variables signifies how both variables change together.

## VARIANCE

Variance refers to the amount by which your estimate of f(X) would change if we estimated it using a diﬀerent training data set.

Since the training data is used to ﬁt the statistical learning method, diﬀerent training data sets will result in a diﬀerent estimation. But ideally the estimate for f(X) should not vary too much between training sets. However, if a method has high variance then small changes in the training data can result in large changes in f(X).

## STANDARD ERROR COEFFICIENT

The standard error of the coefficient measures the precision of the estimates. Lower values indicate more precise estimates.

With increase in SE the confidence interval becomes wider leading to less precise estimates of slope parameters.

With decrease in SE, confidence intervals and prediction intervals become narrower. Also, lower standard errors would cause the associated p-values to be lower than actual. This will make us incorrectly conclude a parameter to be statistically significant.

## FEATURE

Features are individual independent variables that act as the input in your system. In machine learning feature means a property of the training data set. Feature is a column of data in the data set.

For example, if you're trying to predict the type of pet someone will choose, your input features might include age, home region, family income, etc.

## LABEL

The output that we get from the model after training it, is called label.

Once you've trained your model, you will give it sets of new input containing those features; it will return the predicted "label" (pet type like cat, dog etc) for that person.

Thus, we obtain labels as output when provided with features as input. Labels are not associated with unsupervised learning.

## UNLABELED DATA

Typically, unlabeled data consists of samples of natural or human-created artifacts that you can obtain relatively easily from the world. Some examples of unlabeled data might include photos, audio recordings, videos, news articles, tweets, x-rays (if you were working on a medical application), etc. There is no "explanation" for each piece of unlabeled data -- it just contains the data, and nothing else.

## LABELED DATA

Labeled data typically takes a set of unlabeled data and augments each piece of that unlabeled data with some sort of meaningful "tag," "label," or "class" that is somehow informative or desirable to know. For example, labels for the above types of unlabeled data might be whether this photo contains a horse or a cow, which words were uttered in this audio recording, what type of action is being performed in this video, what the topic of this news article is, what the overall sentiment of this tweet is, whether the dot in this x-ray is a tumor, etc.

## COLLINEARITY

Collinearity is a linear association between two explanatory variables.

## MULTICOLLINEARITY

Multicollinearity in a multiple regression model are highly linearly related associations between two or more explanatory variables.

Multicollinearity is defined correlation among the independent variables. Multicollinearity occurs when independent variables in a regression model are correlated. This correlation is a problem because independent variables should be independent. If the degree of correlation between variables is high enough, it can cause problems when you fit the model and interpret the results. It is therefore a type of disturbance in the data, and if present in the data the statistical inferences made about the data may not be reliable.

There are certain reasons why multicollinearity occurs:

* It is caused by an inaccurate use of dummy variables.
* It is caused by the inclusion of a variable which is computed from other variables in the data set.
* Multicollinearity can also result from the repetition of the same kind of variable.
* Generally, occurs when the variables are highly correlated to each other.

Multicollinearity can result in several problems. These problems are as follows:

* The partial regression coefficient due to multicollinearity may not be estimated precisely. **The standard errors are likely to be high.**
* Multicollinearity results in a change in the signs as well as in the magnitudes of the partial regression coefficients from one sample to another sample.
* Multicollinearity makes it tedious to assess the relative importance of the independent variables in explaining the variation caused by the dependent variable.
* Lower precision, switched signs, and a lack of statistical significance are typical problems associated with multicollinearity.

There are certain signals which help the researcher to detect the degree of multicollinearity.

* One such signal is if the individual outcome of a statistic is not significant but the overall outcome of the statistic is significant. In this instance, the researcher might get a mix of significant and insignificant results that show the presence of multicollinearity. Suppose the researcher, after dividing the sample into two parts, finds that the coefficients of the sample differ drastically. This indicates the presence of multicollinearity. This means that the coefficients are unstable due to the presence of multicollinearity.
* Suppose the researcher observes drastic change in the model by simply adding or dropping some variable. This also indicates that multicollinearity is present in the data.
* Multicollinearity can also be detected with the help of tolerance and its reciprocal, called variance inflation factor (VIF). If the value of tolerance is less than 0.2 or 0.1 and, simultaneously, the value of VIF 10 and above, then the multicollinearity is problematic.

## REGRESSION COEFFICIENT

The partial regression coefficient is also called regression coefficient, regression weight, partial regression weight, slope coefficient or partial slope coefficient.

It is used in the context of multiple linear regression (MLR) analysis and gives the amount by which the dependent variable (DV) increases when one independent variable (IV) is increased by one unit and all the other independent variables are held constant.

This coefficient is called partial because its value depends, in general, upon the other independent variables. Specifically, the value of the partial coefficient for one independent variable will vary, in general, depending upon the other independent variables included in the regression equation.



Here b0 is called the intercept and b1 is coefficient of X1 and so on.

## INTERACTION TERM / VARIABLE

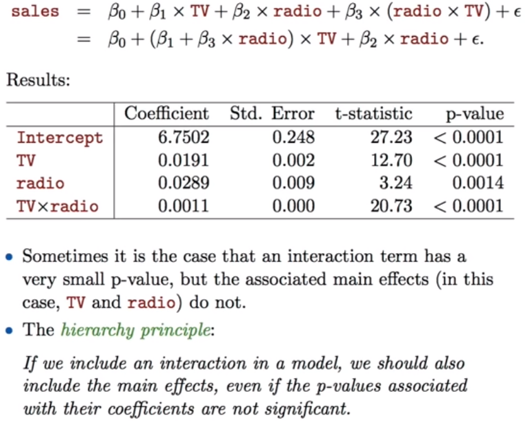
An interaction variable or interaction feature is a variable constructed from an original set of variables to try to represent either all of the interaction present or some part of it.

In exploratory statistical analyses it is common to use products of original variables as the basis of testing whether interaction is present with the possibility of substituting other more realistic interaction variables at a later stage. When there are more than two explanatory variables, several interaction variables are constructed, with pairwise-products representing pairwise-interactions and higher order products representing higher order interactions.



is an example of a model with an interaction between variables x1 and x2 (main terms) namely x1\*x2

Example 2:



## BACKWARD ELIMINATION

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

## 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.

## RECURSIVE FEATURE ELIMINATION

It is a greedy optimization algorithm which 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.

## PARSIMONY

Parsimony means the simplicity of a model to be accurate. At times models with simple parameters are more accurate at prediction than models with many parameters. We use simple models to avoid over fitting.

## INFORMATION

Information is nothing but useful / important patterns in the dataset that the learning algorithm tried to capture. In stats information is directly proportional to variance. More the variance in a variable more is the information it carries.

## REGULARIZATION

The word regularize means to make things regular or acceptable. This is exactly why we use it for. Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.

Consider the training dataset comprising of independent variables X = (x1, x2…. xn) and the corresponding target variables t = (t1, t2,… tn). X are random variables lying uniformly between [0,1]. The target dataset ‘t’ is obtained by substituting the value of X into the function f(x) and then adding some Gaussian noise into it.

Now, our goal is to find patterns in this underlying dataset and generalize it to predict the corresponding target value for some new values of ‘x’. The problem here is, our target dataset is inflicted with some random noise. So, it will be difficult to find the inlying function f(x) in the training data. So, how do we solve it?

Regularization is a form of regression, that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.

A simple relation for linear regression looks like this. Here Y represents the learned relation and β represents the coefficient estimates for different variables or predictors(X).



## MODEL PARAMETER & HYPER PARAMETERS

A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data.

* They are required by the model when making predictions.
* Their values define the skill of the model on your problem.
* They are estimated or learned from data.
* They are often not set manually by the practitioner.
* They are often saved as part of the learned model.

Parameters are key to machine learning algorithms. They are the part of the model that is learned from historical training data.

## SPARSE VS DENSE

In numerical analysis and computer science, a sparse matrix or sparse array is a matrix in which most of the elements are zero. By contrast, if most of the elements are nonzero, then the matrix is considered dense.

## CLOSED-FORM EXPRESSION

A closed-form expression is a mathematical expression that can be evaluated in a finite number of operations. It may contain constants, variables, certain "well-known" operations (e.g., + − × ÷), and functions (e.g., nth root, exponent, logarithm, trigonometric functions, and inverse hyperbolic functions), but usually no limit.

## CLOSED-FORM SOLUTION

A closed-form solution is a general solution to a problem in the form of a closed-form expression.

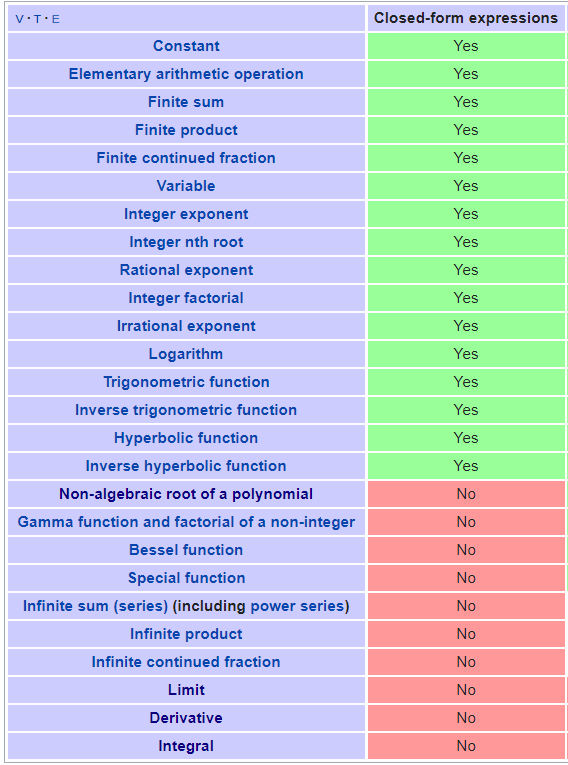
It is for this reason that closed-form expressions and solutions are typically restricted to having the following operations:

* Addition, subtraction, multiplication, and division.
* Exponents and logarithms.
* Trigonometric functions and inverse trigonometric functions.
* Summation of finite numbers.

## IN SAMPLE VS OUT OF SAMPLE PREDICTION

In-sample forecast is the process of formally evaluating the predictive capabilities of the models developed using observed data to see how effective the algorithms are in reproducing data. It is kind of similar to a training set in a machine learning algorithm and the out-of-sample is similar to the test set.

If you are forecasting for an observation that was part of the training set - it is in-sample forecast. If you are forecasting for an observation that was not part test set - it is out-of-sample forecast.



## **ENTROPY**

Entropy is a measure of randomness or uncertainty.

We would say that an unbiased coin has a high entropy, because the uncertainty of the value of X is the highest possible. Finding out the value of the coin flip tells you lots of information you didn’t already know.

A biased coin has a low entropy, because the uncertainty of X is the lowest possible. Finding out the value of the coin flip doesn’t tell you anything, because you already knew that it would land heads.

Entropy is the theoretical limit of a lossless compression of training data. Many machine learning algorithms work with a related concept known as KL divergence, which is a measure of how much one distribution differs from another.

In a classification problem, you are trying to “force” the predicted output distribution to be as close to the true output distribution as much as possible. One way to do this is by minimizing the KL divergence (relative entropy) between the two distributions. If you do this, you derive the cross-entropy cost function, which is the default cost function for classification problems.

## LOSS FUNCTION

Loss function is usually a function defined on a data point, prediction and label, and measures the penalty. For example

* Square loss, used in linear regression
* Hinge loss, used in SVM

## COST FUNCTION

Cost function is usually more general. It might be a sum of loss functions over your training set plus some model complexity penalty (regularization). For example:

* Mean Squared Error
* SVM cost function

## OBJECTIVE FUNCTION

Objective function is the most general term for any function that you optimize during training. For example, a probability of generating training set in maximum likelihood approach is a well-defined objective function

* Least squares for regression
* MLE for classification

A loss function is a part of a cost function which is a type of an objective function.

The loss function computes the error for a single training example whereas the cost function is the average of the loss functions of the entire training set. And objective function is used either to minimize or maximize the objective based on the type of learning algorithm.

## LINK FUNCTION

Generalized linear models and generalized linear mixed models are called generalized linear because they connect a model’s outcome to its predictors in a linear way. The function used to make this connection is called a link function.

For example, Poisson regression (commonly used for outcomes that are counts) makes use of a natural log link function as follows:



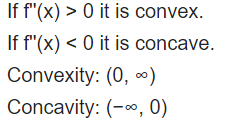
Clearly, there is not a direct linear relationship of the x variables to the average count, but there is a “sort of linear” relationship happening: a function of the mean of y is related to a linear combination of x variables. In other words, the linear model has now been generalized to a bigger type of situation.

The key thing to understand is that the natural log link function is a function of the mean of y, not the y values themselves.

Further reading: <http://www.statsoft.com/Textbook/Nonlinear-Estimation>

## CONVEX AND CONCAVE FUNCTIONS

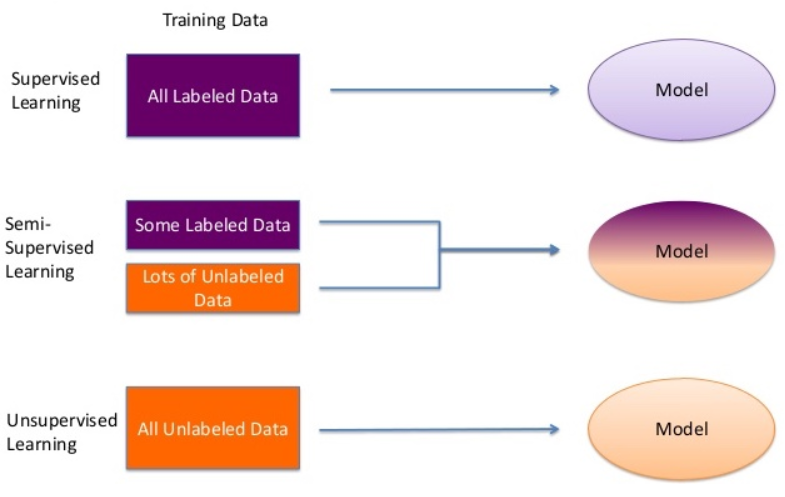
For a twice-differentiable function f, if the second derivative, f ''(x), is positive (or, if the acceleration is positive), then the graph is convex (or concave upward); if the second derivative is negative, then the graph is concave (or concave downward).



# TYPE OF MACHINE LEARNING ALGORITHMS

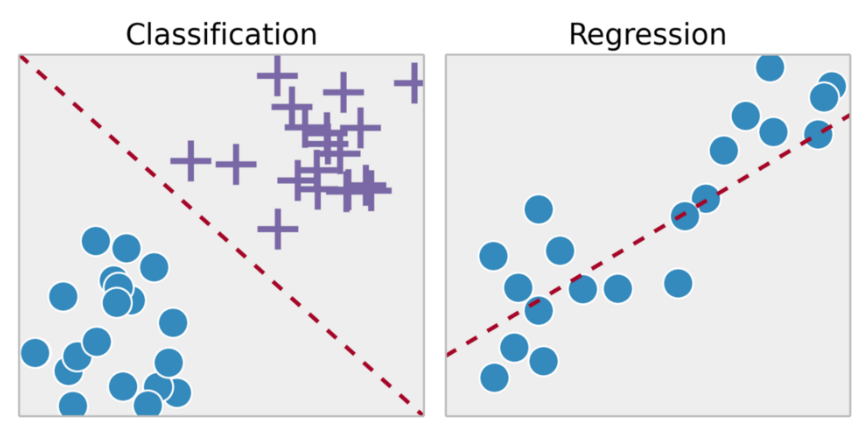
The machine learning algorithms can be broadly classified into four types:

* Supervised Learning
* Unsupervised Learning
* Semi Supervised Learning
* Reinforcement Learning



## SUPERVISED LEARNING

The outcome or output for the given input is known before itself and the machine must be able to map or assign the given input to the output.



KEY POINTS:

* Regression and classification problems are mainly solved here.
* Labelled data is used for training here.
* Popular Regression Algorithms:
  1. Linear Regression
* Popular Classification Algorithms:
  1. Logistic Regression
  2. Support Vector Machines (SVM)
  3. Neural Networks
  4. Decision Trees
  5. Naive Bayes
  6. Nearest Neighbour
* It is mainly used in Predicting Modelling.

### BINARY CLASSIFICATION MODEL

ML models for binary classification problems predict a binary outcome (one of two possible classes). Examples of Binary Classification Problems

* "Is this email spam or not spam?"
* "Will the customer buy this product?"
* "Is this product a book or a farm animal?"
* "Is this review written by a customer or a robot?"

### MULTICLASS CLASSIFICATION MODEL

ML models for multiclass classification problems allow you to generate predictions for multiple classes (predict one of more than two outcomes). Examples of Multiclass Problems

* "Is this product a book, movie, or clothing?"
* "Is this movie a romantic comedy, documentary, or thriller?"
* "Which category of products is most interesting to this customer?"

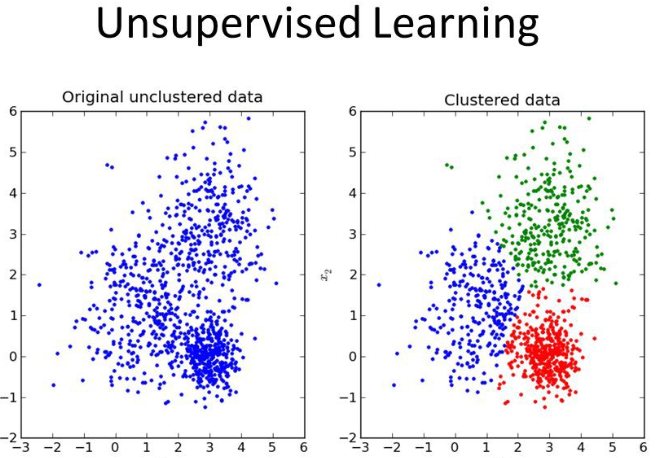
### REGRESSION MODEL

ML models for regression problems predict a numeric value. Examples of Regression Problems

* "What will the temperature be in Seattle tomorrow?"
* "For this product, how many units will sell?"
* "What price will this house sell for?"

## UNSUPERVISED LEARNING

The outcome or output for the given inputs is unknown. Here input data is given and the model is run on it. The image or the input given are grouped together here and insights on the inputs can be found here. E.g. customer grouping / segmentation or profiling.

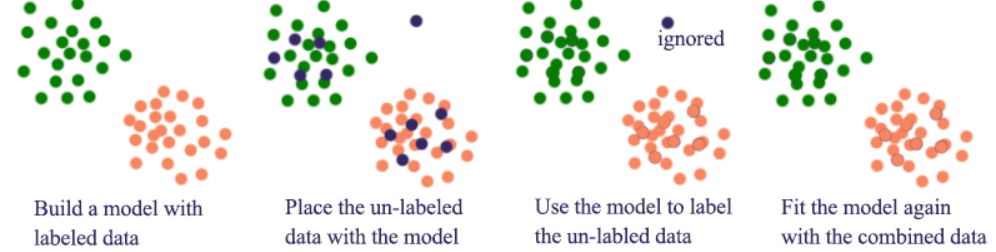


KEY POINTS:

* It is used for Clustering problems(grouping), Anomaly Detection (in banks for unusual transactions) where there is a need for finding relationships among the data given.
* Unlabeled data is used in unsupervised learning.
* Popular Algorithms
  1. k-means clustering
  2. Association rule.
* It is mainly used in Descriptive Modelling.

## SEMI-SUPERVISED LEARNING

It is in-between that of Supervised and Unsupervised Learning. Where the combination is used to produce the desired results and it is the most important in real-world scenarios where all the data available are a combination of labelled and unlabeled data.

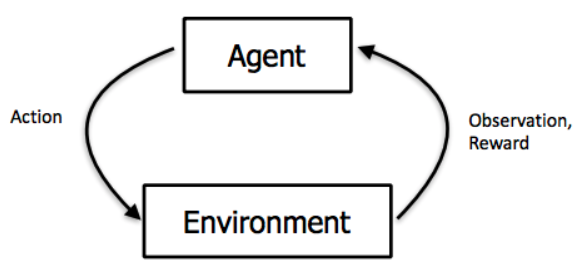


## REINFORCED LEARNING

The machine is exposed to an environment where it gets trained by trial and error method, here it is trained to make a much specific decision. The machine learns from past experience and tries to capture the best possible knowledge to make accurate decisions based on the feedback received.

It is similar to Supervised learning but the output/ outcome is not known instantly. For example, we want to go from place a to place b, we have multiple options that we can take. So, to reach from point a to point b the next step depends on our previous selection. This is an example of reinforcement learning.

Another example is automated game play. For example, an online player playing chess with an online game agent. Here the steps taken by the game agent will depend on the players move.



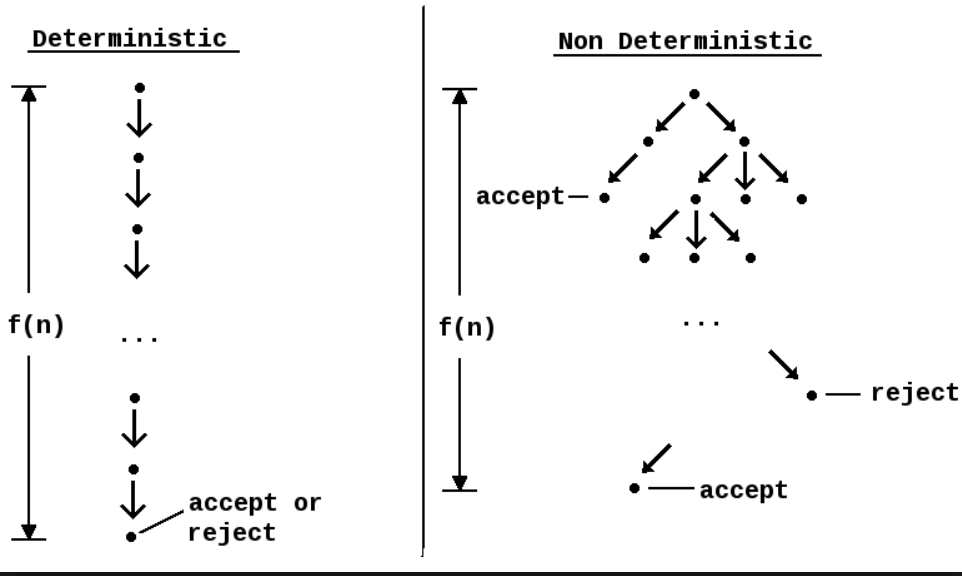
KEY POINTS:

* Basic reinforcement is modelled as Markov Decision Process
* The most popular algorithms used here is Q-Learning, Deep Adversarial Networks.
* Its practical applications include computer playing board games such as chess and GO, Self-driving cars also use this learning.



## DETERMINISTIC VS NONDETERMINISTIC ALGORITHMS

Deterministic algorithm returns same result for same input where as non-deterministic algorithm returns different result for same input.



# TOP ALGORITHMS IN ML

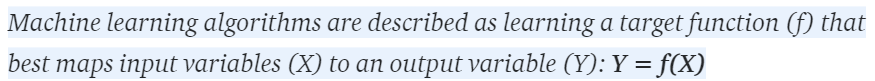
In machine learning, there’s something called the “No Free Lunch” theorem. In a nutshell, it states that no one algorithm works best for every problem, and it’s especially relevant for supervised learning (i.e. predictive modeling).

For example, you can’t say that neural networks are always better than decision trees or vice-versa. There are many factors at play, such as the size and structure of your dataset.

As a result, you should try many different algorithms for your problem, while using a hold-out “validation set” of data to evaluate performance and select the winner.

Of course, the algorithms you try must be appropriate for your problem, which is where picking the right machine learning task comes in.

A common principle that underlies all supervised machine learning algorithms for predictive modeling.



## LINEAR REGRESSION

Linear regression is perhaps one of the most well-known and well-understood algorithms in statistics and machine learning.

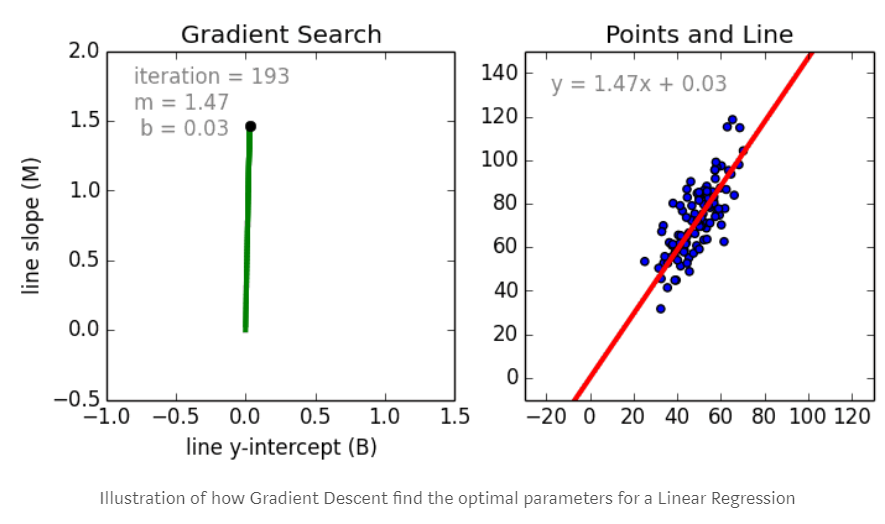
Predictive modeling is primarily concerned with minimizing the error of a model or making the most accurate predictions possible, at the expense of explainability.

The representation of linear regression is an equation that describes a line that best fits the relationship between the input variables (x) and the output variables (y), by finding specific weightings for the input variables called coefficients (B).

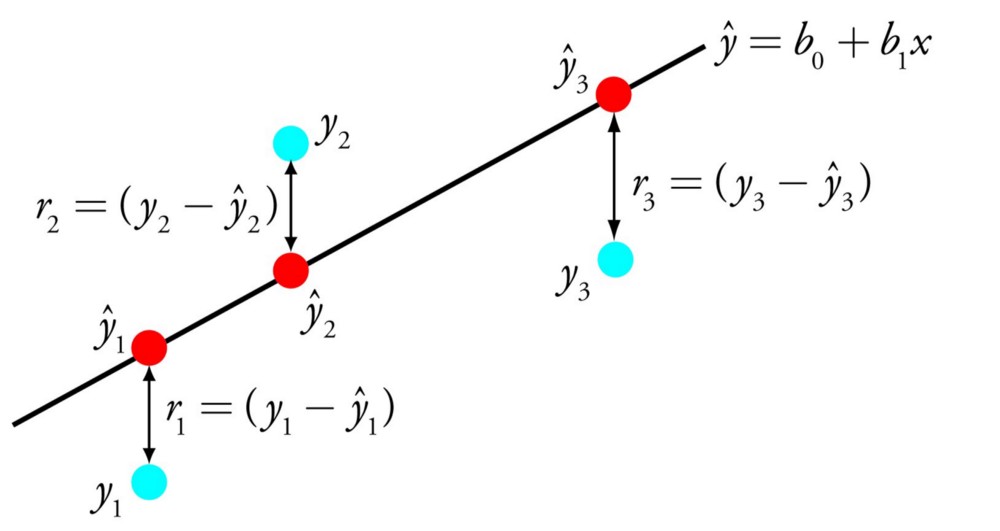


We will predict y given the input x and the goal of the linear regression learning algorithm is to find the values for the coefficients B0 and B1.

It is quite easy to understand as we are simply weighting the importance of each feature variable X\_n using the coefficient weights a\_n. We determine these weights a\_n and the intercept b using a Stochastic Gradient Descent (SGD).



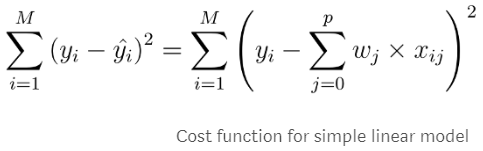
Below figure shows the residuals in linear regression

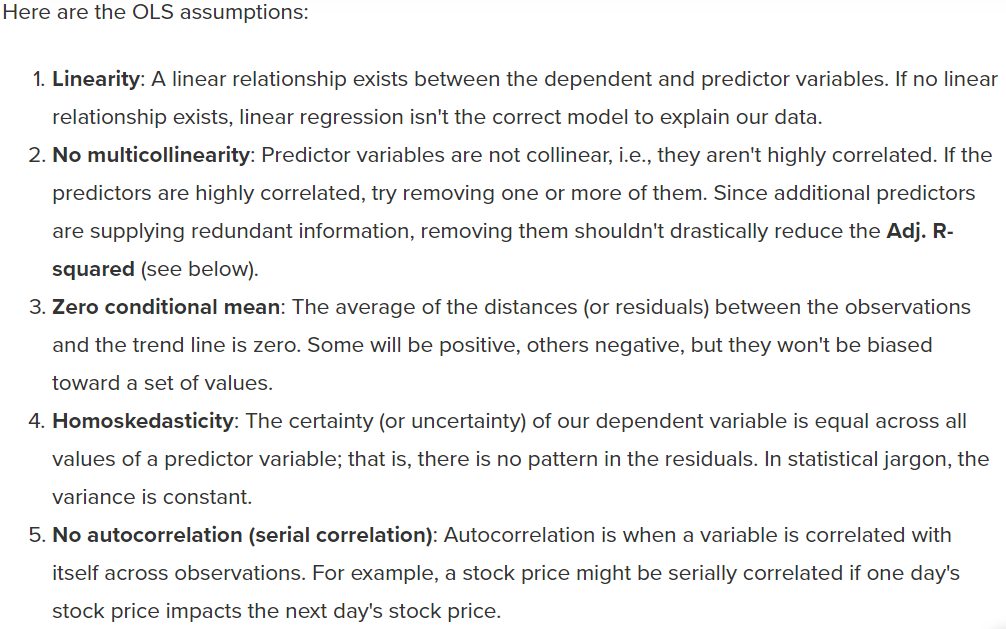


A few key points about Linear Regression:

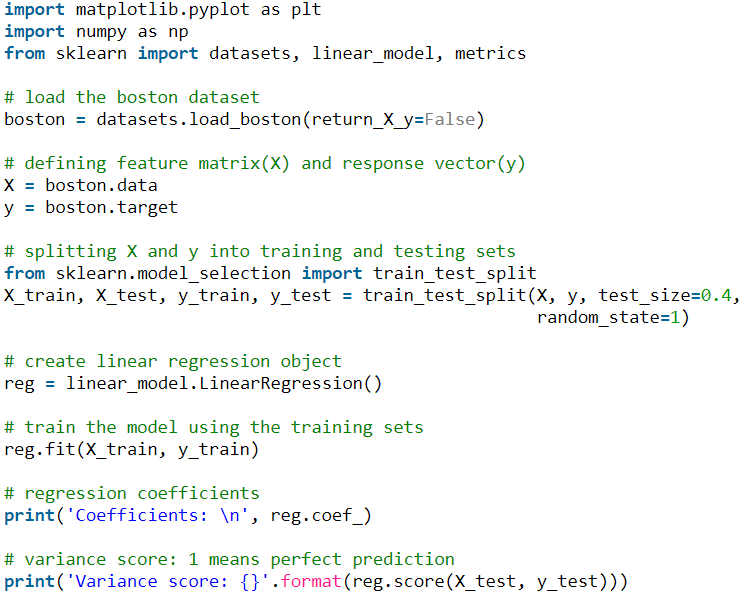
* Fast and easy to model and is particularly useful when the relationship to be modeled is not extremely complex and if you don’t have a lot of data.
* Very intuitive to understand and interpret.
* Linear Regression is very sensitive to outliers.

The cost function for linear model is defined below

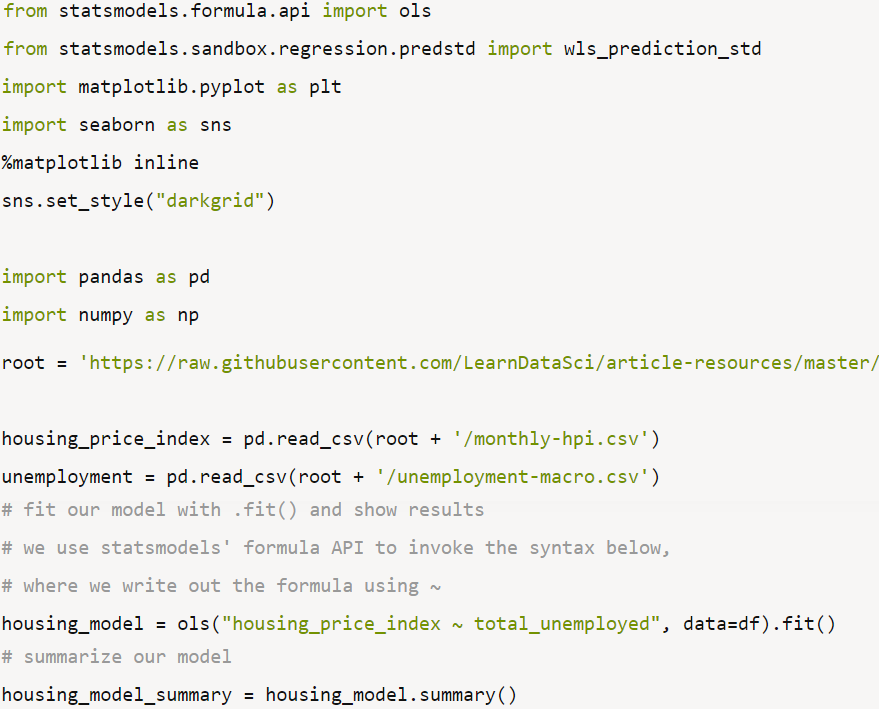




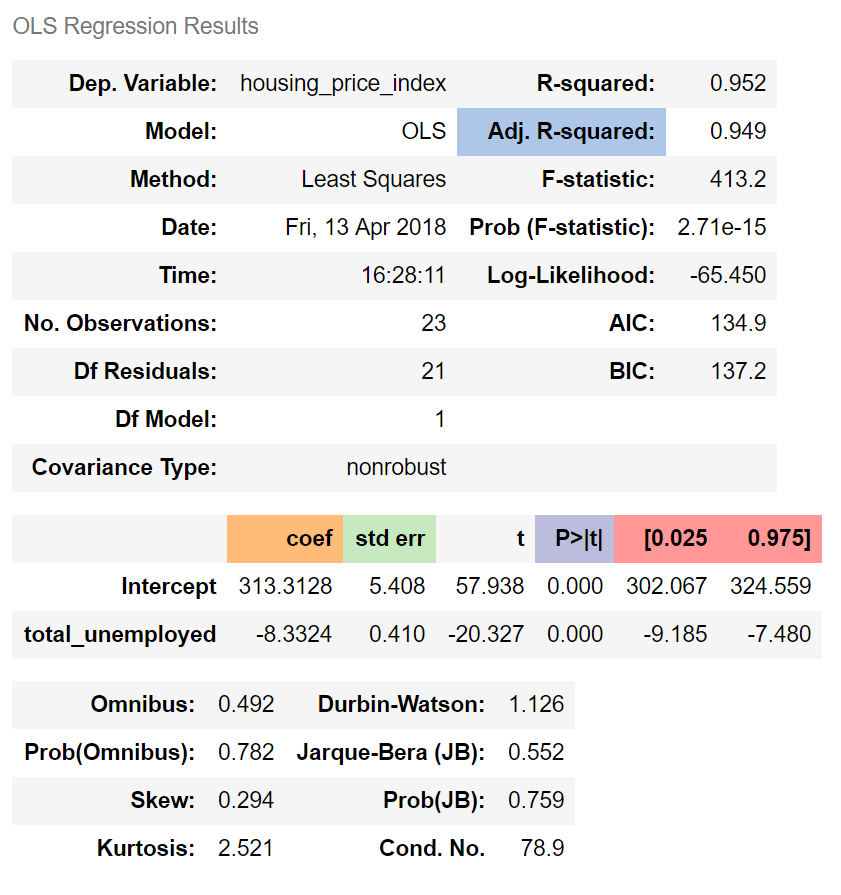
We can perform linear regression using sklearn linear\_model module:



We can also use statsmodels module to perform linear regression using below code:



We can use statsmodels.api to develop and test various algorithms for our model as its gives detailed output report as shown below which is very useful in manual feature selection and checks for multicollinearity, homoscedasticity. We can then build the final model using sklearn.

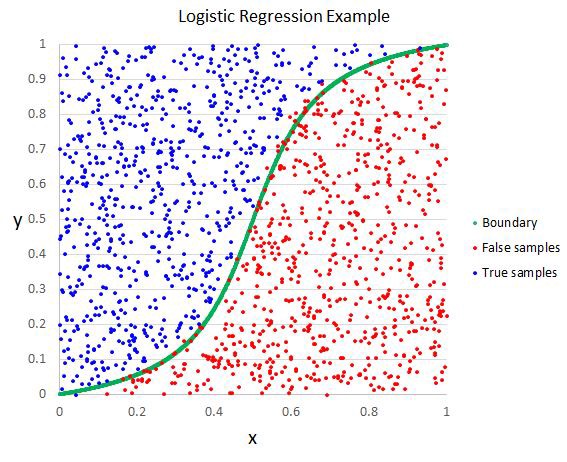


## LOGISTIC REGRESSION

It is used for binary classification problems (problems with two class values).

Logistic regression is like linear regression in that the goal is to find the values for the coefficients that weight each input variable. Unlike linear regression, the prediction for the output is transformed using a non-linear function called the logistic function.

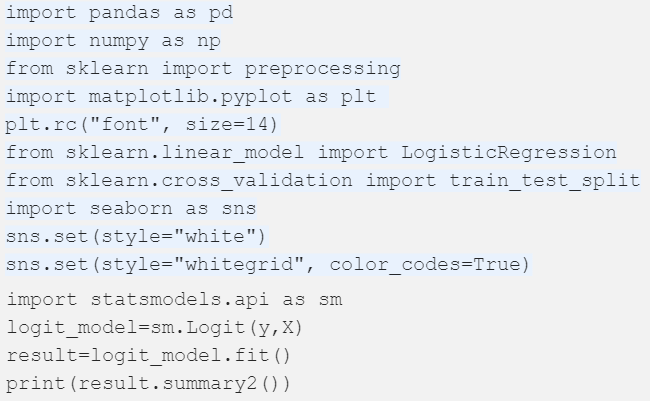
The logistic function (also known as logit or sigmoid function or class function) looks like a big S and will transform any value into the range 0 to 1. This is useful because we can apply a rule to the output of the logistic function to snap values to 0 and 1 (e.g. IF less than 0.5 then output 1) and predict a class value.

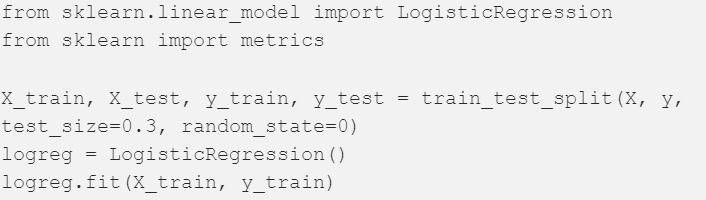


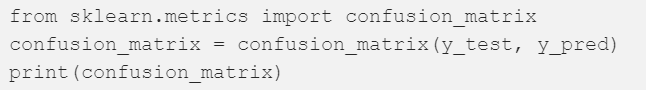
The predictions made by logistic regression can also be used as the probability of a given data instance belonging to class 0 or class 1. This can be useful for problems where you need to give more validation / justification for a prediction.

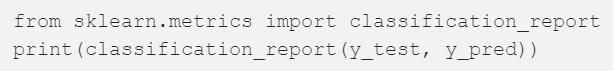
Like linear regression, logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. It’s a fast model to learn and effective on binary classification problems.

We can implement Logistic regression in Python using below code:

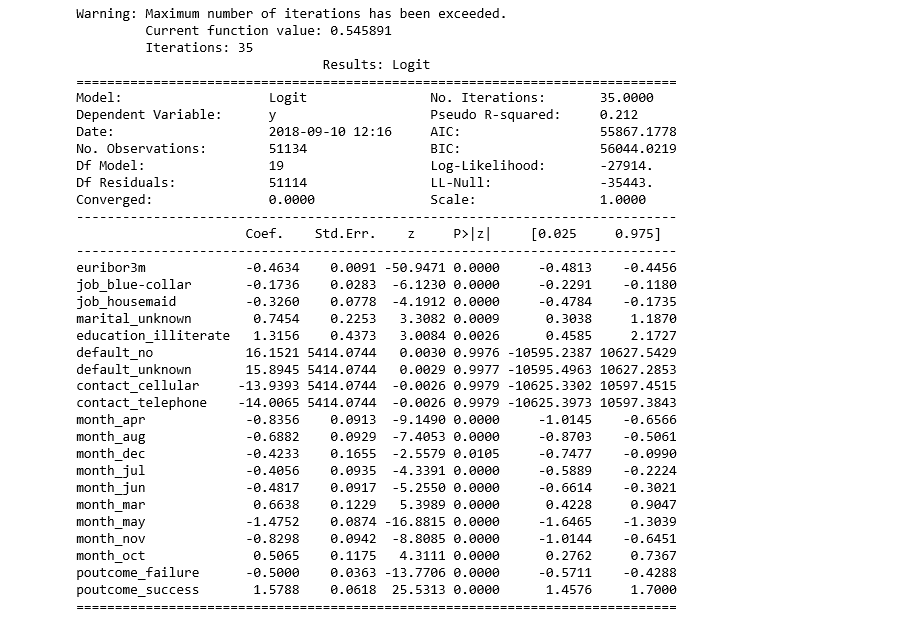




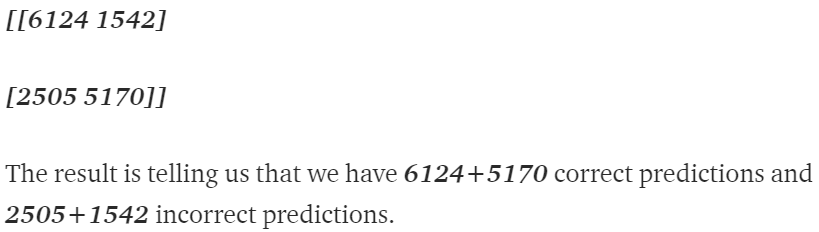




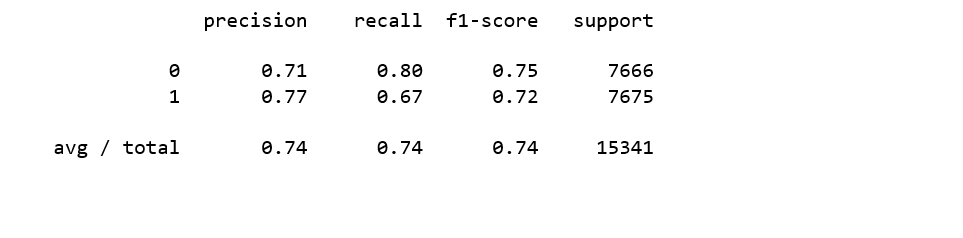
The output from statsmodels.api is shown below:



The outputs from sklearn are shown below (confusion matrix):



PRECISION, RECALL, F-MEASURE AND SUPPORT



## LINEAR DISCRIMINANT ANALYSIS

Logistic Regression is a classification algorithm traditionally limited to only two-class classification problems. If you have more than two classes then the Linear Discriminant Analysis algorithm is the preferred linear classification technique.

The representation of LDA is pretty straight forward. It consists of statistical properties of your data, calculated for each class. For a single input variable this includes:

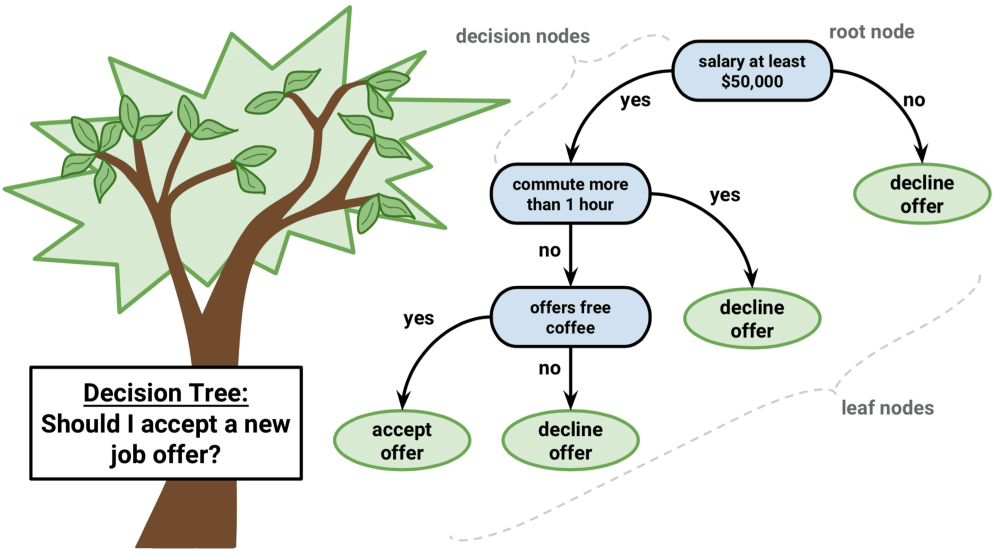
* The mean value for each class.
* The variance calculated across all classes.

Predictions are made by calculating a discriminate value for each class and making a prediction for the class with the largest value. The technique assumes that the data has a Gaussian distribution (bell curve), so it is a good idea to remove outliers from your data beforehand. It’s a simple and powerful method for classification predictive modeling problems.

## CLASSIFICATION AND REGRESSION TREES

Decision Trees are an important type of algorithm for predictive modeling machine learning.

The representation of the decision tree model is a binary tree. This is your binary tree from algorithms and data structures, nothing too fancy. Each node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric).



The leaf nodes of the tree contain an output variable (y) which is used to make a prediction. Predictions are made by walking the splits of the tree until arriving at a leaf node and output the class value at that leaf node.

Trees are fast to learn and very fast for making predictions. They are also often accurate for a broad range of problems and do not require any special preparation for your data.

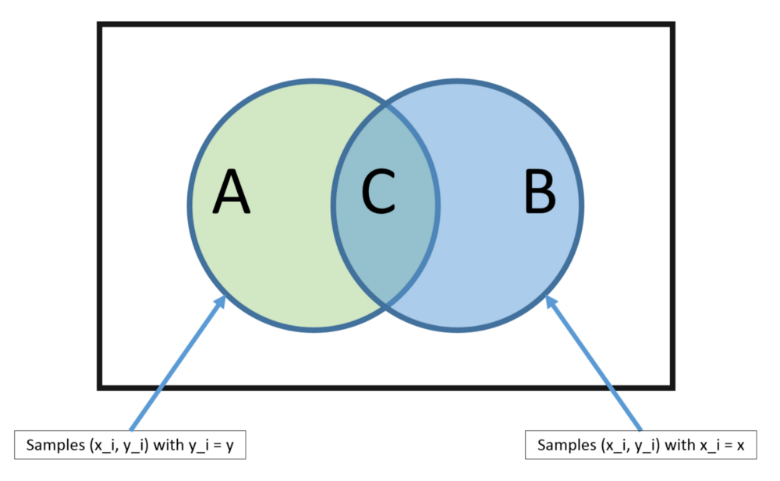
## NAIVE BAYES

Naive Bayes is a simple but surprisingly powerful algorithm for predictive modeling.

The model is comprised of two types of probabilities that can be calculated directly from your training data:

1. The probability of each class; and
2. The conditional probability for each class given each x value.

Once calculated, the probability model can be used to make predictions for new data using Bayes Theorem. When your data is real-valued it is common to assume a Gaussian distribution (bell curve) so that you can easily estimate these probabilities.



Naive Bayes is called naive because it assumes that each input variable is independent. This is a strong assumption and unrealistic for real data, nevertheless, the technique is very effective on a large range of complex problems.

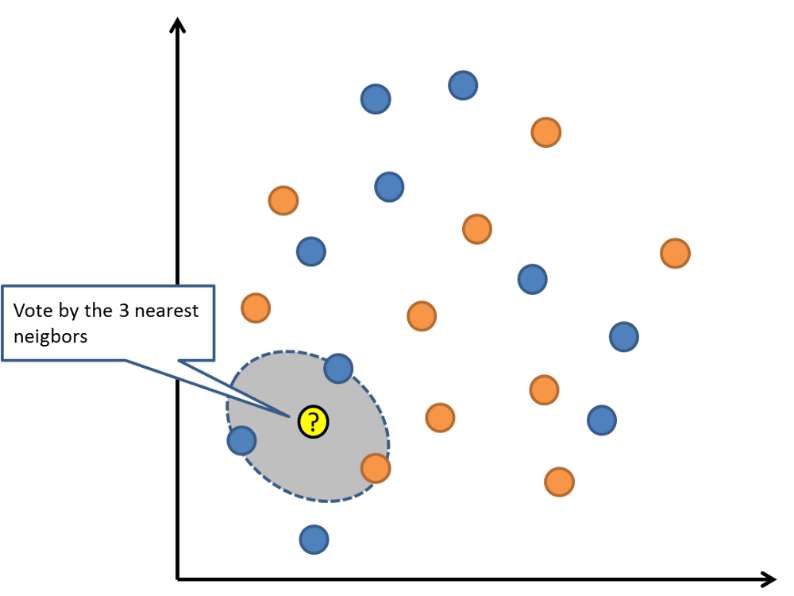
## K-NEAREST NEIGHBOURS

The KNN algorithm is very simple and very effective. The model representation for KNN is the entire training dataset.

Predictions are made for a new data point by searching through the entire training set for the K most similar instances (the neighbours) and summarizing the output variable for those K instances.

For regression problems, this might be the mean output variable, for classification problems this might be the mode (or most common) class value.

The trick is in how to determine the similarity between the data instances. The simplest technique if your attributes are all of the same scale (all in inches for example) is to use the Euclidean distance, a number you can calculate directly based on the differences between each input variable.



KNN can require a lot of memory or space to store all of the data, but only performs a calculation (or learn) when a prediction is needed, just in time. You can also update and curate your training instances over time to keep predictions accurate.

The idea of distance or closeness can break down in very high dimensions (lots of input variables) which can negatively affect the performance of the algorithm on your problem. This is called the curse of dimensionality. It suggests you only use those input variables that are most relevant to predicting the output variable.

## SUPPORT VECTOR MACHINES

Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms.

A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1. In two-dimensions, you can visualize this as a line and let’s assume that all of our input points can be completely separated by this line. The SVM learning algorithm finds the coefficients that results in the best separation of the classes by the hyperplane.



The distance between the hyperplane and the closest data points is referred to as the margin. The best or optimal hyperplane that can separate the two classes is the line that has the largest margin. Only these points are relevant in defining the hyperplane and in the construction of the classifier. These points are called the support vectors. They support or define the hyperplane. In practice, an optimization algorithm is used to find the values for the coefficients that maximizes the margin.

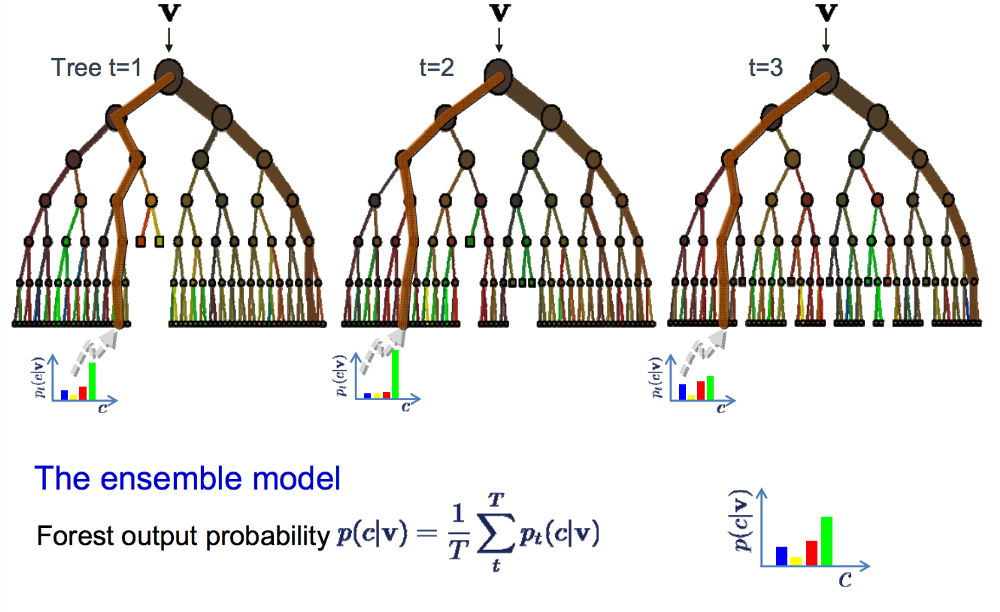
SVM might be one of the most powerful out-of-the-box classifiers and worth trying on your dataset.

## BAGGING AND RANDOM FOREST

Random Forest is one of the most popular and most powerful machine learning algorithms. It is a type of collective machine learning algorithm called Bootstrap Aggregation or bagging.

The bootstrap is a powerful statistical method for estimating a quantity from a data sample. Such as a mean. You take lots of samples of your data, calculate the mean, then average all of your mean values to give you a better estimation of the true mean value.

In bagging, the same approach is used, but instead for estimating entire statistical models, most commonly decision trees. Multiple samples of your training data are taken then models are constructed for each data sample. When you need to make a prediction for new data, each model makes a prediction and the predictions are averaged to give a better estimate of the true output value.

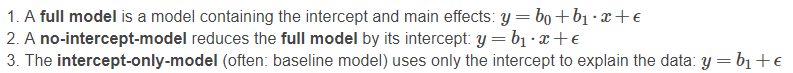


If you get good results with an algorithm with high variance (like decision trees), you can often get better results by bagging that algorithm.

## WHY WE NEED INTERCEPT IN MODELS

A common question is, if the intercept-term may be removed from a regression analysis in case it is not significant. Most of the time the answer to this question should be "No!".

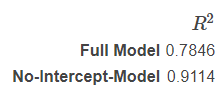
We will use the following terminology:



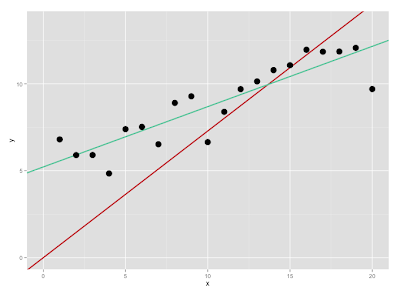
Problems of no-intercept-models

* R2 is not useful any more.
* The slope estimators might be biased.

The example presented in the graph below the R2s are:



It is obvious that this is not a reasonable result. The red line is clearly not the better model!



The second problem that arises is that the least squares estimator for the slopes in a no-intercept model are biased (systematically shifted towards larger or smaller values).

With removing the intercept from the model, we impose a restriction so that the regression line goes through the origin (x=0; y=0). The graph shows what happens to the regression line. The blue line is the common regression line the red line is the no-intercept-regression-line. It is heavily pulled down because it has to go through (0;0).

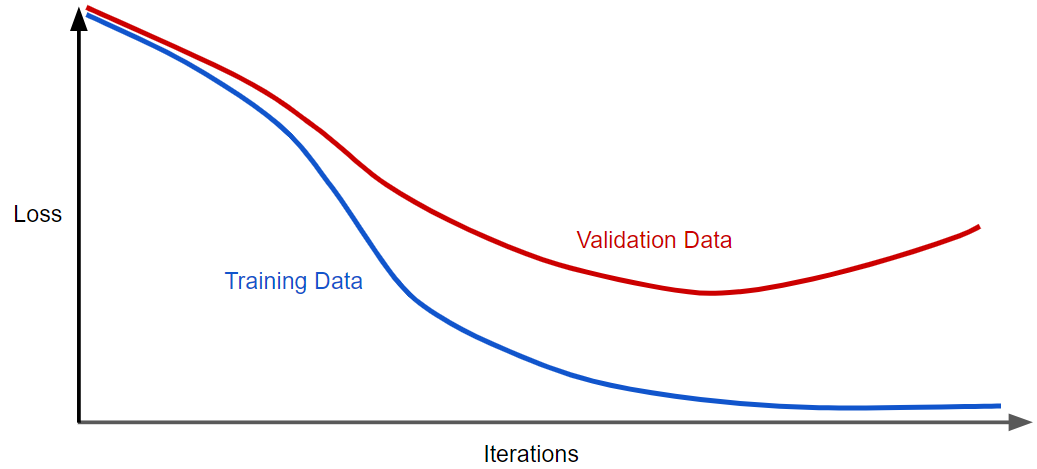
In python, most of the algorithms does not add an intercept by default. Hence, we should we careful in adding a constant term to the training set.

# REGULARIZATION TECHNIQUES

This is a form of regression, that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.

## L2 REGULARIZATION

Below figure shows a model in which training loss gradually decreases, but validation loss eventually goes up. In other words, this generalization curve shows that the model is overfitting to the data in the training set. Perhaps we could prevent overfitting by penalizing complex models, a principle called regularization.



In other words, instead of simply aiming to minimize loss (empirical risk minimization):



we'll now minimize los s+ complexity, which is called structural risk minimization:



Our training optimization algorithm is now a function of two terms: the loss term, which measures how well the model fits the data, and the regularization term, which measures model complexity.

Mainly Machine Learning focuses on two common ways to think of model complexity:

1. Model complexity as a function of the weights of all the features in the model.
2. Model complexity as a function of the total number of features with nonzero weights.

If model complexity is a function of weights, a feature weight with a high absolute value is more complex than a feature weight with a low absolute value.

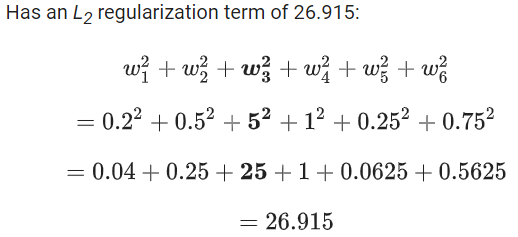
We can quantify complexity using the **L2 regularization** formula, which defines the regularization term as the sum of the squares of all the feature weights:



In this formula, weights close to zero have little effect on model complexity, while outlier weights can have a huge impact.

For example, a linear model with the following weights:





But w3 (bolded above), with a squared value of 25, contributes nearly all the complexity. The sum of the squares of all five other weights adds just 1.915 to the L2 regularization term.

## LAMBDA

Developers tune the overall impact of the regularization term by multiplying its value by a scalar known as lambda (also called the regularization rate). That is, model developers aim to do the following:



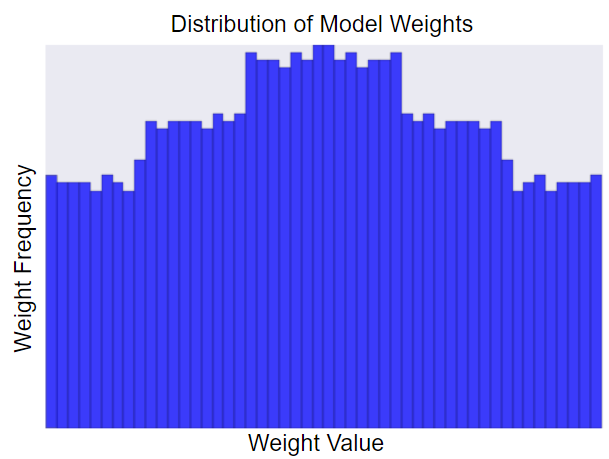
Performing L2 regularization has the following effect on a model

* Encourages weight values toward 0 (but not exactly 0)
* Encourages the mean of the weights toward 0, with a normal (bell-shaped or Gaussian) distribution.

Increasing the lambda value strengthens the regularization effect. For example, the histogram of weights for a high value of lambda might look as shown in figure



Lowering the value of lambda tends to yield a flatter histogram, as shown in Figure:



When choosing a lambda value, the goal is to strike the right balance between simplicity and training-data fit:

* If your lambda value is too high, your model will be simple, but you run the risk of underfitting your data. Your model won't learn enough about the training data to make useful predictions.
* If your lambda value is too low, your model will be more complex, and you run the risk of overfitting your data. Your model will learn too much about the particularities (trends and patterns) of the training data, and won't be able to generalize to new data.

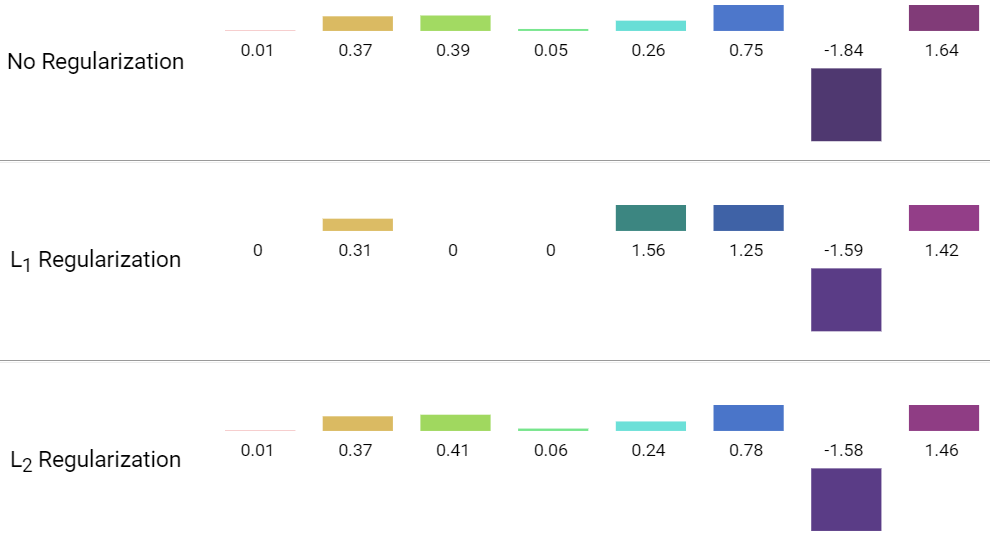
Setting lambda to zero removes regularization completely. In this case, training focuses exclusively on minimizing loss, which poses the highest possible overfitting risk.

## WHAT DOES REGULARIZATION ACHIEVE

A standard least squares model tends to have some variance in it, i.e. this model won’t generalize well for a data set different than its training data. Regularization, significantly reduces the variance of the model, without substantial increase in its bias. So, the tuning parameter λ, used in the regularization techniques described above, controls the impact on bias and variance. As the value of λ rises, it reduces the value of coefficients and thus reducing the variance. Till a point, this increase in λ is beneficial as it is only reducing the variance (hence avoiding overfitting), without losing any important properties in the data. But after certain value, the model starts losing important properties, giving rise to bias in the model and thus underfitting. Therefore, the value of λ should be carefully selected.



Coefficient weights after 500 iterations:

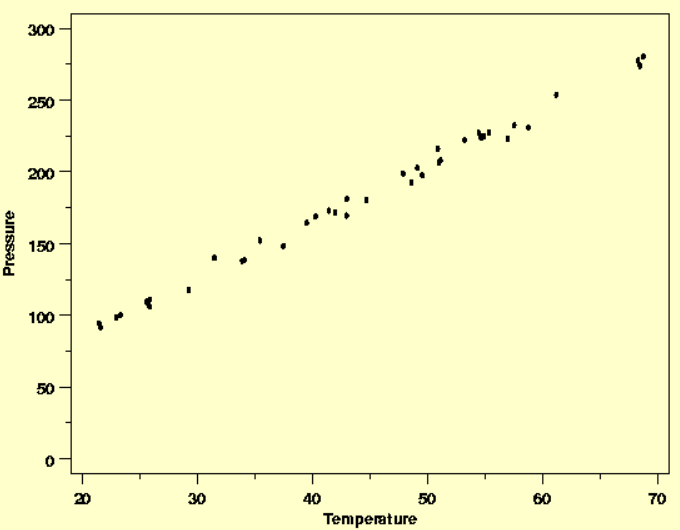


# SELECTING AN APPROPRIATE FUNCTION

The best way to select an initial model is to plot the data. Even if you have a good idea of what the form of the regression function will be, plotting allows a preliminary check of the underlying assumptions required for the model fitting to succeed.

## UNIVARIATE

The data from the Pressure/Temperature example is plotted below. From the plot it looks like a straight-line model will fit the data well. In this case there are no signs of any problems with the process or data collection.

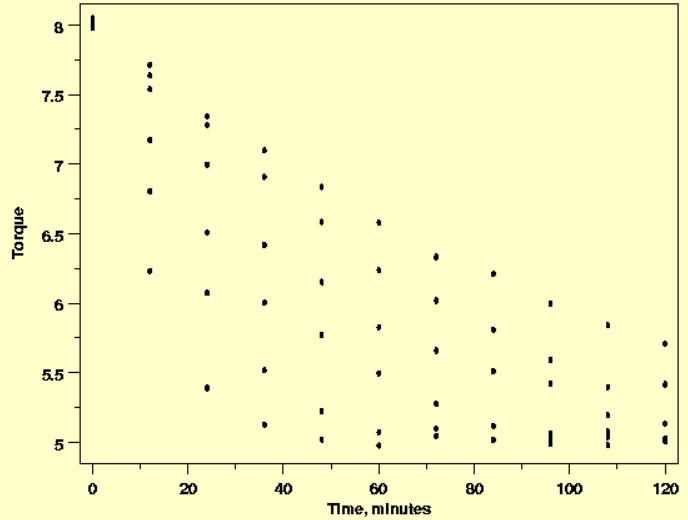


A key point when selecting a model is to start with the simplest function that looks as though it will describe the structure in the data. Complex models are fine if required, but they should not be used unnecessarily. Fitting models that are more complex than necessary means that random noise in the data will be modeled as deterministic structure. This will unnecessarily reduce the amount of data available for estimation of the residual standard deviation, potentially increasing the uncertainties of the results obtained when the model is used to answer engineering or scientific questions.

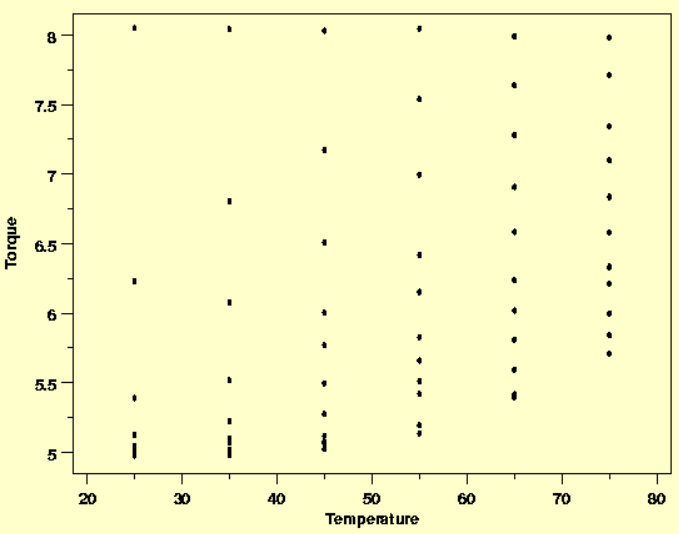
## MULTIVARIATE

When the function describing the deterministic variability in the response variable depends on several predictor (input) variables, it can be difficult to see how the different variables relate to one another. One way to tackle this problem that often proves useful is to plot cross-sections of the data and build up a function one dimension at a time.

This approach will often shed more light on the relationships between the different predictor variables and the response than plots that lump different levels of one or more predictor variables together on plots of the response variable versus another predictor variable.



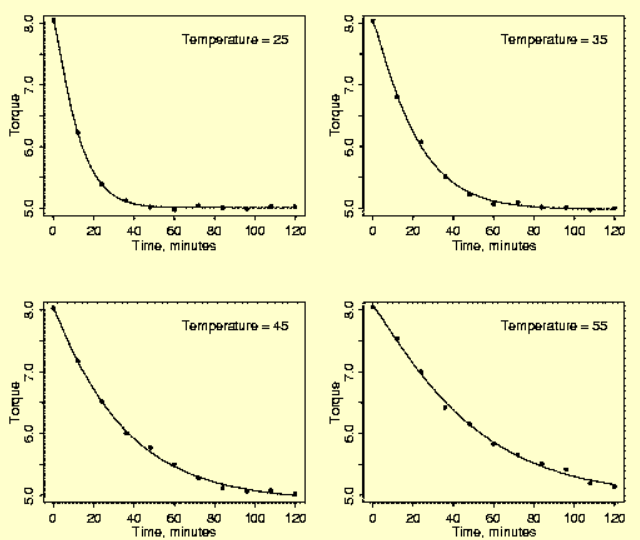
When the torque is plotted against time, the nature of the relationship is not clearly shown.



Similarly, when torque is plotted versus the temperature the effect of temperature is also unclear.

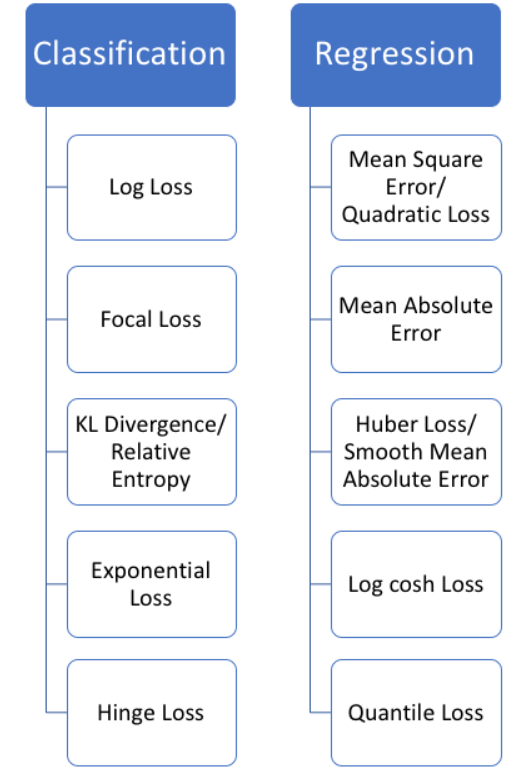
The difficulty in interpreting these plots arises because the plot of torque versus time includes data for several different temperatures and the plot of torque versus temperature includes data observed at different times. If both temperature and time are necessary parts of the function that describes the data, these plots are collapsing what really should be displayed as a three-dimensional surface onto a two-dimensional plot, muddying the picture of the data.

If cross-sections of the data are plotted in multiple plots instead of lumping different explanatory variable values together, the relationships between the variables can become much clearer.



# LOSS FUCNTIONS

Below are few loss functions that are used for regression and classification problems:



# UNDERSTANDING NOISE IN DATA

Machine learning works on a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, I mean noise in data. By noise we mean the data points that don’t really represent the true properties of your data, but random chance. Learning such data points, makes your model more flexible, at the risk of overfitting.

This becomes even more important when the number of features are very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. I have myself witnessed feature subsets giving better results than complete set of features for the same algorithm.

Not only in the competitions but this can be very useful in industrial applications as well. You not only reduce the training time and the evaluation time, you also have less things to worry about!

Noise (in the data science space) is unwanted data items, features or records which don’t help in explaining the feature itself or the relationship between feature & target. Noise often causes the algorithms to miss out patterns in the data.

In this section we will discuss about the noise in the dataset which we would encounter while working on data (tabular data) in data science projects and possible approaches we can take to deal with such noise.

One of the ways of avoiding overfitting is using cross validation, that helps in estimating the error over test set, and in deciding what parameters work best for your model.

We can encounter below type of noises in a dataset:

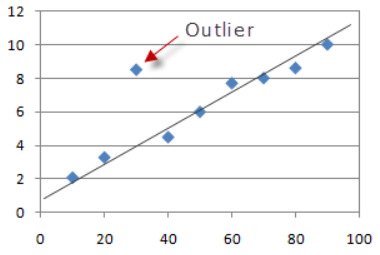
* Noise in data
  + Noise as an item (Noise 1)
  + Noise as a feature (Noise 2)
  + Noise as a record (Noise 3)
* Unsupervised methods

Noise in tabular data can be of three types:

* Anomalies in certain data items (Noise 1: certain anomalies in features & target)
* Features that don’t help in explaining the target (Noise 2: irrelevant/weak features)
* Records which don’t follow the form or relation which rest of the records do (Noise 3: noisy records)

## NOISE AS AN ITEM

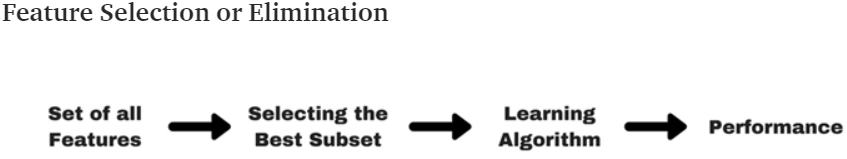
We can analyse the features & target and identify the noise in terms of outliers.



SOLUTION: Outlier detection & treatment: either remove the records or put upper and lower ceiling.

## NOISE AS A FEATURE

This type of noise is introduced when there are features in the data which are not related to target or doesn’t help explaining target.

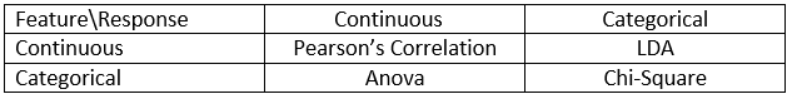


SOLUTION: Not all features are important, so we can use various methods to find the best subset of features:

### FILTER METHOD

Filter methods are generally used as a pre-processing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable.

We can perform various statistical tests between feature & response to identify which features are more relevant than others.



Please note that above methods don’t identify or deal with multicollinearity, we need to figure that out separately.

### 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.

Add/remove features to baseline model and compare the performance of the model:

* Forward selection
* Backward elimination
* Recursive elimination

### EMBEDDED METHODS (REGULARIZATION)

Embedded methods combine the qualities of filter and wrapper methods. It’s implemented by algorithms that have their own built-in feature selection methods.

Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce overfitting.

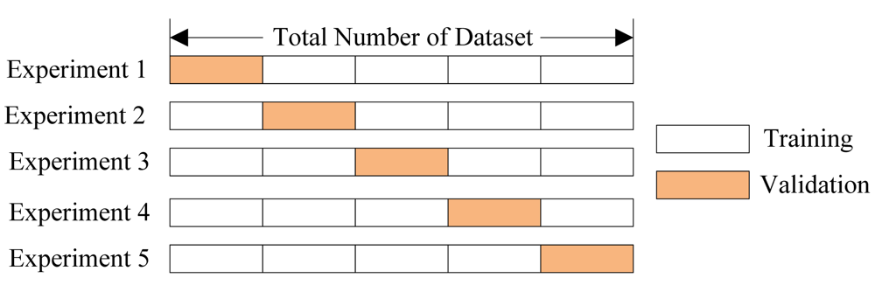
* Lasso regression performs L1 regularization which adds penalty equivalent to absolute value of the magnitude of coefficients.
* Ridge regression performs L2 regularization which adds penalty equivalent to square of the magnitude of coefficients.

## NOISE AS A RECORD

In these methods, we can try to find the set of records which have noise.

### K-FOLD VALIDATION (Cross Validation)

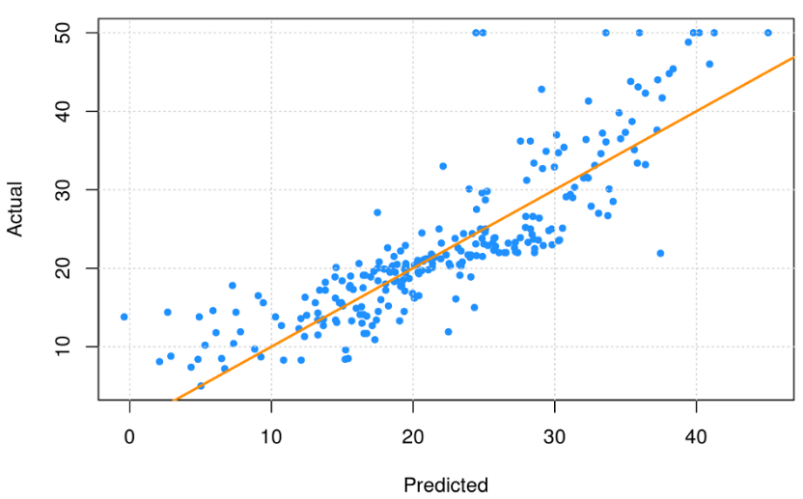
In this method, we can look at the cross-validation score of each fold and analyse the folds which have poor CV scores, what are the common attributes of records having poor scores, etc.



There are some other CV techniques which can be used.

### MANUAL METHOD

Here we can evaluate CV of each record (predicted vs. actual) and filter/analyse the records having a poor CV score. This will help us in analyzing why this is happening in the first place.



## UNSUPERVISED METHODS (ANOMALY DETECTION)

We can also use unsupervised learning algorithms to identify anomalies in data, these are mostly categorized as Anomaly Detection techniques.

### DENSITY-BASED ANOMALY DETECTION

This method assumes normal data points occur around a dense neighbourhood and abnormalities are far away. i.e. kNN & LOF based methods

### CLUSTERING-BASED ANOMALY DETECTION

Using clustering technique, we can analyse the clusters to analyse which has noise. Data instances falling outside the clusters can be marked as anomalies. i.e. k-Means clustering

### SVM-BASED ANOMALY DETECTION

This technique uses SVM to learn the soft boundary in the training set and tune on validation set to identify anomalies. In this approach, the need of large samples by the previous approach is reduced by using Support Vector Machine while maintaining the high quality of clustering-based anomaly detection methods. i.e. One-class SVM

### AUTOENCODER-BASED ANOMALY DETECTION

Auto-encoders are used in deep learning for unsupervised learning, we can use them for anomaly detection to identify noisy data-set. These methods are advanced and outperforms traditional anomaly detection methods. i.e. Variational Autoencoder based Anomaly Detection using Reconstruction Probability.

## BENEFITS OF TREATING NOISE IN DATA

* Enables the DS algorithm to train faster.
* Reduces the complexity of a model and makes it easier to interpret
* Improves the accuracy of a model if the right subset is chosen
* Reduces overfitting

# CONFOUNDING VARIABLE BIAS

Omitted variable bias occurs when a regression model leaves out relevant independent variables, which are known as confounding variables. This condition forces the model to attribute the effects of omitted variables to variables that are in the model, which biases the coefficient estimates.

Omitting confounding variables from your regression model can bias the coefficient estimates. What does that mean exactly? When you’re assessing the effects of the independent variables in the regression output, this bias can produce the following problems:

* Overestimate the strength of an effect.
* Underestimate the strength of an effect.
* Change the sign of an effect.
* Mask an effect that actually exists



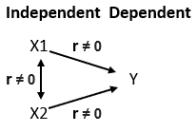


## OMITTED VARIABLE BIAS

How does this bias occur? How can variables you leave out of the model affect the variables that you include in the model? At first glance, this problem might not make sense.

* For omitted variable bias to occur, the following two conditions must exist:
* The omitted variable must correlate with the dependent variable.
* The omitted variable must correlate with at least one independent variable that is in the regression model.

The diagram below illustrates these two conditions. There must be non-zero correlations (r) on all three sides of the triangle.



This correlation structure causes confounding variables that are not in the model to bias the estimates that appear in your regression results. For example, removing either X variable will bias the other X variable.

The amount of bias depends on the strength of these correlations. Strong correlations produce greater bias. If the relationships are weak, the bias might not be severe. And, if the omitted variable is not correlated with another independent variable at all, excluding it does not produce bias.

## EXAMPLE

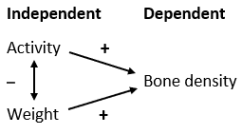
One study assessed the effects of physical activity on bone density. We measured various characteristics including the subjects’ activity levels, their weights, and bone densities among many others. Theories about how our bodies build bone suggest that there should be a positive correlation between activity level and bone density. In other words, higher activity produces greater bone density.

Early in the study, I wanted to validate our initial data quickly by using simple regression analysis to determine whether there is a relationship between activity and bone density. If our data were valid, there should be a positive relationship. To my great surprise, there was no relationship at all!

What was happening? The theory is well established in the field. Maybe our data was messed up somehow? Long story short, thanks to a confounding variable, the model was exhibiting omitted variable bias.

To perform the quick assessment, I included activity level as the only independent variable, but it turns out there is another variable that correlates with both activity and bone density—the subject’s weight.

After including weight in the regression model, along with activity, the results indicated that both activity and weight are statistically significant and have positive correlations with bone density. The diagram below shows the signs of the correlations between the variables.



## HOW THE OMITTED CONFOUNDING VARIABLE HID THE RELATIONSHIP

This correlation structure produces two opposing effects of activity. More active subjects get a bone density boost. However, they also tend to weigh less, which reduces bone density.

When I fit a regression model with only activity, the model had to attribute both opposing effects to activity alone. Hence, the zero correlation. However, when I fit the model with both activity and weight, it could assign the opposing effects to each variable separately.

For this example, when I omitted weight from the model, it produced a negative bias because the model underestimated the effect of activity. The results said there is no correlation when there is, in fact, a positive correlation.

The important takeaway here is that leaving out an important variable not only reduces the goodness-of-fit (larger residuals), but it can also bias the coefficient estimates.

# FEATURE SCALING

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing it is also known as data normalization and is generally performed during the data pre-processing step.

Since the range of values of raw data varies widely in some machine learning algorithms objective functions will not work properly without normalization.

For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes. To supress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.

Machine learning algorithms make assumptions about the dataset you are modeling (such as the IV’s should follow normal distribution). Often raw data is comprised of attributes with varying scales. For example, one attribute may be in kilograms and another may be a count. Although not required, you can often get a boost in performance by carefully choosing methods to rescale your data.

Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

Usually we adopt two techniques / methods for feature scaling:

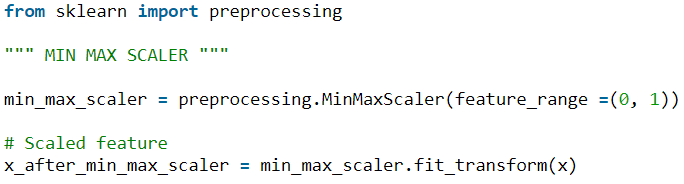
1. Either Normalize the numeric attributes between the range of 0 and 1.
2. Or standardize the numeric attributes to have a 0 mean and unit variance.

## NORMALIZING NUMERIC ATTRIBUTES

Data normalization is the process of rescaling one or more attributes to the range of 0 to 1. This means that the largest value for each attribute is 1 and the smallest value is 0.

When to use Normalization?

* Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve).
* Normalization is useful when your data has varying scales and the algorithm you are using does not make assumptions about the distribution of your data, such as k-nearest neighbours and artificial neural networks.



## STANDARDIZING NUMERIC ATTRIBUTES

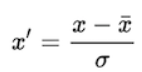
Data standardization is the process of rescaling one or more attributes so that they have a mean value of 0 and a standard deviation of 1.

Standardization (aka Z-Score normalization) assumes that your data has a Gaussian (bell curve) distribution. This does not strictly have to be true, but the technique is more effective if your attribute distribution is Gaussian.

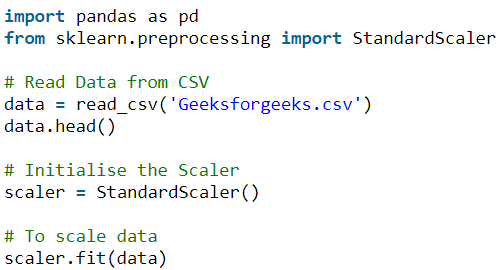
When to use Standardization?

* Standardization is useful when your data has varying scales and the algorithm you are using does make assumptions about your data having a Gaussian distribution, such as linear regression, logistic regression and linear discriminant analysis.

Standardisation replaces the values by their Z scores.



This redistributes the features with mean = 0 and standard deviation = 1. In Python we can implement standardization using sklearn preprocessing module



## WHEN TO SCALE

The rule of thumb for using feature scaling is any algorithm that computes distance or assumes normality or when using regularisation algorithms scale the features.

Some examples of algorithms where feature scaling matters are:

* **k-nearest neighbours** with a Euclidean distance measure is sensitive to magnitudes and hence should be scaled for all features to weigh in equally.
* Scaling is critical, while performing **Principal Component Analysis** (PCA). PCA tries to get the features with maximum variance and the variance is high for high magnitude features. This skews the PCA towards high magnitude features.
* While using iterative algorithm (gradient descent for example) scaling can make convergence faster.
* We can speed up **Gradient Descent** by scaling. This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.
* When ever using a **regularization technique** (like Lasso or Ridge) we should scale the features.

Algorithms like **Linear Discriminant Analysis (LDA)**, **Naive Bayes** are by design equipped to handle this and gives weights to the features accordingly. Performing a feature scaling in these algorithms may not have much effect.

Tree based models are not distance based models and can handle varying ranges of features. Hence, Scaling is not required while modelling trees.

More details can be found below:

<https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py>

## BENEFITS OF SCALING

Scaling means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range (for example, 0 to 1 or -1 to +1). If a feature set consists of only a single feature, then scaling provides little to no practical benefit. If, however, a feature set consists of multiple features, then feature scaling provides the following benefits:

* Helps gradient descent converge more quickly.
* Helps avoid the "NaN trap," in which one number in the model becomes a NaN (e.g., when a value exceeds the floating-point precision limit during training), and—due to math operations—every other number in the model also eventually becomes a NaN.
* Helps the model learn appropriate weights for each feature. Without feature scaling, the model will pay too much attention to the features having a wider range.

You don't have to give every floating-point feature exactly the same scale. Nothing terrible will happen if Feature A is scaled from -1 to +1 while Feature B is scaled from -3 to +3. However, your model will react poorly if Feature B is scaled from 5000 to 100000.

# TRAIN VALIDATE & TESTING FRAMEWORK FOR ML

One of the most likely culprits for this disconnect between results in development vs results in production is a poorly chosen validation set (or even worse, no validation set at all). Depending on the nature of your data, choosing a validation set can be the most important step. Although sklearn offers a train\_test\_split method, this method takes a random subset of the data, which is a poor choice for many real-world problems.

Sklearn has a train\_test\_split method, but no train\_validation\_test\_split. Kaggle only provides training and test sets, yet to do well, you will need to split their training set into your own validation and training sets. Also, it turns out that Kaggle’s test set is actually sub-divided into two sets.

* **The training set is used to train a given model.**
* **The validation set is used to choose between models** (for instance, does a random forest or a neural network better for your problem? Do you want a random forest with 40 trees or 50 trees?)
* **The test set tells you how you’ve done**. If you’ve tried out a lot of different models, you may get one that does well on your validation set just by chance, and having a test set helps make sure that is not the case.

A key property of the validation and test sets is that they must be representative of the new data you will see in the future.

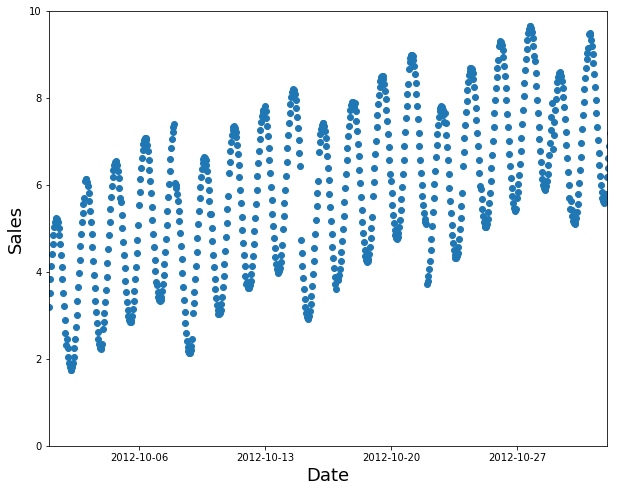
WHEN IS A RANDOM SUBSET NOT GOOD ENOUGH?

It’s informative to look at a few examples. Although many of these examples come from Kaggle competitions, they are representative of problems you would see in the workplace.

## TIME SERIES DATA

If your data is a time series, choosing a random subset of the data will be both too easy and not representative of most business use cases (where you are using historical data to build a model for use in the future). If your data includes the date and you are building a model to use in the future, you will want to choose a continuous section with the latest dates as your validation set (for instance, the last two weeks or last month of the available data).

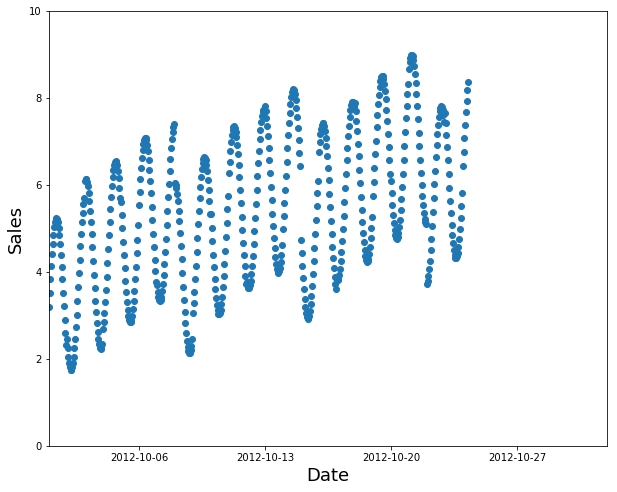
Suppose you want to split the time series data below into training and validation sets:



A random subset is a poor choice (too easy to fill in the gaps, and not indicative of what you’ll need in production):



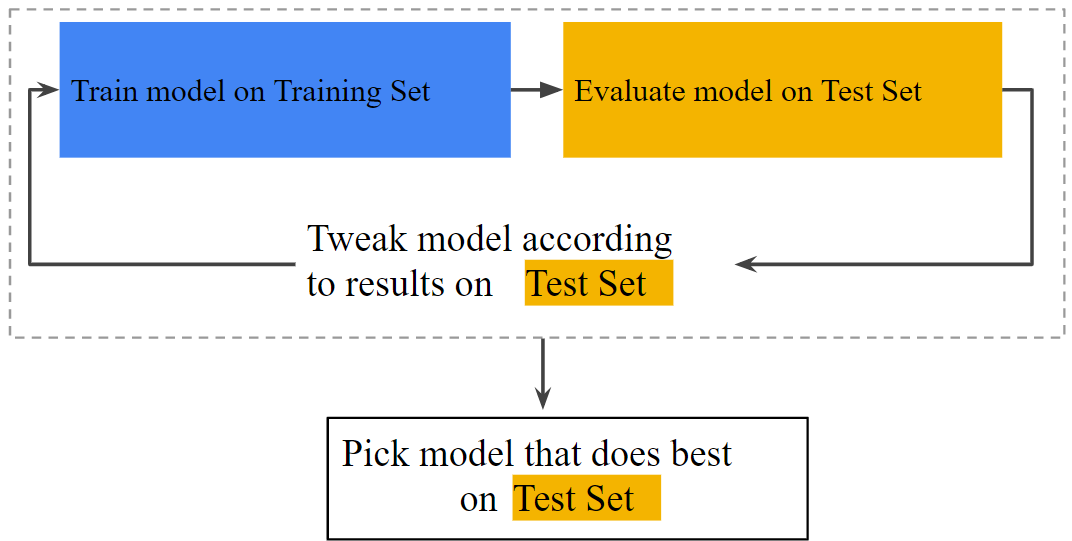
Use the earlier data as your training set (and the later data for the validation set):



**Example:** Kaggle currently has a competition to predict the sales in a chain of Ecuadorian grocery stores. Kaggle’s “training data” runs from Jan 1 2013 to Aug 15 2017 and the test data spans Aug 16 2017 to Aug 31 2017. A good approach would be to use Aug 1 to Aug 15 2017 as your validation set, and all the earlier data as your training set.



A training set, which includes the independent variables, as well as the dependent variable (what you are trying to predict).



A test set, which just has the independent variables. You will make predictions for the test set, which you can submit to Kaggle and get back a score of how well you did.

# PREPROCESSING DATA

The sklearn.preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalers or transformers are more appropriate.

The behaviours of the different scalers, transformers, and normalizers on a dataset containing marginal outliers is highlighted in Compare the effect of different scalers on data with outliers.

<https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing>

# TRANSFORMING THE PREDICTION TARGET

These are transformers that are not intended to be used on features, only on supervised learning targets.

<https://scikit-learn.org/stable/modules/preprocessing_targets.html#preprocessing-targets>

# WHEN TO USE LOG TRANSFORMATION

There are three reasons to transform a variable by taking the natural logarithm. The reason for logging the variable will determine whether you want to log the independent variable(s), dependent or both.

1. Firstly, to improve model fit. For instance, if your residuals aren't normally distributed then taking the logarithm of a skewed variable may improve the fit by altering the scale and making the variable more "normally" distributed. For instance, earnings are truncated at zero and often exhibits positive skew. If the variable has negative skew you could firstly invert the variable before taking the logarithm.

While this usually applies to the dependent variable you occasionally have problems with the residuals (e.g. heteroscedasticity) caused by an independent variable which can be sometimes corrected by taking the logarithm of that variable.

1. The second reason for logging one or more variables in the model is for interpretation.

If you log both your dependent (Y) and independent (X) variable(s) your regression coefficients (β) will be elasticities and interpretation would go as follows: a 1% increase in X would lead to a ceteris paribus β% increase in Y (on average). Logging only one side of the regression "equation" would lead to alternative interpretations as outlined below:

* Y and X -- a one unit increase in X would lead to a β increase/decrease in Y
* Log Y and Log X -- a 1% increase in X would lead to a β% increase/decrease in Y
* Log Y and X -- a one unit increase in X would lead to a β∗100 % increase/decrease in Y
* Y and Log X -- a 1% increase in X would lead to a β/100 increase/decrease in Y

1. And finally, there could be a theoretical reason for doing so. For example, some models that we would like to estimate are multiplicative and therefore nonlinear. Taking logarithms allows these models to be estimated by linear regression.

# PREDICTION ASSUMPTIONS

The following three basic assumptions guide generalization:

* We draw examples independently and identically (i.i.d) at random from the distribution. In other words, examples don't influence each other. (An alternate explanation: i.i.d. is a way of referring to the randomness of variables.)
* The distribution is stationary; that is the distribution doesn't change within the data set.
* We draw examples from partitions from the same distribution.

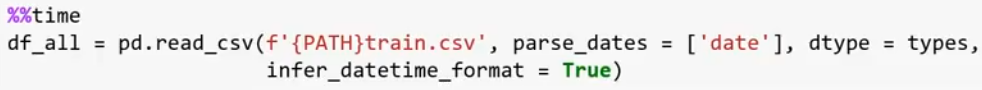
In practice, we sometimes violate these assumptions. For example:

* Consider a model that chooses ads to display. The i.i.d. assumption would be violated if the model bases its choice of ads, in part, on what ads the user has previously seen.
* Consider a data set that contains retail sales information for a year. User's purchases change seasonally, which would violate stationarity.

When we know that any of the preceding three basic assumptions are violated, we must pay careful attention to metrics.

# CODE OPTIMIZATION

1. While working with larger datasets, we might get poor performance as Python will take time to load the data. We can provide the dtypes of the columns / features while reading the file to fast the import process. This will significantly improve the performance.

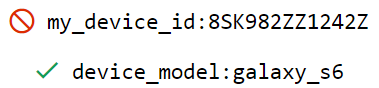


1. Sometimes when we split the data into train and test, we use random sampling. Instead of using random sampling if we use subset technique, it will improve the performance significantly.
2. We can use %prun (profiler) to identify the bottleneck while fitting the model as which command (code of line) is taking most of the resources and time to process.

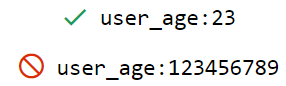


# PROPERTIES OF GOOD FEATURES

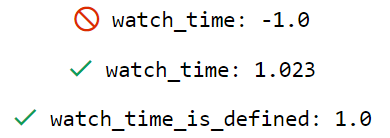
1. Feature values should appear with non-zero values more than a handful of times in the dataset.



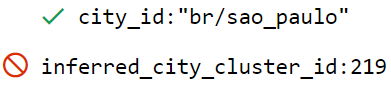
1. Feature should not have more zeros. If there exists a feature, It will be of no use and hence we can drop it.
2. Feature should have a clear and obvious meaning



1. Feature should not have magic values



1. Feature values should not change over time



1. Feature should not have crazy outliers.
2. If possible, we can bin a particular feature. For example, in housing pricing dataset there is not direct relationship between house prices in north and south. Hence, we can create bins on longitude values and create different bins. After creating the bins, we can use OneHotEncoding to assign them categories. Now this allows the model to fit different values for each bin.
3. In time series, check for no of examples over time.

# MONOTONICITY IN MACHINE LEARNING

In practical machine learning and data science tasks, an ML model is often used to quantify a global, semantically meaningful relationship between two or more values. For example, a hotel chain might want to use ML to optimize their pricing strategy and use a model to estimate the likelihood of a room being booked at a given price and day of the week.

For a relationship like this the assumption is that, all other things being equal, a cheaper price is preferred by a user, so demand is higher at a lower price. However, what might easily happen is that upon building the model, the data scientist discovers that the model is behaving unexpectedly. For example, the model predicts that on Tuesdays, the clients would rather pay $110 than $100 for a room.

The reason is that while there is an expected monotonic relationship between price and the likelihood of booking, the model is unable to (fully) capture it, due to noisiness of the data and confounds in it.

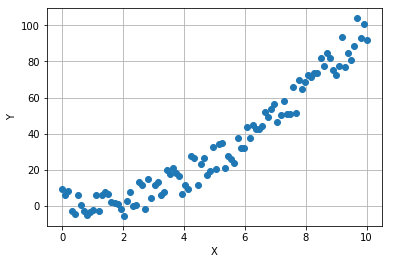
such constraints are ignored by practitioners, especially when non-linear models such as random forests, gradient boosted trees or neural networks are used.

In recent years there has been a lot of progress in various ML libraries to allow setting monotonicity constraints for the models, including in LightGBM and XGBoost, two of the most popular libraries for gradient boosted trees.

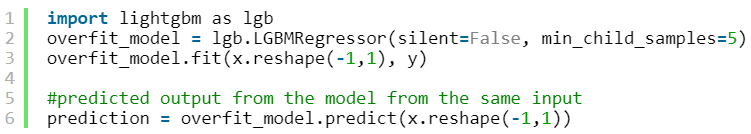
## MONOTONICITY CONSTRAINTS IN LIGHGBM AND XGBOOST

For tree-based methods (decision trees, random forests, gradient boosted trees), monotonicity can be forced during the model learning phase by not creating splits on monotonic features that would break the monotonicity constraint.

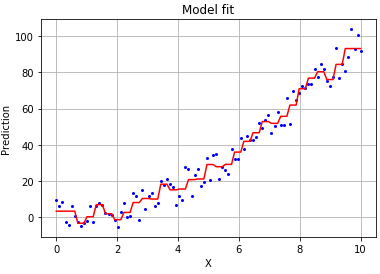
In the following example, let’s train too models using LightGBM on a toy dataset where we know the relationship between X and Y to be monotonic (but noisy) and compare the default and monotonic model.



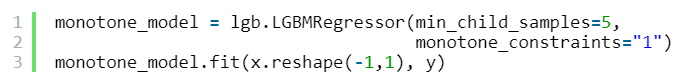
Let’s fit a fit a gradient boosted model on this data, setting min\_child\_samples to 5.



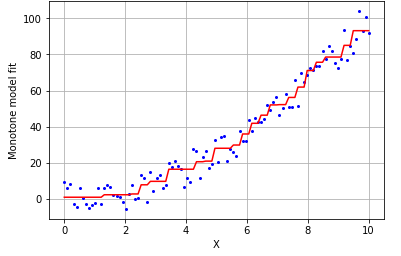
The model will slightly overfit (due to small min\_child\_samples), which we can see from plotting the values of X against the predicted values of Y: the red line is not monotonic as we’d like it to be.



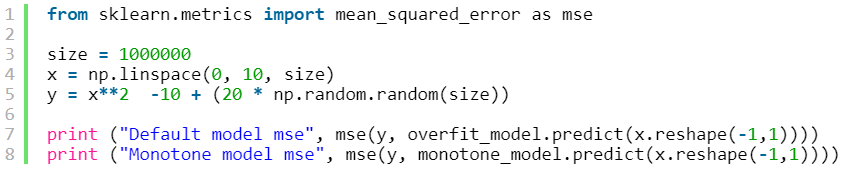
Since we know that that the relationship between X and Y should be monotonic, we can set this constraint when specifying the model.



The parameter monotone\_constraints=”1″ states that the output should be monotonically increasing wrt. the first features (which in our case happens to be the only feature). After training the monotone model, we can see that the relationship is now strictly monotone.



And if we check the model performance, we can see that not only does the monotonicity constraint provide a more natural fit, but the model generalizes better as well (as expected).





Measuring the mean squared error on new test data, we see that error is smaller for the monotone model.

# OPTIMIZING MODEL PERFORMANCE

Once the model has been finalized, we can use below techniques for tuning the model.

## OPTIMIZATION ALGORITHM

Change the algorithm to use in the optimization if available for the model. For example, we can change the solver in logistic regression.

## DIMENSIONALITY REDUCTION

In general, dimensionality reduction does not only help reducing computational costs for a given classification task, but it can also be helpful to avoid overfitting by minimizing the error in parameter estimation (“curse of dimensionality”).

One of the most important applications of PCA is for speeding up machine learning algorithms.

## %PRUN MAGIC COMMAND

We can use %prun magic command in front of the code which is taking longer to get executed and it will point to the code which is causing the issue.

## IMPORT AS REQUIRED

We can import only required classes/ functions from the various models and hence it will take less time to execute.

## DASK

We can use DASK to import heavy files and run the code efficiently. Dask supports the Pandas dataframe and Numpy array data structures and is able to either be run on your local computer or be scaled up to run on a cluster.

<https://towardsdatascience.com/why-every-data-scientist-should-use-dask-81b2b850e15b>

## VARIABLE TYPES

We can change the datatype while reading the files for efficient storage and faster processing.

## OBJECTIVE FUNCTION

We must choose appropriate objective function and pass in appropriate hyperoperators else objective function will take time to converge.

## CHOOSE APPROPRIATE UTILITY

For example, if I wish to perform Hyperparameter tuning then I can use either GridSearchCV or RandomizedSearchCV. If we compare the performance of both RandomizedSearchCV is faster compared to GridSearchCV.

## ML BEST PRACTICES

Below are the coding best practises which one should follow to design a good machine learning model.

1. We should use numpy data structures (list, arrays etc) when needed as they are faster compared to python DS as they are written in C language.
2. We should use df.column instead of df[‘column’] whenever applicable as the latter is slow.
3. Avoid Dummy variable trap (drop 1 column when converting categorical variables to numeric).
4. Scale the variables whenever required. If you scale the training set, then you scale test set as well.
5. If Y is scaled then, you will have to inverse transform the predicted to get it in original scale.
6. If you are scaling X and Y, then use diff Scaler objects to fit and transform. As we would need to inverse transform the predicted.
7. Constant should always be the 1st column and label should be the last column in dataset. And order of variables in test and train should be same.
8. Its good to have random\_state = seed while regression as it makes the code rerunnable and valid for comparison with different versions of model.
9. After importing the dataset, drop duplicates if any.
10. Use imputation for missing values. Fill in the missing values with mean values (if there are categories then base on groups mean).
11. Drop any features with ID values (unique values).
12. If you have a numeric categorical column in the dataset (example Passenger class 1,2, and 3) then change the same using dummy variables.

# IMPROVING MODEL PERFORMANCE

We can improve model performance using below techniques:

## HYPER-PARAMETER TUNING / GRID SERACH

Parameter Tuning allow us to improve the performance of the models, by tuning them. Each model is composed of two types of parameters:

* The parameters that are learnt, for example the coefficients in Linear Regression.
* The hyperparameters.

The hyperparameters are the parameters that are not learnt and that are fixed values inside the model equations. For example, the regularization parameter lambda or the penalty parameter ‘C’ are hyperparameters. Finding their optimal value is exactly what Parameter Tuning is about.

We can use **Grid Search** technique for finding out the optimal values of the hyper parameters or we can manually try hit and try method.

Here are a few rules of thumb that may help guide you:

* Training error should steadily decrease, steeply at first, and should eventually plateau as training converges.
* If the training has not converged, try running it for longer.
* If the training error decreases too slowly, increasing the learning rate may help it decrease faster.
* But sometimes the exact opposite may happen if the learning rate is too high.
* If the training error varies wildly, try decreasing the learning rate.
* Lower learning rate plus larger number of steps or larger batch size is often a good combination.
* Very small batch sizes can also cause instability. First try larger values like 100 or 1000, and decrease until you see degradation.

## FEATURE SCALING

Feature scaling is very important and required when the dataset used for learning is having features with different scales and we are using an algorithm which tries to calculate Euclidean distance between datapoints or using an algorithm which make assumptions regarding feature distribution.

## FEATURE ENGINEERING

Creating new features from existing features can very useful and can impact the performance of the model. For example, if we have a date feature, we can extract year, quarter, month, week and other attributes from same. Another example would be if we are given total income and total spent, we can calculate profit or loss.

# MODEL EVALUATION METRICS

Below are some metrics for scoring the model performance



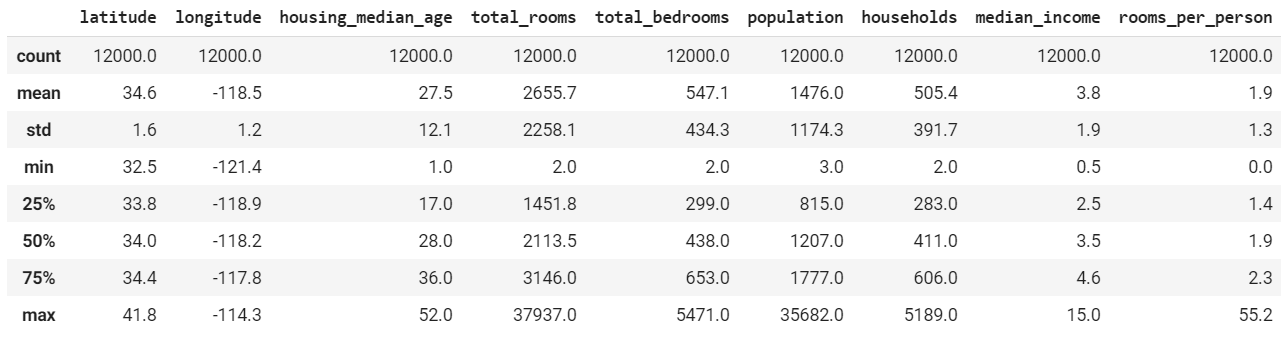
# STEPS TO FOLLOW WHILE BUILDING A MODEL

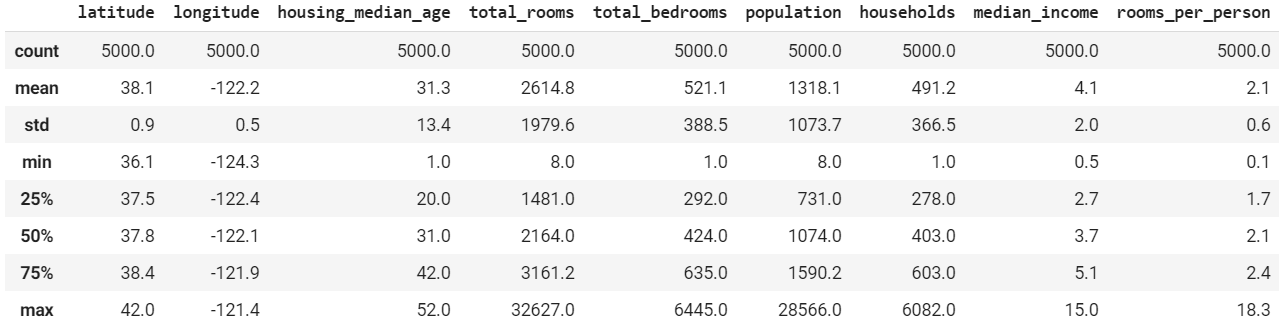
Below are the steps that we can follow to build a model.

1. Before creating the test and train split, shuffle the dataset, then we can subset the df.



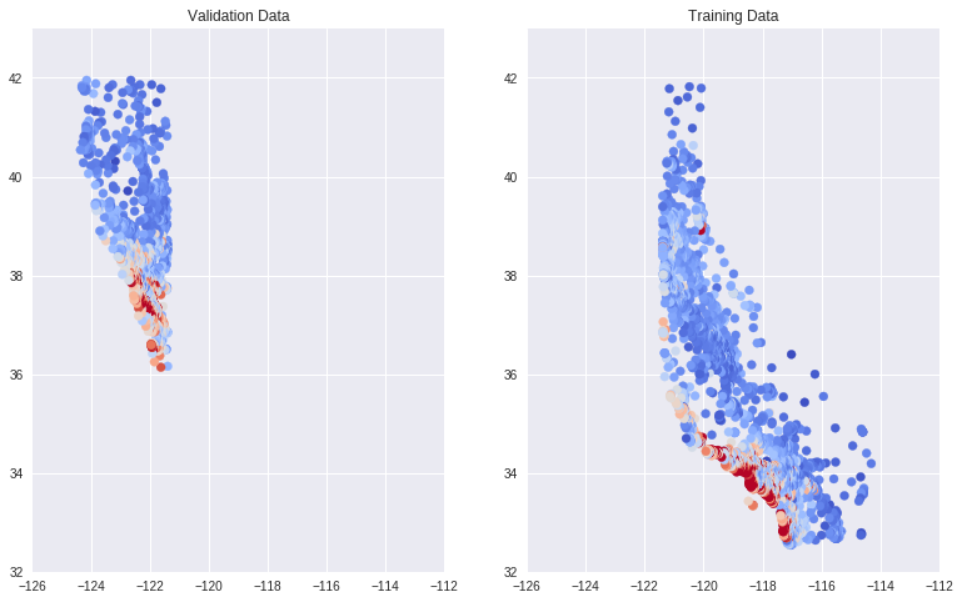
1. Compare the basic stats for train and test dataset. The key thing to notice is that for any given feature or column, the distribution of values between the train and validation splits should be roughly equal.





If you see above example, the 75th percentile and the max for rooms per person feature in train and test are not comparable. The fact that this is not the case is a real worry, and shows that we likely have a fault in the way that our train and validation split was created.

If we make a plot, its evident from the graph.



While creating the train and test splits, we should make sure that both are representative of the population. As seen below training set is representative of the population but validation set is not.

# LARGE FEATURE DATASET

It is very common to work on a dataset containing thousands of features. To handle categorical features having large distinct values we can use

* Bin counting scheme
* Hashing scheme

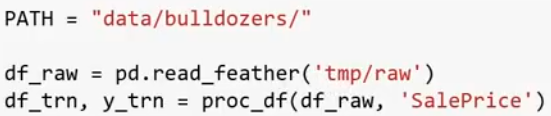
# ENCODING COMPARISON

https://medium.com/data-design/visiting-categorical-features-and-encoding-in-decision-trees-53400fa65931

# FEATHER FILE

Please see below code to create a feather file





# NAN vs NONE

Below are some of the differences:

* np.nan allows for vectorized operations, it’s a float value
* while None by definition forces object type, which basically disables all efficiency in numpy.

# REFERENCES

Top 8 common machine learning tasks

<https://vitalflux.com/7-common-machine-learning-tasks-related-methods/>

Key machine learning definitions

<https://medium.com/technology-nineleaps/some-key-machine-learning-definitions-b524eb6cb48>

Regularization

<http://enhancedatascience.com/2017/07/04/machine-learning-explained-regularization/>

Feature Interactions

<https://xgboost.readthedocs.io/en/latest/tutorials/feature_interaction_constraint.html>

# MACHINE LEARNING WORK FLOW

