**UNIT II**

**What Is Model Selection**

Model selection is the process of selecting one [final machine learning model](https://machinelearningmastery.com/train-final-machine-learning-model/) from among a collection of candidate machine learning models for a training dataset.

Model selection is a process that can be applied both across different types of models (e.g. logistic regression, SVM, KNN, etc.) and across models of the same type configured with different model hyperparameters (e.g. different kernels in an SVM).

For example, we may have a dataset for which we are interested in developing a classification or regression predictive model. We do not know beforehand as to which model will perform best on this problem, as it is unknowable. Therefore, we fit and evaluate a suite of different models on the problem.

**Model selection** is the process of choosing one of the models as the final model that addresses the problem.

Model selection is different from **model assessment**.

For example, we evaluate or assess candidate models in order to choose the best one, and this is model selection. Whereas once a model is chosen, it can be evaluated in order to communicate how well it is expected to perform in general; this is model assessment.

**Considerations for Model Selection**

Fitting models is relatively straightforward, although selecting among them is the true [challenge of applied machine learning](https://machinelearningmastery.com/applied-machine-learning-is-hard/).

Firstly, we need to get over the idea of a “*best*” model.

All models have some predictive error, given the statistical noise in the data, the incompleteness of the data sample, and the limitations of each different model type. Therefore, the notion of a perfect or best model is not useful. Instead, we must seek a model that is “*good enough*.”

**What do we care about when choosing a final model?**

The project stakeholders may have specific requirements, such as maintainability and limited model complexity. As such, a model that has lower skill but is simpler and easier to understand may be preferred.

Alternately, if model skill is prized above all other concerns, then the ability of the model to perform well on out-of-sample data will be preferred regardless of the computational complexity involved.

Therefore, a “*good enough*” model may refer to many things and is specific to your project, such as:

* A model that meets the requirements and constraints of project stakeholders.
* A model that is sufficiently skillful given the time and resources available.
* A model that is skillful as compared to naive models.
* A model that is skillful relative to other tested models.
* A model that is skillful relative to the state-of-the-art.

Next, we must consider what is being selected.

For example, we are not selecting a fit model, as all models will be discarded. This is because once we choose a model, we will fit a new final model on all available data and start using it to make predictions.

Therefore, are we choosing among algorithms used to fit the models on the training dataset?

Some algorithms require specialized data preparation in order to best expose the structure of the problem to the learning algorithm. Therefore, we must go one step further and consider **model selection as the process of selecting among model development pipelines**.

Each pipeline may take in the same raw training dataset and outputs a model that can be evaluated in the same manner but may require different or overlapping computational steps, such as:

* Data filtering.
* Data transformation.
* Feature selection.
* Feature engineering.
* And more…

The closer you look at the challenge of model selection, the more nuance you will discover.

Now that we are familiar with some considerations involved in model selection, let’s review some common methods for selecting a model.

**Model Selection Techniques**

The best approach to model selection requires “*sufficient*” data, which may be nearly infinite depending on the complexity of the problem.

In this ideal situation, we would split the data into [training, validation, and test sets](https://machinelearningmastery.com/difference-test-validation-datasets/), then fit candidate models on the training set, evaluate and select them on the validation set, and report the performance of the final model on the test set.

*If we are in a data-rich situation, the best approach […] is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model.*

*In many applications, however, the supply of data for training and testing will be limited, and in order to build good models, we wish to use as much of the available data as possible for training. However, if the validation set is small, it will give a relatively noisy estimate of predictive performance.*

Instead, there are two main classes of techniques to approximate the ideal case of model selection; they are:

* **Probabilistic Measures**: Choose a model via in-sample error and complexity.
* **Resampling Methods**: Choose a model via estimated out-of-sample error.

Let’s take a closer look at each in turn.

### Probabilistic Measures

[Probabilistic measures](https://machinelearningmastery.com/probabilistic-model-selection-measures) involve analytically scoring a candidate model using both its performance on the training dataset and the complexity of the model.

It is known that training error is optimistically biased, and therefore is not a good basis for choosing a model. The performance can be penalized based on how optimistic the training error is believed to be. This is typically achieved using algorithm-specific methods, often linear, that penalize the score based on the complexity of the model.

*Historically various ‘information criteria’ have been proposed that attempt to correct for the bias of maximum likelihood by the addition of a penalty term to compensate for the over-fitting of more complex models.*

Four commonly used probabilistic model selection measures include:

* Akaike Information Criterion (AIC).
* Bayesian Information Criterion (BIC).
* Minimum Description Length (MDL).
* Structural Risk Minimization (SRM).

Probabilistic measures are appropriate when using simpler linear models like linear regression or logistic regression where the calculating of model complexity penalty (e.g. in sample bias) is known and tractable.

### Resampling Methods

[Resampling methods](https://machinelearningmastery.com/statistical-sampling-and-resampling/) seek to estimate the performance of a model (or more precisely, the model development process) on out-of-sample data.

This is achieved by splitting the training dataset into sub train and test sets, fitting a model on the sub train set, and evaluating it on the test set. This process may then be repeated multiple times and the mean performance across each trial is reported.

It is a type of [Monte Carlo estimate](https://machinelearningmastery.com/monte-carlo-sampling-for-probability) of model performance on out-of-sample data, although each trial is not strictly independent as depending on the resampling method chosen, the same data may appear multiple times in different training datasets, or test datasets.

Three common resampling model selection methods include:

* Random train/test splits.
* [Cross-Validation](https://machinelearningmastery.com/k-fold-cross-validation/) (k-fold, LOOCV, etc.).
* [Bootstrap](https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/).

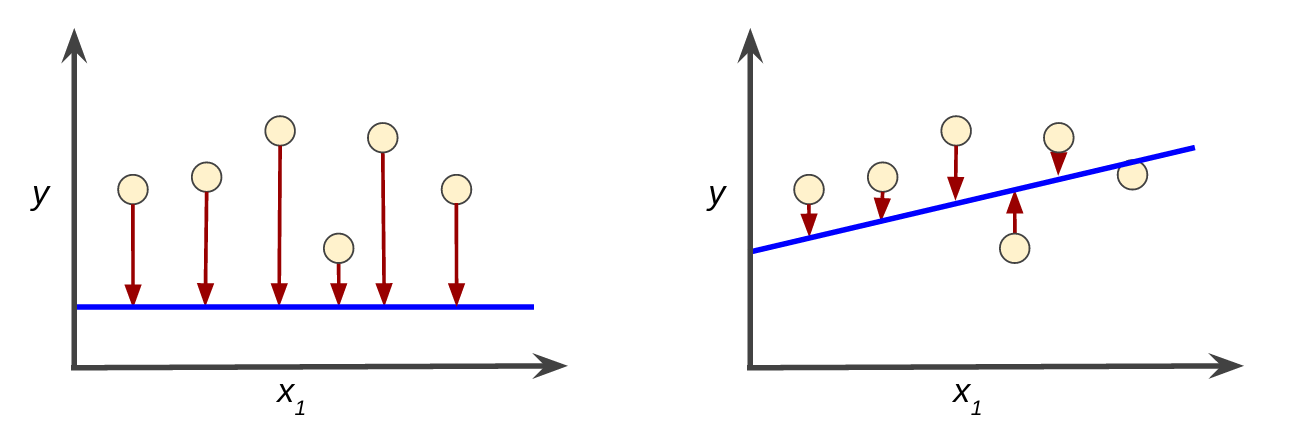
Most of the time probabilistic measures (described in the previous section) are not available, therefore resampling methods are used.

**Training Model:**

**Training** a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called **empirical risk minimization**.

Loss is the penalty for a bad prediction. That is, **loss** is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples. For example, Figure 3 shows a high loss model on the left and a low loss model on the right. Note the following about the figure:

* The arrows represent loss.
* The blue lines represent predictions.



**Figure 3. High loss in the left model; low loss in the right model.**

Notice that the arrows in the left plot are much longer than their counterparts in the right plot. Clearly, the line in the right plot is a much better predictive model than the line in the left plot.

You might be wondering whether you could create a mathematical function—a loss function—that would aggregate the individual losses in a meaningful fashion.

### Squared loss: a popular loss function

The linear regression models we'll examine here use a loss function called **squared loss** (also known as **L2 loss**). The squared loss for a single example is as follows:

= the square of the difference between the label and the prediction

= (observation - prediction(**x**))2

= (y - y')2

**Mean square error** (**MSE**) is the average squared loss per example over the whole dataset. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

MSE=1N∑(x,y)∈D(y−prediction(x))2

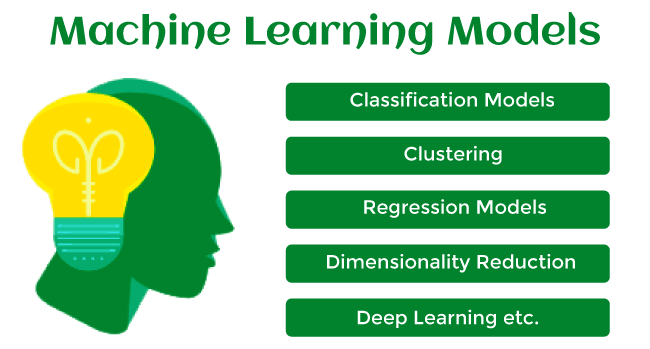
where:

* (x,y) is an example in which
  + x is the set of features (for example, chirps/minute, age, gender) that the model uses to make predictions.
  + y is the example's label (for example, temperature).
* prediction(x) is a function of the weights and bias in combination with the set of features x.
* D is a data set containing many labeled examples, which are (x,y) pairs.
* N is the number of examples in D.

Although MSE is commonly-used in machine learning, it is neither the only practical loss function nor the best loss function for all circumstances.

# Machine Learning Models

**A machine learning model is defined as a mathematical representation of the output of the training process.** Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.



Let's understand an example of the ML model where we are creating an app to recognize the user's emotions based on facial expressions. So, creating such an app is possible by Machine learning models where we will train a model by feeding images of faces with various emotions labeled on them. Whenever this app is used to determine the user's mood, it reads all fed data then determines any user's mood.

Hence, in simple words, we can say that a machine learning model is a simplified representation of something or a process. In this topic, we will discuss different machine learning models and their techniques and algorithms.

## What is Machine Learning Model?

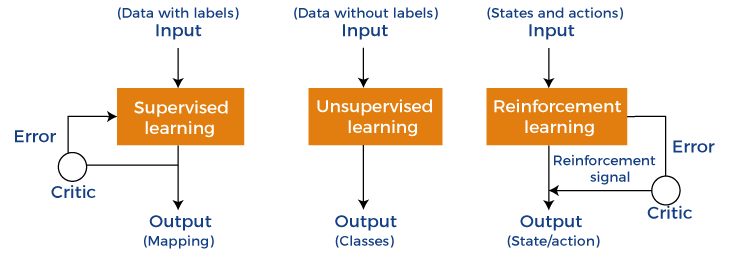
Machine Learning models can be understood as a program that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response. First, these models are trained over a set of data, and then they are provided an algorithm to reason over data, extract the pattern from feed data and learn from those data. Once these models get trained, they can be used to predict the unseen dataset.

There are various types of machine learning models available based on different business goals and data sets.

### Classification of Machine Learning Models:

Based on different business goals and data sets, there are three learning models for algorithms. Each machine learning algorithm settles into one of the three models:

* Supervised Learning
* Unsupervised Learning
* Reinforcement Learning



**Supervised Learning is further divided into two categories:**

* Classification
* Regression

**Unsupervised Learning is also divided into below categories:**

* Clustering
* Association Rule
* Dimensionality Reduction

## 1. Supervised Machine Learning Models

Supervised Learning is the simplest machine learning model to understand in which input data is called training data and has a known label or result as an output. So, it works on the principle of input-output pairs. It requires creating a function that can be trained using a training data set, and then it is applied to unknown data and makes some predictive performance. Supervised learning is task-based and tested on labeled data sets.

We can implement a supervised learning model on simple real-life problems. For example, we have a dataset consisting of age and height; then, we can build a supervised learning model to predict the person's height based on their age.

Supervised Learning models are further classified into two categories:

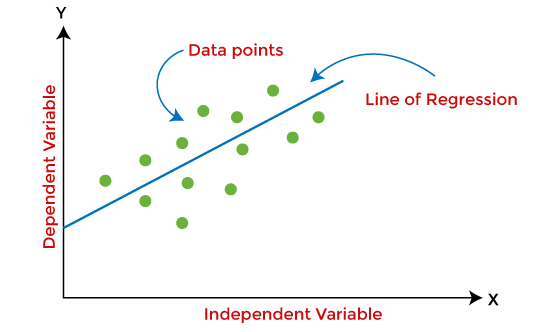
### Regression

In regression problems, the output is a continuous variable. Some commonly used Regression models are as follows:

**a) Linear Regression**

Linear regression is the simplest machine learning model in which we try to predict one output variable using one or more input variables. The representation of linear regression is a linear equation, which combines a set of input values(x) and predicted output(y) for the set of those input values. It is represented in the form of a line:

Y = bx+ c.



The main aim of the linear regression model is to find the best fit line that best fits the data points.

Linear regression is extended to multiple linear regression (find a plane of best fit) and polynomial regression (find the best fit curve).

**b) Decision Tree**

Decision trees are the popular machine learning models that can be used for both regression and classification problems.

A decision tree uses a tree-like structure of decisions along with their possible consequences and outcomes. In this, each internal node is used to represent a test on an attribute; each branch is used to represent the outcome of the test. The more nodes a decision tree has, the more accurate the result will be.

The advantage of decision trees is that they are intuitive and easy to implement, but they lack accuracy.

Decision trees are widely used in **operations research, specifically in decision analysis, strategic planning**, and mainly in machine learning.

**c) Random Forest**

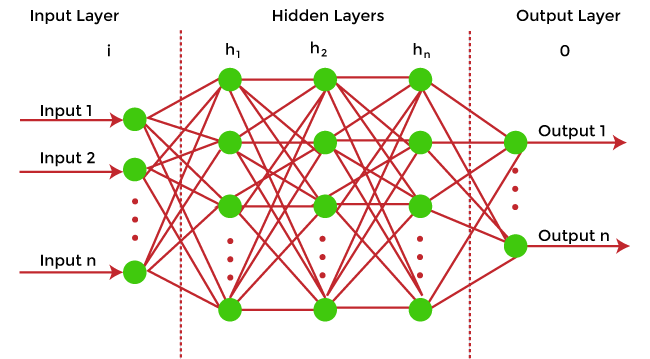
Random Forest is the ensemble learning method, which consists of a large number of decision trees. Each decision tree in a random forest predicts an outcome, and the prediction with the majority of votes is considered as the outcome.

A random forest model can be used for both regression and classification problems.

For the classification task, the outcome of the random forest is taken from the majority of votes. Whereas in the regression task, the outcome is taken from the mean or average of the predictions generated by each tree.

**d) Neural Networks**

Neural networks are the subset of machine learning and are also known as artificial neural networks. Neural networks are made up of artificial neurons and designed in a way that resembles the human brain structure and working. Each artificial neuron connects with many other neurons in a neural network, and such millions of connected neurons create a sophisticated cognitive structure.



Neural networks consist of a multilayer structure, containing one input layer, one or more hidden layers, and one output layer. As each neuron is connected with another neuron, it transfers data from one layer to the other neuron of the next layers. Finally, data reaches the last layer or output layer of the neural network and generates output.

Neural networks depend on training data to learn and improve their accuracy. However, a perfectly trained & accurate neural network can cluster data quickly and become a powerful machine learning and AI tool. One of the best-known neural networks is **Google's search algorithm.**

### Classification

Classification models are the second type of Supervised Learning techniques, which are used to generate conclusions from observed values in the categorical form. For example, the classification model can identify if the email is spam or not; a buyer will purchase the product or not, etc. Classification algorithms are used to predict two classes and categorize the output into different groups.

In classification, a classifier model is designed that classifies the dataset into different categories, and each category is assigned a label.

There are two types of classifications in machine learning:

* **Binary classification**: If the problem has only two possible classes, called a binary classifier. For example, cat or dog, Yes or No,
* **Multi-class classification**: If the problem has more than two possible classes, it is a multi-class classifier.

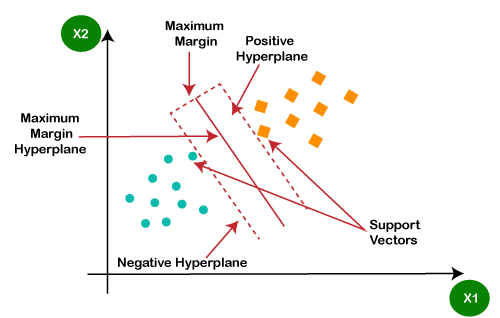
Some popular classification algorithms are as below:

**a) Logistic Regression**

Logistic Regression is used to solve the classification problems in machine learning. They are similar to linear regression but used to predict the categorical variables. It can predict the output in either Yes or No, 0 or 1, True or False, etc. However, rather than giving the exact values, it provides the probabilistic values between 0 & 1.

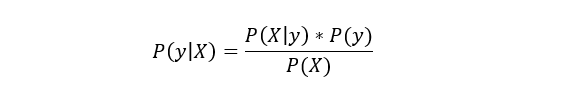
**b) Support Vector Machine**

Support vector machine or SVM is the popular machine learning algorithm, which is widely used for classification and regression tasks. However, specifically, it is used to solve classification problems. The main aim of SVM is to find the best decision boundaries in an N-dimensional space, which can segregate data points into classes, and the best decision boundary is known as Hyperplane. SVM selects the extreme vector to find the hyperplane, and these vectors are known as support vectors.



**c) Naïve Bayes**

Naïve Bayes is another popular classification algorithm used in machine learning. It is called so as it is based on Bayes theorem and follows the naïve(independent) assumption between the features which is given as:

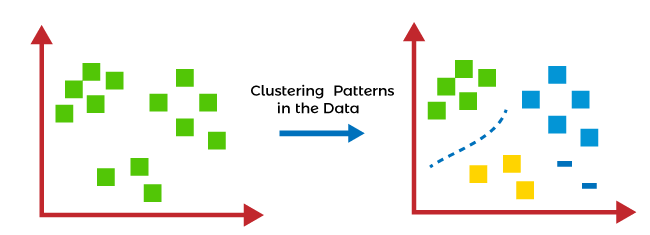


Each naïve Bayes classifier assumes that the value of a specific variable is independent of any other variable/feature. For example, if a fruit needs to be classified based on color, shape, and taste. So yellow, oval, and sweet will be recognized as mango. Here each feature is independent of other features.

## 2. Unsupervised Machine learning models

Unsupervised Machine learning models implement the learning process opposite to supervised learning, which means it enables the model to learn from the unlabeled training dataset. Based on the unlabeled dataset, the model predicts the output. Using unsupervised learning, the model learns hidden patterns from the dataset by itself without any supervision.

Unsupervised learning models are mainly used to perform three tasks, which are as follows:

* **Clustering**  
  Clustering is an unsupervised learning technique that involves clustering or groping the data points into different clusters based on similarities and differences. The objects with the most similarities remain in the same group, and they have no or very few similarities from other groups.  
  Clustering algorithms can be widely used in different tasks such as **Image segmentation, Statistical data analysis, Market segmentation**, etc.  
  Some commonly used Clustering algorithms are K-means Clustering, hierarchal Clustering, DBSCAN, etc.  
  
* **Association Rule Learning**  
  Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.  
  Some popular algorithms of Association rule learning are **Apriori Algorithm, Eclat, FP-growth algorithm.**
* **Dimensionality Reduction**  
  The number of features/variables present in a dataset is known as the dimensionality of the dataset, and the technique used to reduce the dimensionality is known as the dimensionality reduction technique.  
  Although more data provides more accurate results, it can also affect the performance of the model/algorithm, such as overfitting issues. In such cases, dimensionality reduction techniques are used.  
  "**It is a process of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information**."  
  Different dimensionality reduction methods such **as PCA(Principal Component Analysis), Singular Value Decomposition, etc.**

## Reinforcement Learning

In reinforcement learning, the algorithm learns actions for a given set of states that lead to a goal state. It is a feedback-based learning model that takes feedback signals after each state or action by interacting with the environment. This feedback works as a reward (positive for each good action and negative for each bad action), and the agent's goal is to maximize the positive rewards to improve their performance.

The behavior of the model in reinforcement learning is similar to human learning, as humans learn things by experiences as feedback and interact with the environment.

Below are some popular algorithms that come under reinforcement learning:

* **Q-learning:** Q-learning is one of the popular model-free algorithms of reinforcement learning, which is based on the Bellman equation.

It aims to learn the policy that can help the AI agent to take the best action for maximizing the reward under a specific circumstance. It incorporates Q values for each state-action pair that indicate the reward to following a given state path, and it tries to maximize the Q-value.

* **State-Action-Reward-State-Action (SARSA):** SARSA is an On-policy algorithm based on the Markov decision process. It uses the action performed by the current policy to learn the Q-value. The SARSA algorithm stands **for State Action Reward State Action, which symbolizes the tuple (s, a, r, s', a').**
* **Deep Q Network:** DQN **or Deep Q Neural network is Q-learning** within the neural network. It is basically employed in a big state space environment where defining a Q-table would be a complex task. So, in such a case, rather than using Q-table, the neural network uses Q-values for each action based on the state.

## Training Machine Learning Models

Once the Machine learning model is built, it is trained in order to get the appropriate results. To train a machine learning model, one needs a huge amount of pre-processed data. Here pre-processed data means data in structured form with reduced null values, etc. If we do not provide pre-processed data, then there are huge chances that our model may perform terribly.

## How to choose the best model?

In the above section, we have discussed different machine learning models and algorithms. But one most confusing question that may arise to any beginner that "which model should I choose?". So, the answer is that it depends mainly on the business requirement or project requirement. Apart from this, it also depends on associated attributes, the volume of the available dataset, the number of features, complexity, etc. However, in practice, it is recommended that we always start with the simplest model that can be applied to the particular problem and then gradually enhance the complexity & test the accuracy with the help of parameter tuning and cross-validation.

## Difference between Machine learning model and Algorithms

One of the most confusing questions among beginners is that are machine learning models, and algorithms are the same? Because in various cases in machine learning and data science, these two terms are used interchangeably.

The answer to this question is No, and the machine learning model is not the same as an algorithm. In a simple way, an **ML algorithm is like a procedure or method that runs on data to discover patterns from it** and generate the model. At the same time, a **machine learning model is like a computer program that generates output or makes predictions**. More specifically, when we train an algorithm with data, it becomes a model.

# Interpretability in Machine Learning

## Why we need to understand how our models make predictions

Should we always trust a model that performs well? A model could reject your application for a mortgage or diagnose you with cancer. The consequences of these decisions are serious and, even if they are correct, we would expect an explanation. A human would be able to tell you that your income is too low for a mortgage or that a specific cluster of cells is likely malignant. A model that provided similar explanations would be more useful than one that just provided predictions.



Source: [flaticon](https://www.flaticon.com/premium-icon/cyborg_901032" \t "_blank)

By obtaining these explanations, we say we are interpreting a machine learning model. In the rest of this article, we’ll explain in more detail what is meant by interpretability. We’ll then move on to the importance and benefits of being able to interpret your models. There are, however, still some downsides. We’ll end off by discussing these and why, in some cases, you may prefer a less interpretable model.

# What do we mean by interpretability?

[In a previous article,](https://towardsdatascience.com/interperable-vs-explainable-machine-learning-1fa525e12f48) I discuss the concept of model interpretability and how it relates to interpretable and explainable machine learning. To summarise, interpretability is the degree to which a model can be understood in human terms. Model A is more interpretable than model B if it is easier for a human to understand how model A makes predictions. For example, a Convolutional Neural Network is less interpretable than a Random Forest which is less interpretable than a Decision Tree.

With this in mind, we say a model is an interpretable model if it can be understood without any other aids/techniques. Interpretable models are highly interpretable. In comparison, explainable models are too complicated to be understood without the help of additional techniques. We say these models have low interpretability. We can see how these concepts are related in Figure 1. Generally, models can be classified as either interpretable or explainable but there are grey areas where people would disagree.

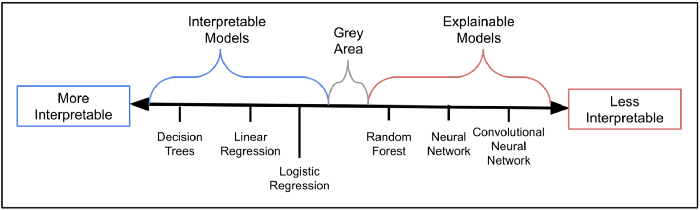
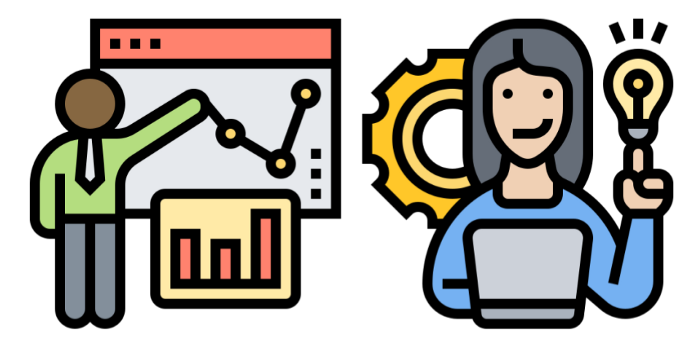


Figure 1: The Interpretability Spectrum (Source: Author)

# Why is interpretability important?

As mentioned, we require additional techniques, such as [feature importance](https://machinelearningmastery.com/calculate-feature-importance-with-python/#:~:text=Feature%20importance%20refers%20to%20techniques,at%20predicting%20a%20target%20variable.) or [LIME](https://github.com/marcotcr/lime), to understand how explainable models work. Implementing these techniques can be a lot of effort and, importantly, they only provide approximations for how a model works. So, we cannot be completely certain that we understand an explainable model. We can have a similar situation when comparing interpretable models.



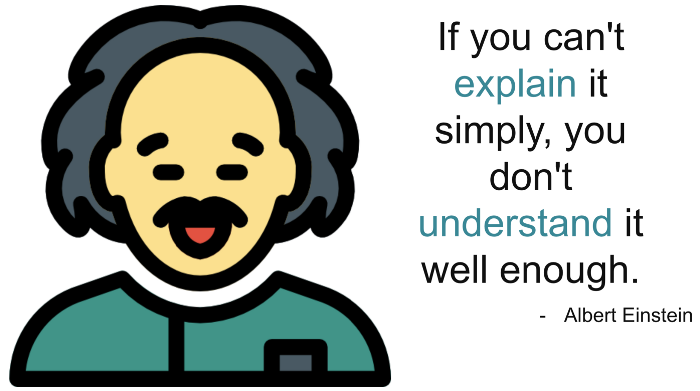
Source: [flaticon](https://www.flaticon.com/free-icon/interpretation_1935181" \t "_blank)

For example, logistic regression and decision trees. Neither of these requires additional techniques but logistic regression may still require more effort to interpret. We would need an understanding of the sigmoid function and how coefficients are related to odds/probability. This complexity may also lead to errors in our interpretations. In general, the more interpretable a model; the easier it is to understand and the more certain we can be that our understanding is correct. Interpretability is important because of the many benefits that flow from this.

## Easier to explain

Our first benefit is that interpretable models are easier to explain to other people. For any topic, the better we understand it the easier it is to explain. We should also be able to explain it in simple terms (i.e. without mentioning the technical details). In industry, there are many people who may expect simple explanations for how your model works. These people will not necessarily have technical backgrounds or experience with machine learning.

For example, suppose we have created a model that predicts whether someone will make a life insurance claim. We want to use this model to automate life insurance underwriting at our company. To sign off on the model, our boss would require a detailed explanation of how it works. A disgruntled customer may rightly demand an explanation for why they were not approved for life cover. A regulator could even require such an explanation by law.



Source: [flaticon](https://www.flaticon.com/free-icon/einstein_892712" \t "_blank)

Trying to explain to these people how a neural network makes predictions may cause a lot of confusion. Due to the uncertainty, they may not even accept the explanation. In comparison, interpretable models like logistic regression can be understood in human terms. This means they can be explained in human terms. For example, we could explain precisely how much the customer’s smoking habit has increased their probability of dying.

## Easier to sense check and fix errors

The relationship described above is causal (i.e. smoking causes cancer/death). In general, machine learning models only care about associations. For example, a model could use someone’s country of origin to predict if they had skin cancer. However, as with smoking, can we say someone’s country causes cancer? The reason for this is that skin cancer is caused by sunshine and some countries are just sunnier than others. So we can only say skin cancer is associated with certain countries.

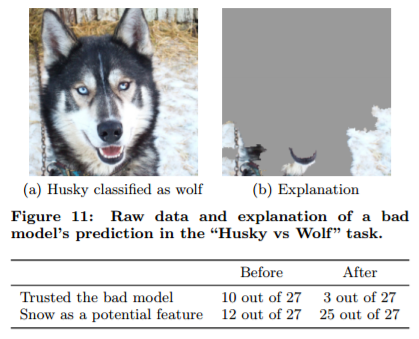
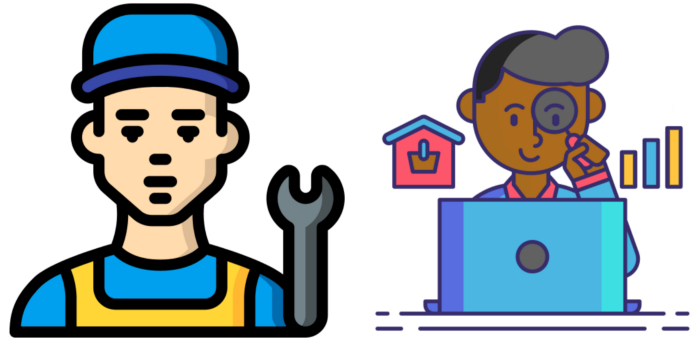


Figure 2: Wolf vs husky experiment (Source: [M. Tulio Ribeiro, S. Singh & C. Guestrin](https://arxiv.org/abs/1602.04938))

A good example of where associations can go wrong comes from an experiment performed by researches at the University of Washington. The researches trained an image recognition model to classify animals as a husky or wolf. Using LIME, they tried to understand how their model made predictions. In Figure 2, we can see that the model was basing its predictions on image backgrounds. If the background had snow, the animal was always classified as a wolf. They had essentially built a model that detects snow.

The issue is that wolves are associated with snow. Wolves will usually be found in the snow whereas huskies are not. What this example shows us is that models can, not only make incorrect predictions, but they can also make correct predictions in the wrong way. As data scientists, we need to sense check our models to make sure they are not making predictions in this way. The more interpretable your model the easier this is to do.



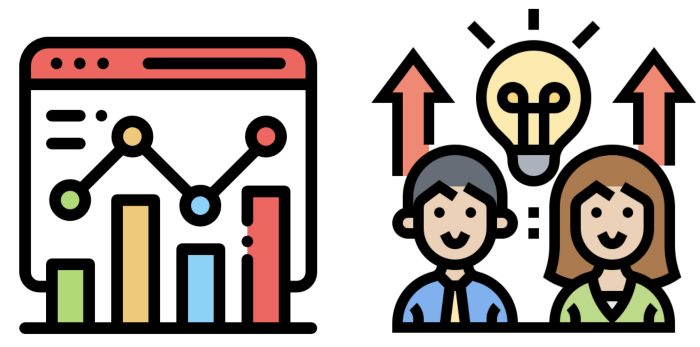
Source: [flaticon](https://www.flaticon.com/premium-icon/man_3418023" \t "_blank)

## Easier to determine future performance

As time goes on, a model’s prediction power may deteriorate. This is because relationships between model features and the target variable can change. For example, due to the wage gap, income may currently be a good predictor of gender. As society becomes more equal, income would lose its predictive strength. We need to be aware of these potential changes and their impact on our models. This is harder to do for explainable models. As it is less clear how features are being used, even if we know the impact on individual features, we may not be able to tell the impact on the model as a whole.

## Easier to learn from the model

It is human nature to try to find meaning in the unknown. Machine learning can help us discover patterns in our data we didn’t know existed. However, we cannot identify these patterns by just looking at the model’s predictions. Any lessons are lost if we can not interpret our model. Ultimately, the less interpretable a model the harder it is to learn from it.



Source: [flaticon](https://www.flaticon.com/premium-icon/cooperation_3317601" \t "_blank)

## Algorithm Fairness

It is important that your models make unbiased decisions so that they do not perpetuate any historical injustices. Identifying sources of bias can be difficult. It often comes from associations between model features and protected variables (e.g. race or gender). For example, due to a history of forced segregation in South Africa, race is highly associated with someones’ location\neighbourhood. Location can act as a proxy for race. A model that uses location may be biased towards a certain race.

Using an interpretable model will not necessarily mean that you will have an unbiased model. It also does not mean that it will be easier to determine if the model is fair or not. This is because most measures of fairness (e.g. false positive rate, disparate impact) are model agnostic. They are just as easy to calculate for any model. What using an interpretable model does do is make it easier to identify and correct the source of bias. We know what features are being used and we can check which of these are associated with the protected variables.



Source: [flaticon](https://www.flaticon.com/free-icon/fair_3260927?term=fairness&page=1&position=14" \t "_blank)

# Downsides to interpretability

Okay, we get it… interpretable models are great. They are easier to understand, explain and learn from. They also allow us to better sense check current performance, future performance and model fairness. There are however downsides to interpretability and situations where we would prefer an explainable model.

## Open to manipulation

Systems based on ML are open to manipulation or fraud. For example, suppose we have a system that automatically gives out car loans. An important feature could be the number of credit cards. The more cards a customer has the risker she is. If a customer knew this they could temporarily cancel all their cards, take out a car loan and then reapply for all the credit cards.



Source: [flaticon](https://www.flaticon.com/free-icon/hacker_3380378" \t "_blank)

The probability of the customer repaying the loan does not change when she cancels her cards. The customer has manipulated the model to make an incorrect prediction. The more interpretable a model the more transparent and easy it is to manipulate. This is the case even if the inner working of a model are kept secret. The relationships between features and target variable are usually simpler making them easier to guess.

## Less to learn

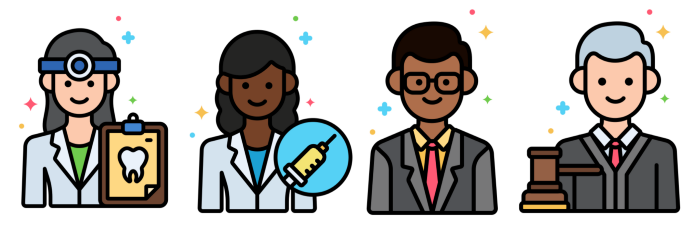
We mentioned that interpretable models are easier to learn from. The flip side is that they are less likely to teach us something new. An explainable model like a neural network can automatically model interactions and non-linear relationships in data. By interpreting these models we can uncover these relationships that we never knew existed.

In comparison, algorithms like linear regression can only model linear relationships. To model non-linear relationships, we would have to use feature engineering to include any relevant variable in our dataset. This would require prior knowledge of the relationships defeating the purpose of interpreting the model.

## Domain knowledge/ expertise requirement

Building interpretable models can require significant domain knowledge and expertise. Generally, interpretable models, like regression, can only model linear relationships in your data. To model non-linear relationships we have to perform feature engineering. For example, for a medical diagnosis model, we may want to calculate BMI using height and weight. Knowing what features will be predictive and, therefore, what features to create requires domain knowledge in a particular field.

Your team may not have this knowledge. Alternatively, you could use an explainable model which will automatically model non-linear relationships in your data. This removes the need to create any new features; essentially leaving the thinking up to the computer. The downside, as we’ve discussed thoroughly above, is a poorer understanding of how the features are being used to make predictions.



## Complexity-Accuracy Trade-off

What we can see from the above is that, generally, the less complicated a model the more interpretable. So, for higher interpretability, there can be the trade-off of lower accuracy. This is because, in some cases, simpler models can make less accurate predictions. This really depends on the problem you are trying to solve. For instance, you would get poor results using logistic regression to do image recognition.

**Evaluating Performance of a Model :**

we will discuss the various ways to check the performance of our machine learning or deep learning model and why to use one in place of the other. We will discuss terms like:

1. Confusion matrix
2. Accuracy
3. Precision
4. Recall
5. Specificity
6. F1 score
7. Precision-Recall or PR curve
8. **ROC** (**R**eceiver **O**perating **C**haracteristics) curve
9. PR vs ROC curve.

For simplicity, we will mostly discuss things in terms of a binary classification problem where let’s say we’ll have to find if an image is of a cat or a dog. Or a patient is having cancer (positive) or is found healthy (negative). Some common terms to be clear with are:

**True positives (TP)**: Predicted positive and are actually positive.

**False positives (FP)**: Predicted positive and are actually negative.

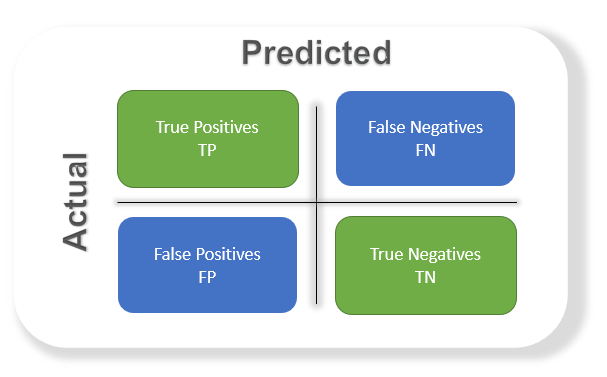
**True negatives (TN)**: Predicted negative and are actually negative.

**False negatives (FN)**: Predicted negative and are actually positive.

So let's get started!

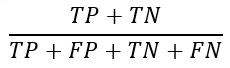
# ****Confusion matrix****

It’s just a representation of the above parameters in a matrix format. Better visualization is always good :)



# ****Accuracy****

The most commonly used metric to judge a model and is actually not a clear indicator of the performance. The worse happens when classes are imbalanced.

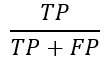


Take for example a cancer detection model. The chances of actually having cancer are very low. Let’s say out of 100, 90 of the patients don’t have cancer and the remaining 10 actually have it. We don’t want to miss on a patient who is having cancer but goes undetected (false negative). Detecting everyone as not having cancer gives an accuracy of 90% straight. The model did nothing here but just gave cancer free for all the 100 predictions.

We surely need better alternatives.

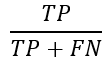
# ****Precision****

Percentage of positive instances out of the **total predicted positive**instances. Here denominator is the model prediction done as positive from the whole given dataset. Take it as to find out ‘how much the model is right when it says it is right’.



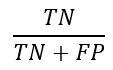
# ****Recall/Sensitivity/True Positive Rate****

Percentage of positive instances out of the**total actual positive** instances. Therefore denominator (TP + FN) here is the actual number of positive instances present in the dataset. Take it as to find out ‘how much extra right ones, the model missed when it showed the right ones’.



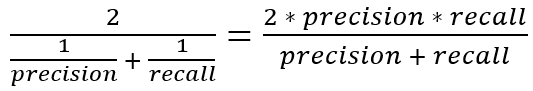
# ****Specificity****

Percentage of negative instances out of the**total actual negative** instances. Therefore denominator (TN + FP) here is the actual number of negative instances present in the dataset. It is similar to recall but the shift is on the negative instances. Like finding out how many healthy patients were not having cancer and were told they don’t have cancer. Kind of a measure to see how separate the classes are.



# ****F1 score****

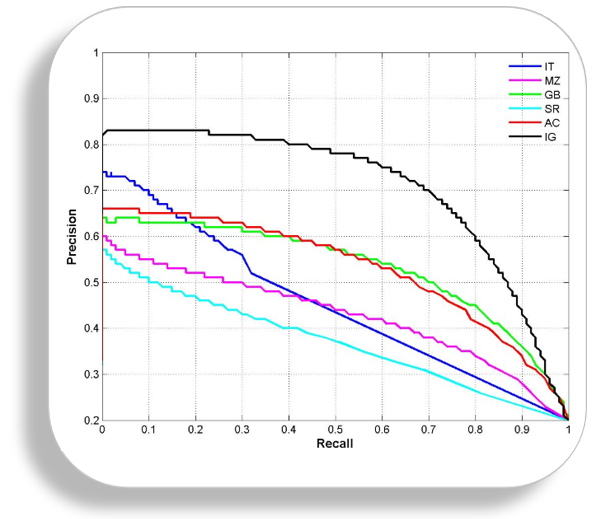
Itis the harmonic mean of precision and recall. This takes the contribution of both, so higher the F1 score, the better. See that due to the product in the numerator if one goes low, the final F1 score goes down significantly. So a model does well in F1 score if the positive predicted are actually positives (precision) and doesn't miss out on positives and predicts them negative (recall).



One drawback is that both precision and recall are given equal importance due to which according to our application we may need one higher than the other and F1 score may not be the exact metric for it. Therefore either weighted-F1 score or seeing the PR or ROC curve can help.

# ****PR curve****

It is the curve between precision and recall for various threshold values. In the figure below we have 6 predictors showing their respective precision-recall curve for various threshold values. The top right part of the graph is the ideal space where we get high precision and recall. Based on our application we can choose the predictor and the threshold value. PR AUC is just the area under the curve. The higher its numerical value the better.

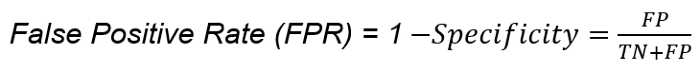


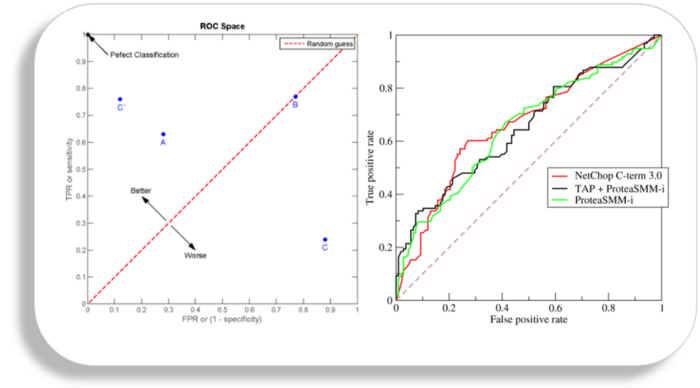
[image link](https://github.com/MenuPolis/MLT/wiki/PR-Curve)

# ****ROC curve****

ROC stands for receiver operating characteristic and the graph is plotted against TPR and FPR for various threshold values. As TPR increases FPR also increases. As you can see in the first figure, we have four categories and we want the threshold value that leads us closer to the top left corner. Comparing different predictors (here 3) on a given dataset also becomes easy as you can see in figure 2, one can choose the threshold according to the application at hand. ROC AUC is just the area under the curve, the higher its numerical value the better.





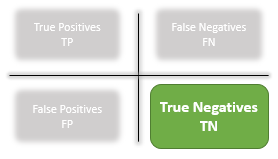


[wiki link](https://en.wikipedia.org/wiki/Receiver_operating_characteristic)

# PR vs ROC curve

Both the metrics are widely used to judge a models performance.

**Which one to use PR or ROC?**



The answer lies in TRUE NEGATIVES.

**Due to the absence of TN in the precision-recall equation, they are useful in imbalanced classes**. In the case of class imbalance when there is a majority of the negative class. The metric doesn’t take much into consideration the high number of TRUE NEGATIVES of the negative class which is in majority, giving better resistance to the imbalance. This is important when the detection of the positive class is very important.

Like to detect cancer patients, which has a high class imbalance because very few have it out of all the diagnosed. We certainly don’t want to miss on a person having cancer and going undetected (recall) and be sure the detected one is having it (precision).

**Due to the consideration of TN or the negative class in the ROC equation, it is useful when both the classes are important to us.**Like the detection of cats and dog. The importance of true negatives makes sure that both the classes are given importance, like the output of a CNN model in determining the image is of a cat or a dog.

# Conclusion

The evaluation metric to use depends heavily on the task at hand. For a long time, accuracy was the only measure I used, which is really a vague option. I hope this blog would have been useful for you. That's all from my side. Feel free to suggest corrections and improvements.

**Improving Performance of a Model:**

Machine learning development would be not difficult for ML engineers, but ensuring its performance is important to get accurate and most reliable results. Though, there are various methods you can improve your machine learning model performance.

Basically developed on python, machine learning models need to develop while considering the various factors that affect its performance. But right here we brought the list of most important parameters that you can consider while developing the ML model.

## ****5 Ways to Improve Performance of ML Models****

### ****1. Choosing the Right Algorithms****

Algorithms are the key factor used to train the ML models. The data feed into this that helps the model to learn from and predict with accurate results. Hence, choosing the right algorithm is important to ensure the performance of your machine learning model.

Linear Regression, Logistic Regression, Decision Tree, SVM, Naive Bayes, kNN, K-Means, Random Forest and Dimensionality Reduction Algorithms and Gradient Boosting are the leading ML algorithms you can choose as per your ML model compatibility.

### ****2. Use the Right Quantity of Data****

The next important factor you can consider while developing a machine learning model is choosing the right quantity of data sets. And there are multirole factors and for deep learning-based ML models, a huge quantity of datasets is required for algorithms.

Depending on the complexities of problem and learning algorithms, model skill, data size evaluation and use of statistical heuristic rule are the leading factors determine the quantity and [**types of training data sets**](https://www.cogitotech.com/blog/what-are-the-various-types-of-data-sets-used-in-machine-learning/) that help in improving the performance of the model.

### ****3. Quality of Training Data Sets****

Just like quantity, the quality of machine learning training data set is another key factor, you need to keep in mind while developing an ML model. If the quality of [**machine learning training data sets**](https://www.cogitotech.com/services/machine-learning/) is not good or accurate your model will never give accurate results, affecting the overall performance of the model not suitable to use in real-life.

Actually, there are different methods to measure the quality of the training data set. Standard quality-assurance methods and detailed for in-depth quality assessment are the leading two popular methods you can use to ensure the quality of data sets. Quality of data is important to get unbiased decisions from the ML models, so you need to make sure to use the right quality of training data sets to improve the performance of your ML model.

### ****4. Supervised or Unsupervised ML****

Moreover, the above-discussed ML algorithms, the performance of such AI-based models are affected by methods or process of machine learning. And supervised, unsupervised and reinforcement learning are the algorithm consist of a target/outcome variable (or dependent variable) which is to be predicted from a given set of predictors (independent variables).

In unsupervised machine learning, a model is given any target or outcome variable to predict/estimate. And, it is used for clustering population in different groups, which is widely used for segmenting customers in different groups for specific intervention. For supervised ML, labeled or annotated data is required, while for unsupervised ML the approach is different.

Similarly, reinforcement Learning is another important algorithm, used to train the model to make specific decisions. In this training process, the machine learns from previous experiences and tries to store the best suitable knowledge for the right predictions.

### ****5. Model Validation and Testing****

Building a machine learning model is not enough to get the right predictions, as you have to check the accuracy and need to validate the same to ensure get the precise results. And validating the model will improve the performance of the ML model.

Actually, there are various types of validation techniques you can follow but you need to make sure choose the best one that is suitable for your [**ML model validation**](https://www.cogitotech.com/ml-model-validation-services/) and help you to improve the overall performance of your ML model and predict in an unbiased manner. Similarly, testing of the model is also important to ensure its accuracy and performance.

#### ****Summing-up****

**Improving machine learning model performance** will not only make the model predict in an unbiased manner but make it one of the most reliable and acceptable in the AI world. Hence, a machine learning engineer and data scientist need to make sure all these points while working on such models to improve the overall performance of the AI model.

Anolytics, is one the leading [**data annotation company**](https://www.anolytics.ai/), provides the training data sets for machine learning models. You will get right here the best data sets for different types of machine learning models and improve their performance. **Anolytics**, is skilled in [**image annotation services**](https://www.anolytics.ai/image-annotation-services/) to annotate the images and provide high-quality training data for computer vision-based AI models for different fields like [**healthcare**](https://www.anolytics.ai/solutions/healthcare/), [**retail**](https://www.anolytics.ai/solutions/retail/), [**automotive**](https://www.anolytics.ai/solutions/self-driving/) and [**agriculture**](https://www.anolytics.ai/solutions/agriculture/), etc.

**Feature Engineering: Feature Transformation** **:**

The life cycle of the [Machine Learning](https://towardsai.net/ai/machine-learning) model can be broken down into the following steps.

1. Data Collection
2. Data Preprocessing
3. Feature Engineering
4. [Feature Selection](https://towardsai.net/p/data-science/principal-component-analysis-pca-with-python-examples-tutorial-67a917bae9aa)
5. Model Building
6. Hyper Parameter Tuning
7. Model Deployment

To build a model we have to preprocess data. Feature Transformation is one of the most important tasks in this process. In the [dataset](https://towardsai.net/p/machine-learning/best-datasets-for-machine-learning-and-data-science-d80e9f030279), there will be data with different magnitudes most of the time. So to make our predictions better we have to scale down different features to the same range of magnitude or some specific data distribution. because most of the [algorithms](https://towardsai.net/p/machine-learning/machine-learning-algorithms-for-beginners-with-python-code-examples-ml-19c6afd60daa) will give more importance to the features with high volume rather than giving the same importance to all features. This will lead to wrong predictions and faulty models and we don’t want that.

1. In [algorithms](https://towardsai.net/p/machine-learning/machine-learning-algorithms-for-beginners-with-python-code-examples-ml-19c6afd60daa) like K-Nearest- Neighbors, [SVM](https://towardsai.net/p/machine-learning/support-vector-machine-svm-introduction-machine-learning-8c56b7da63f1), and K-means which are Distance-based algorithms, They will give more weightage to features with large values because the distance is calculated with values of data points to find out similarities between them. If we feed the algorithm unscaled features the prediction will suffer dramatically.
2. Because we are scaling features down, the area of calculations will also be brought down to a smaller range and a little less time will be spent on calculating. Hence faster model training.
3. In the algorithms like [linear regression](https://towardsai.net/p/machine-learning/calculating-simple-linear-regression-and-linear-best-fit-an-in-depth-tutorial-with-math-and-python-804a0cb23660) and logistic regression which work upon gradient-descent optimization, Feature scaling becomes crucial because if we feed data with different magnitudes, it will be way too hard to converge to Global Minima. With values of the same range, it becomes less burden for algorithms to learn.

Most of the ensemble method doesn’t require feature scaling because even if we perform feature transformation, the depth of distribution wouldn’t change much. So in such algorithms, there is no need for scaling unless specifically needed.

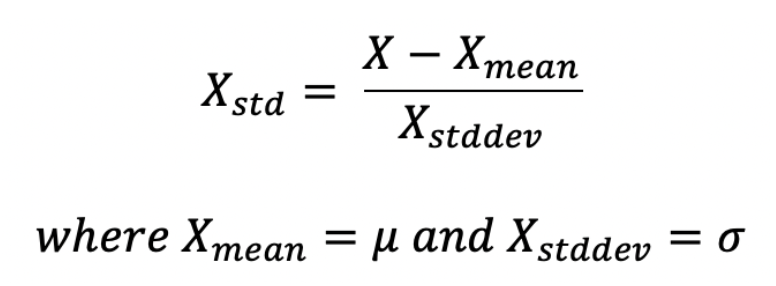
#### There are so many types of feature transformation methods, we will talk about the most useful and popular ones.

1. Standardization
2. Min — Max Scaling/ Normalization
3. Robust Scaler
4. Logarithmic Transformation
5. Reciprocal Transformation
6. Square Root Translation
7. Box-Cox Transformation
8. Johnson transformation

Standardization should be used when the features of the input [dataset](https://towardsai.net/p/machine-learning/best-datasets-for-machine-learning-and-data-science-d80e9f030279) have large differences between ranges or when they are measured in different measurements units like Height, Weight, Meters, Miles, etc.

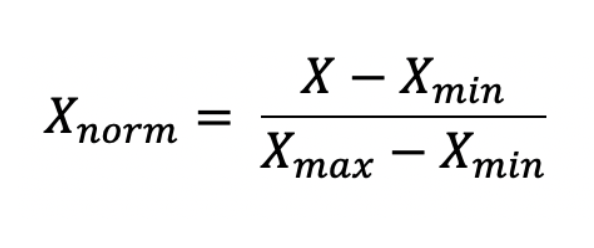
We bring all the variables or features to a similar scale. Where the mean is 0 and the Standard Deviation is 1.

In Standardization, we subtract feature values by their mean and then divide by standard deviation which gives exactly standard normal distribution.



In simple terms, min-max scaling brings down feature values to a range of 0 to 1. Until we specify the range we want it to be scaled down to.

In Normalization, we subtract the feature value by its minimum value and then divide it by the range of features (range of feature= maximum value of feature — minimum value of feature).



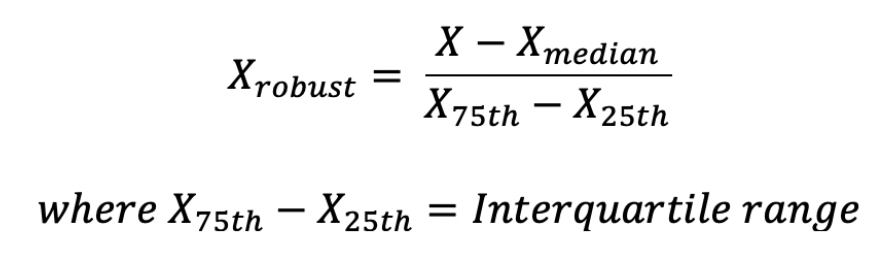
If the dataset has too many outliers, both Standardization and Normalization can be hard to depend on, in such case you can use Robust Scaler for feature scaling.

You can also say Robust Scaler is robust to outliers 😂.

It scales values using median and interquartile range therefore it doesn’t get affected by very large or very small values of features.

The robust scaler subtracts feature values by their median and then divides by its IQR.

* 25th percentile = 1st quartile
* 50th percentile = 2nd quartile (also called the median)
* 75th percentile = 3rd quartile
* 100th percentile = 4th quartile (also called the maximum)
* IQR= Inter Quartile Range
* IQR= 3rd quartile — 1st quartile

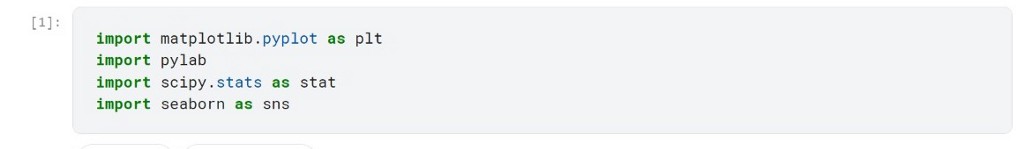


### 4. Gaussian Transformations

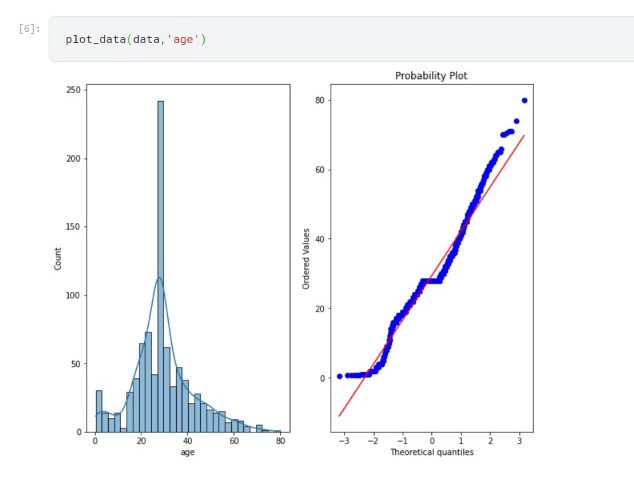
Some [Machine Learning](https://towardsai.net/ai/machine-learning) algorithms like linear and logistic regression assumes that the data we are feeding them is normally distributed. If the data is normally distributed such algorithms tend to perform better and give higher accuracy. Normalizing a skewed distribution becomes an important part here.

But most of the time data would be skewed so You have to transform it into Gaussian distribution using the following algorithms, you might have to try out a few methods before selecting one since different datasets tend to have different requirements and we can’t fit one method to all the data.

We will use this function to plot data throughout the rest of the article. I will be using the age feature only from the titanic dataset to plot histogram and QQ Plot.

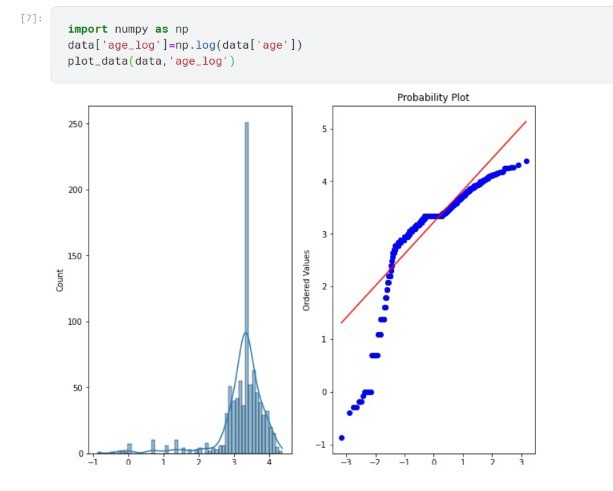


Below the graph is of age feature before feature scaling



### A. Logarithmic Transformation

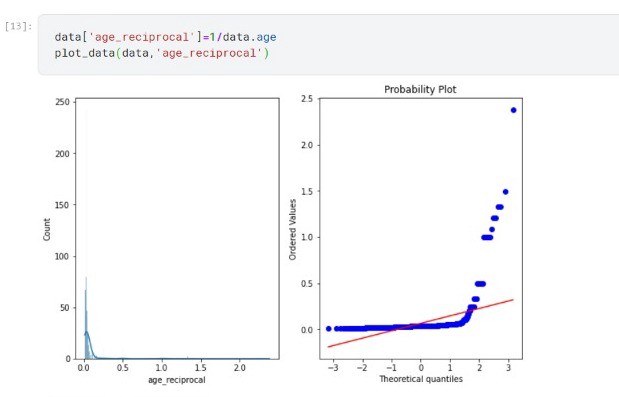
In the Logarithmic Transformation, we will apply log to all values of features using NumPy and store it in the new feature.



Using Log Transformation doesn’t seem to fit very well in this dataset, it even worsens the distribution by making data left-skewed. So we have to rely on other methods to achieve normal distribution.

### B. Reciprocal Transformation

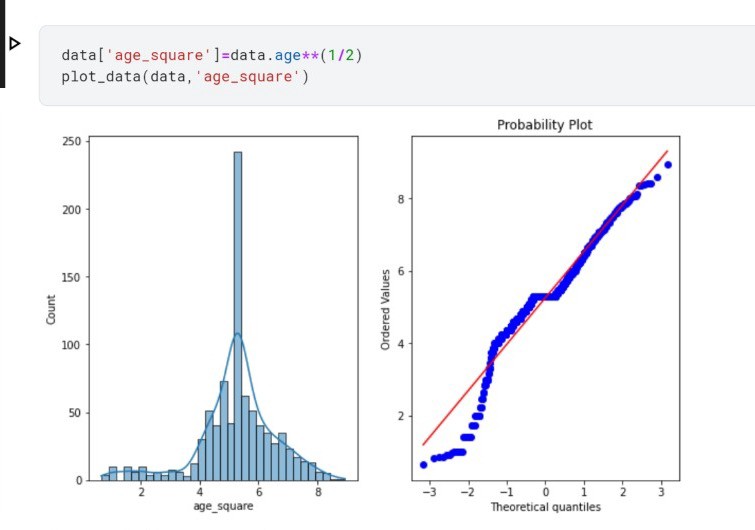
In Reciprocal Transformation, we divide each value of a feature by 1(reciprocal) and store it in the new feature.



Reciprocal Transformation doesn’t work well with this data, It doesn’t give normal distribution instead it made data even more right-skewed.

### C. Square Root Translation

In square root transformation, we raise the values of feature to the power of fraction(1/2) to achieve the square root of a value. We can also use NumPy for this transformation.



Square root transformation seems to perform better than reciprocal and log transformation with this data but yet it is a bit left-skewed.

### D. Box-Cox Transformation

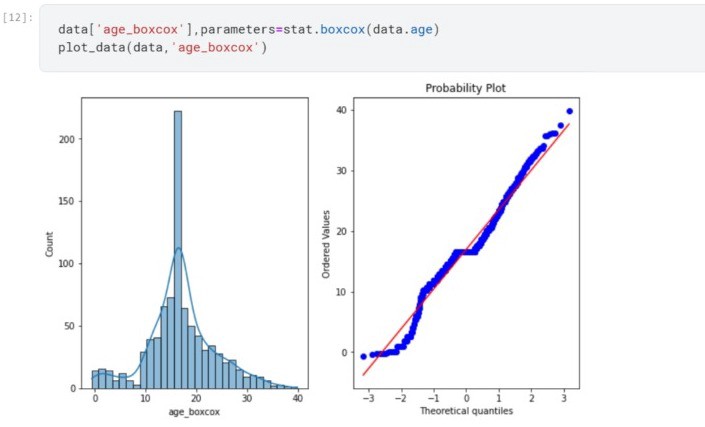
Box-Cox transformation is one of the most useful scaling techniques to transfer data distribution in a normal distribution.

The Box-Cox transformation can be defined as:

T(Y)=(Y exp(λ)−1)/λ

Where Y is the response variable and λ is the transformation parameter. λ varies from -5 to 5. In the transformation, all values of λ are considered and the optimal value for a given variable is selected.

We can calculate box cox transformation using stats from the SciPy module.



So far box cox transformation seems to be the best fit for the age feature to transform.

### Conclusion

There are other techniques also to perform to get Gaussian distribution but most of the time one of these methods seems to fit well on the dataset.

# Feature Subset Selection Process:

Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features.

While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning.

Feature selection is one of the important concepts of machine learning, which highly impacts the performance of the model. As machine learning works on the concept of "Garbage In Garbage Out", so we always need to input the most appropriate and relevant dataset to the model in order to get a better result.

In this topic, we will discuss different feature selection techniques for machine learning. But before that, let's first understand some basics of feature selection.

* **What is Feature Selection?**
* **Need for Feature Selection**
* **Feature Selection Methods/Techniques**
* **Feature Selection statistics**

## What is Feature Selection?

**A** feature is an attribute that has an impact on a problem or is useful for the problem, and choosing the important features for the model is known as feature selection. Each machine learning process depends on feature engineering, which mainly contains two processes; which are Feature Selection and Feature Extraction. Although feature selection and extraction processes may have the same objective, both are completely different from each other. The main difference between them is that feature selection is about selecting the subset of the original feature set, whereas feature extraction creates new features. Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

So, we can define feature Selection as, "**It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building**." Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.

## Need for Feature Selection

Before implementing any technique, it is really important to understand, need for the technique and so for the Feature Selection. As we know, in machine learning, it is necessary to provide a pre-processed and good input dataset in order to get better outcomes. We collect a huge amount of data to train our model and help it to learn better. Generally, the dataset consists of noisy data, irrelevant data, and some part of useful data. Moreover, the huge amount of data also slows down the training process of the model, and with noise and irrelevant data, the model may not predict and perform well. So, it is very necessary to remove such noises and less-important data from the dataset and to do this, and Feature selection techniques are used.

Selecting the best features helps the model to perform well. For example, Suppose we want to create a model that automatically decides which car should be crushed for a spare part, and to do this, we have a dataset. This dataset contains a Model of the car, Year, Owner's name, Miles. So, in this dataset, the name of the owner does not contribute to the model performance as it does not decide if the car should be crushed or not, so we can remove this column and select the rest of the features(column) for the model building.

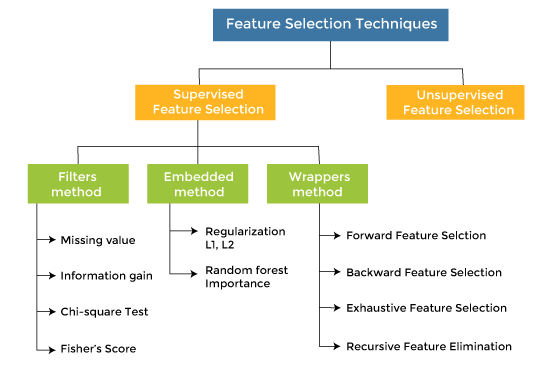
Below are some benefits of using feature selection in machine learning:

* **It helps in avoiding the curse of dimensionality.**
* **It helps in the simplification of the model so that it can be easily interpreted by the researchers.**
* **It reduces the training time.**
* **It reduces overfitting hence enhance the generalization.**

## Feature Selection Techniques

There are mainly two types of Feature Selection techniques, which are:

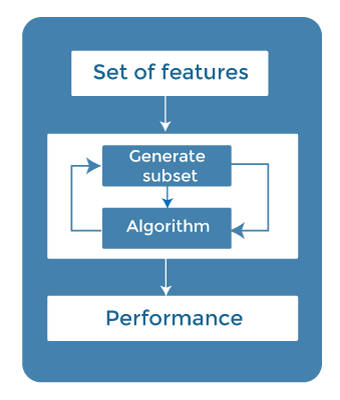
* **Supervised Feature Selection technique**  
  Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
* **Unsupervised Feature Selection technique**  
  Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.



There are mainly three techniques under supervised feature Selection:

### 1. Wrapper Methods

In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations. It trains the algorithm by using the subset of features iteratively.



On the basis of the output of the model, features are added or subtracted, and with this feature set, the model has trained again.

Some techniques of wrapper methods are:

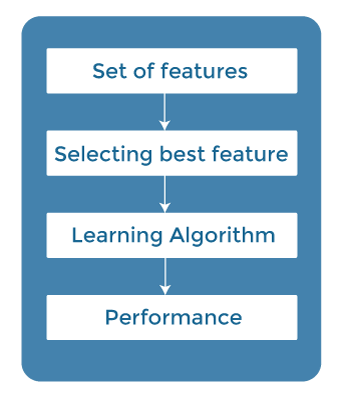
* **Forward selection** - Forward selection is an iterative process, which begins with an empty set of features. After each iteration, it keeps adding on a feature and evaluates the performance to check whether it is improving the performance or not. The process continues until the addition of a new variable/feature does not improve the performance of the model.
* **Backward elimination** - Backward elimination is also an iterative approach, but it is the opposite of forward selection. This technique begins the process by considering all the features and removes the least significant feature. This elimination process continues until removing the features does not improve the performance of the model.
* **Exhaustive Feature Selection-** Exhaustive feature selection is one of the best feature selection methods, which evaluates each feature set as brute-force. It means this method tries & make each possible combination of features and return the best performing feature set.
* **RecursiveFeatureElimination-**  
  Recursive feature elimination is a recursive greedy optimization approach, where features are selected by recursively taking a smaller and smaller subset of features. Now, an estimator is trained with each set of features, and the importance of each feature is determined using coef\_attribute or through a feature\_importances\_attribute.

### 2. Filter Methods

In Filter Method, features are selected on the basis of statistics measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step.

The filter method filters out the irrelevant feature and redundant columns from the model by using different metrics through ranking.

The advantage of using filter methods is that it needs low computational time and does not overfit the data.



Some common techniques of Filter methods are as follows:

* Information Gain
* Chi-square Test
* Fisher's Score
* Missing Value Ratio

**Information Gain:** Information gain determines the reduction in entropy while transforming the dataset. It can be used as a feature selection technique by calculating the information gain of each variable with respect to the target variable.

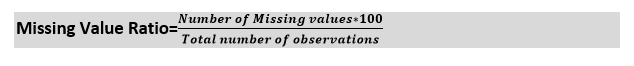
**Chi-square Test:** Chi-square test is a technique to determine the relationship between the categorical variables. The chi-square value is calculated between each feature and the target variable, and the desired number of features with the best chi-square value is selected.

**Fisher's Score:**

Fisher's score is one of the popular supervised technique of features selection. It returns the rank of the variable on the fisher's criteria in descending order. Then we can select the variables with a large fisher's score.

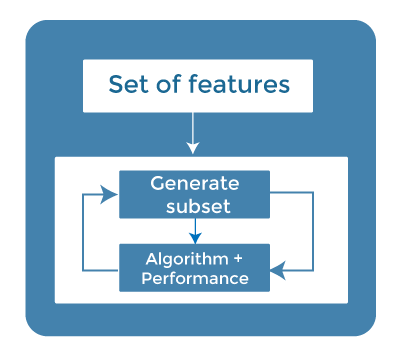
**Missing Value Ratio:**

The value of the missing value ratio can be used for evaluating the feature set against the threshold value. The formula for obtaining the missing value ratio is the number of missing values in each column divided by the total number of observations. The variable is having more than the threshold value can be dropped.



### 3. Embedded Methods

Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost. These are fast processing methods similar to the filter method but more accurate than the filter method.

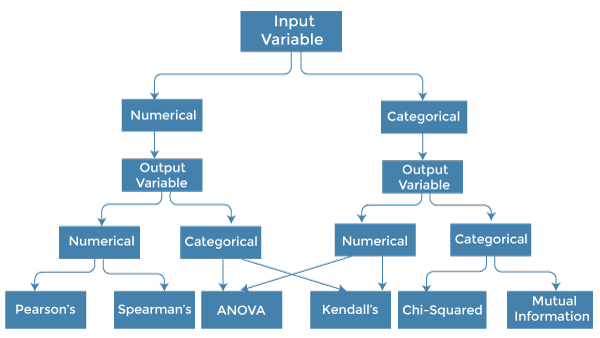


These methods are also iterative, which evaluates each iteration, and optimally finds the most important features that contribute the most to training in a particular iteration. Some techniques of embedded methods are:

* **Regularization**- Regularization adds a penalty term to different parameters of the machine learning model for avoiding overfitting in the model. This penalty term is added to the coefficients; hence it shrinks some coefficients to zero. Those features with zero coefficients can be removed from the dataset. The types of regularization techniques are L1 Regularization (Lasso Regularization) or Elastic Nets (L1 and L2 regularization).
* **Random Forest Importance** - Different tree-based methods of feature selection help us with feature importance to provide a way of selecting features. Here, feature importance specifies which feature has more importance in model building or has a great impact on the target variable. Random Forest is such a tree-based method, which is a type of bagging algorithm that aggregates a different number of decision trees. It automatically ranks the nodes by their performance or decrease in the impurity (Gini impurity) over all the trees. Nodes are arranged as per the impurity values, and thus it allows to pruning of trees below a specific node. The remaining nodes create a subset of the most important features.

## How to choose a Feature Selection Method?

For machine learning engineers, it is very important to understand that which feature selection method will work properly for their model. The more we know the datatypes of variables, the easier it is to choose the appropriate statistical measure for feature selection.



To know this, we need to first identify the type of input and output variables. In machine learning, variables are of mainly two types:

* **Numerical Variables:** Variable with continuous values such as integer, float
* **Categorical Variables:** Variables with categorical values such as Boolean, ordinal, nominals.

Below are some univariate statistical measures, which can be used for filter-based feature selection:

**1. Numerical Input, Numerical Output:**

Numerical Input variables are used for predictive regression modelling. The common method to be used for such a case is the Correlation coefficient.

* Pearson's correlation coefficient (For linear Correlation).
* Spearman's rank coefficient (for non-linear correlation).

**2. Numerical Input, Categorical Output:**

Numerical Input with categorical output is the case for classification predictive modelling problem**s.** In this case, also, correlation-based techniques should be used, but with categorical output.

* **ANOVA correlation coefficient (linear).**
* **Kendall's rank coefficient (nonlinear).**

**3. Categorical Input, Numerical Output:**

This is the case of regression predictive modelling with categorical input. It is a different example of a regression problem. We can use the same measures as discussed in the above case but in reverse order.

**4. Categorical Input, Categorical Output:**

This is a case of classification predictive modelling with categorical Input variables.

The commonly used technique for such a case is Chi-Squared Test. We can also use Information gain in this case.

**We can summarise the above cases with appropriate measures in the below table:**

|  |  |  |
| --- | --- | --- |
| **Input Variable** | **Output Variable** | **Feature Selection technique** |
| Numerical | Numerical | * Pearson's correlation coefficient (For linear Correlation). * Spearman's rank coefficient (for non-linear correlation). |
| Numerical | Categorical | * ANOVA correlation coefficient (linear). * Kendall's rank coefficient (nonlinear). |
| Categorical | Numerical | * Kendall's rank coefficient (linear). * ANOVA correlation coefficient (nonlinear). |
| Categorical | Categorical | * Chi-Squared test (contingency tables). * Mutual Information. |

## Conclusion

Feature selection is a very complicated and vast field of machine learning, and lots of studies are already made to discover the best methods. There is no fixed rule of the best feature selection method. However, choosing the method depend on a machine learning engineer who can combine and innovate approaches to find the best method for a specific problem. One should try a variety of model fits on different subsets of features selected through different statistical Measures.