**Unit 3 Classification**

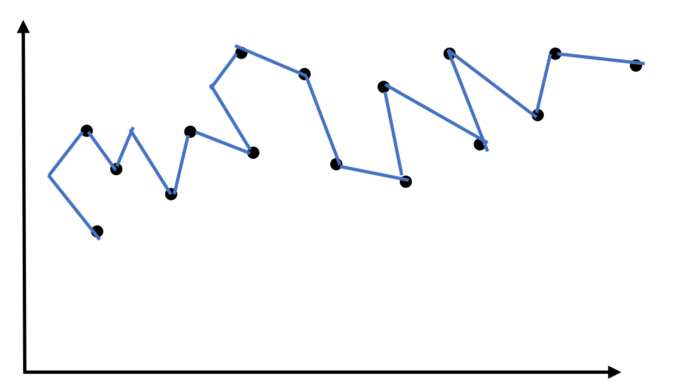
**Topics** : Logistic Regression : Cost function, problem of overfitting, Regularization, Support vector machine : support vector, kernel, K-nearest Neighbor(KNN)

**Logistic Regression**

**Cost Function**

**Problem of Overfitting**

When a model performs very well for training [data](https://www.simplilearn.com/what-is-data-article) but gives a poor performance with test data (new data), it is known as overfitting. Performing very well means giving very good accuracies during training and less accuracy when checked on testing data. Overfitting happens when the model learns the detail and noise in the training data which ultimately leads to negative impacts on the performance of the model on new data. This is because the data model becomes more complex with the presence of noise in a data set. *The noise here means irrelevant or meaningless data like outliers,*[*missing values*](https://www.naukri.com/learning/articles/handling-missing-values-beginners-tutorial/)*, and extra features. So model picks up*noise or random fluctuations in the training data and learns them as concepts. The model fits all the data points so well that noise is interpreted as patterns in the data as shown in the figure.



**Examples of Overfitting**

Let’s go with examples,

Let’s say we need to predict if a student will land a job interview based on his resume. Now assume we train a model from a dataset of 20,000 resumes and their outcomes.

Then we try a model out on the original dataset and it predicts outcomes with 98% Accuracy… Wow! It’s Amazing, but not in Reality.

But now comes the bad news. When we run a model out on the new dataset of resumes, we only get 50% of Accuracy.

Our model doesn’t get generalized well from our training data to see unseen data. This is known an **Overfitting**and it is a common problem in Data Science.

In fact, **Overfitting**occurs in the real world all the time. We need to handle it to generalize the model.

### Reasons for Overfitting are as follows:

1. High variance and low bias
2. The model is too complex
3. The size of the training data

**How to find Overfitting?**

The primary challenge in machine learning and in data science is that we can’t able to evaluate the model performance until we test it. So the first step to finding the Overfitting is to split the data into the Training and Testing set.

**If our model does much better on the training set than on the test set, then we’re likely overfitting.**

The performance can be measured using the percentage of accuracy observed in both data sets to conclude on the presence of **overfitting**. If the model performs better on the training set than on the test set, it means that the model is likely **overfitting**. For example, it would be a big Alert if our model saw 99% accuracy on the training set but only 50% accuracy on the test set.

**How to prevent Overfitting?**

1. Training with more data
2. Data Augmentation
3. Cross-Validation
4. Feature Selection
5. Regularization

**Regularization :**

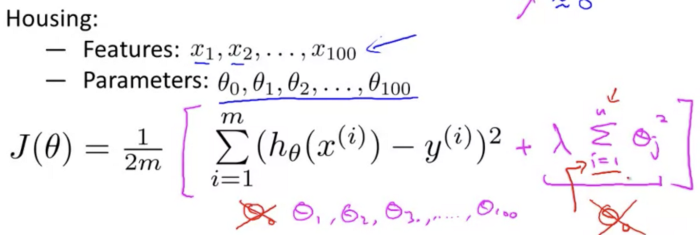
* keep all features but reduce the magnitude/value of parameters (theta j) to make the value smaller

\*Works well when we have a lot of features, each of which contributes a bit to predicting y.

**How does regularization work?**

Regularization (makes values smaller)

* make the “simpler” hypothesis
* less prone to overfitting



Make theta 3 and theta 4 close to 0

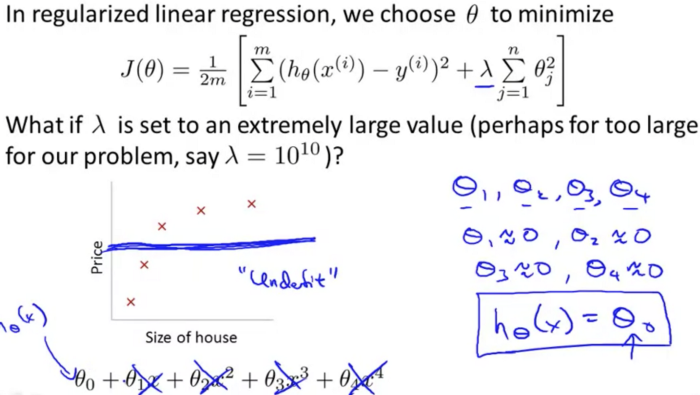
**Modify the cost function** by **adding an extra regularization term**in the end to **shrink every single parameter** (e.g. close to 0)

**lambda (regularization parameter)** controls the tradeoff between two goals:

former formula — 1st goal: fit the training data well

extra lambda (purple) — 2nd goal: keep the parameters small to avoid overfitting

**In regularized linear regression**



If all parameters (theta) are close to 0, the result will be close to 0. -> it will generate a **flat straight line** that **fails to fit the features wel**l → **underfit**

* To sum up, if**lambda** is chosen to be too **large,** it may **smooth out the function too much**and cause **underfitting**.

**Support Vector Machine**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



You can use them to detect cancerous cells based on millions of images or you can use them to predict future driving routes with a well-fitted regression model.

**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



## Types of SVM

**SVM can be of two types:**

* **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

## Hyperplane and Support Vectors in the SVM algorithm:

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

**Support Vectors:**

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

## How does SVM works?

**Linear SVM:**

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



**Non-Linear SVM:**

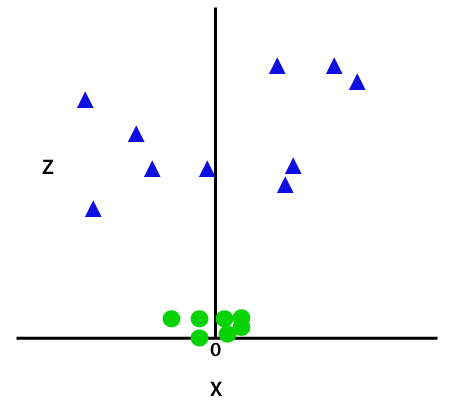
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

z=x2 +y2

By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:

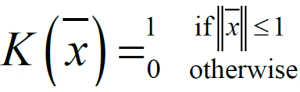


**SVM Kernel:**

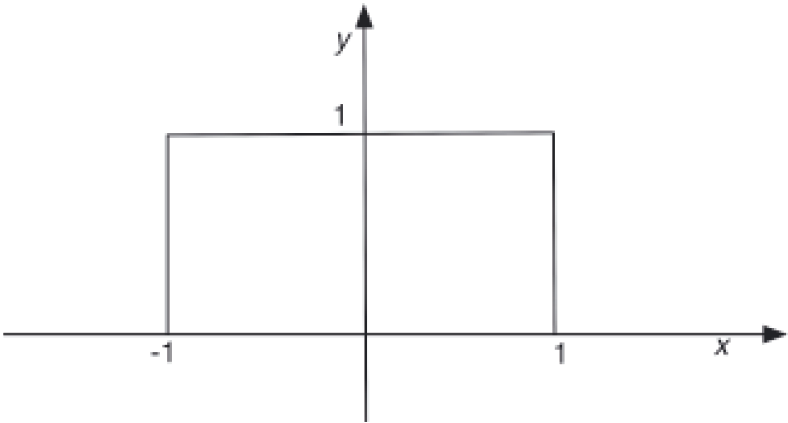
The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts non separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put the kernel, it does some extremely complex data transformations then finds out the process to separate the data based on the labels or outputs defined.

So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces.

**Standard Kernel Function Equation :**



This value of this function is 1 inside the closed ball of radius 1 centered at the origin, and 0 otherwise . As shown in the figure below:



* **Gaussian Kernel:**It is used to perform transformation when there is no prior knowledge about data.

Gaussian kernel equation

### Polynomial kernel

It is popular in image processing.  
Equation is:

[Polynomial kernel equation](https://data-flair.training/blogs/wp-content/uploads/sites/2/2017/08/polynomial-kernel.png)

*Polynomial kernel equation*

where d is the degree of the polynomial.

### 4.4. Laplace RBF(radial basis function) kernel

It is general-purpose kernel; used when there is no prior knowledge about the data.  
Equation is:

[Laplace RBF kernel equation](https://data-flair.training/blogs/wp-content/uploads/sites/2/2017/08/laplace-RBF-kernel.png)

*Laplace RBF kernel equation*

### 4.6. Sigmoid kernel

We can use it as the proxy for neural networks. Equation is

[ Sigmoid kernel equation](https://data-flair.training/blogs/wp-content/uploads/sites/2/2017/08/sigmoid-kernel.png)

*Sigmoid kernel equation*

### Hyperbolic tangent kernel

We can use it in neural networks.  
Equation is:

[Hyperbolic tangent kernel equation](https://data-flair.training/blogs/wp-content/uploads/sites/2/2017/08/hyperbolic-tangent-kernel.png)

*Hyperbolic tangent kernel equation*

, for some (not every) k>0 and c<0.

**Major Kernel Functions :-**

**Advantages of SVM:**

* Effective in high dimensional cases
* Its memory efficient as it uses a subset of training points in the decision function called support vectors
* Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

**KNN**

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



## Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



## How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
* **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

## How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.

## Advantages of KNN Algorithm:

* It is simple to implement.
* It is robust to the noisy training data
* It can be more effective if the training data is large.

## Disadvantages of KNN Algorithm:

* Always needs to determine the value of K which may be complex some time.
* The computation cost is high because of calculating the distance between the data points for all the training samples.

**Cross Validation**

# Cross-Validation in Machine Learning

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. **We can also say that it is a technique to check how a statistical model generalizes to an independent dataset**.

In [machine learning](https://www.javatpoint.com/machine-learning), there is always the need to test the stability of the model. It means based only on the training dataset; we can't fit our model on the training dataset. For this purpose, we reserve a particular sample of the dataset, which was not part of the training dataset. After that, we test our model on that sample before deployment, and this complete process comes under cross-validation. This is something different from the general train-test split.

Hence the basic steps of cross-validations are:

* Reserve a subset of the dataset as a validation set.
* Provide the training to the model using the training dataset.
* Now, evaluate model performance using the validation set. If the model performs well with the validation set, perform the further step, else check for the issues.

## Why do we need Cross-Validation?

Suppose you build a machine learning model to solve a problem, and you have trained the model on a given dataset. When you check the accuracy of the model on the training data, it is close to 95%. Does this mean that your model has trained very well, and it is the best model because of the high accuracy?

No, it’s not! Because your model is trained on the given data, it knows the data well, captured even the minute variations(noise), and has generalized very well over the given data. If you expose the model to completely new, unseen data, it might not predict with the same accuracy and it might fail to generalize over the new data. This problem is called over-fitting.

Sometimes the model doesn’t train well on the training set as it’s not able to find patterns. In this case, it wouldn’t perform well on the test set as well. This problem is called Under-fitting.

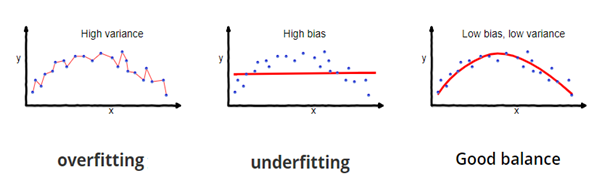


Image Source: fireblazeaischool.in

To overcome over-fitting problems, we use a technique called Cross-Validation.

**Cross-Validation** is a resampling technique with the fundamental idea of splitting the dataset into 2 parts- training data and test data. Train data is used to train the model and the unseen test data is used for prediction. If the model performs well over the test data and gives good accuracy, it means the model hasn’t overfitted the training data and can be used for prediction.

## Methods used for Cross-Validation

There are some common methods that are used for cross-validation. These methods are given below:

1. **Validation Set Approach**
2. **Leave one out cross-validation**
3. **K-fold cross-validation**
4. **Stratified(Bias-Variance) k-fold cross-validation**

### 1.Validation Set Approach

We divide our input dataset into a training set and test or validation set in the validation set approach. Both the subsets are given 50% of the dataset.

But it has one of the big disadvantages that we are just using a 50% dataset to train our model, so the model may miss out to capture important information of the dataset. It also tends to give the underfitted model.

### Steps Involved in the Validation Set Approach

1. A random splitting of the dataset into a certain ratio(generally 70-30 or 80-20 ratio is preferred)
2. Training of the model on the training data set
3. The resultant model is applied to the validation set
4. Model’s accuracy is calculated through prediction error by using model performance metrics

This article discusses the **step by step method of implementing the Validation set approach** as a cross-validation technique for both **classification** and **regression** machine learning models.

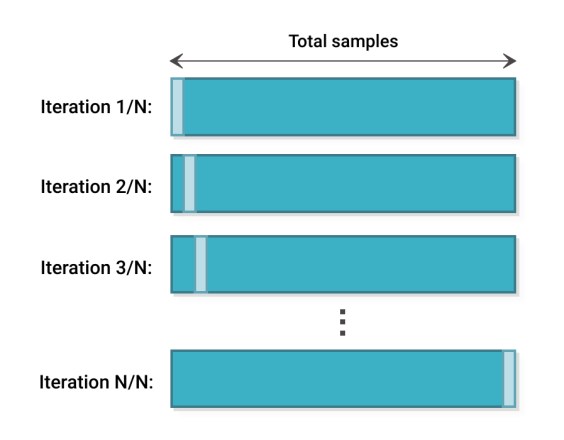
2. **Leave one out cross-validation**

## Leave-one-out cross-validation

**Leave-one-out сross-validation** (**LOOCV**) is an extreme case of**k-Fold CV**. Imagine if **k** is equal to **n** where n is the number of samples in the dataset. Such **k-Fold** case is equivalent to **Leave-one-out** technique.

The algorithm of LOOCV technique:

1. Choose one sample from the dataset which will be the test set
2. The remaining n – 1 samples will be the training set
3. Train the model on the training set. On each iteration, a new model must be trained
4. Validate on the test set
5. Save the result of the validation
6. Repeat steps 1 – 5 n times as for n samples we have n different training and test sets
7. To get the final score average the results that you got on step 5.



For LOOCV sklearn also has a built-in method. It can be found in the model\_selection library – sklearn.model\_selection.LeaveOneOut.

**import** numpy **as** np

**from** sklearn.model\_selection **import** LeaveOneOut

X = np.array([[1, 2], [3, 4]])

y = np.array([1, 2])

loo = LeaveOneOut()

**for** train\_index, test\_index **in** loo.split(X):

print("TRAIN:", train\_index, "TEST:", test\_index)

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

The greatest advantage of Leave-one-out cross-validation is that it doesn’t waste much data. We use only one sample from the whole dataset as a test set, whereas the rest is the training set. But when compared with k-Fold CV, LOOCV requires building n models instead of k models, when we know that n which stands for the number of samples in the dataset is much higher than k. It means LOOCV is more computationally expensive than k-Fold, it may take plenty of time to cross-validate the model using LOOCV.

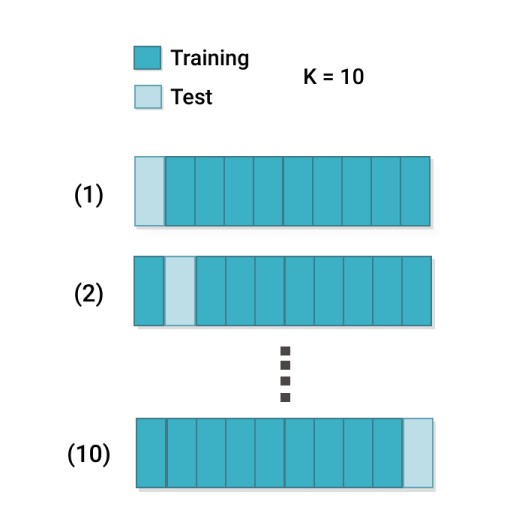
Thus, the Data Science community has a general rule based on empirical evidence and different researches, which suggests that 5- or 10-fold cross-validation should be preferred over LOOCV.

**3.K-fold cross-validation**

K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

The algorithm of the k-Fold technique:

1. Pick a number of folds – k. Usually, k is 5 or 10 but you can choose any number which is less than the dataset’s length.
2. Split the dataset into k equal (if possible) parts (they are called folds)
3. Choose k – 1 folds as the training set. The remaining fold will be the test set
4. Train the model on the training set. On each iteration of cross-validation, you must train a new model independently of the model trained on the previous iteration
5. Validate on the test set
6. Save the result of the validation
7. Repeat steps 3 – 6 k times. Each time use the remaining  fold as the test set. In the end, you should have validated the model on every fold that you have.
8. To get the final score average the results that you got on step 6.



To perform k-Fold cross-validation you can use sklearn.model\_selection.KFold.

**import** numpy **as** np

**from** sklearn.model\_selection **import** KFold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])

y = np.array([1, 2, 3, 4])

kf = KFold(n\_splits=2)

**for** train\_index, test\_index **in** kf.split(X):

print("TRAIN:", train\_index, "TEST:", test\_index)

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

In general, it is always better to use k-Fold technique instead of hold-out. In a head to head, comparison k-Fold gives a more stable and trustworthy result since training and testing is performed on several different parts of the dataset. We can make the overall score even more robust if we increase the number of folds to test the model on many different sub-datasets.

Still, k-Fold method has a disadvantage. Increasing k results in training more models and the training process might be really expensive and time-consuming.

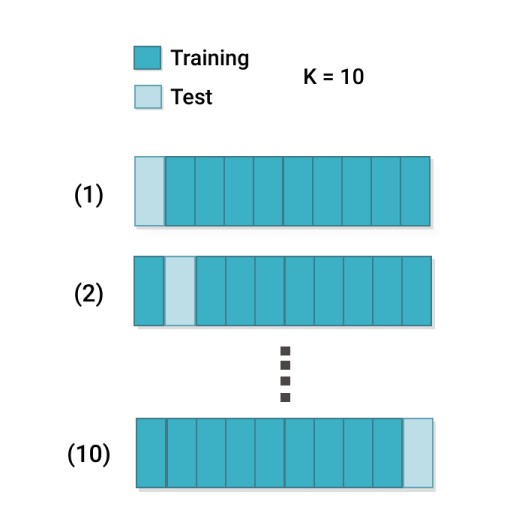
### Stratified k-fold cross-validation

This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.

It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

The algorithm of Stratified k-Fold technique:

1. Pick a number of folds – k
2. Split the dataset into k folds. Each fold must contain approximately the same percentage of samples of each target class as the complete set
3. Choose k – 1 folds which will be the training set. The remaining fold will be the test set
4. Train the model on the training set. On each iteration a new model must be trained
5. Validate on the test set
6. Save the result of the validation
7. Repeat steps 3 – 6 k times. Each time use the remaining  fold as the test set. In the end, you should have validated the model on every fold that you have.
8. To get the final score average the results that you got on step 6.



As you may have noticed, the algorithm for Stratified k-Fold technique is similar to the standard k-Folds. You don’t need to code something additionally as the method will do everything necessary for you.

Stratified k-Fold also has a built-in method in sklearn – sklearn.model\_selection.StratifiedKFold.

**import** numpy **as** np

**from** sklearn.model\_selection **import** StratifiedKFold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])

y = np.array([0, 0, 1, 1])

skf = StratifiedKFold(n\_splits=2)

**for** train\_index, test\_index **in** skf.split(X, y):

print("TRAIN:", train\_index, "TEST:", test\_index)

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

All mentioned above about k-Fold CV is true for Stratified k-Fold technique. When choosing between different CV methods, make sure you are using the proper one. For example, you might think that your model performs badly simply because you are using k-Fold CV to validate the model which was trained on the dataset with a [class imbalance](https://neptune.ai/blog/how-to-deal-with-imbalanced-classification-and-regression-data). To avoid that you should always do a proper exploratory data analysis on your data.

## 

**Bias-Variance tradeoff**

# ML Models

A machine learning model’s performance is evaluated based on how accurate is its prediction and how well it generalizes on another independent dataset it has not seen.

The errors in a machine learning model can be broken down into 2 parts:

* **Reducible Error**
* **Irreducible Error (noise)**

Irreducible errors are errors that cannot be reduced even if you use any other machine learning model.

Reducible errors, on the other hand, is further broken down into square of bias and variance. Due to this bias-variance, it causes the machine learning model to either overfit or underfit the given data

**Low-variance ML algorithms**: Linear Regression, Logistic Regression, Linear Discriminant Analysis.

**High-variance ML algorithms**: Decision Trees, k-NN, and Support Vector Machines.

**What is bias?**

* Bias is the difference between the average prediction of our model and the correct value which we are trying to predict.

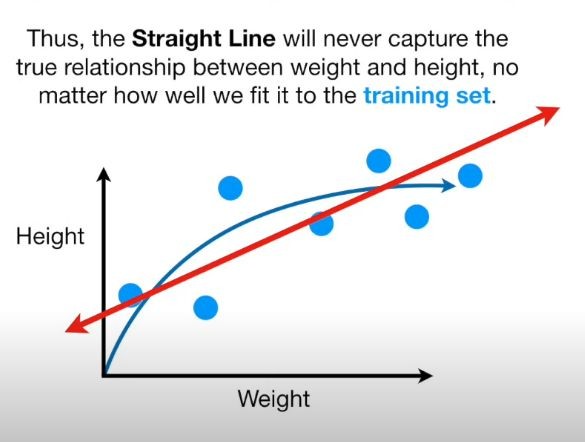
The Error between average model prediction and ground truth.

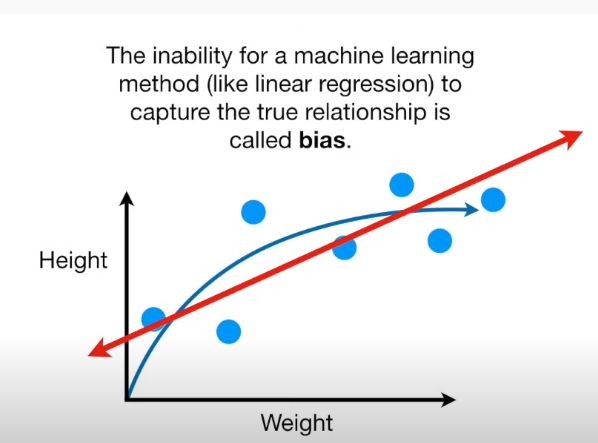
* Model with high bias pays very little attention to the training data and oversimplifies the model.
* It always leads to high error on training and test data.
* Bias is considered to be a systematic error that occurs in the machine learning model automatically due to incorrect assumptions in the ML process.
* It depends on the quality and size of training data being used to teach it.

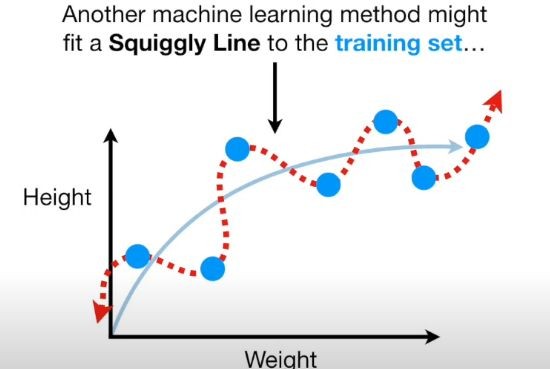
Using faulty, poor or incomplete data will always result in inaccurate predictions, further describing the saying of “garbage in, garbage out”.

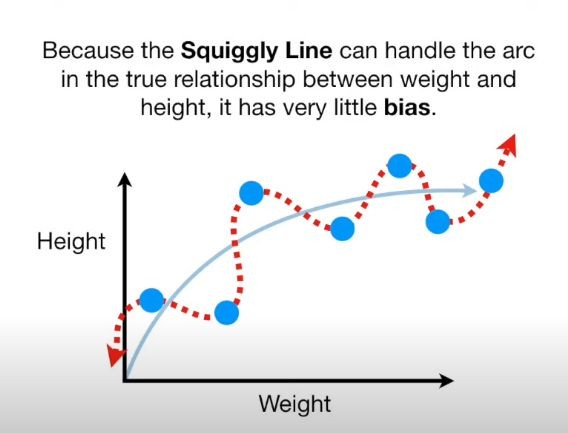
# What is Variance

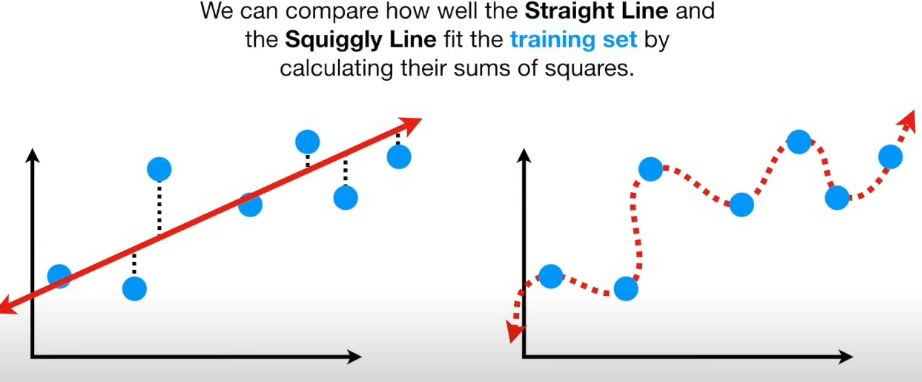
* Variance is the complete opposite of Bias
* Average Variability in the model prediction for the given dataset
* Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before. As a result, such models perform very well on training data but has high error rates on test data.
* variance as a model’s sensitivity to fluctuations in the data.
* The higher the variance of the model, the more complex the model will become and the more will it be able to learn complex functions.

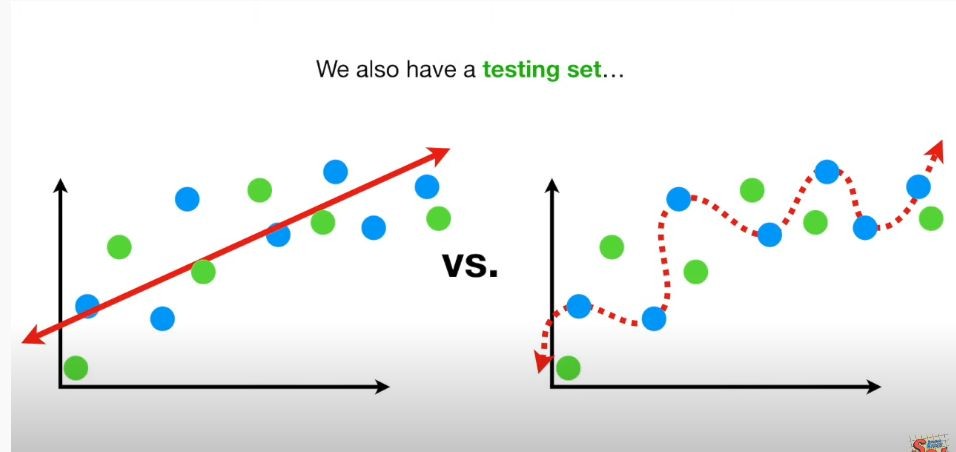


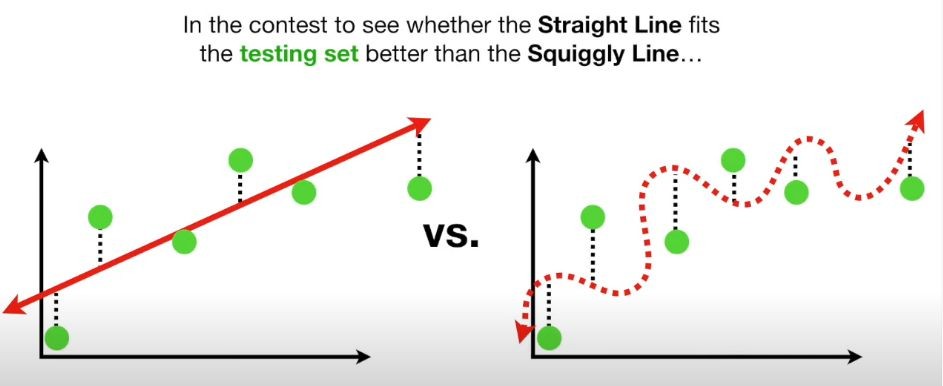


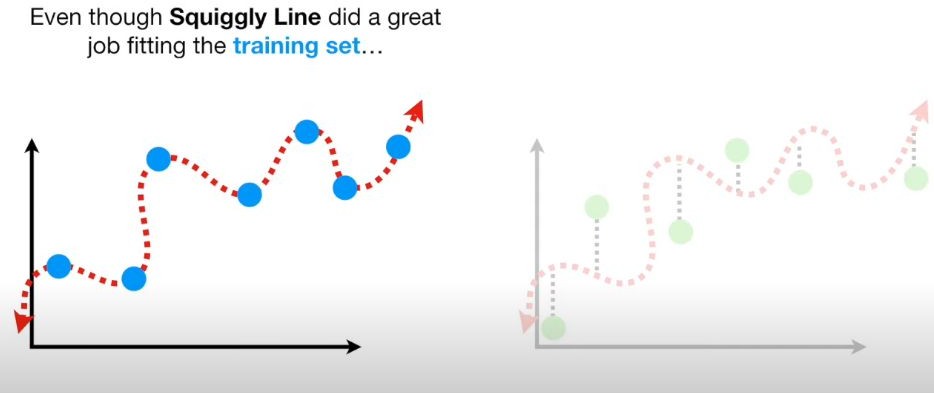


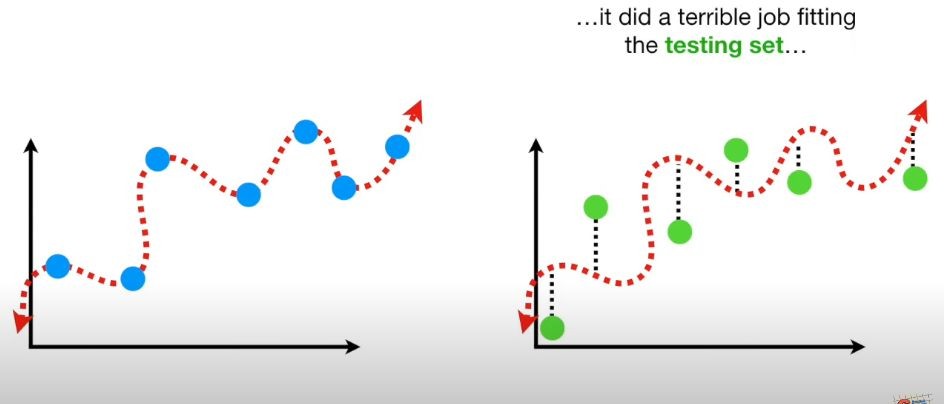


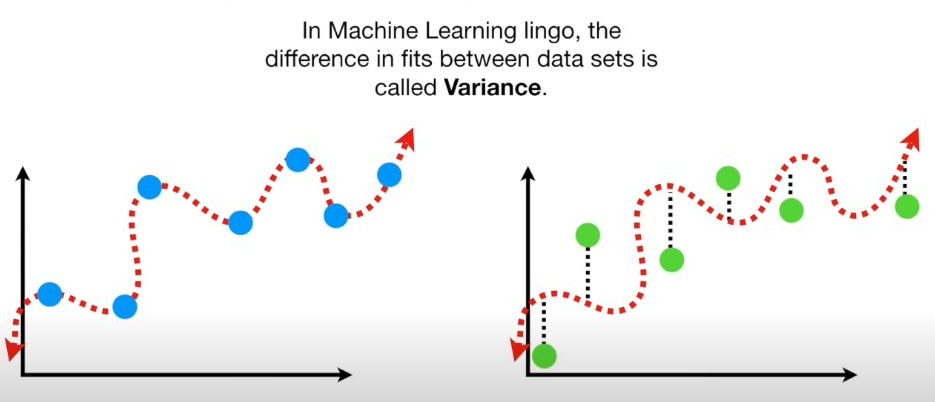


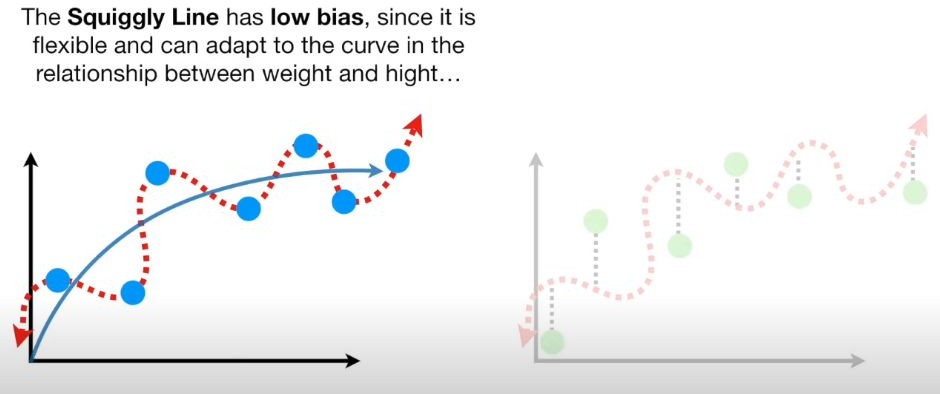


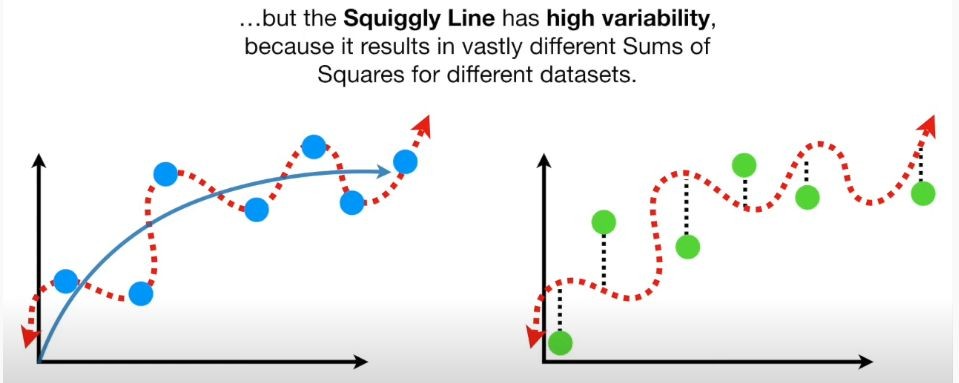


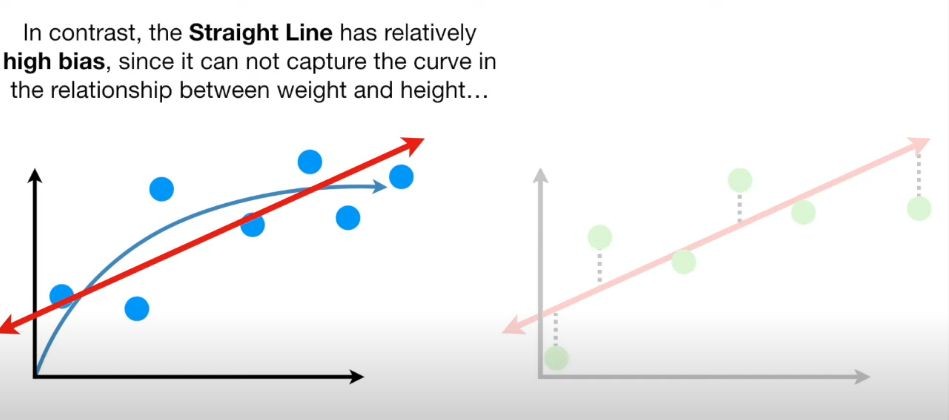


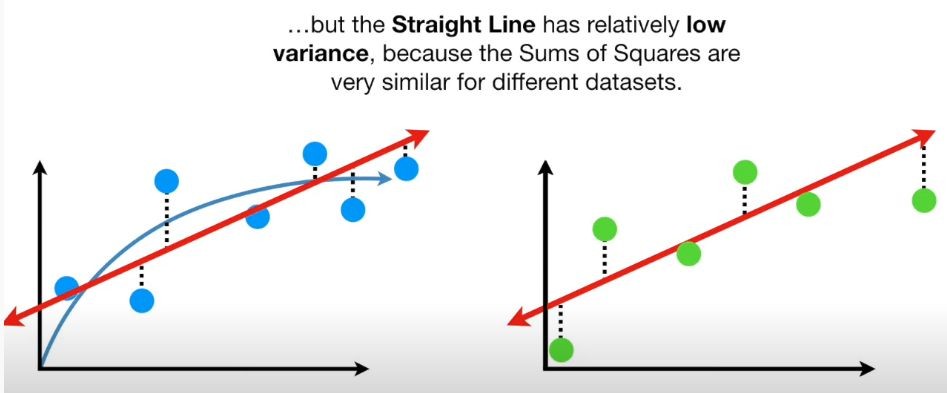


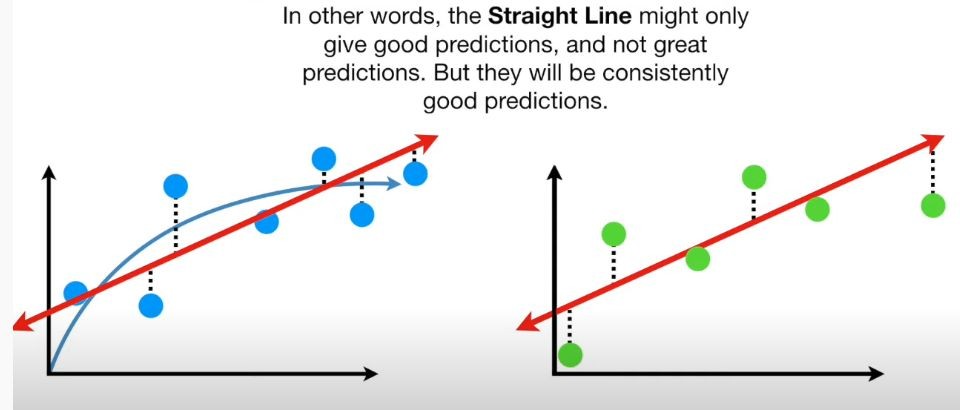


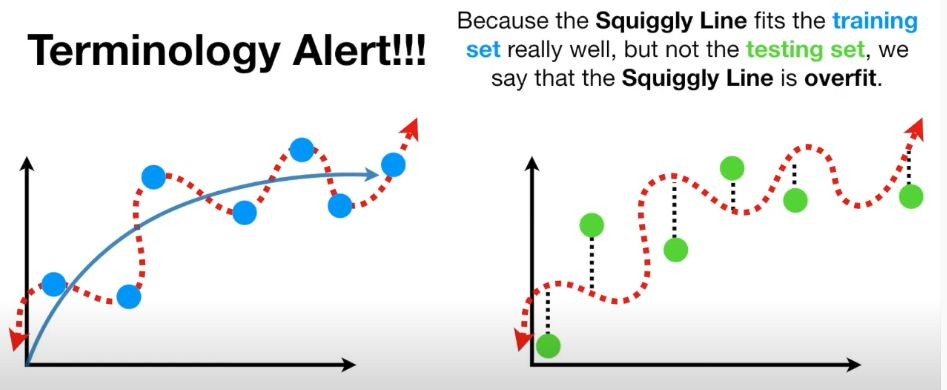


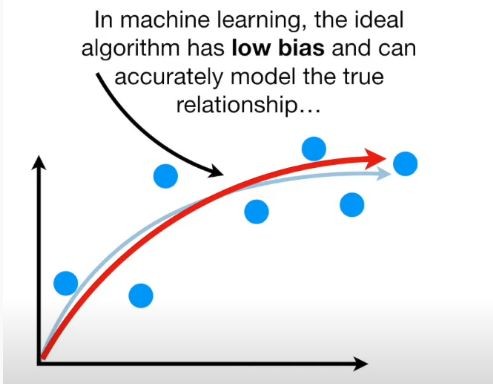


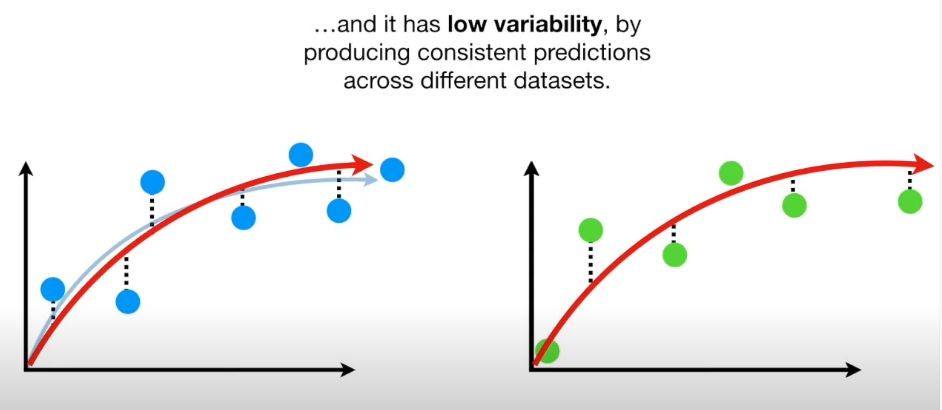


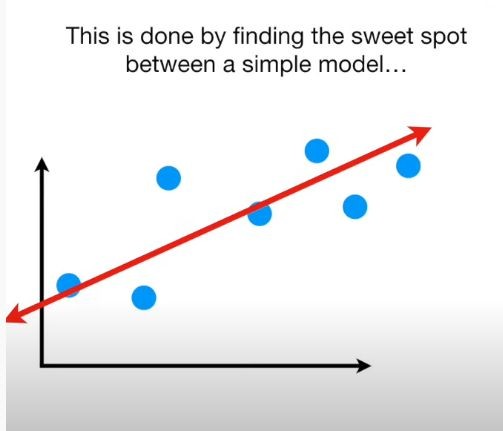


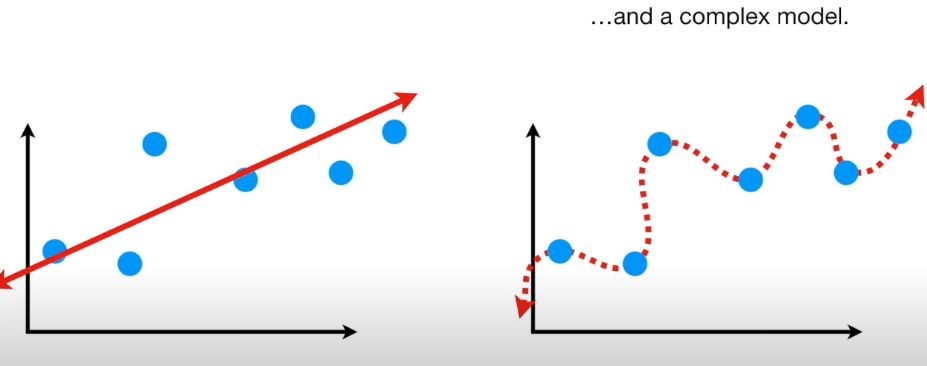












**How to Choose K**

In general, the more folds we use in k-fold cross-validation the lower the bias of the test MSE but the higher the variance. Conversely, the fewer folds we use the higher the bias but the lower the variance. This is a classic example of [the bias-variance tradeoff](https://www.statology.org/bias-variance-tradeoff/) in machine learning.

In practice, we typically choose to use between 5 and 10 folds. As noted in [*An Introduction to Statistical Learning*,](https://www.ime.unicamp.br/~dias/Intoduction%20to%20Statistical%20Learning.pdf) this number of folds has been shown to offer an optimal balance between bias and variance and thus provide reliable estimates of test MSE:

**3-methods to find sweet spot**

Regularization

Bagging

Boosting

**Learning algorithm comprises of 3 parts:**

1. Bias error
2. Variance error
3. The noise

While the noise is the irreducible error that we cannot eliminate, the other two i.e. Bias and Variance are reducible errors that we can attempt to minimize as much as possible.

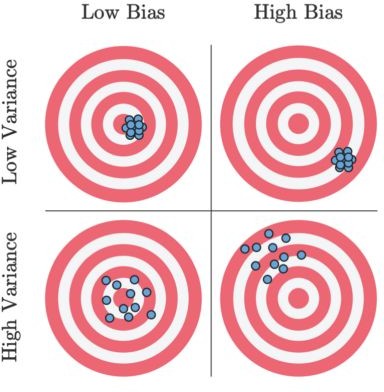
**Effect of High Bias**

* Overly-Simplified Model
* Under-Fitting
* High error on both test and train data

**High Variance**

* Overly-complex model
* Overly-Fitting
* Low error on train data
* High error on test data
* Starts modeling the noise in the input

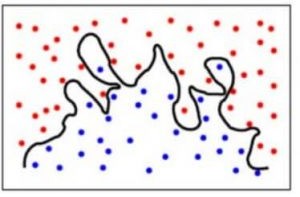
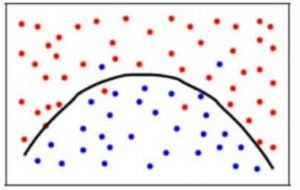
**Bias Variance Trade-Off**



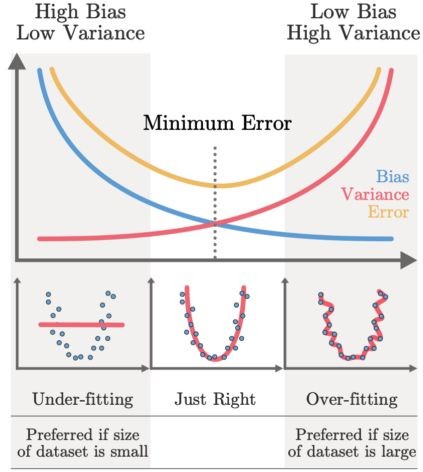
**Bias Variance Trade-Off**

* + Increasing bias reduces variance and vice-versa
  + Error = Bias² + Variance + irreducible error
  + The best model is where the error is reduced.
  + Compromise between bias and variance.

# Overfitting model



**Reasonable model**

****

**Bootstrap**

## What is Bootstrap Sampling?

Here’s a formal definition of Bootstrap Sampling:

In statistics, Bootstrap Sampling is a method that involves drawing of sample data repeatedly with replacement from a data source to estimate a population parameter.

Wait – that’s too complex. Let’s break it down and understand the key terms:

* **Sampling:** With respect to statistics, sampling is the process of selecting a subset of items from a vast collection of items (population) to estimate a certain characteristic of the entire population
* **Sampling with replacement:** It means a data point in a drawn sample can reappear in future drawn samples as well
* **Parameter estimation:** It is a method of estimating parameters for the population using samples. A parameter is a measurable characteristic associated with a population. For example, the average height of residents in a city, the count of red blood cells, etc.

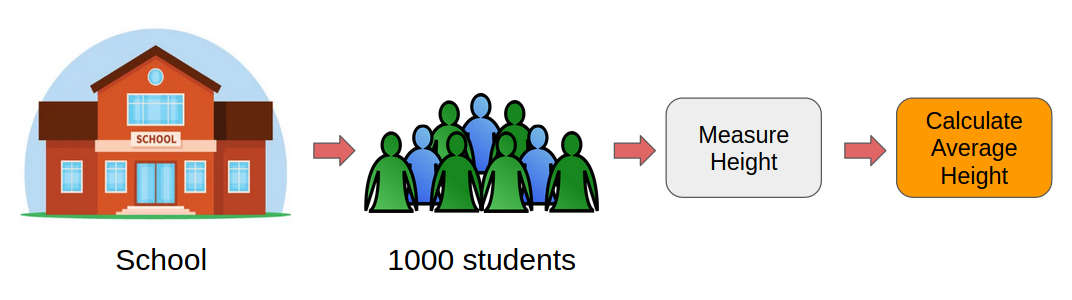
With that knowledge, go ahead and re-read the above definition again. It’ll make much more sense now!

## Why Do We Need Bootstrap Sampling?

This is a fundamental question I’ve seen machine learning enthusiasts grapple with. What is the point of Bootstrap Sampling? Where can you use it? Let me take an example to explain this.

Let’s say we want to find the mean height of all the students in a school (which has a total population of 1,000). So, how can we perform this task?

One approach is to measure the height of all the students and then compute the mean height. I’ve illustrated this process below:

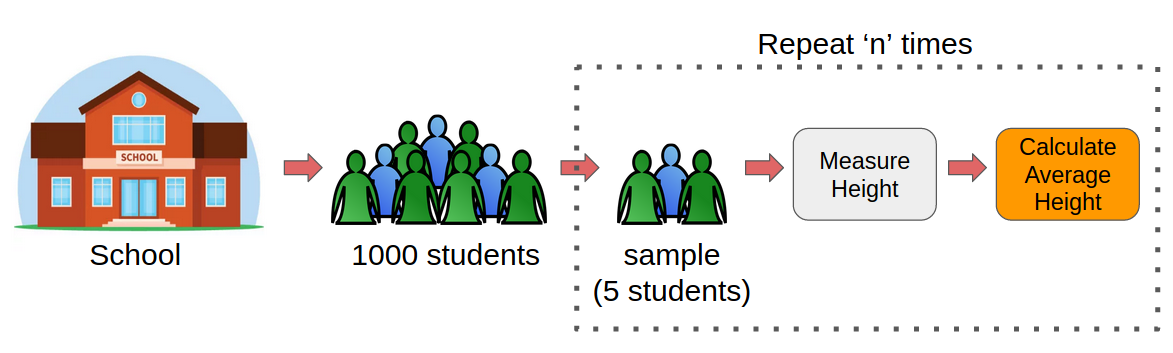
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/img_1-1.png)

However, this would be a tedious task. Just think about it, we would have to individually measure the heights of 1,000 students and then compute the mean height. It will take days! We need a smarter approach here.

This is where Bootstrap Sampling comes into play.

Instead of measuring the heights of all the students, we can draw a random sample of 5 students and measure their heights. We would repeat this process 20 times and then average the collected height data of 100 students (5 x 20). This average height would be an estimate of the mean height of all the students of the school.

Pretty straightforward, right? This is the basic idea of Bootstrap Sampling.

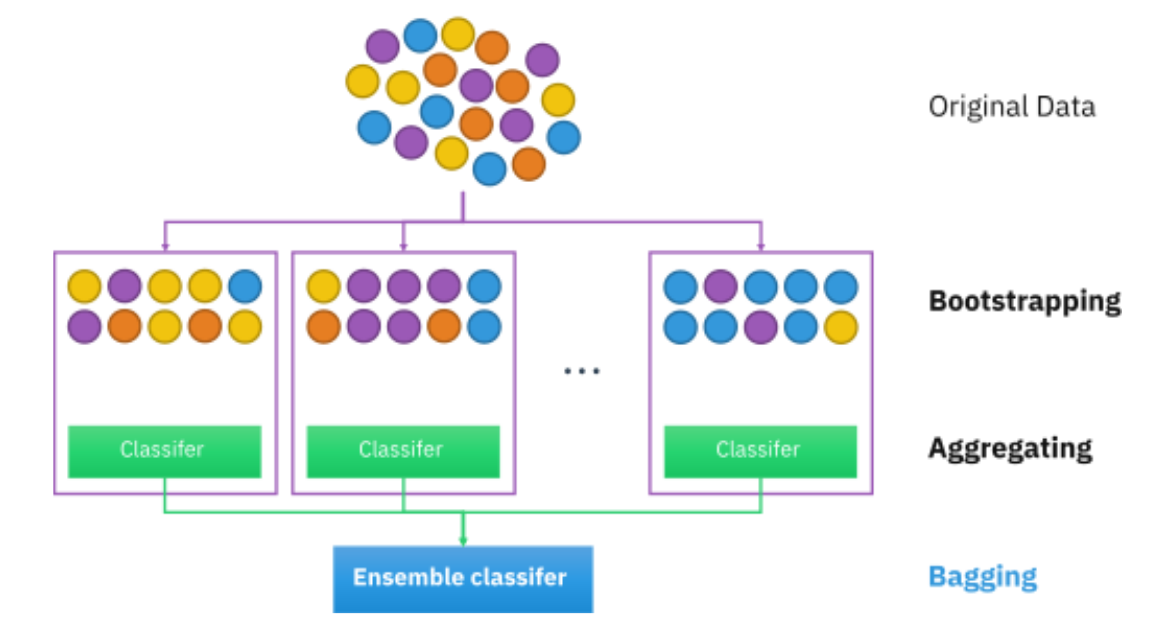
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/img_2-1.png)

**Hence, when we have to estimate a parameter of a large population, we can take the help of Bootstrap Sampling.**

## Bootstrap Sampling in Machine Learning

Bootstrap sampling is used in a machine learning ensemble algorithm called bootstrap aggregating (also called bagging). It helps in avoiding [overfitting](https://www.analyticsvidhya.com/blog/2020/02/underfitting-overfitting-best-fitting-machine-learning/) and improves the stability of [machine learning algorithms](https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/).

In bagging, a certain number of equally sized subsets of a dataset are extracted with replacement. Then, a machine learning algorithm is applied to each of these subsets and the outputs are ensembled as I have illustrated below:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/Bagging.png)

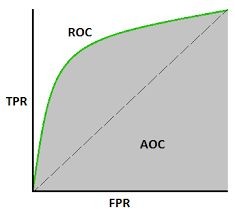
**ROC-AUC**

# What is Statistics?

If we have to define statistics, we will simply say that statistics is a discipline of applied mathematics that deals with gathering, describing, analyzing, and inferring conclusions from numerical data.

Differential and integral calculus, linear algebra, and probability theory are all used substantially in statistics' mathematical theories. Statisticians are particularly interested in learning how to make trustworthy inferences about big groups and general phenomena from the observable features of small samples that reflect just a tiny share of the big group or a small number of instances of a general occurrence.

ROC curve can be used to select a threshold for a classifier, which maximizes the true positives and in turn minimizes the false positives.

ROC Curves help determine the exact trade-off between the true positive rate and false-positive rate for a model using different measures of probability thresholds.

# ROC-AUC curve

# ROC-AUC

* Classification performance metrics such as Log-Loss, Average Accuracy, AUC, metric used for evaluating the performance in classification models.
* capability of a model in distinguishing the classes.
  + The judging criteria being - Higher the AUC, better the model.
* graphical way the connection and trade-off between sensitivity and specificity
* ROC curve, also known as Receiver Operating Characteristics Curve,
* depicts the rate of true positives with respect to the rate of false positives.
* it is a comparison of two operating characteristics, the True Positive Rate and the False Positive Rate

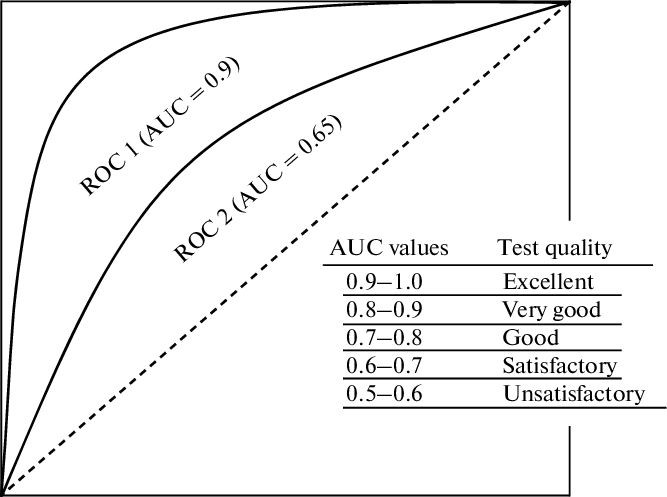
**True Positive**: Actual Positivity and Positivity Predicted True **Negative**: Actual Negative and Negatively Predicted

**Type I Error (False Positive)**: Although the situation is actually negative, it is anticipated to be positive.

**False Negative** :True Positive but anticipated as Negative (Type II Error): Actual Positive but anticipated as Negative.

# AUC

* Area Under Curve or AUC is one of the most widely used metrics for model evaluation
* It is generally used for binary classification problems.
* AUC measures the entire two-dimensional area present underneath the entire ROC curve. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than that of a randomly chosen negative example.
* The Area Under the Curve provides the ability for a classifier to distinguish between classes and is used as a summary of the ROC curve.
* The higher the AUC, it is assumed that the better the performance of the model at distinguishing between the positive and negative classes.

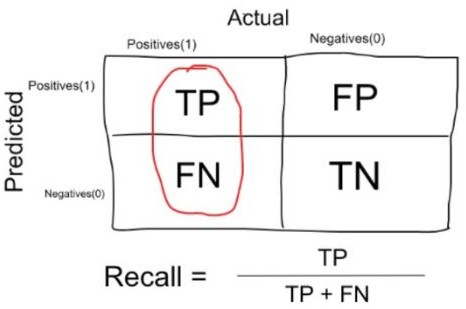


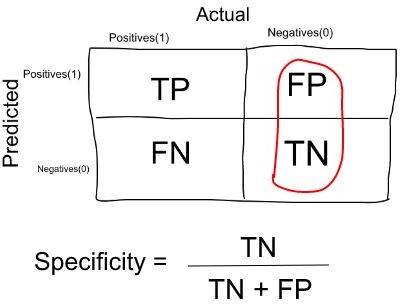
interpretation of AUC-ROC curve

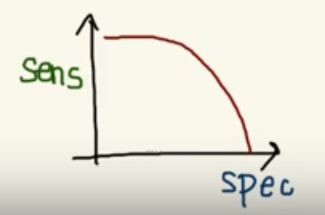
An excellent model poses an AUC near to the 1 which tells that it has a good measure of separability. A poor model will have an AUC near 0 which describes that it has the worst measure of separability. In fact, it means it is reciprocating the result and predicting 0s as 1s and 1s as 0s. When an AUC is 0.5, it means the model has no class separation capacity present whatsoever.

Higher the AUC, better the model.

# Confusion Matrix

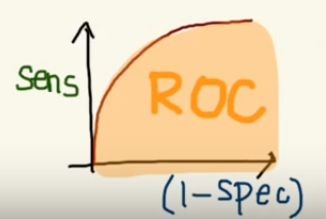




**Tradeoff**

As **Sensitivity ** **Specificity **

As **Specificity Sensitivity**



AUC-ROC Curve in Machine Learning

In Machine Learning, only developing an ML model is not sufficient as we also need to see whether it is performing well or not. It means that after building an ML model, we need to evaluate and validate how good or bad it is, and for such cases, we use different Evaluation Metrics. *AUC-ROC curve is such an evaluation metric that is used to visualize the performance of a classification model*. It is one of the popular and important metrics for evaluating the performance of the classification model. In this topic, we are going to discuss more details about the AUC-ROC curve.

## What is AUC-ROC Curve?

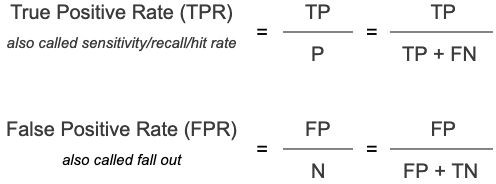
AUC-ROC curve is a performance measurement metric of a classification model at different **threshold** values. Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve.

### ROC Curve

**ROC or Receiver Operating Characteristic curve represents a probability graph to show the performance of a classification model at different threshold levels**. The curve is plotted between two parameters, which are:

* **True Positive Rate or TPR**
* **False Positive Rate or FPR**

In the curve, TPR is plotted on Y-axis, whereas FPR is on the X-axis.



**TPR**

TPR or True Positive rate is a synonym for Recall, which can be calculated as:



FPR or False Positive Rate can be calculated as:



Here, **TP**: True Positive

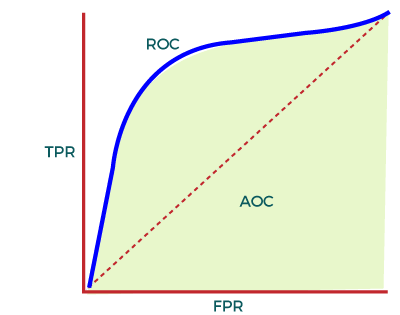
**FP**: False Positive

**TN**: True Negative

**FN**: False Negative

Now, to efficiently calculate the values at any threshold level, we need a method, which is AUC.

### AUC: Area Under the ROC curve



In the ROC curve, AUC computes the performance of the binary classifier across different thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1, which means an excellent model will have AUC near 1, and hence it will show a good measure of Separability.

### When to Use AUC-ROC

**AUC is preferred due to the following cases:**

* AUC is used to measure how well the predictions are ranked instead of giving their absolute values. Hence, we can say AUC is **Scale-Invariant.**
* It measures the quality of predictions of the model without considering the selected classification threshold. It means AUC is **classification-threshold-invariant.**

### When not to use AUC-ROC

* AUC is not preferable when we need to calibrate probability output.
* Further, AUC is not a useful metric when there are wide disparities in the cost of false negatives vs false positives, and it is difficult to minimize one type of classification error.

## How AUC-ROC curve can be used for the Multi-class Model?

Although the AUC-ROC curve is only used for binary classification problems, we can also use it for multiclass classification problems. For multi-class classification problems, we can plot N number of AUC curves for N number of classes with the One vs ALL method.

For example, if we have three different classes, X, Y, and Z, then we can plot a curve for X against Y & Z, a second plot for Y against X & Z, and the third plot for Z against Y and X.

## Applications of AUC-ROC Curve

Although the AUC-ROC curve is used to evaluate a classification model, it is widely used for various applications. Some of the important applications of AUC-ROC are given below:

**1.Classification of 3D model**

The curve is used to classify a 3D model and separate it from the normal models. With the specified threshold level, the curve classifies the non-3D and separates out the 3D models.

**2. Healthcare**

The curve has various applications in the healthcare sector. It can be used to detect cancer disease in patients. It does this by using false positive and false negative rates, and accuracy depends on the threshold value used for the curve.

**3.Binary Classification**

AUC-ROC curve is mainly used for binary classification problems to evaluate their performance.

## Confusion Matrix

Have you been in a situation where you expected your machine learning model to perform really well but it sputtered out a poor accuracy? You’ve done all the hard work – so where did the classification model go wrong? How can you correct this?

There are plenty of ways to gauge the performance of your classification model but none have stood the test of time like the confusion matrix. It helps us evaluate how our model performed, where it went wrong and offers us guidance to correct our path.

In this article, we will explore what is confusion matrix in machine learning and how a Confusion matrix gives a holistic view of the performance of your model. And unlike its name, you will realize that a Confusion matrix python is a pretty simple yet powerful concept. So let’s unravel the mystery around the confusion matrix!

**What is a Confusion Matrix?**

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:



Let’s decipher the matrix:

The target variable has two values: **Positive**or **Negative**

The **columns**represent the **actual values** of the target variable

The **rows**represent the **predicted values**of the target variable

* But wait – what’s TP, FP, FN and TN here? That’s the crucial part of a confusion matrix. Let’s understand each term below.

**True Positive (TP)**

The predicted value matches the actual value

The actual value was positive and the model predicted a positive value

**True Negative (TN)**

The predicted value matches the actual value

The actual value was negative and the model predicted a negative value

**False Positive (FP) – Type 1 error**

The predicted value was falsely predicted

The actual value was negative but the model predicted a positive value

Also known as the **Type 1 error**

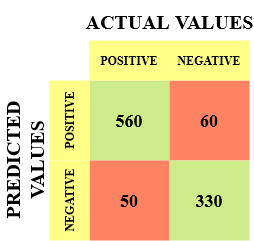
**False Negative (FN) – Type 2 error**

The predicted value was falsely predicted

The actual value was positive but the model predicted a negative value

Also known as the **Type 2 error**

Let me give you an example to better understand this. Suppose we had a classification dataset with 1000 data points. We fit a classifier on it and get the below confusion matrix:



The different values of the Confusion matrix would be as follows:

* True Positive (TP) = 560; meaning 560 positive class data points were correctly classified by the model
* True Negative (TN) = 330; meaning 330 negative class data points were correctly classified by the model
* False Positive (FP) = 60; meaning 60 negative class data points were incorrectly classified as belonging to the positive class by the model
* False Negative (FN) = 50; meaning 50 positive class data points were incorrectly classified as belonging to the negative class by the model

This turned out to be a pretty decent classifier for our dataset considering the relatively larger number of true positive and true negative values.

## Why Do We Need a Confusion Matrix?

Before we answer this question, let’s think about a hypothetical classification problem.

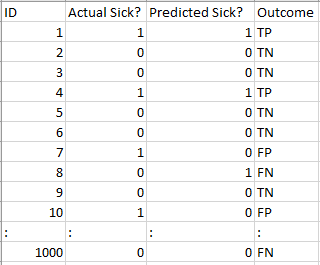
Let’s say you want to predict how many people are infected with a contagious virus in times before they show the symptoms, and isolate them from the healthy population (ringing any bells, yet? ). The two values for our target variable would be: Sick and Not Sick.

Now, you must be wondering – why do we need a confusion matrix when we have our all-weather friend – Accuracy? Well, let’s see where accuracy falters.

Our dataset is an example of an [**imbalanced dataset**](https://www.analyticsvidhya.com/blog/2017/03/imbalanced-data-classification/?utm_source=blog&utm_medium=confusion-matrix-machine-learning). There are 947 data points for the negative class and 3 data points for the positive class. This is how we’ll calculate the accuracy:

Equation_Accuracy

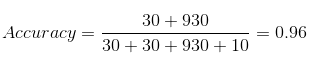
Let’s see how our model performed:



The total outcome values are:

TP = 30, TN = 930, FP = 30, FN = 10

So, the accuracy for our model turns out to be:



96%! Not bad!

But it is giving the wrong idea about the result. Think about it.

Our model is saying “I can predict sick people 96% of the time”. However, it is doing the opposite. It is predicting the people who will not get sick with 96% accuracy while the sick are spreading the virus!

Do you think this is a correct metric for our model given the seriousness of the issue? Shouldn’t we be measuring how many positive cases we can predict correctly to arrest the spread of the contagious virus? Or maybe, out of the correctly predicted cases, how many are positive cases to check the reliability of our model?

This is where we come across the dual concept of Precision and Recall.

## Precision vs. Recall

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

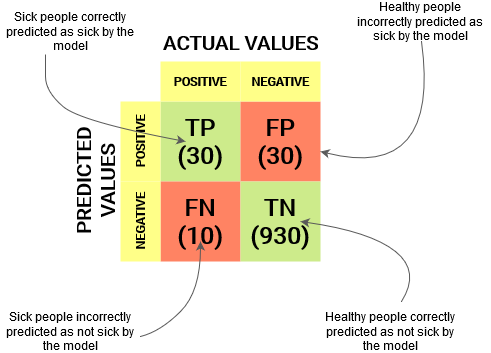
Here’s how to calculate Precision:

https://cdn.analyticsvidhya.com/wp-content/uploads/2020/04/Confusion-matrix_Precision.png

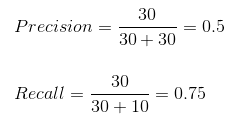
This would determine whether our model is reliable or not.

Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

**And here’s how we can calculate Recall:**



We can easily calculate Precision and Recall for our model by plugging in the values into the above questions:



50% percent of the correctly predicted cases turned out to be positive cases. Whereas 75% of the positives were successfully predicted by our model. Awesome!

Precision is a useful metric in cases where False Positive is a higher concern than False Negatives.

Precision is important in music or video recommendation systems, e-commerce websites, etc. Wrong results could lead to customer churn and be harmful to the business.

Recall is a useful metric in cases where False Negative trumps False Positive.

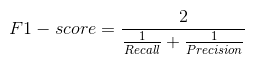
Recall is important in medical cases where it doesn’t matter whether we raise a false alarm but the actual positive cases should not go undetected!

In our example, Recall would be a better metric because we don’t want to accidentally discharge an infected person and let them mix with the healthy population thereby spreading the contagious virus. Now you can understand why accuracy was a bad metric for our model.

But there will be cases where there is no clear distinction between whether Precision is more important or Recall. What should we do in those cases? We combine them!

## F1-Score

In practice, when we try to increase the precision of our model, the recall goes down, and vice-versa. The F1-score captures both the trends in a single value:



**F1-score is a harmonic mean of Precision and Recall**, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.

But there is a catch here. The interpretability of the F1-score is poor. This means that we don’t know what our classifier is maximizing – precision or recall? So, we use it in combination with other evaluation metrics which gives us a complete picture of the result.

* True Positive

True Negative

* False Positive – Type 1 Error

False Negative – Type 2 Error