1. In the sense of machine learning, what is a model? What is the best way to train a model?

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

Once you have trained the model, you can use it to reason over data that it hasn't seen before, and make predictions about those data. For example, let's say you want to build an application that can recognize a user's emotions based on their facial expressions. You can train a model by providing it with images of faces that are each tagged with a certain emotion, and then you can use that model in an application that can recognize any user's emotion

The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with training data to learn from. The term ML model refers to the model artifact that is created by the training process.

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

You can use the ML model to get predictions on new data for which you do not know the target. For example, let's say that you want to train an ML model to predict if an email is spam or not spam. You would provide Amazon ML with training data that contains emails for which you know the target (that is, a label that tells whether an email is spam or not spam). Amazon ML would train an ML model by using this data, resulting in a model that attempts to predict whether new email will be spam or not spam.

1. In the sense of machine learning, explain the "No Free Lunch" theorem.

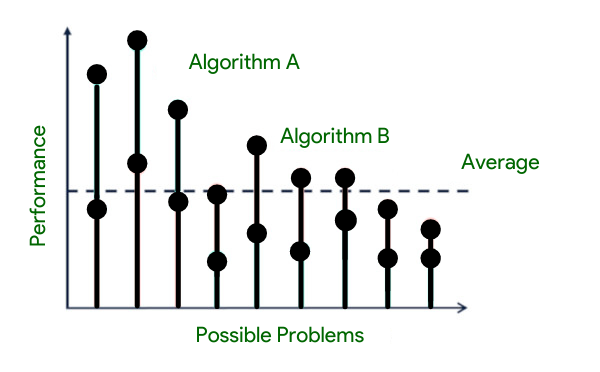
The No Free Lunch Theorem is often used in optimization and machine learning, with little comprehension of what it means or implies.

The theory asserts that when the performance of all optimization methods is averaged across all conceivable problems, they all perform equally well. It indicates that no one optimum optimization algorithm exists. Because of the strong link between optimization, search, and machine learning, there is no one optimum machine learning method for predictive modelling tasks like classification and regression.

They all agree on one point: there is no “best” algorithm for specific kinds of algorithms, since they all perform similarly on average. Mathematically, the computing cost of finding a solution is the same for any solution technique when averaged across all problems in the class. As a result, no solution provides a shortcut.

*There are****two No Free Lunch (NFL) theorems in general:****one for machine learning and one for search and optimization. These two theorems are connected and are frequently combined into a single general postulate (the folklore theorem).*

Although many other scholars have contributed to the collective writings on the No Free Lunch theorems, David Wolpert is the most well-known name connected with these studies.  
Surprisingly, the concept that may have inspired the NFL theorem was first offered by a 1700s philosopher. Yes, you read that correctly! A philosopher, not a mathematician or a statistician..



*Figure 1. Understanding NFL.*

David Hume, a Scottish philosopher, presented the issue of induction in the mid-1700s. This is a philosophical question about whether inductive reasoning leads to true knowledge.

*Inductive reasoning is a type of thinking in which we make inferences about the world based on previous observations.*

According to the “No Free Lunch” theory, there is no one model that works best for every situation. Because the assumptions of a great model for one issue may not hold true for another, it is typical in machine learning to attempt many models to discover the one that performs best for a specific problem. This is especially true in supervised learning, where validation or cross-validation is frequently used to compare the prediction accuracy of many models of various complexity in order to select the optimal model. A good model may also be trained using several methods — for example, linear regression can be learned using normal equations or gradient descent.

**According to the “No Free Lunch” theorem,** all optimization methods perform equally well when averaged over all optimization tasks without re-sampling. This fundamental theoretical notion has had the greatest impact on optimization, search, and supervised learning. The first theorem, No Free Lunch, was rapidly formulated, resulting in a series of research works, which defined a whole field of study with meaningful outcomes across different disciplines of science where the effective exploration of a search region is a vital and crucial activity.

In general, its usefulness is as important as the algorithm. An effective solution is created by matching the utility with the algorithm. If no good conditions for the objective function are known, and one is just working with a black box, no guarantee can be made that this or that method outperforms a (pseudo)random search.

A framework is being created to investigate the relationship between successful optimization algorithms and the issues they solve. A series of “no free lunch” (NFL) theorems are provided, establishing that any improved performance over one class of tasks is compensated by improved performance over another. These theorems provide a geometric explanation of what it means for an algorithm to be well matched to an optimization issue.

*The NFL theorems are also applied to information-theoretic elements of optimization and benchmark measurements of performance.*

There is no such thing as a free lunch, since adding alternatives to a project incurs both direct and opportunity expenses. As a result, incorporating actual alternatives may increase the original development cost. Direct costs are the expenses of additional development effort required to include certain flexibilities into the project’s architecture. Opportunity costs are the expenses of not being able to do anything else (for example, add a feature) as a result of the time and effort spent on generating that flexibility.

**Conclusion:**  
Machine learning models adhere to the Garbage in, Garbage out (GIGO) principle (i.e. Predictions rely on the data quality on which our model is trained). And a lot of study went into these theorems, and others may claim that this theorem does not apply in all instances. It is preferable that we concentrate on the aspects that will help us better comprehend the data and construct the best performing models.

1. Describe the K-fold cross-validation mechanism in detail.

Generalization is a frequent term that is used in conversations about Machine Learning. It refers to how able the model can adapt to new, unseen data and how it works effectively using various inputs. It is understandable to say that if new unseen data is inputted into a model, if this unseen data has similar characteristics to the training data, it will perform well.

Generalizing things is easy to us humans, however it can be challenging to Machine Learning models. This is where Cross-Validation comes into the picture.

**Cross-Validation**

You may see cross-validation also being referred to as rotation estimation and/or out-of-sample testing. The overall aim of Cross-Validation is to use it as a tool to evaluate machine learning models, by training a number of models on different subsets of the input data.

Cross-validation can be used to detect overfitting in a model which infers that the model is not effectively generalizing patterns and similarities in the new inputted data.

**A typical Cross-Validation workflow**

In order to perform cross-validation, the following steps are typically taken:

1. Split the dataset into training data and test data
2. The parameters will undergo a Cross-Validation test to see which are the best parameters to select.
3. These parameters will then be implemented into the model for retraining
4. Final evaluation will occur and this will depend if the cycle has to go again, depending on the accuracy and the level of generalization that the model performs.

There are different types of Cross-Validation techniques

* Hold-out
* K-folds
* Leave-one-out
* Leave-p-out

However, we will be particularly focusing on K-folds.

**K-fold Cross-Validation**

K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data. K refers to the number of groups the data sample is split into. For example, if you see that the k-value is 5, we can call this a 5-fold cross-validation. Each fold is used as a testing set at one point in the process.

**K-fold Cross-Validation Process:**

1. Choose your k-value
2. Split the dataset into the number of k folds.
3. Start off with using your k-1 fold as the test dataset and the remaining folds as the training dataset
4. Train the model on the training dataset and validate it on the test dataset
5. Save the validation score
6. Repeat steps 3 – 5, but changing the value of your k test dataset. So we chose k-1 as our test dataset for the first round, we then move onto k-2 as the test dataset for the next round.
7. By the end of it you would have validated the model on every fold that you have.
8. Average the results that were produced in step 5 to summarize the skill of the model.

You can easily implement this using sklearn.model\_selection.KFold

4. Describe the bootstrap sampling method. What is the aim of it?

5. What is the significance of calculating the Kappa value for a classification model? Demonstrate how to measure the Kappa value of a classification model using a sample collection of results.

1. Describe the model ensemble method. In machine learning, what part does it play?

Ensemble models are a machine learning approach to combine multiple other models in the prediction process. These models are referred to as base estimators. Ensemble models offer a solution to overcome the technical challenges of building a single estimator.

Anytime we’re trying to make an important decision, we try to collect as much information as possible and reach out to experts for advice. The more information we can gather, the more we (and those around us) trust the decision-making process.

Machine learning predictions follow a similar behavior. Models process given inputs and produce an outcome. The outcome is a prediction based on what pattern the models see during the training process.

One model is not enough in many cases, and this article sheds light on this point. When and why do we need multiple models? How do we train those models? What kind of diversity should those models provide? So, let’s jump right in.

The technical challenges of building a single estimator include:

**High variance**: The model is very sensitive to the provided inputs for the learned features.

**Low accuracy**: One model (or one algorithm) to fit the entire training data might not provide you with the nuance your project requires.

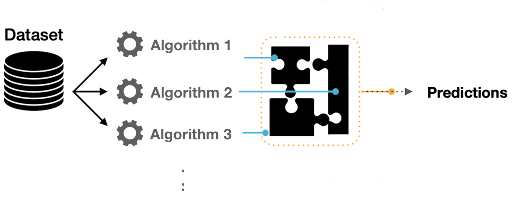
**Features noise and bias**: The model relies heavily on too few features while making a prediction.

A single algorithm may not make the perfect prediction for a given data set. Machine learning algorithms have their limitations and producing a model with high accuracy is challenging. If we build and combine multiple models, we have the chance to boost the overall accuracy. We then implement the combination of models by aggregating the output from each model with two objectives:

Reducing the model error

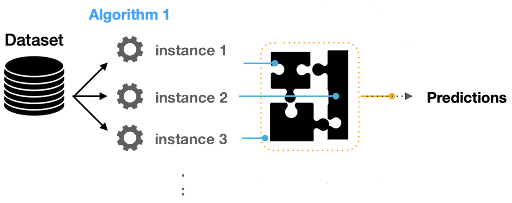
Maintaining the model’s generalization

You can implement such aggregation using different techniques, sometimes referred to as meta-algorithms.

Figure 1: Diversifying the model predictions using multiple algorithms

When we’re building ensemble models, we’re not only focusing on the algorithm’s variance. For instance, we could build multiple C45 models where each model is learning a specific pattern specialized in predicting any given thing. Models we can use to obtain a meta-model are called weak learners. In this ensemble learning architecture, the inputs are passed to each weak learner while also collecting their predictions. We can use the combined prediction to build a final ensemble model.

One important thing to mention is that weak learners can have different ways of mapping features with variant decision boundaries.

Figure 2: Aggregated predictions using multiple weak learners of the same algorithm

**TYPES OF ENSEMBLE MODELING TECHNIQUES**

1. Bagging
2. Boosting
3. Stacking
4. Blending

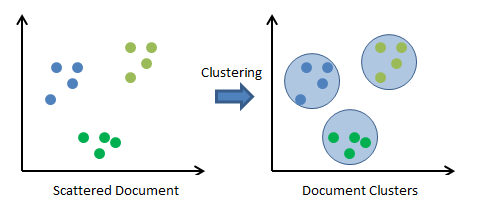
7. What is a descriptive model's main purpose? Give examples of real-world problems that descriptive models were used to solve.

A **descriptive model** is used for tasks that would benefit from the insight gained from summarizing data in new and interesting ways. As opposed to predictive models that predict a target of interest, in a descriptive model, no single feature is more important than any other. In fact, because there is no target to learn, the process of training a descriptive model is called **unsupervised learning**. Although it can be more difficult to think of applications for descriptive models, what good is a learner that isn't learning anything in particular - they are used quite regularly for *data mining*.

*So, in unsupervised learning algorithm, we do not have any target or outcome variable to predict / estimate. It is used for clustering population in different groups, which is widely used for segmenting customers in different groups for specific intervention. Examples of Unsupervised Learning: Apriori algorithm, K-means.*

For example, the descriptive modeling task called **pattern discovery** is used to identify useful associations within data. Pattern discovery is often used for **market basket analysis** on retailers' transactional purchase data. Here, the goal is to identify items that are frequently purchased together, such that the learned information can be used to refine marketing tactics. For instance, if a retailer learns that swimming trunks are commonly purchased at the same time as sunglasses, the retailer might reposition the items more closely in the store or run a promotion to "up-sell" customers on associated items.

The descriptive modeling task of dividing a dataset into homogeneous groups is called **clustering**. This is sometimes used for **segmentation analysis** that identifies groups of individuals with similar behavior or demographic information, so that advertising campaigns could be tailored for particular audiences. Although the machine is capable of identifying the clusters, human intervention is required to interpret them. For example, given five different clusters of shoppers at a grocery store, the marketing team will need to understand the differences among the groups in order to create a promotion that best suits each group.



1. Describe how to evaluate a linear regression model.

# Residual plots — Before evaluation of a model

We know that linear regression tries to fit a line that produces the smallest difference between predicted and actual values, where these differences are unbiased as well. This difference or error is also known as **residual. (**Unbiased means there is no systematic pattern of distribution of the predicted values)

**Residual = actual value — predicted value**

e = y — ŷ

It is important to note that, before assessing or evaluating our model with evaluation metrics like R-squared, we must make use of residual plots.

**Residual plots expose a biased model than any other evaluation metric. If your residual plots look normal, go ahead, and evaluate your model with various metrics.**

Residual plots show the residual values on the y-axis and predicted values on the x-axis. If your model is biased you cannot trust the results.

Residual plot showing the errors corresponding to the predicted values must be randomly distributed. However, if there are any signs of a systematic pattern, then your model is biased.

**But what does it mean by randomly distributed errors?**One of the assumptions of a linear regression model is that the errors must be normally distributed. This means, make sure your residuals are distributed around zero for the entire range of predicted values. Thus, if the residuals are evenly scattered, then your model may perform well.

# Evaluation metrics for a linear regression model

Evaluation metrics are a measure of how good a model performs and how well it approximates the relationship. Let us look at**MSE, MAE, R-squared, Adjusted R-squared, and RMSE.**

## Mean Squared Error (MSE)

The most common metric for regression tasks is MSE. It has a convex shape. It is the average of the squared difference between the predicted and actual value. Since it is differentiable and has a convex shape, it is easier to optimize.

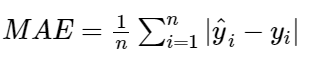
https://miro.medium.com/v2/resize:fit:315/1*aLt5bWtuBr_V7TrBILe3qQ.png

Mean squared error. Image by the author.

MSE penalizes large errors.

## Mean Absolute Error (MAE)

This is simply the average of the absolute difference between the target value and the value predicted by the model. Not preferred in cases where outliers are prominent.



Mean absolute error. Image by the author.

MAE does not penalize large errors.

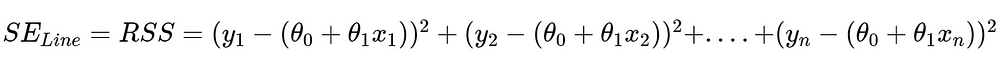
## R-squared or Coefficient of Determination

This metric represents the part of the variance of the dependent variable explained by the independent variables of the model. It measures the strength of the relationship between your model and the dependent variable.

To understand what R-square really represents let us consider the following case where we measure the error of the model with and without the knowledge of the independent variables.

**Calculating regression error**When we know the values of the independent variables, we can calculate the regression error.

We know that residual is the difference between actual and predicted value. Thus, RSS (Residual sum of squares) can be calculated as follows.



Residual sum of squares. Image by the author.

**Calculating squared residual error**Consider the case where we don't know the values of the independent variables. We only have the **y** values. With this, we calculate the **mean of the y values.** This point can be represented as a horizontal line. Now we calculate the sum of squared error between the **mean y value** and that of every other **y** value.

The total variation in Y can be given as a sum of squared differences of the distance between every point and the arithmetic mean of Y values. This can be termed as **TSS**(Total sum of squares).

https://miro.medium.com/v2/resize:fit:634/1*nmhlSwtVwD3d244JcdBTvQ.png

Total variation in y or TSS. Image by the author.

**Calculating the coefficient of determination with RSS & TSS**So we wanna find out the percentage of the total variation of Y, described by the independent variables X. If we know the percentage of the total variation of Y, that is **not**described by the regression line, we could just subtract the same from 1 to get the **coefficient of determination or R-squared.**

https://miro.medium.com/v2/resize:fit:1000/1*b0d8Q-gZ-2KL31Y7t4QK3Q.png

Image by the author.

https://miro.medium.com/v2/resize:fit:1000/1*1q9TBl7kDjol_Xq35OHmrg.png

Coefficient of determination. Image by the author.

If the data points are very close to the regression line, then the model accounts for a good amount of variance, thus resulting in a high R² value.

*However do not let the R² value fool you. A good model can have low R² value and a biased model can have a high R² value as well. That is the reason you should make use of residual plots.*

To summarize, the ratio of the residual error (RSS) against the total error (TSS) tells you how much of the total error remains in your regression model. Subtracting that ratio from 1 gives how much error you removed using the regression analysis. That is the R-squared error.

**If R² is high (say 1), then the model represents the variance of the dependent variable.**

**If R² is very low, then the model does not represent the variance of the dependent variable and regression is no better than taking the mean value, i.e. you are not using any information from the other variables.**

**A Negative R² means you are doing worse than the mean value. It can have a negative value if the predictors do not explain the dependent variables at all such that RSS ~ TSS.**

Thus R² evaluates the scattered data points about the regression line.

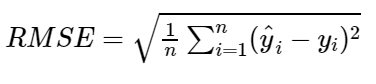
It is not possible to see a model with an R² of 1. In that case, all predicted values are the same as actual values and this essentially means that all values fall on the regression line.

## Root Mean Squared Error (RMSE)

This is the square root of the average of the squared difference of the predicted and actual value.

R-squared error is better than RMSE. This is because R-squared is a relative measure while RMSE is an absolute measure of fit (highly dependent on the variables — not a normalized measure).

Basically, RMSE is just the root of the average of squared residuals. We know that residuals are a measure of how distant the points are from the regression line. Thus, RMSE measures the scatter of these residuals.



Root mean square error. Image by the author.

RMSE penalizes large errors.

9. Distinguish :

1. Descriptive vs. predictive models

# Difference Between Descriptive and Predictive Data Mining

The descriptive and predictive data mining techniques have huge applications in data mining; they are used to mine the types of patterns. The descriptive analysis is used to mine data and specify the current data on past events. In contrast, the predictive analysis gives the answers to all queries related to recent or previous data that move across using historical data as the main principle for decision.

The task of data mining can be predictive, descriptive and prescriptive. In this article, we will discuss the two terms, predictive data mining and descriptive data mining, separately. In laymen language, you can say that descriptive mining involves finding interesting patterns or associations relating to data. In contrast, predictive mining involves the prediction and classification of the data gathered in past or current. Read the article to learn the difference between descriptive and predictive data mining.

## What is descriptive data mining?

Descriptive mining is usually used to provide correlation, cross-tabulation, frequency, etc. These techniques are used to determine the data regularities and to reveal patterns. It targets the summarization and conversion of data into meaningful data for reporting and monitoring.

As the name suggests, descriptive mining "describe" the data. Once the data is captured, we convert it into human interpretable form. Descriptive analytics focus on answering "What has happened in the past?" Descriptive analytics is useful because it enables us to learn from the past.

How descriptive analytics is used in learning analytics

* Comparing pre-test and post-test assessments.
* Tracking course enrollments.
* Collating course survey results.
* Recording which learning resources and accessed and how often.
* Summarizing the number of times, a learner posts on a discussion board.

## What is Predictive data Mining?

The term 'Predictive' means to predict something, so predictive data mining is the analysis done to predict the future event or other data or trends. Predictive data mining can enable business analysts to make decisions and add value to the analytics team efforts. Predictive data mining supports predictive analytics. As we know, predictive analytics is the use of information to predict outcomes.

Let's understand this concept with the help of an example;

Any retail shop may use algorithm-based tools to go through a customer database to look at the previous transactions to predict future transactions. In other words, the previous data may enable the shopkeeper to project what will happen in future in the business, enabling business people to plan accordingly.

### **Advantages of Predictive mining in business**

Following are the given most important business benefits of Predictive mining.

1. It increases the company production.
2. It reduces risks in business.
3. It helps business analysts to make better decisions in a business organization.
4. It helps to maintain a competitive environment.

## Difference between predictive and descriptive data mining

|  |  |
| --- | --- |
| **Descriptive data mining** | **Predictive data mining** |
| Descriptive mining is usually used to provide correlation, cross-tabulation, frequency, etc. | The term 'Predictive' means to predict something, so predictive data mining is the analysis done to predict the future event or other data or trends. |
| It is based on the reactive approach. | It is based on the proactive approach. |
| It specifies the characteristics of the data in a target data set. | It executes the induction over the current and past data so that prediction can happen. |
| It needs data aggregation and data mining. | It needs statistics and data forecasting procedures. |
| It provides precise data. | It produces outcomes without ensuring accuracy. |

1. Underfitting vs. overfitting the model

Overfitting and Underfitting are the two main problems that occur in machine learning and degrade the performance of the machine learning models.

The main goal of each machine learning model is **to generalize well**. Here **generalization** defines the ability of an ML model to provide a suitable output by adapting the given set of unknown input. It means after providing training on the dataset, it can produce reliable and accurate output. Hence, the underfitting and overfitting are the two terms that need to be checked for the performance of the model and whether the model is generalizing well or not.

Before understanding the overfitting and underfitting, let's understand some basic term that will help to understand this topic well:

* **Signal:** It refers to the true underlying pattern of the data that helps the machine learning model to learn from the data.
* **Noise:** Noise is unnecessary and irrelevant data that reduces the performance of the model.
* **Bias:** Bias is a prediction error that is introduced in the model due to oversimplifying the machine learning algorithms. Or it is the difference between the predicted values and the actual values.
* **Variance:** If the machine learning model performs well with the training dataset, but does not perform well with the test dataset, then variance occurs.

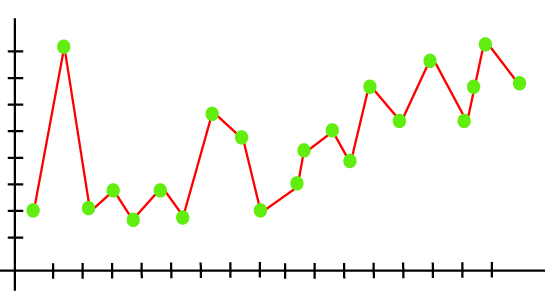
## Overfitting

Overfitting occurs when our [machine learning](https://www.javatpoint.com/machine-learning) model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model. The overfitted model has **low bias** and **high variance.**

The chances of occurrence of overfitting increase as much we provide training to our model. It means the more we train our model, the more chances of occurring the overfitted model.

Overfitting is the main problem that occurs in [supervised learning](https://www.javatpoint.com/supervised-machine-learning).

**Example:** The concept of the overfitting can be understood by the below graph of the linear regression output:



As we can see from the above graph, the model tries to cover all the data points present in the scatter plot. It may look efficient, but in reality, it is not so. Because the goal of the regression model to find the best fit line, but here we have not got any best fit, so, it will generate the prediction errors.

### **How to avoid the Overfitting in Model**

Both overfitting and underfitting cause the degraded performance of the machine learning model. But the main cause is overfitting, so there are some ways by which we can reduce the occurrence of overfitting in our model.

* **Cross-Validation**
* **Training with more data**
* **Removing features**
* **Early stopping the training**
* **Regularization**
* **Ensembling**

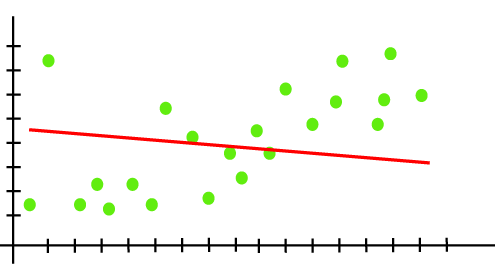
## Underfitting

Underfitting occurs when our machine learning model is not able to capture the underlying trend of the data. To avoid the overfitting in the model, the fed of training data can be stopped at an early stage, due to which the model may not learn enough from the training data. As a result, it may fail to find the best fit of the dominant trend in the data.

In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and produces unreliable predictions.

An underfitted model has high bias and low variance.

**Example:** We can understand the underfitting using below output of the linear regression model:



As we can see from the above diagram, the model is unable to capture the data points present in the plot.

### **How to avoid underfitting:**

* By increasing the training time of the model.
* By increasing the number of features.

## Goodness of Fit

The "Goodness of fit" term is taken from the statistics, and the goal of the machine learning models to achieve the goodness of fit. In statistics modeling, it defines how closely the result or predicted values match the true values of the dataset.

The model with a good fit is between the underfitted and overfitted model, and ideally, it makes predictions with 0 errors, but in practice, it is difficult to achieve it.

As when we train our model for a time, the errors in the training data go down, and the same happens with test data. But if we train the model for a long duration, then the performance of the model may decrease due to the overfitting, as the model also learn the noise present in the dataset. The errors in the test dataset start increasing, so the point, just before the raising of errors, is the good point, and we can stop here for achieving a good model.

There are two other methods by which we can get a good point for our model, which are the **resampling method** to estimate model accuracy and **validation dataset**.

1. Bootstrapping vs. cross-validation

### Bootstrapping or Cross-Validation

1. Bootstrapping selects samples with replacements that can be as big as the dataset.
2. Cross-validation samples are smaller than the dataset.
3. Bootstrapping contains repeated elements in every subset. Bootstrapping relies on random sampling.
4. Cross-validation does not rely on random sampling, just splitting the dataset into k unique subsets.
5. Cross-validation is usually used to test an ML model's generalization capabilities.
6. Bootstrapping is used more for statistical tests, ensemble machine learning, and parameter estimation.

### Boostraping

Bootstrapping is a resampling technique with replacement; that is, we can choose on every sample a subset of elements that might be repeated.

Cross-validations is a very similar technique to Bootstrapping, with the difference that it selects its samples without replacement; that is, there are no repeated elements in every subset. Selecting k-samples with Cross-Validation is called K-Fold CrossValidation. Usually, on each cross-validation exercise, we define a portion of the selected sample to be the train and the other the test set (for example 70/30 or 80/20). The type of cross-validation that selects a test set with one single example is called LOOCV (leave one out cross-validation).

Subsetting a dataset using LOOCV is computationally expensive, so with usually use k-fold cross-validation with a k = {4, 5, 7, 10}

10. Make quick notes on:

1. LOOCV.

The **Leave-One-Out Cross-Validation**, or **LOOCV**, procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a computationally expensive procedure to perform, although it results in a reliable and unbiased estimate of model performance. Although simple to use and no configuration to specify, there are times when the procedure should not be used, such as when you have a very large dataset or a computationally expensive model to evaluate.

Cross-validation, or k-fold cross-validation, is a procedure used to estimate the performance of a machine learning algorithm when making predictions on data not used during the training of the model.

The cross-validation has a single hyperparameter “*k*” that controls the number of subsets that a dataset is split into. Once split, each subset is given the opportunity to be used as a test set while all other subsets together are used as a training dataset.

This means that k-fold cross-validation involves fitting and evaluating *k* models. This, in turn, provides k estimates of a model’s performance on the dataset, which can be reported using summary statistics such as the mean and standard deviation. This score can then be used to compare and ultimately select a model and configuration to use as the “*final model*” for a dataset.

Typical values for k are k=3, k=5, and k=10, with 10 representing the most common value. This is because, given extensive testing, 10-fold cross-validation provides a good balance of low computational cost and low bias in the estimate of model performance as compared to other k values and a single train-test split.

For more on k-fold cross-validation, see the tutorial:

* [A Gentle Introduction to k-fold Cross-Validation](https://machinelearningmastery.com/k-fold-cross-validation/)

Leave-one-out cross-validation, or LOOCV, is a configuration of k-fold cross-validation where *k* is set to the number of examples in the dataset.

LOOCV is an extreme version of k-fold cross-validation that has the maximum computational cost. It requires one model to be created and evaluated for each example in the training dataset.

The benefit of so many fit and evaluated models is a more robust estimate of model performance as each row of data is given an opportunity to represent the entirety of the test dataset.

Given the computational cost, LOOCV is not appropriate for very large datasets such as more than tens or hundreds of thousands of examples, or for models that are costly to fit, such as neural networks.

* **Don’t Use LOOCV**: Large datasets or costly models to fit.

Given the improved estimate of model performance, LOOCV is appropriate when an accurate estimate of model performance is critical. This particularly case when the dataset is small, such as less than thousands of examples, can lead to model overfitting during training and biased estimates of model performance.

Further, given that no sampling of the training dataset is used, this estimation procedure is deterministic, unlike train-test splits and other k-fold cross-validation confirmations that provide a stochastic estimate of model performance.

* **Use LOOCV**: Small datasets or when estimated model performance is critical.

Once models have been evaluated using LOOCV and a final model and configuration chosen, a final model is then fit on all available data and used to make predictions on new data.

2. F-measurement

Fbeta-measure is a configurable single-score metric for evaluating a binary classification model based on the predictions made for the positive class.

The Fbeta-measure is calculated using precision and recall.

**Precision** is a metric that calculates the percentage of correct predictions for the positive class. **Recall** calculates the percentage of correct predictions for the positive class out of all positive predictions that could be made. Maximizing precision will minimize the false-positive errors, whereas maximizing recall will minimize the false-negative errors.

The **F-measure** is calculated as the harmonic mean of precision and recall, giving each the same weighting. It allows a model to be evaluated taking both the precision and recall into account using a single score, which is helpful when describing the performance of the model and in comparing models.

The **Fbeta-measure** is a generalization of the F-measure that adds a configuration parameter called beta. A default beta value is 1.0, which is the same as the F-measure. A smaller beta value, such as 0.5, gives more weight to precision and less to recall, whereas a larger beta value, such as 2.0, gives less weight to precision and more weight to recall in the calculation of the score.

It is a useful metric to use when both precision and recall are important but slightly more attention is needed on one or the other, such as when false negatives are more important than false positives, or the reverse.

In this tutorial, you will discover the Fbeta-measure for evaluating classification algorithms for machine learning.

After completing this tutorial, you will know:

* Precision and recall provide two ways to summarize the errors made for the positive class in a binary classification problem.
* F-measure provides a single score that summarizes the precision and recall.
* Fbeta-measure provides a configurable version of the F-measure to give more or less attention to the precision and recall measure when calculating a single score.

## Precision and Recall

Before we can dive into the Fbeta-measure, we must review the basics of the precision and recall metrics used to evaluate the predictions made by a classification model.

### Confusion Matrix

A [confusion matrix](https://machinelearningmastery.com/ufaqs/what-is-a-confusion-matrix/) summarizes the number of predictions made by a model for each class, and the classes to which those predictions actually belong. It helps to understand the types of prediction errors made by a model.

The simplest confusion matrix is for a two-class classification problem, with negative (class 0) and positive (class 1) classes.

In this type of confusion matrix, each cell in the table has a specific and well-understood name, summarized as follows:

|  |  |
| --- | --- |
| 1  2  3 | | Positive Prediction | Negative Prediction  Positive Class | True Positive (TP)  | False Negative (FN)  Negative Class | False Positive (FP) | True Negative (TN) |

The precision and recall metrics are defined in terms of the cells in the confusion matrix, specifically terms like true positives and false negatives.

### Precision

Precision is a metric that quantifies the number of correct positive predictions made.

It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that were predicted.

* Precision = TruePositives / (TruePositives + FalsePositives)

The result is a value between 0.0 for no precision and 1.0 for full or perfect precision.

The intuition for precision is that it is not concerned with false negatives and it **minimizes false positives**

### Recall

Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made.

It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that could be predicted.

* Recall = TruePositives / (TruePositives + FalseNegatives)

The result is a value between 0.0 for no recall and 1.0 for full or perfect recall.

**F-Measure**

Precision and recall measure the two types of errors that could be made for the positive class.

Maximizing precision minimizes false positives and maximizing recall minimizes false negatives.

F-Measure or F-Score provides a way to combine both precision and recall into a single measure that captures both properties.

* F-Measure = (2 \* Precision \* Recall) / (Precision + Recall)

This is the [harmonic mean](https://en.wikipedia.org/wiki/Harmonic_mean) of the two fractions.

The result is a value between 0.0 for the worst F-measure and 1.0 for a perfect F-measure.

The intuition for F-measure is that both measures are balanced in importance and that only a good precision and good recall together result in a good F-measure.

### 50% Precision, Perfect Recall

It is not possible to have perfect precision and no recall, or no precision and perfect recall. Both precision and recall require true positives to be predicted.

Consider the case where we predict the positive class for all cases.

This would give us 50 percent precision as half of the predictions are false positives. It would give us perfect recall because we would no false negatives.

For the balanced dataset we are using in our examples, half of the predictions would be true positives, half would be false positives; therefore, the precision ratio would be 0.5 or 50 percent. Combining 50 percept precision with perfect recall will result in a penalized F-measure, specifically the harmonic mean between 50 percent and 100 percent.

**Fbeta-Measure**

The F-measure balances the precision and recall.

On some problems, we might be interested in an F-measure with more attention put on precision, such as when false positives are more important to minimize, but false negatives are still important.

On other problems, we might be interested in an F-measure with more attention put on recall, such as when false negatives are more important to minimize, but false positives are still important.

The solution is the Fbeta-measure.

The Fbeta-measure measure is an abstraction of the F-measure where the balance of precision and recall in the calculation of the [harmonic mean](https://en.wikipedia.org/wiki/Harmonic_mean) is controlled by a coefficient called *beta*.

* Fbeta = ((1 + beta^2) \* Precision \* Recall) / (beta^2 \* Precision + Recall)

The choice of the beta parameter will be used in the name of the Fbeta-measure.

For example, a beta value of 2 is referred to as F2-measure or F2-score. A beta value of 1 is referred to as the F1-measure or the F1-score.

Three common values for the beta parameter are as follows:

* **F0.5-Measure** (beta=0.5): More weight on precision, less weight on recall.
* **F1-Measure** (beta=1.0): Balance the weight on precision and recall.
* **F2-Measure** (beta=2.0): Less weight on precision, more weight on recall

The impact on the calculation for different beta values is not intuitive, at first.

### F1-Measure

The F-measure discussed in the previous section is an example of the Fbeta-measure with a beta value of 1.

Specifically, F-measure and F1-measure calculate the same thing; for example:

* F-Measure = ((1 + 1^2) \* Precision \* Recall) / (1^2 \* Precision + Recall)
* F-Measure = (2 \* Precision \* Recall) / (Precision + Recall)

Consider the case where we have 50 percept precision and perfect recall. We can manually calculate the F1-measure for this case as follows:

* F-Measure = (2 \* Precision \* Recall) / (Precision + Recall)
* F-Measure = (2 \* 0.5 \* 1.0) / (0.5 + 1.0)
* F-Measure = 1.0 / 1.5
* F-Measure = 0.666

We can confirm this calculation using the [fbeta\_score() function](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html) in scikit-learn with the “beta” argument set to 1.0.

### F0.5-Measure

The F0.5-measure is an example of the Fbeta-measure with a beta value of 0.5.

It has the effect of raising the importance of precision and lowering the importance of recall.

If maximizing precision minimizes false positives, and maximizing recall minimizes false negatives, then the **F0.5-measure puts more attention on minimizing false positives** than minimizing false negatives.

The F0.5-Measure is calculated as follows:

* F0.5-Measure = ((1 + 0.5^2) \* Precision \* Recall) / (0.5^2 \* Precision + Recall)
* F0.5-Measure = (1.25 \* Precision \* Recall) / (0.25 \* Precision + Recall)

Consider the case where we have 50 percent precision and perfect recall. We can manually calculate the F0.5-measure for this case as follows:

* F0.5-Measure = (1.25 \* Precision \* Recall) / (0.25 \* Precision + Recall)
* F0.5-Measure = (1.25 \* 0.5 \* 1.0) / (0.25 \* 0.5 + 1.0)
* F0.5-Measure = 0.625 / 1.125
* F0.5-Measure = 0.555

We would expect that a beta value of 0.5 would result in a lower score for this scenario given that precision has a poor score and the recall is excellent.

This is exactly what we see, where an F0.5-measure of 0.555 is achieved for the same scenario where an F1-score was calculated as 0.667. Precision played more of a role in the calculation.

### F2-Measure

The F2-measure is an example of the Fbeta-measure with a beta value of 2.0.

It has the effect of lowering the importance of precision and increase the importance of recall.

If maximizing precision minimizes false positives, and maximizing recall minimizes false negatives, then the **F2-measure puts more attention on minimizing false negatives** than minimizing false positives.

The F2-measure is calculated as follows:

* F2-Measure = ((1 + 2^2) \* Precision \* Recall) / (2^2 \* Precision + Recall)
* F2-Measure = (5 \* Precision \* Recall) / (4 \* Precision + Recall)

Consider the case where we have 50 percent precision and perfect recall.

We can manually calculate the F2-measure for this case as follows:

* F2-Measure = (5 \* Precision \* Recall) / (4 \* Precision + Recall)
* F2-Measure = (5 \* 0.5 \* 1.0) / (4 \* 0.5 + 1.0)
* F2-Measure = 2.5 / 3.0
* F2-Measure = 0.833

We would expect that a beta value of 2.0 would result in a higher score for this scenario given that recall has a perfect score, which will be promoted over that of the poor performance of precision.

This is exactly what we see where an F2-measure of 0.833 is achieved for the same scenario where an F1-score was calculated as 0.667. Recall played more of a role in the calculation.

1. The width of the silhouette

One of the fundamental steps of an unsupervised learning algorithm is to determine the number of clusters into which the data may be divided. The silhouette algorithm is one of the many algorithms to determine the optimal number of clusters for an unsupervised learning technique.

In the Silhouette algorithm, we assume that the data has already been clustered into k clusters by a clustering technique(Typically [K-Means Clustering technique](https://www.geeksforgeeks.org/k-means-clustering-introduction/)). Then for each data point, we define the following:-

C(i) -The cluster assigned to the ith data point

|C(i)| – The number of data points in the cluster assigned to the ith data point

a(i) – It gives a measure of how well assigned the ith data point is to it’s cluster

b(i) – It is defined as the average dissimilarity to the closest cluster which is not it’s cluster

The silhouette coefficient s(i) is given by:-

We determine the average silhouette for each value of k and for the value of k which has the **maximum value of s(i)** is considered the optimal number of clusters for the unsupervised learning algorithm.

Let us consider the following data:-

| **S.No** | **X1** | **X2** |
| --- | --- | --- |
| 1. | -7.36 | 6.37 |
| 2. | 3.08 | -6.78 |
| 3. | 5.03 | -8.31 |
| 4. | -1.93 | -0.92 |
| 5. | -8.86 | 6.60 |

We now iterate the values of k from 2 to 5. We assume that no practical data exists for which all the data points can be optimally clustered into 1 cluster.

We construct the following tables for each value of k:-

**k = 2**

| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | 5.31 | 14.1 | 0.62 |
| 2. | 2.47 | 13.15 | 0.81 |
| 3. | 2.47 | 14.97 | 0.84 |
| 4. | 9.66 | 8.93 | -0.076 |
| 5. | 5.88 | 19.16 | 0.69 |

**Average value of s(i) = 0.58**

**k = 3**

| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | 1.52 | 9.09 | 0.83 |
| 2. | 2.47 | 7.71 | 0.68 |
| 3. | 2.47 | 10.15 | 0.76 |
| 4. | 0 | 7.71 | 1 |
| 5. | 1.52 | 17.93 | 0.92 |

**Average value of s(i) = 0.84**

**k = 4**

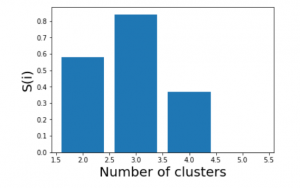
| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | 1.52 | 9.09 | 0.83 |
| 2. | infinite | 2.47 | 0 |
| 3. | infinite | 2.47 | 0 |
| 4. | infinite | 7.71 | 0 |
| 5. | 1.52 | 10.23 | 0.85 |

**Average value of s(i) = 0.37**

**k = 5**

| **S.No** | **a(i)** | **b(i)** | **s(i)** |
| --- | --- | --- | --- |
| 1. | infinite | 1.52 | 0 |
| 2. | infinite | 2.47 | 0 |
| 3. | infinite | 2.47 | 0 |
| 4. | infinite | 7.71 | 0 |
| 5. | infinite | 1.52 | 0 |

**Average value of s(i) = 0**



We see that the highest value of s(i) exists for k = 3. Therefore we conclude that the optimal number of clusters for the given data is 3.

4. Receiver operating characteristic curve

**ROC curve**

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

* True Positive Rate
* False Positive Rate

**True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:

���=����+��

**False Positive Rate** (**FPR**) is defined as follows:

���=����+��

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

**Figure 4. TP vs. FP rate at different classification thresholds.**

To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds, but this would be inefficient. Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called AUC.

**AUC: Area Under the ROC Curve**

**AUC** stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

**Figure 5. AUC (Area under the ROC Curve).**

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:

**Figure 6. Predictions ranked in ascending order of logistic regression score.**

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

AUC is desirable for the following two reasons:

* AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.
* AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

However, both these reasons come with caveats, which may limit the usefulness of AUC in certain use cases:

* **Scale invariance is not always desirable.** For example, sometimes we really do need well calibrated probability outputs, and AUC won’t tell us about that.
* **Classification-threshold invariance is not always desirable.** In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize one type of classification error. For example, when doing email spam detection, you likely want to prioritize minimizing false positives (even if that results in a significant increase of false negatives). AUC isn't a useful metric for this type of optimization.