# K-Nearest Neighbor(KNN) Algorithm

Last Updated : 25 Jan, 2024

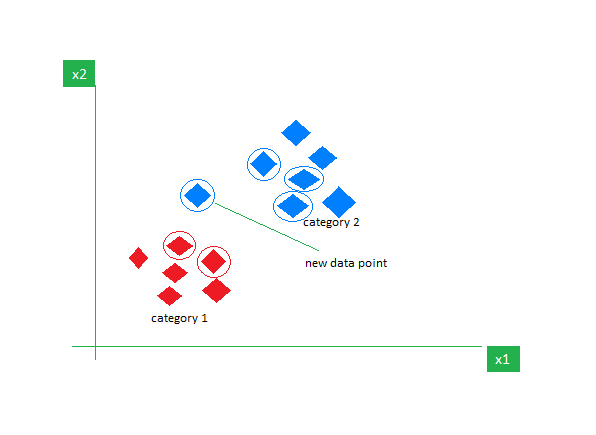
The **K-Nearest Neighbors (KNN) algorithm** is a supervised machine learning method employed to tackle classification and regression problems. Evelyn Fix and Joseph Hodges developed this algorithm in 1951, which was subsequently expanded by Thomas Cover. The article explores the fundamentals, workings, and implementation of the KNN algorithm.

## What is the K-Nearest Neighbors Algorithm?

KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the [supervised learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) domain and finds intense application in pattern recognition, [data mining](https://www.geeksforgeeks.org/data-mining/), and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a [Gaussian distribution](https://www.geeksforgeeks.org/mathematics-probability-distributions-set-3-normal-distribution/) of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:



*KNN Algorithm working visualization*

Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set. Note that the unclassified points are marked as ‘White’.

## ****Intuition Behind KNN Algorithm****

If we plot these points on a graph, we may be able to locate some clusters or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbors belong to. This means a point close to a cluster of points classified as ‘Red’ has a higher probability of getting classified as ‘Red’.

Intuitively, we can see that the first point (2.5, 7) should be classified as ‘Green’, and the second point (5.5, 4.5) should be classified as ‘Red’.

## Why do we need a KNN algorithm?

(K-NN) algorithm is a versatile and widely used machine learning algorithm that is primarily used for its simplicity and ease of implementation. It does not require any assumptions about the underlying data distribution. It can also handle both numerical and categorical data, making it a flexible choice for various types of datasets in classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of data points in a given dataset. K-NN is less sensitive to outliers compared to other algorithms.

The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

## Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

### Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. [Euclidean distance](https://www.geeksforgeeks.org/calculate-the-euclidean-distance-using-numpy/) can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

### Manhattan Distance

[Manhattan Distance](https://www.geeksforgeeks.org/how-to-calculate-manhattan-distance-in-r/) metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

### Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the [Minkowski distance](https://www.geeksforgeeks.org/minkowski-distance-python/).

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

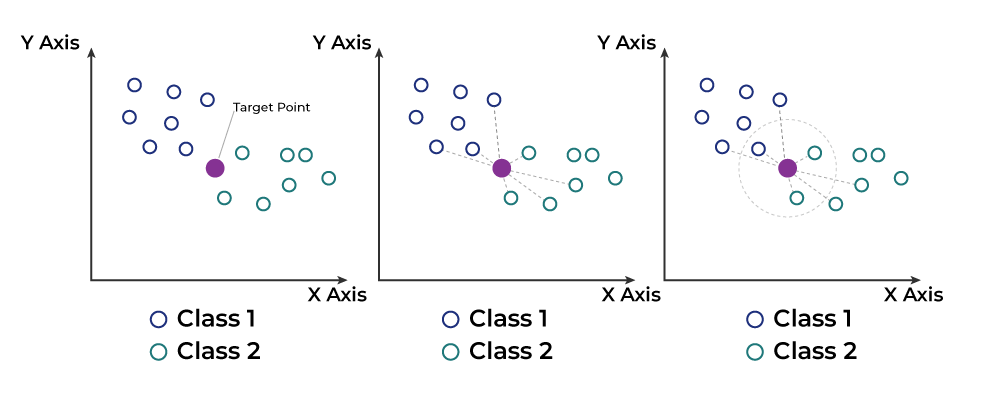
The above-discussed metrics are most common while dealing with a [Machine Learning](https://www.geeksforgeeks.org/machine-learning/) problem but there are other distance metrics as well like [Hamming Distance](https://www.geeksforgeeks.org/hamming-distance-two-strings/) which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be Boolean as well as string values.

## How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. [Cross-validation](https://www.geeksforgeeks.org/cross-validation-machine-learning/) methods can help in selecting the best k value for the given dataset.

## Workings of KNN algorithm

Thе K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



Step-by-Step explanation of how KNN works is discussed below:

### Step 1: Selecting the optimal value of K

* K represents the number of nearest neighbors that needs to be considered while making prediction.

### Step 2: Calculating distance

* To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

### Step 3: Finding Nearest Neighbors

* The k data points with the smallest distances to the target point are the nearest neighbors.

### Step 4: Voting for Classification or Taking Average for Regression

* In the classification problem, the class labels of are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
* In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Let X be the training dataset with n data points, where each data point is represented by a d-dimensional feature vector  and Y be the corresponding labels or values for each data point in X. Given a new data point x, the algorithm calculates the distance between x and each data point  in X using a distance metric, such as Euclidean distance:

The algorithm selects the K data points from X that have the shortest distances to x. For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x. For regression tasks, the algorithm calculates the average or weighted average of the values y of the K nearest neighbors and assigns it as the predicted value for x.

## Advantages of the KNN Algorithm

* **Easy to implement** as the complexity of the algorithm is not that high.
* **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
* **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

## Disadvantages of the KNN Algorithm

* **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
* **Curse of Dimensionality** – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the [curse of dimensionality](https://www.geeksforgeeks.org/videos/curse-of-dimensionality-in-machine-learning/) which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.
* **Prone to Overfitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally [feature selection](https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/) as well as [dimensionality reduction](https://www.geeksforgeeks.org/dimensionality-reduction/) techniques are applied to deal with this problem.

# Support Vector Machine (SVM) Algorithm

Last Updated : 10 Jun, 2023

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships.

SVM algorithms are very effective as we try to find the maximum separating hyperplane between the different classes available in the target feature.

## Support Vector Machine

Support Vector Machine (SVM) is a [supervised machine learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The main objective of the SVM algorithm is to find the optimal [hyperplane](https://www.geeksforgeeks.org/separating-hyperplanes-in-svm/) in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Let’s consider two independent variables x1, x2, and one dependent variable which is either a blue circle or a red circle.



*Linearly Separable Data points*

From the figure above it’s very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregate our data points or do a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points?

### How does SVM work?

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.



*Multiple hyperplanes separate the data from two classes*

So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the **maximum-margin hyperplane/hard margin**. So from the above figure, we choose L2. Let’s consider a scenario like shown below



*Selecting hyperplane for data with outlier*

Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It’s simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



*Hyperplane which is the most optimized one*

So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called **soft margins**. When there is a soft margin to the data set, the SVM tries to minimize *(1/margin+∧(∑penalty))*. Hinge loss is a commonly used penalty. If no violations no hinge loss.If violations hinge loss proportional to the distance of violation.

Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable by a straight line/linear line). What to do if data are not linearly separable?



*Original 1D dataset for classification*

Say, our data is shown in the figure above. SVM solves this by creating a new variable using a **kernel**. We call a point xion the line and we create a new variable yi as a function of distance from origin o.so if we plot this we get something like as shown below



*Mapping 1D data to 2D to become able to separate the two classes*

In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as a kernel.

### Support Vector Machine Terminology

1. **Hyperplane:**Hyperplane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of linear classifications, it will be a linear equation i.e. wx+b = 0.
2. **Support Vectors:**Support vectors are the closest data points to the hyperplane, which makes a critical role in deciding the hyperplane and margin.
3. **Margin**: Margin is the distance between the support vector and hyperplane. The main objective of the support vector machine algorithm is to maximize the margin.  The wider margin indicates better classification performance.
4. **Kernel**: Kernel is the mathematical function, which is used in SVM to map the original input data points into high-dimensional feature spaces, so, that the hyperplane can be easily found out even if the data points are not linearly separable in the original input space. Some of the common kernel functions are linear, polynomial, radial basis function(RBF), and sigmoid.
5. **Hard Margin:** The maximum-margin hyperplane or the hard margin hyperplane is a hyperplane that properly separates the data points of different categories without any misclassifications.
6. **Soft Margin:**When the data is not perfectly separable or contains outliers, SVM permits a soft margin technique. Each data point has a slack variable introduced by the soft-margin SVM formulation, which softens the strict margin requirement and permits certain misclassifications or violations. It discovers a compromise between increasing the margin and reducing violations.
7. **C:**Margin maximisation and misclassification fines are balanced by the regularisation parameter C in SVM. The penalty for going over the margin or misclassifying data items is decided by it. A stricter penalty is imposed with a greater value of C, which results in a smaller margin and perhaps fewer misclassifications.
8. **Hinge Loss:** A typical loss function in SVMs is hinge loss. It punishes incorrect classifications or margin violations. The objective function in SVM is frequently formed by combining it with the regularisation term.
9. **Dual Problem:** A dual Problem of the optimisation problem that requires locating the Lagrange multipliers related to the support vectors can be used to solve SVM. The dual formulation enables the use of kernel tricks and more effective computing.

### Mathematical intuition of Support Vector Machine

Consider a binary classification problem with two classes, labeled as +1 and -1. We have a training dataset consisting of input feature vectors X and their corresponding class labels Y.

The equation for the linear hyperplane can be written as:

The vector W represents the normal vector to the hyperplane. i.e the direction perpendicular to the hyperplane. The parameter **b** in the equation represents the offset or distance of the hyperplane from the origin along the normal vector **w**.

The distance between a data point x\_i and the decision boundary can be calculated as:

where ||w|| represents the Euclidean norm of the weight vector w. Euclidean norm of the normal vector W

For Linear SVM classifier :

#### ****Optimization:****

* **For Hard margin linear SVM classifier:**

The target variable or label for the ith training instance is denoted by the symbol ti in this statement. And ti=-1 for negative occurrences (when yi= 0) and ti=1positive instances (when yi = 1) respectively. Because we require the decision boundary that satisfy the constraint:

* **For Soft margin linear SVM classifier:**
* **Dual Problem:** A dual Problem of the optimisation problem that requires locating the Lagrange multipliers related to the support vectors can be used to solve SVM. The optimal Lagrange multipliers α(i) that maximize the following dual objective function

where,

* αi is the Lagrange multiplier associated with the ith training sample.
* K(xi, xj) is the kernel function that computes the similarity between two samples xi and xj. It allows SVM to handle nonlinear classification problems by implicitly mapping the samples into a higher-dimensional feature space.
* The term ∑αi represents the sum of all Lagrange multipliers.

The SVM decision boundary can be described in terms of these optimal Lagrange multipliers and the support vectors once the dual issue has been solved and the optimal Lagrange multipliers have been discovered. The training samples that have i > 0 are the support vectors, while the decision boundary is supplied by:

### Types of Support Vector Machine

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

* **Linear SVM:**Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.
* **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

### Popular kernel functions in SVM

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

### ****Advantages of SVM****

* Effective in high-dimensional cases.
* Its memory is 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.

### ****SVM implementation in Python****

Predict if cancer is Benign or malignant. Using historical data about patients diagnosed with cancer enables doctors to differentiate malignant cases and benign ones are given independent attributes.

#### Steps

* Load the breast cancer dataset from sklearn.datasets
* Separate input features and target variables.
* Buil and train the SVM classifiers using RBF kernel.
* Plot the scatter plot of the input features.
* Plot the decision boundary.
* Plot the decision boundary

#### Recall

Recall, also called sensitivity, measures the model's ability to detect positive events correctly. It is the percentage of accurately predicted positive events out of all actual positive events. To calculate the recall of a classification model, the formula is

blog image

Using the classification model outcomes from Table 1 above, recall is calculated as

blog image

A high recall score indicates that the classifier predicts the majority of the relevant results correctly. However, the recall metric does not take into account the potential repercussions of false positives, i.e., occurrences that are wrongly identified as positive – a false alarm. Typically, we would like to avoid such cases, especially in mission-critical applications such as intrusion detection, where a non-malicious false alarm increases the workload of overburdened security teams.

While precision and recall give useful information on their own, they also have limitations when viewed separately.

Ideally, we want to build classifiers with high precision and recall. But that’s not always possible. A classifier with high recall may have low precision, meaning it captures the majority of positive classes but produces a considerable number of false positives. Hence, we use the F1 score metric to balance this precision-recall trade-off.

## F1 Score Metric

The[**F1 score or F-measure**](https://encord.com/glossary/f1-score-definition/) is described as the harmonic mean of the precision and recall of a classification model. The two metrics contribute equally to the score, ensuring that the F1 metric correctly indicates the reliability of a model.

It’s important to note that calculating the F1 score using arithmetic mean may not appropriately represent the model's overall performance, especially when precision and recall have considerably varied values. That’s because the arithmetic mean focuses on the sum of values and their average.

On the other hand, the **harmonic mean** emphasizes the reciprocal of values. It is computed by dividing the total number of values by the sum of their reciprocals. Hence, it enhances the effect of the smaller value on the overall calculation to achieve a balanced measurement. As a result, the F1 score takes into account both precision-recall while avoiding the overestimation that the arithmetic mean might cause.

The F1 score formula is

blog image

Using the classification model outcomes from Table 1, the F1 score is calculated as

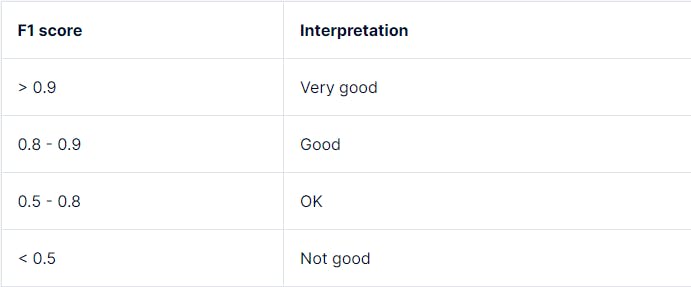
blog image

Here, you can observe that the harmonic mean of precision and recall creates a balanced measurement, i.e., the model's precision is not optimized at the price of recall, or vice versa. Hence, the F1 score shows a strong performance in recognizing positive cases while minimizing false positives and false negatives. This makes it a suitable metric when recall and precision must be optimized simultaneously, especially in imbalanced datasets. As a result, the F1 score metric directs real-world decision-making more accurately.

## Interpreting the F1 Score

The F1 score ranges between 0 and 1, with 0 denoting the lowest possible result and 1 denoting a flawless result, meaning that the model accurately predicted each label.

A **high F1 score generally** indicates a well-balanced performance, demonstrating that the model can concurrently attain high precision and high recall. A **low F1 score** often signifies a trade-off between recall and precision, implying that the model has trouble striking that balance. As a general rule of thumb, the F1 score value can be interpreted as follows:



[*What is a good F1 score and how do I interpret it?*](https://stephenallwright.com/good-f1-score/)

However, depending on the task requirements, model use case, and the tolerance for mistakes, the precise threshold for what is considered “low” might also change. For instance, a simple[**decision tree**](https://encord.com/glossary/decision-tree-definition/) classifier and a multi-layered deep learning[**neural network**](https://encord.com/glossary/neural-networks-definition/) would have different ranges for high or low F1 scores.

Now, let's consider various ML applications where model evaluation requires a balance of precision and recall, deeming the F1 score as a more suitable evaluation metric.

## ML Applications of F1 Score

### Medical Diagnostics

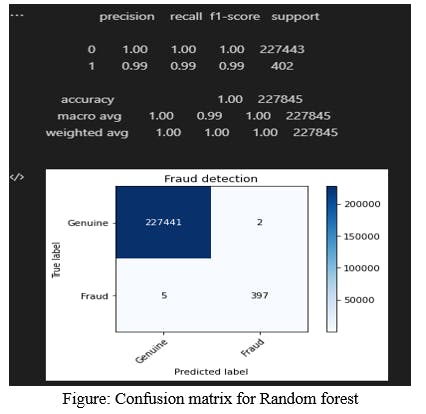
In medical diagnostics, it is important to acquire a high recall while correctly detecting positive occurrences, even if doing so necessitates losing precision. For instance, the F1 score of a cancer detection classifier should minimize the possibility of false negatives, i.e., patients with malignant cancer, but the classifier wrongly predicts as benign.

### Sentiment Analysis

For natural language processing (NLP) tasks like sentiment analysis, recognizing both positive and negative sentiments in textual data allow businesses to assess public opinion, consumer feedback, and brand sentiment. Hence, the F1 score allows for an efficient evaluation of sentiment analysis models by taking precision and recall into account when categorizing sentiments.

### Fraud Detection

In fraud detection, by considering both precision (the accuracy with which fraudulent cases are discovered) and recall (the capacity to identify all instances of fraud), the F1 score enables practitioners to assess fraud detection models more accurately. For instance, the figure below shows the evaluation metrics for a credit card fraud detection model.



[*Implementation of Credit Card Fraud Detection Using Random Forest Algorithm*](https://www.ijraset.com/research-paper/credit-card-fraud-detection-using-random-forest-algorithm)

## Limitations of F1 Score

ML practitioners must be aware of the following limits and caveats of the F1 score when interpreting its results.

### Dataset Class Imbalance

For imbalanced data, when one class significantly outweighs the other, the regular F1 scoremetric might not give a true picture of the model's performance. This is because the regular F1 score gives precision and recall equal weight, but in datasets with imbalances, achieving high precision or recall for the minority class may result in a lower F1 score due to the majority class's strong influence.

💡 Interested in learning more about class imbalance in datasets? Read our [**Introductory Blog on Balanced and Imbalanced Datasets in Machine Learning**](https://encord.com/blog/an-introduction-to-balanced-and-imbalanced-datasets-in-machine-learning/).

### Cost Associated with False Prediction Outcomes

False positives and false negatives can have quite diverse outcomes depending on the application. In medical diagnostics, as discussed earlier, a false negative is more dangerous than a false positive. Hence, the F1 score must be interpreted carefully.

### Contextual Dependence

The evaluation of the F1 score varies depending on the particular problem domain and task objectives. Various interpretations of what constitutes a high or low F1 score for different applications require various precision-recall criteria. Hence, a thorough understanding of the domain and the task at hand is needed to use and interpret the F1 score properly.

## F-score Variants

To resolve severe class imbalance issues and achieve an appropriate balance between precision and recall, practitioners often use the following two variants of the F-score metric:

### F2 Score

This variant places more emphasis on recall than precision. It is suitable for circumstances where detecting true positives is crucial. During the harmonic mean computation, recall is given more weightage.

The F2 score formula is as follows:

blog image

### F-beta Score

This variant offers a dynamic blend of recall and precision by changing the beta parameter — weight coefficient which should be greater than 0. Based on the particular task requirements, practitioners can change the beta value, i.e., beta < 1 favors precision, and beta > 1 favors recall.

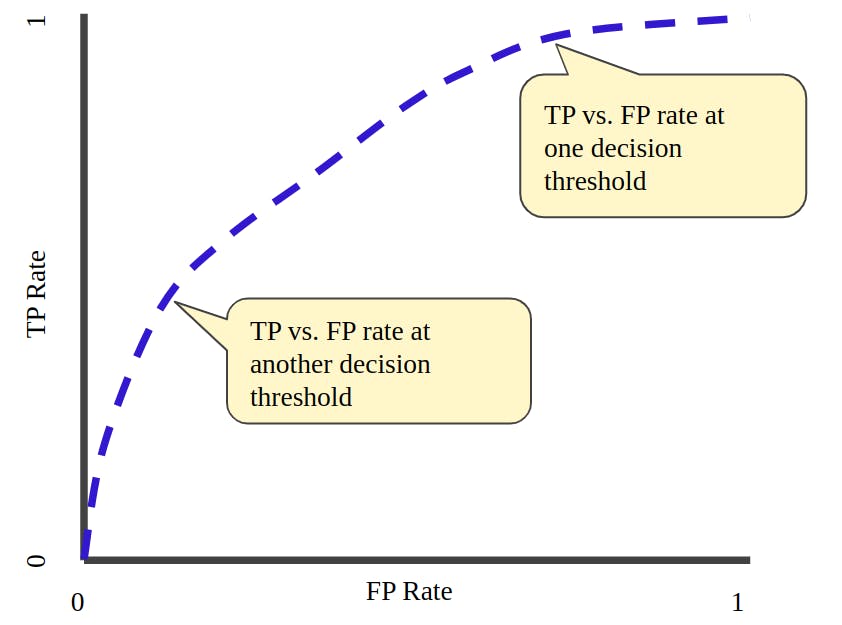
The F-beta score is calculated using the same formula as the F2 score, with beta dictating the importance of recall against precision.

blog image

## Supplementing the F1 Score

Other performance metrics, such as the Area Under the Curve-Receiver Operating Characteristic Curve (AUC-ROC) can be used in addition to the F1 score to offer supplementary insights into an[**artificial intelligence**](https://encord.com/glossary/artificial-intelligence-definition/) model performance.

The AUC-ROC metric evaluates the model's capability to differentiate between positive and negative classes across various classification criteria or decision thresholds by plotting the true positive rate (TPR) versus the false positive rate (FPR), as illustrated below.



[*TP vs. FP rate at different classification thresholds*](https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc)

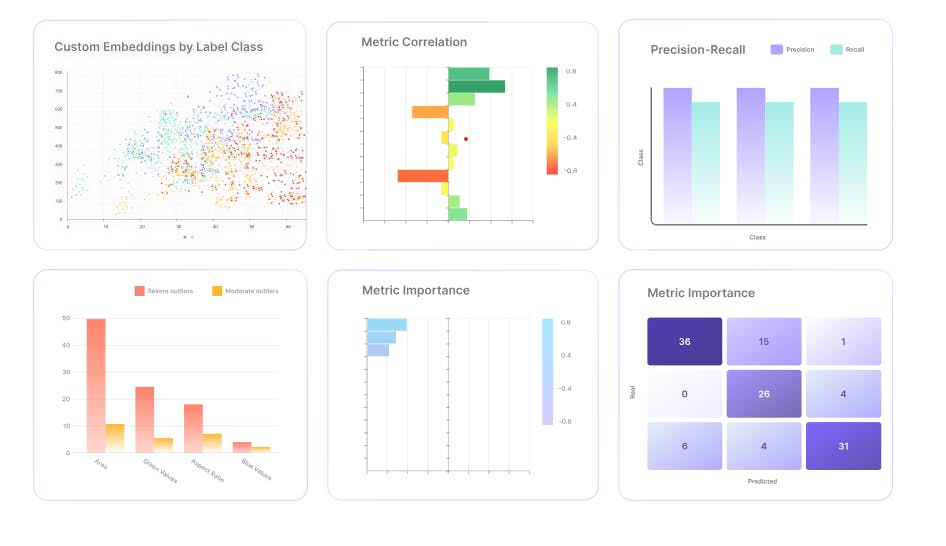
## Future Research for F1 Score

As data science grows, researchers and practitioners continue to investigate the[**challenges posed by imbalanced datasets**](https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0274-4). Modified F1 scores that account for class imbalance are being developed to improve performance evaluation.

Another important area of focus for evaluation measures is[**fairness and ethics**](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8913820/). The goal is to ensure that metrics take into account fairness towards specific subgroups or protected traits in addition to overall performance.

Moreover, another [**research**](https://www.sciencedirect.com/science/article/abs/pii/S0957417422003967) proposes a new discriminant metric to gauge how well AI models perform in maximizing risk-adjusted return for financial tasks after arguing that metrics like the F1 score were unsatisfactory for evaluation.

## Model Evaluation with Encord Active



[**Encord Active**](https://encord.com/active/) is an ML platform that helps practitioners build better models. It offers the following features:

* Evaluation metrics visualization with intuitive charts and graphs
* Auto-identification of[**labeling errors**](https://encord.com/glossary/label-errors-definition/)
* Search and curate high-value visual data using natural language search
* Find and fix bias,[**drift**](https://encord.com/glossary/data-drift-definition/), and dataset errors
* Find model failure modes with automated robustness tests
* Compare your datasets and models based on a detailed metrics evaluation

Hence, it provides practitioners with a variety of [**evaluation approaches and measures**](https://docs.encord.com/docs/active-quality-metrics), including the well-known F1 score, accuracy, precision, and recall. With Encord Active, data scientists can evaluate machine learning mod

# F-score2

## ****Understanding the F-Score and its Significance****

The F-score, also known as F-measure or F1 score, is a powerful tool used to gauge the performance of a Machine Learning model. It amalgamates precision and recall into a solitary score. The F-score algorithm can be defined as: F-score = 2 \* (precision \* recall) / (precision + recall).

## ****Precision and Recall Explained****

The accuracy of positive prediction is revealed by recall, and precision signifies the identification of all positive instances in the dataset. The score spans from 0 to 1 where a higher number represents better performance. The F-measure is frequently employed when it is essential to strike a correct balance between precision and recall and specifically when the positive class occurrence is unusual.

## ****The F-beta and F-2 Score****

Depending on your goals, it is sometimes necessary to place different weights on precision and recall, tweaking the F-score to become a weighted harmonic mean of precision and recall, which is recognized as the F-beta score, where beta is the assigned weight to the recall.

The F-2 score is essentially the F1 score's variant, merging precision and recall into a single score. However, in F-2 score, there's a more pronounced emphasis on recall compared to the conventional F-1 score. The F-2 equation is: F-2 score = (1 + 2^2) \* (precision \* recall) / (2^2 \* precision + recall). Just like the F-score, the F-2 score ranges from 0 to 1, always symbolizing better performance.

## ****Applications and Limitations****

The F-score is applied in various situations, including classification operations, where it is an effective evaluator for classifier performance. It is useful in information retrieval tasks, such as powering search engines, or to enhance a Machine Learning model's performance. Many also find it useful in model comparison, specifically in selecting the best model for a specific application or task.

However, it is crucial to realize that the F-score is only one of many metrics available to assess the performance of a Machine Learning model. Performance can also be evaluated by alternative metrics like accuracy, AUC (Area Under the Curve), and log loss, with the choice of metrics primarily depending on the task specifics and model objectives.

‍

## Practices of Science: Precision vs. Accuracy

*NGSS Science and Engineering Practices*

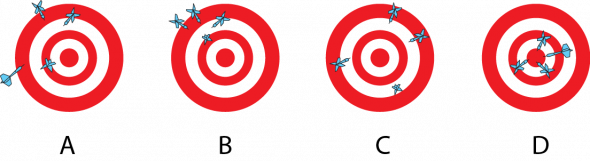
[Planning and Carrying Out Investigations](https://manoa.hawaii.edu/exploringourfluidearth/curriculum-alignment/ngss/sep/planning-and-carrying-out-investigations)

Precision and accuracy are two ways that scientists think about error. **Accuracy** refers to how close a measurement is to the true or accepted value. Precision refers to how close measurements of the same item are to each other. **Precision** is independent of accuracy. That means it is possible to be very precise but not very accurate, and it is also possible to be accurate without being precise. The best quality scientific observations are both accurate and precise.

A classic way of demonstrating the difference between precision and accuracy is with a dartboard. Think of the bulls-eye (center) of a dartboard as the true value. The closer darts land to the bulls-eye, the more accurate they are.

* If the darts are neither close to the bulls-eye, nor close to each other, there is neither accuracy, nor precision (SF Fig. 1.5 A).
* If all of the darts land very close together, but far from the bulls-eye, there is precision, but not accuracy (SF Fig. 1.5 B).
* If the darts are all about an equal distance from and spaced equally around the bulls-eye there is mathematical accuracy because the average of the darts is in the bulls-eye. This represents data that is accurate, but not precise (SF Fig. 1.5 C). However, if you were actually playing darts this would not count as a bulls-eye!
* If the darts land close to the bulls-eye and close together, there is both accuracy and precision (SF Fig. 1.5 D).

**Image**

[[](https://manoa.hawaii.edu/exploringourfluidearth/sites/default/files/M1U1-SF%201-5_Accuracy%20and%20Precision.png)](https://manoa.hawaii.edu/exploringourfluidearth/sites/default/files/M1U1-SF%201-5_Accuracy%20and%20Precision.png" \t "_blank)

**Image caption**

**SF Fig. 1.5.** Dartboards showing different accuracy and precision scenarios.

**Image copyright and source**

Image by Byron Inouye

**Question Set**

1. An oceanographer needs to go out in a boat to collect an important temperature and salinity data logger that is attached to an underwater buoy. How does each of the following situations illustrate the differences between precision and accuracy?
   1. The oceanographer checks the weather forecast the night before her trip so she knows what to wear on the boat. The TV forecaster says it will be between 26 and 31 degrees (°) Celsius (C) at noon the next day. The actual temperature reading the next day on the boat at noon is 28° C.
   2. When the oceanographer’s Global Positioning System (GPS) indicates that she is at the location of the underwater buoy, she anchors the boat and jumps in the water to collect the data logger. However, she can’t see the buoy. The other GPS units belonging to her colleagues on the boat also indicate that they are at the correct location. After an extensive search, the oceanographer finds the buoy 50 meters (m) from the boat.
   3. While on the way back to shore, the oceanographer throws in a fishing line to see if she can catch anything for dinner. She is lucky enough to catch a mahi-mahi. When she pulls it out of the water, her colleagues estimate the weight of the fish. Their estimates are 16.1 kilograms (kg), 16.8 kg, and 15.9 kg. When they weigh the fish upon returning to shore, the actual weight is 18.2 kg.
2. Write your own scenario illustrating the difference between accuracy and precision. Swap your scenario with a classmate. Identify your classmate’s scenario measurements as accurate or inaccurate and precise or imprecise.
3. A dart player can see how accurate his or her dart throws are by comparing the location of the thrown darts to the target, the bulls-eye of the dartboard.
   1. How is this model different from scientists who are measuring a natural phenomenon?
   2. Is there a way for scientists to determine how accurate their measurements are? Explain your answer

# **KFold**

***class*sklearn.model\_selection.KFold(*n\_splits****=5***,***\****, *shuffle****=False***, *random\_state****=None***)**[**[source]**](https://github.com/scikit-learn/scikit-learn/blob/2621573e6/sklearn/model_selection/_split.py#L441)

K-Fold cross-validator.

Provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds (without shuffling by default).

Each fold is then used once as a validation while the k - 1 remaining folds form the training set.

Read more in the [User Guide](https://scikit-learn.org/1.5/modules/cross_validation.html#k-fold).

For visualisation of cross-validation behaviour and comparison between common scikit-learn split methods refer to [Visualizing cross-validation behavior in scikit-learn](https://scikit-learn.org/1.5/auto_examples/model_selection/plot_cv_indices.html#sphx-glr-auto-examples-model-selection-plot-cv-indices-py)

**Parameters:**

**n\_splits*int, default=5***

Number of folds. Must be at least 2.

***Changed in version 0.22:***n\_splits default value changed from 3 to 5.

**shuffle*bool, default=False***

Whether to shuffle the data before splitting into batches. Note that the samples within each split will not be shuffled.

**random\_state*int, RandomState instance or None, default=None***

When shuffle is True, random\_state affects the ordering of the indices, which controls the randomness of each fold. Otherwise, this parameter has no effect. Pass an int for reproducible output across multiple function calls. See [Glossary](https://scikit-learn.org/1.5/glossary.html#term-random_state).

See also

[**StratifiedKFold**](https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold)

Takes class information into account to avoid building folds with imbalanced class distributions (for binary or multiclass classification tasks).

[**GroupKFold**](https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.GroupKFold.html#sklearn.model_selection.GroupKFold)

K-fold iterator variant with non-overlapping groups.

[**RepeatedKFold**](https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.RepeatedKFold.html#sklearn.model_selection.RepeatedKFold)

Repeats K-Fold n times.

**Notes**

The first n\_samples % n\_splits folds have size n\_samples // n\_splits + 1, other folds have size n\_samples // n\_splits, where n\_samples is the number of samples.

Randomized CV splitters may return different results for each call of split. You can make the results identical by setting random\_state to an integer.

**Examples**

>>> **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)**

>>> kf**.**get\_n\_splits**(**X**)**

*2*

>>> print**(**kf**)**

*KFold(n\_splits=2, random\_state=None, shuffle=False)*

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

... print**(**f"Fold {i}:"**)**

... print**(**f" Train: index={train\_index}"**)**

... print**(**f" Test: index={test\_index}"**)**

*Fold 0:*

*Train: index=[2 3]*

*Test: index=[0 1]*

*Fold 1:*

*Train: index=[0 1]*

*Test: index=[2 3]*

**get\_metadata\_routing()**[**[source]**](https://github.com/scikit-learn/scikit-learn/blob/2621573e6/sklearn/utils/_metadata_requests.py#L1495)

Get metadata routing of this object.

Please check [User Guide](https://scikit-learn.org/1.5/metadata_routing.html#metadata-routing) on how the routing mechanism works.

**Returns:**

**routing*MetadataRequest***

A [**MetadataRequest**](https://scikit-learn.org/1.5/modules/generated/sklearn.utils.metadata_routing.MetadataRequest.html#sklearn.utils.metadata_routing.MetadataRequest) encapsulating routing information.

**get\_n\_splits(*X****=None***, *y****=None***, *groups****=None***)**[**[source]**](https://github.com/scikit-learn/scikit-learn/blob/2621573e6/sklearn/model_selection/_split.py#L419)

Returns the number of splitting iterations in the cross-validator.

**Parameters:**

**X*object***

Always ignored, exists for compatibility.

**y*object***

Always ignored, exists for compatibility.

**groups*object***

Always ignored, exists for compatibility.

**Returns:**

**n\_splits*int***

Returns the number of splitting iterations in the cross-validator.

**split(*X*, *y****=None***, *groups****=None***)**[**[source]**](https://github.com/scikit-learn/scikit-learn/blob/2621573e6/sklearn/model_selection/_split.py#L67)

Generate indices to split data into training and test set.

**Parameters:**

**X*array-like of shape (n\_samples, n\_features)***

Training data, where n\_samples is the number of samples and n\_features is the number of features.

**y*array-like of shape (n\_samples,)***

The target variable for supervised learning problems.

**groups*object***

Always ignored, exists for compatibility.

**Yields:**

**train*ndarray***

The training set indices for that split.

**test*ndarray***

The testing set indices for that split.