Hristina Hristova

Machine Learning with K-Nearest Neighbors

Course Notes

365 DataScience

Table of Contents

Abstract

K-nearest neighbors is a supervised classification and regression machine learning algorithm. It is famous for being one of the most intuitive and easy-to-implement machine learning algorithms out there. Despite its simplicity, it could outperform other, more sophisticated methods. It has been and still is, a subject of research in terms of optimization and applications.

The following notes serve as a complement to the "Machine Learning with K-Nearest Neighbors" course. They list the algorithm's pros and cons, outline the working of the KNN classification and regression algorithms, cover in greater detail the more involved topic of distance metrics, and summarize the most commonly used performance metrics.

Keywords: machine learning algorithm, KNN, classification, regression, Minkowski, Manhattan, Euclidean, distance, confusion matrix, accuracy, precision, recall, F_1 score

1. Motivation

In this section, we summarize the advantages and disadvantages of the K-nearest neighbors (KNNs) algorithm (Table 1) together with a couple of the most common use cases.

Table 1: Pros and cons of KNN classifiers.

Pros	Cons
Intuitive	Not a preferable choice for extrapolation tasks
Easy to implement	Needs more data to make good predictions, compared to parametric models
The fitting process is very time-efficient	The fitting process could take up too much memory
Non-parametric, therefore easily adjustable to new data	Testing could be slow for big datasets
Hyperparameter tuning is straightforward	Could suffer from the "curse of dimensionality"
	Not a preferable choice for datasets with categorical features
	Sensitive to irrelevant features

2. Math Prerequisites: Distance Metrics

One of the parameters that can be changed in **sklearn**'s KNN implementation is the distance metric. Starting with two dimensions, we first introduce the most general form of the distance metric, the Minkowski distance, followed by two special cases – the Manhattan and the Euclidean distances. Next, we extend these metrics to three dimensions and conclude by generalizing to N number of dimensions.

2.1 The Minkowski Distance

Consider point A having coordinates (x_1, y_1) and point B having coordinates (x_2, y_2), as shown in Figure 1.

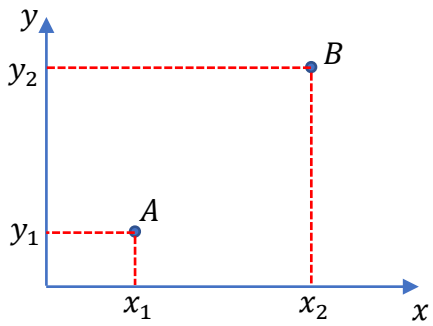


Figure 1: Points A and B positioned on a coordinate system.

In its general form, the 2-dimensional Minkowski metric looks as follows:

$$d = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$$

Let's discuss each element that enters the expression.

- Absolute value (modulus) of a number

The *modulus* is an operation that, in simple terms, removes the sign of a number.

Example:

$$|3| = 3$$

$$|-3| = 3$$

The modulus can be very useful when calculating distances. Consider the case of calculating the distance between the points x_1 and x_2, with $x_1 \neq x_2$. Imagine we do not know their numerical values and, therefore, do not know which one is greater, x_1 or x_2. The distance can then be calculated in two ways:

$$d = x_1 - x_2$$

or

$$d = x_2 - x_1$$

One of these calculations will inevitably give a negative result. However, a negative distance is unphysical. To ensure that the result will be positive, we introduce the absolute value of a number.

Example:

Consider $x_1 = 3$ and $x_2 = 1$. Then

$$d = |x_1 - x_2| = |3 - 1| = |2| = 2$$

or alternatively

$$d = |x_2 - x_1| = |1 - 3| = |-2| = 2$$

- Power: a^p

Raising a number, a, to a certain power, $p \neq 0$, is equivalent to multiplying the number p many times. When raising a number, $a \neq 0$, to the power of $p = 0$, however, the result is always 1.

Example:

$$2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 2 \times 2 = 4$$

$$2^3 = 2 \times 2 \times 2 = 8$$

$$2^4 = 2 \times 2 \times 2 \times 2 = 16$$

- Power: $b^{1/p} = \sqrt[p]{b}$

Raising a number, a, to the power of $p \neq 0$ has its inverse operation – taking the p^{th} root:

$$a^p = b \quad \Rightarrow \quad \sqrt[p]{b} = a$$

Taking the p^{th} root returns the number, a, that, when multiplied p number of times, gives b. Taking the p^{th} root can be also written down as raising a number to the power of 1 over p:

$$a^p = b \quad \Rightarrow \quad b^{1/p} = a$$

Let's discuss the terminology in short. A number, a, raised to the power of 2 can be recovered by taking the *square root*:

$$a^2 = b \quad \Rightarrow \quad \sqrt{b} = b^{1/2} = a$$

The square root is the most commonly used one, so the 2 above the root sign is omitted. Analogously, a number raised to the power of 3 can be recovered by taking the *cube root*:

$$a^3 = b \quad \Rightarrow \quad \sqrt[3]{b} = b^{1/3} = a$$

Continuing in the same fashion, a number raised to the powers of 4, 5, etc., can be recovered by taking the fourth root, the fifth root, etc.

$$a^4 = b \quad \Rightarrow \quad \sqrt[4]{b} = b^{1/4} = a$$

$$a^5 = b \quad \Rightarrow \quad \sqrt[5]{b} = b^{1/5} = a$$

Example:

$$2^2 = 4 \quad \Rightarrow \quad \sqrt{4} = 4^{1/2} = \frac{4}{2} = 2$$

$$2^3 = 8 \quad \Rightarrow \quad \sqrt[3]{8} = 8^{1/3} = \frac{8}{2 \times 2} = 2$$

$$2^4 = 16 \quad \Rightarrow \quad \sqrt[4]{16} = 16^{1/4} = \frac{16}{2 \times 2 \times 2} = 2$$

Note that

$$\sqrt{2^2} = 2$$

$$\sqrt[3]{2^3} = 2$$

$$\sqrt[4]{2^4} = 2$$

Example:

Consider the Minkowski distance:

$$d = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$$

Consider also $x_1 = 5$, $x_2 = 2$, $y_1 = 4$, $y_2 = 8$, *and* $p = 2$. *The distance* d *is then calculated as follows:*

$$d = (|5 - 2|^2 + |4 - 8|^2)^{1/2}$$

$$d = (|3|^2 + |-4|^2)^{1/2}$$

$$d = (3^2 + 4^2)^{1/2}$$

$$d = (9 + 16)^{1/2}$$

$$d = 25^{1/2}$$

$$d = \sqrt{25}$$

$$d = \sqrt{5^2}$$

$$d = 5$$

2.2 The Manhattan Distance

Now, we consider a special case of the Minkowski distance – the so-called *Manhattan*, or *taxicab*, distance. It is obtained by substituting the parameter p in the expression with 1:

$$d = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$$

$$d = (|x_1 - x_2|^1 + |y_1 - y_2|^1)^{1/1}$$

$$\boxed{d = |x_1 - x_2| + |y_1 - y_2|}$$

Looking closely at this expression, we realize that this is the sum of the distances along the x-direction and the y-direction. Graphically, it is depicted in Figure 2.

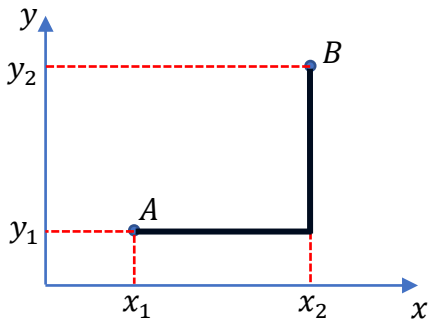


Figure 2: The blue solid line represents the Manhattan distance in two dimensions.

The solid line connecting points A and B represents the Manhattan distance.

2.3 The Euclidean Distance

Now, we consider a second special case of the Minkowski distance, namely, the *Euclidean* one. It is obtained by substituting the parameter p in the expression with 2:

$$d = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$$

$$d = (|x_1 - x_2|^2 + |y_1 - y_2|^2)^{1/2}$$

$$\boxed{d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}$$

This expression is the famous Pythagorean theorem which calculates the length of the hypothenuse of a right triangle having one cathetus equal to $|x_1 - x_2|$ and the other one having a length $|y_1 - y_2|$. The Euclidean distance between two points in two dimensions is represented in Figure 3.

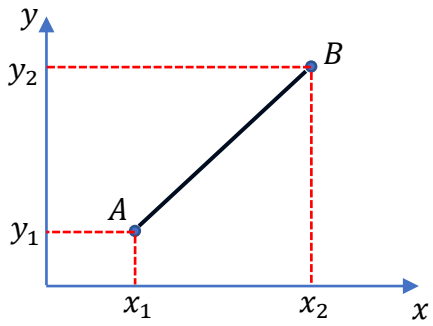


Figure 3: The blue solid line represents the Euclidean distance in two dimensions.

2.4 The Distance Metrics in Three Dimensions

So far, we have been working only with the x- and y-axes which create a coordinate system for the 2-dimensional space. However, we do not live in Flatland[1] but rather live in a 3-dimensional world. It is therefore worth expanding

[1] Referring to Edwin A. Abbott's novel "Flatland".

the distances introduced earlier to three dimensions. This is done rather straightforwardly by considering an additional axis, call it z. The coordinates of points A and B then become (x_1, y_1, z_1) and (x_2, y_2, z_2). Let's write down the distance using each of the three distance metrics, taking into consideration the third coordinate.

- Minkowski distance

$$d = (|x_1 - x_2|^p + |y_1 - y_2|^p + |z_1 - z_2|^p)^{1/p}$$

- Manhattan distance

$$d = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$$

- Euclidean distance

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

2.5 Generalizing to N Dimensions

Despite being impossible to geometrically visualize more than three dimensions, it is indeed possible to extend the algebra accordingly. In the examples above, x is our first dimension, y is the second, and z is the third. This notation, however, is not particularly convenient when aiming to extend the theory to N number of dimensions – we would run out of letters if we go to high enough dimensions. Moreover, the notation would become rather cumbersome.

The way to solve this problem is to label the dimensions $1, 2, 3, \ldots N$, rather than x, y, z, etc. With this new definition, let's initialize two points in the N-dimensional space, call them X and Y. Let point X have coordinates

$(x_1, x_2, x_3, \dots, x_N)$. Point Y would, in turn, have coordinates $(y_1, y_2, y_3, \dots, y_N)$. Let's write down the expressions for the three distance metrics considered above. For each one, we provide two forms, the second one being a short-hand version of the first.

- Minkowski distance

$$d = (|x_1 - y_1|^p + |x_2 - y_2|^p + \cdots + |x_N - y_N|^p)^{1/p}$$

$$d = \left(\sum_{i=1}^{N} |x_i - y_i|^p\right)^{1/p}$$

- Manhattan distance

$$d = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_N - y_N|$$

$$d = \sum_{i=1}^{N} |x_i - y_i|$$

- Euclidean distance

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_N - y_N)^2}$$

$$d = \left[\sum_{i=1}^{N} (x_i - y_i)^2\right]^{1/2}$$

3. How Does a KNN Classification Work?

KNN is an algorithm that classifies a sample based on the classes of the samples that come closest to it.

Consider the following 2-class classification problem (Figure 4). The aim is to determine the class of the sample represented by a cross.

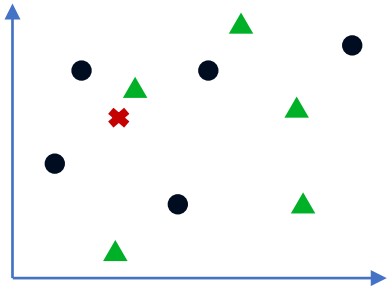


Figure 4: A two-class classification problem. We aim to classify the sampe denoted by a cross.

The way KNN approaches this problem is by first choosing K – the number of nearest neighbors. The value of K is determined by the programmer. For this simple example, let's consider two values – 1 and 3. For K = 1 (Figure 5), we search for the first nearest neighbor and consider its class. In this case, it is a triangle, so the sample gets assigned to the triangles. For K = 3, on the other hand, (Figure 6) one of the neighbors is a triangle, while the other two are circles. Since the number of representatives from the circle class prevails, the sample is classified as a circle.

This example shows that different values of K result in different outcomes. The way to determine the best value of K is by performing cross-validation and minimizing the error.

Notice that, had we chosen K = 2, we would have had a tie – one representative from the circles class and one from the triangles class. The way this use case is handled depends on the implementation of the algorithm.

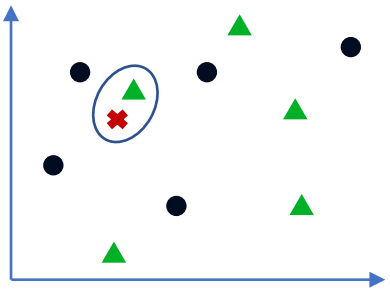


Figure 5: KNN classification with K = 1. The sample is assigned to the triangles class.

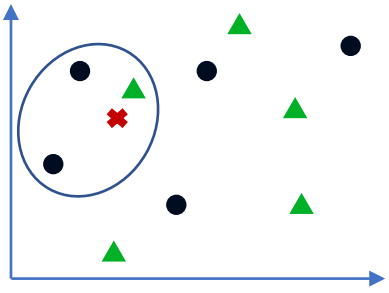


Figure 6: KNN classification with K = 3. The sample is assigned to the circles class.

4. How Does a KNN Regression Work?

The idea behind the KNN regression algorithm is very similar to the one of the KNN classifier.

Consider the following simple regression problem (Figure 7). It represents four points that have a single feature, x, and a single output, y. The sample that we need to predict the y-value of is denoted by a cross and is given a certain value of x.

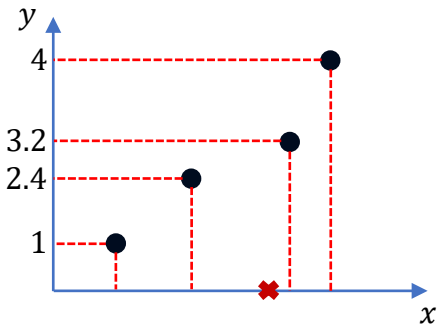


Figure 7: KNN regression problem. Given an x-value, calculate the y-value of the sample represented by a cross.

Just as before, we need to choose a value of K. Let's consider three scenarios: K = 1, 2, and 3. Choosing K = 1 (Figure 8) searches for the point whose x-component comes closest to the x-component of the cross. we see that this is the third point from left to right. Next, the y-value of the cross is calculated by taking the y-values of all neighbors and finding their arithmetic mean. In this case, the neighbor is only one, so the sample adopts the y-value of its nearest neighbor.

Next, choosing K = 2 (Figure) searches for the two points whose x-components come closest to the x-value of the cross. These are the two circles to the right of the cross. Now, take their y-values and find their arithmetic mean. The result is $y = \frac{3.2+4}{2} = 3.6$. So this is the predicted y-value for the cross.

Lastly, let's calculate the predicted y-value when K = 3 (Figure 10). The three nearest neighbors are the two to the right and the one to the left of the cross. The arithmetic mean of their y-values is again $y = \frac{3.2+4+2.4}{3} = 3.2$.

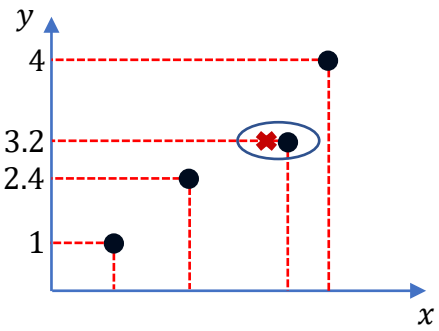


Figure 8: KNN regression with K = 1. The sample is predicted to have a y-value of 3.2.

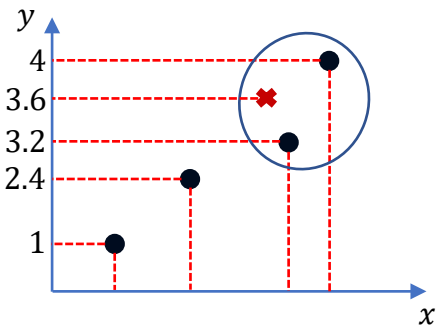


Figure 9: KNN regression with K = 2. The sample is predicted to have a y-value of 3.6.

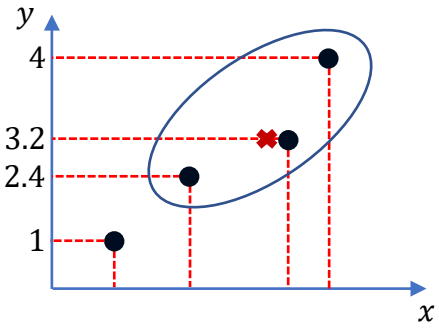


Figure 10: KNN regression with K = 3. The sample is predicted to have a y-value of 3.2.

5. Important Steps to Creating a Model

In this section, we outline the most important steps that need to be executed when creating a machine learning model. It is important that these steps are executed in the order given below.

1. Create the DataFrame

First and foremost, we need to create a **pandas** DataFrame where all inputs and targets are organized. Of course, a **pandas** DataFrame is not the only way to store a database, but it proves to be very useful. You are welcome to experiment with other means, but keep in mind that the **train_test_split()** method accepts the inputs and targets in the form of lists, **NumPy** arrays, **SciPy**-sparse matrices, as well as **pandas** DataFrames.

2. Data cleansing – check for null values

This step is part of the data pre-processing procedure. Before moving forward to the next step, do check for any null values in the data. There are various techniques to deal with this issue. One would be to remove the samples containing missing values altogether. This, however, can be done only if the number of such samples is much smaller than the number of all samples in the dataset. As a rule of thumb, if the number of samples containing null values is no more than 5% of the total number of samples, then removing them from the database should be safe. If that is not the case, statistical methods for filling up the missing values can be used instead.

3. Data cleansing – identify outliers

This step is part of the data pre-processing procedure. One should identify and remove any outliers in the data. The presence of samples with obscure values could cause misclassification of samples that would otherwise be classified correctly.

4. Split the data

Next, split the data into training and testing sets using, for example, **sklearn**'s **train_test_split()** method. An 80:20 split is very common. It would dedicate 80% of the data to the training set and 20% to the test set. Other splits such as 90:10, or 70:30 could, of course, be used as well. Use the training data to fit the model and the test data to evaluate its performance.

This step of splitting the data is one of the most common ways to avoid overfitting. Overfitting is a phenomenon where a model learns the data so well, that it also captures random noise in the data which affects its predictions. This is undesirable as random noise would inevitably be present in completely new datasets as well. An overtrained model would therefore perform poorly by misclassifying many of the points.

5. Data wrangling

This step is part of the data pre-processing procedure. In this step, we prepare the data for the classifier. Classifiers such as KNN, are based on distances between the samples. Such algorithms require standardized inputs, which usually imply transforming the data. Others, such as the Multinomial Naïve Bayes classifier, which is mainly used for text analysis, require vocabulary dictionaries in

the form of sparse matrices. This would require transforming the inputs accordingly.

Such transformations carry information about the data. In the case of standardization, for example, the knowledge of the mean and standard deviation is gained. It is, therefore, dangerous to perform such transformations on the whole dataset, that is, before train-test splitting. Doing so could lead to *data leakage*.

6. Perform the classification

In this step, the appropriate classifier for the task is chosen, it is fit to the training data, and hyperparameters are tuned to achieve maximum performance.

7. Evaluate the performance of the model

Once the model is created and finetuned, it is time to test it on a new dataset. Metrics such as accuracy, precision, recall, and F_1 score are studied in the next section.

6. Relevant Metrics

In this section, we introduce some of the relevant metrics that could be used to evaluate the performance of a machine learning model dealing with a classification task.

6.1 The Confusion Matrix

> *A confusion matrix, C, is constructed such that each entry, C_{ij}, equals the number of observations known to be in group i and predicted to be in group j.*

A confusion matrix is a square 2 × 2, or larger, matrix showing the number of (in)correctly predicted samples from each class.

Consider a classification problem where each sample in a dataset belongs to only one of two classes. We denote these two classes by 0 and 1 and, for the time being, define 1 to be the *positive* class. This would result in the confusion matrix from Figure 11 .

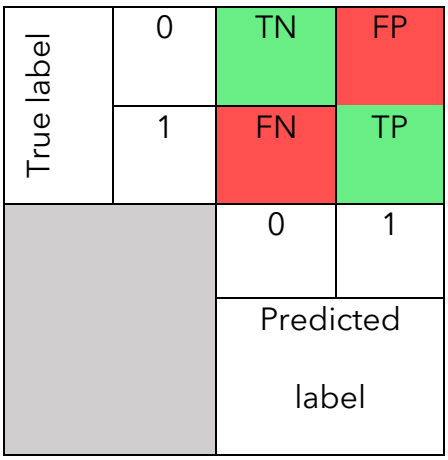


Figure 11: A 2 × 2 confusion matrix denoting the cells representing the true and false positives and negatives. Here, class 1 is defined as the positive one.

The matrix consists of the following cells:

- Top-left cell – **true negatives** (TN). This is the number of samples whose *true* class is 0 and the model has correctly classified them as such.
- Top-right cell – **false positives** (FP). This is the number of samples whose *true* class is 0 but have been incorrectly classified as 1s.
- Bottom-left cell – **false negatives** (FN). This is the number of samples whose *true* class is 1 but have been incorrectly classified as 0s.
- Bottom-right cell – **true positives** (TP). This is the number of samples whose *true* class is 1 and the model has correctly classified them as such.

Consider now a classification problem where each sample in a dataset belongs to one of three classes, 0, 1, or 2, with class 1 again defined as the *positive* class. This makes classes 0 and 2 *negative*. The confusion matrix would then look like the one in Figure 12.

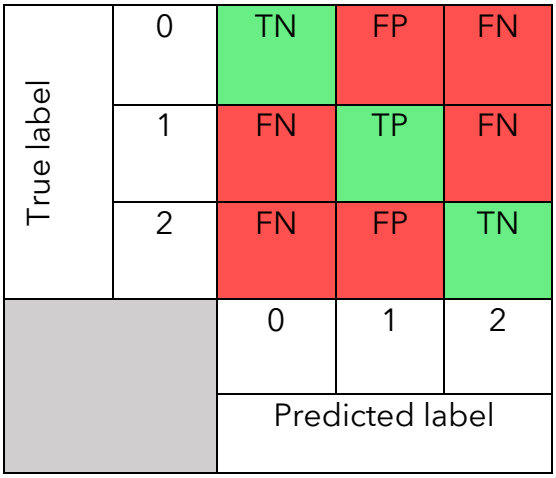


Figure 12: A 3 × 3 confusion matrix denoting the cells representing the true and false positives and negatives. Here, class 1 is defined as the positive one.

Making use of these confusion matrices, we introduce four useful metrics for evaluating the performance of a classifier.

6.2 Accuracy

The ratio between the number of all correctly predicted samples and the number of all samples.

$$\text{Accuracy} = \frac{TN + TP}{TN + FN + FP + TP}$$

6.3 Precision

The ratio between the number of true positives and the number of all samples classified as positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

6.4 Recall

The ratio between the number of true positives and the number of all samples whose true class is the positive one.

$$\text{Recall} = \frac{TP}{TP + FN}$$

6.5 F1 Score

The harmonic mean of precision and recall.

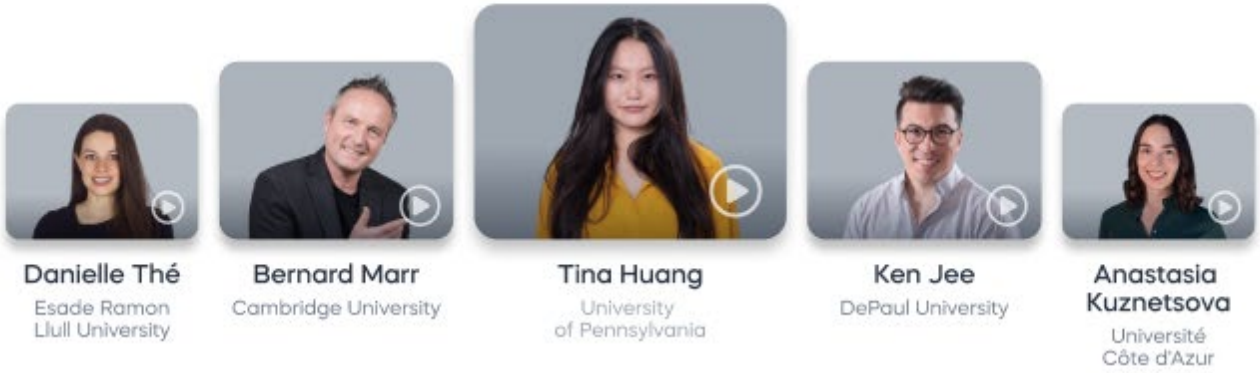
$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

The F_1 score can be thought of as putting precision and recall into a single metric. Contrary to taking the simple arithmetic mean of precision and recall, the F_1 score penalizes low values more heavily. That is to say, if either precision or recall is very low, while the other is high, the F_1 score would be significantly lower compared to the ordinary arithmetic mean.

Learn **DATA SCIENCE** anytime, anywhere, at your own pace.

If you found this resource useful, check out our **e-learning program**. We have everything you need to succeed in data science.

Learn the most sought-after data science skills from **the best experts in the field**! Earn a **verifiable certificate** of achievement trusted by employers worldwide and future proof your career.



Danielle Thé
Esade Ramon Llull University

Bernard Marr
Cambridge University

Tina Huang
University of Pennsylvania

Ken Jee
DePaul University

Anastasia Kuznetsova
Université Côte d'Azur

Comprehensive training, exams, certificates.

- ✓ 162 hours of video
- ✓ 599+ Exercises
- ✓ Downloadables
- ✓ Exams & Certification
- ✓ Personalized support
- ✓ Resume Builder & Feedback
- ✓ Portfolio advice
- ✓ New content
- ✓ Career tracks

Join a global community of 1.8 M successful students with an annual subscription at 60% OFF with coupon code **365RESOURCES**.

~~$432~~ **$172.80**/year

Start at 60% Off

VAT may be applied

365√DataScience

Hristina Hristova

Email: team@365datascience.com

365 DataScience