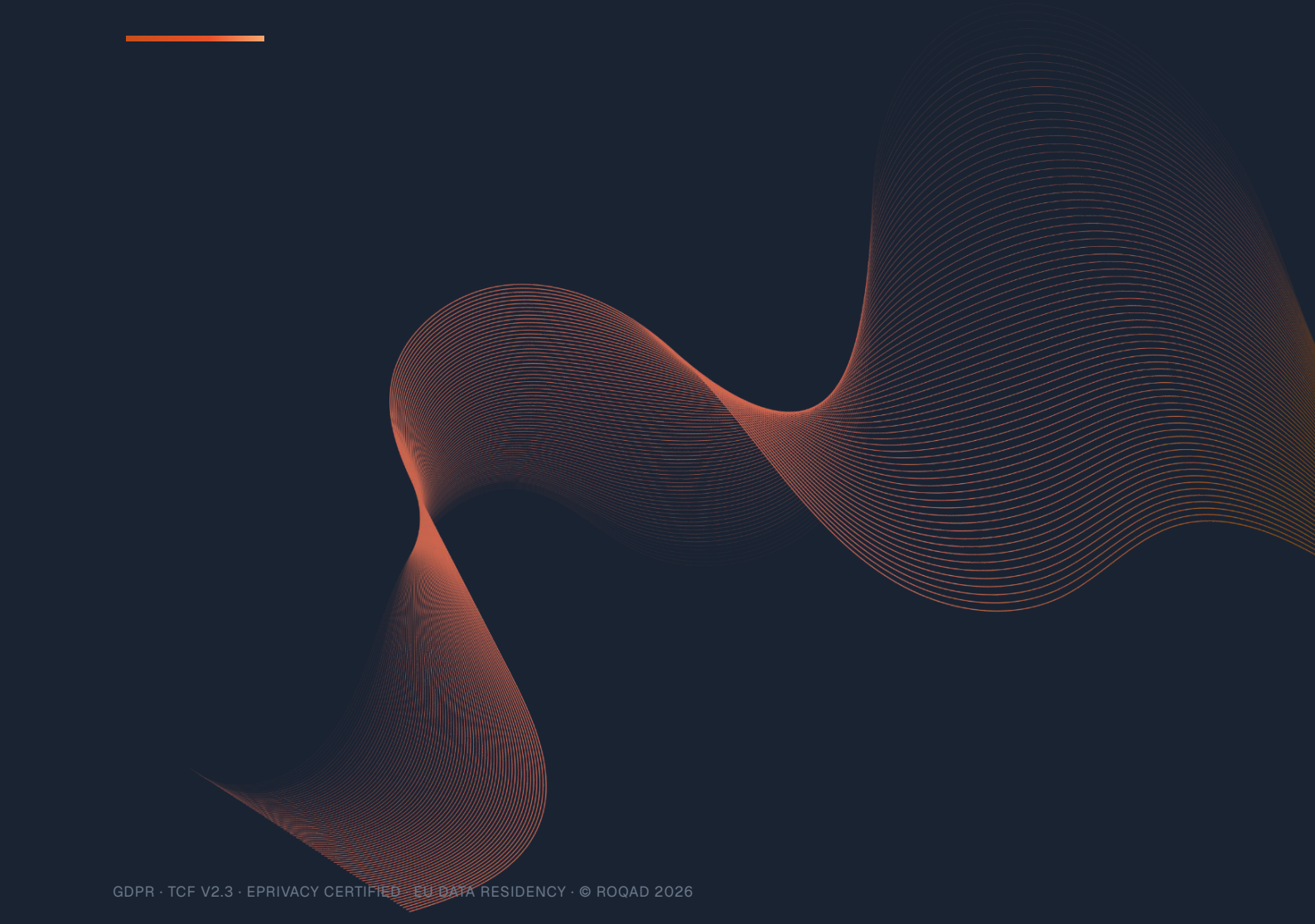
ROQAD

WHITE PAPER · MACHINE LEARNING

Machine Learning

Actionable insights into today's cross-device reality



GDPR · TCF V2.3 · EPRIVACY CERTIFIED · EU DATA RESIDENCY · © ROQAD 2026

FROM BIG DATA TO SMART DATA

01 Big data in a nutshell

In today's high-tech society, an enormous amount of data is being collected around the clock. That has paved the way for big data to be transformed into real-life, actionable knowledge, and that knowledge improves decision making.

For this to happen, the right smart tools need to be employed. The key differentiator between a tool and a smart tool is that the latter always requires a spark: a smart idea on how to use collected data in a new and innovative way. In a nutshell, this is how big data becomes smart data.

Cross-device advertising is a good example, and it is an area where Roqad excels at turning big data into smart data. The solution consists of identifying devices that belong to the same user. To understand it, imagine a huge graph whose nodes correspond to devices. Devices are recognised by different types of identifiers, such as cookies (for desktop and mobile web) and mobile device IDs (IDFA, Android Advertising ID). The goal is to uncover the edges in the graph that link devices belonging to the same user. With a clique of such devices, you can identify an anonymous user. This graph of anonymous users and their devices is better known as an **identity graph**.

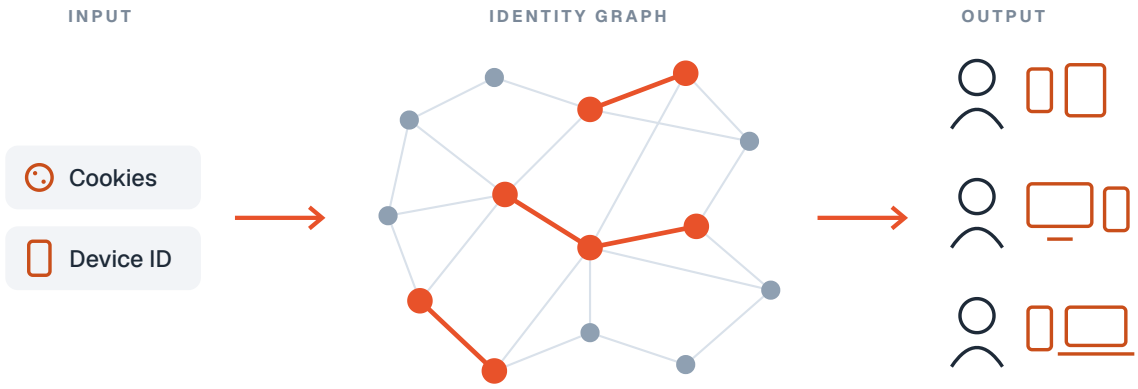


Figure 1. Device IDs can be cookies (for desktop and mobile web), IDFA (Identifier for Advertisers) or AAID (Android Advertising ID). IDFA and AAID identify mobile devices. The identity graph links them to anonymous users.

To find connections, device information collected from web activity needs to be applied. This can be difficult, because sometimes you only receive tiny traces of device activity, which you need to follow to do your job as a digital detective. Think of it as peeking through a keyhole to understand what's happening in a room. These traces are generally HTTP requests received from partners, and they may contain the following information:



Device ID



Time stamp



Hashed IP



Geo location



Operating system



Visited URL

Any sensitive personal information is anonymised, so there is no possibility of directly identifying devices or users.

Unfortunately, the collected requests can be very noisy. They can be generated by bots, be biased towards specific services (and therefore users), and device IDs may not really be actual device IDs. On that last point, it's essential to be aware that a single device can be seen through many cookies, and each cookie is valid for a short period (about 30 days on average). This can create quite a headache when analysing cross-device data.

MATCHING DEVICES AT SCALE

02 Machine learning

Once the database of HTTP requests is created, we look for similarities in activity between devices. If similar behaviour is observed, we can decide whether a given pair of devices belongs to the same user.

The problem of matching devices can be solved in the following way. Given two devices, A and B, one can manually develop a formula for computing the probability that A and B are connected (that is, belong to the same user) using information from the HTTP requests such as visited URLs, IPs used, and the name and properties of a browser. This formula can be guessed, or based on intuition or deeper expert knowledge. By working on it, you'll probably end up with something that you hope gives good estimates of the probability that A and B are connected.

However, there is a more effective way of deriving this formula. Instead of explicitly coding the solution for cross-device matching, it can be done implicitly using machine learning. For this approach, examples of connected and disconnected devices need to be collected. This information comes from **deterministic data**: HTTP requests with additional information that indirectly identifies users, for example a hashed email address. By feeding a learning algorithm with these examples, the formula is written implicitly, and a machine solves the difficult problem of cross-device matching for you.

To put this into perspective, let's look briefly at the Roqad supervised algorithm. It works in two steps:

01

Candidate selection

Selects candidate connections: pairs of devices that are likely to belong to the same user.

→

02

Probabilistic scoring

Assigns each candidate a precise probability of being a true connection, using a classifier trained on part of the deterministic data.

The rest of the deterministic data is used to test the algorithm. The data are split into training and test sets by user identifier, so all requests for a given user are either in the training set or in the test set. This split ensures we obtain proper estimates of the performance of the supervised algorithm.

The two-step approach also lets us control the scalability of the algorithm. Here's an example. Let's assume we have 10 million devices. For this number of devices, we have:

$$\sim 50 \times 10^{12} \text{ possible connections } \left(\text{since } \frac{10^7 \times (10^7 - 1)}{2}\right)$$

No algorithmic solution can deal with such a number of entities. By using smart candidate selection, we end up with only a tiny fraction of the possible connections, and for those we can call the probabilistic classifier.

This is not the final solution, however. So far we have probability estimates for candidate connections, which produce a graph of devices connected by weighted edges. The next step is to translate this into a graph of users and their devices. For this, we use domain knowledge about the distribution of devices per user. The result is a clean structure in which users can be recognised (indirectly) as sets of connected devices.

MEASURING QUALITY

03 Testing cross-device matching solutions

Is the formula, and the resulting graph, actually high quality? That leads to a more general question: how do you test and assess cross-device products?

There are two ways of testing a cross-device matching solution: unsupervised and supervised. The former can always be used, even without deterministic data, while the latter requires it.

In a typical application, an identity graph is queried using a set of device IDs. These device IDs may be relatively well known to you, for example if they come from a client. Three metrics describe the quality of the result:

METRIC 1

Overlap

How many of the queried devices are in the identity graph, as a fraction of all devices in the query.

METRIC 2

Deduplication

Which queried device IDs belong to the same users, as a fraction of deduplicated devices to all devices in the query.

METRIC 3

Enrichment

Whether new, unknown device IDs are connected to the known ones, as a fraction of enriched devices to all devices in the query.

Once you're satisfied with the overlap, you can check whether the identity graph is able to deduplicate the device IDs. It is also possible to ask whether other device IDs exist that are connected to the known ones. This is device enrichment.

To clarify these definitions, let's consider a simplified example. Assume you query an identity graph with six device IDs: a, b, c, d, e and f, and that the identity graph contains five cliques of devices: (a, b, m), (c, d), (e), (g, h, i) and (j, k, l).

5/6

OVERLAP

a, b, c, d and e are in the graph

4/6

DEDUPLICATION

(a, b) and (c, d) are connected

2/6

ENRICHMENT

a and b connect to unknown device m

In this case the overlap is 5/6, because five of the device IDs are also contained in the identity graph. The deduplication rate is 4/6, as the connections (a, b) and (c, d) have been found, which together contain four devices from the query. Enrichment is 2/6, since only two devices are connected to an unknown device: a and b are connected to m.

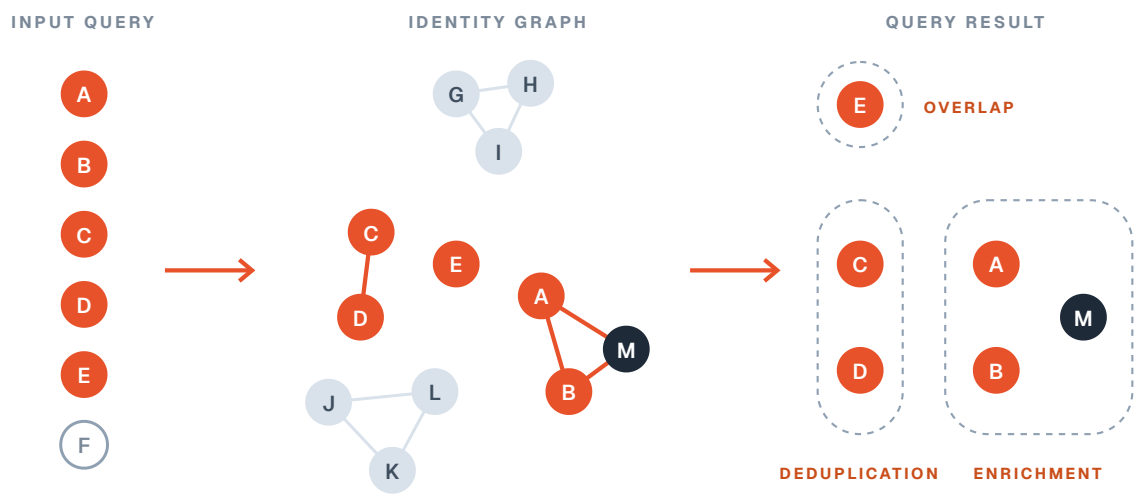


Figure 2. Querying an identity graph with six devices. Orange devices appear in the query, the dark device (M) is new to the client.

These metrics say nothing about the predictive performance of the identity graph, however. For that, a separate test set of deterministic data is needed, against which precision and recall can be checked.

Precision

The fraction of predicted matches between devices that are actual, true matches.

Recall

The fraction of all actual matches that are correctly identified.

There is a natural trade-off between precision and recall, which can be managed by setting a cut-off threshold on prediction scores, removing connections with a probability below the threshold. This lets you tune an identity graph for different business strategies.

LOW THRESHOLD

High recall, lower precision

Best suited to **audience amplification**.

HIGH THRESHOLD

High precision, lower recall

Best suited to **attribution analysis**.

The harmonic mean of precision and recall, often referred to as the **F-score**, is a good measure of this trade-off. The figure below shows precision, recall and F1 score as a function of the prediction cut-off threshold.

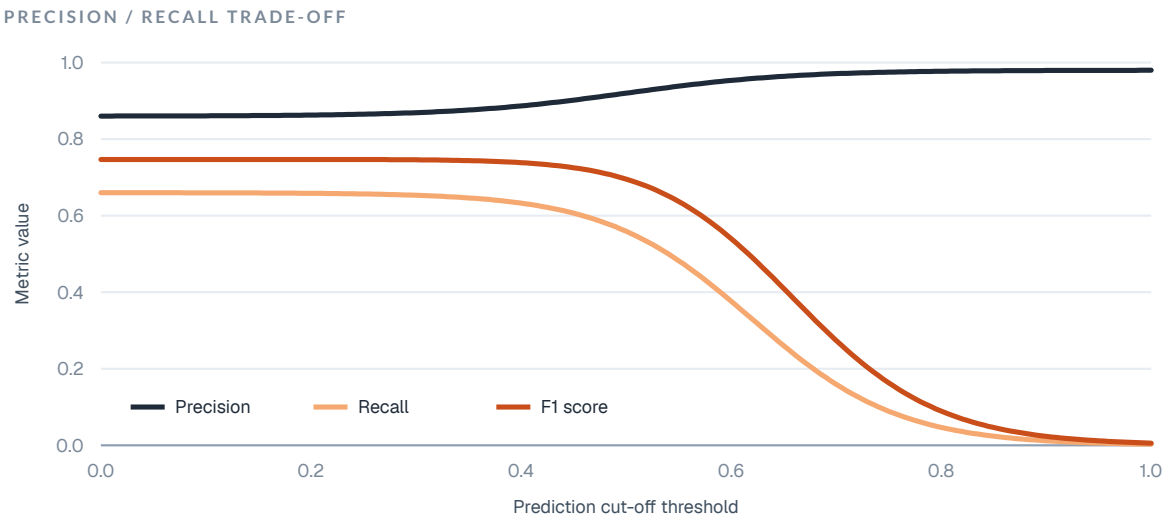


Figure 3. Precision, recall and F1 score against the prediction cut-off threshold.

04 · FINAL WORDS

There's more to come.

There are many other places in cross-device technology where machine learning can be applied. This paper is part of a new series, so stay tuned for more on how Roqad uses the latest advances in artificial intelligence to build and enhance its cross-device products.

Talk to us →