Unlocking Data Insights - Introduction to Data-Centric AI

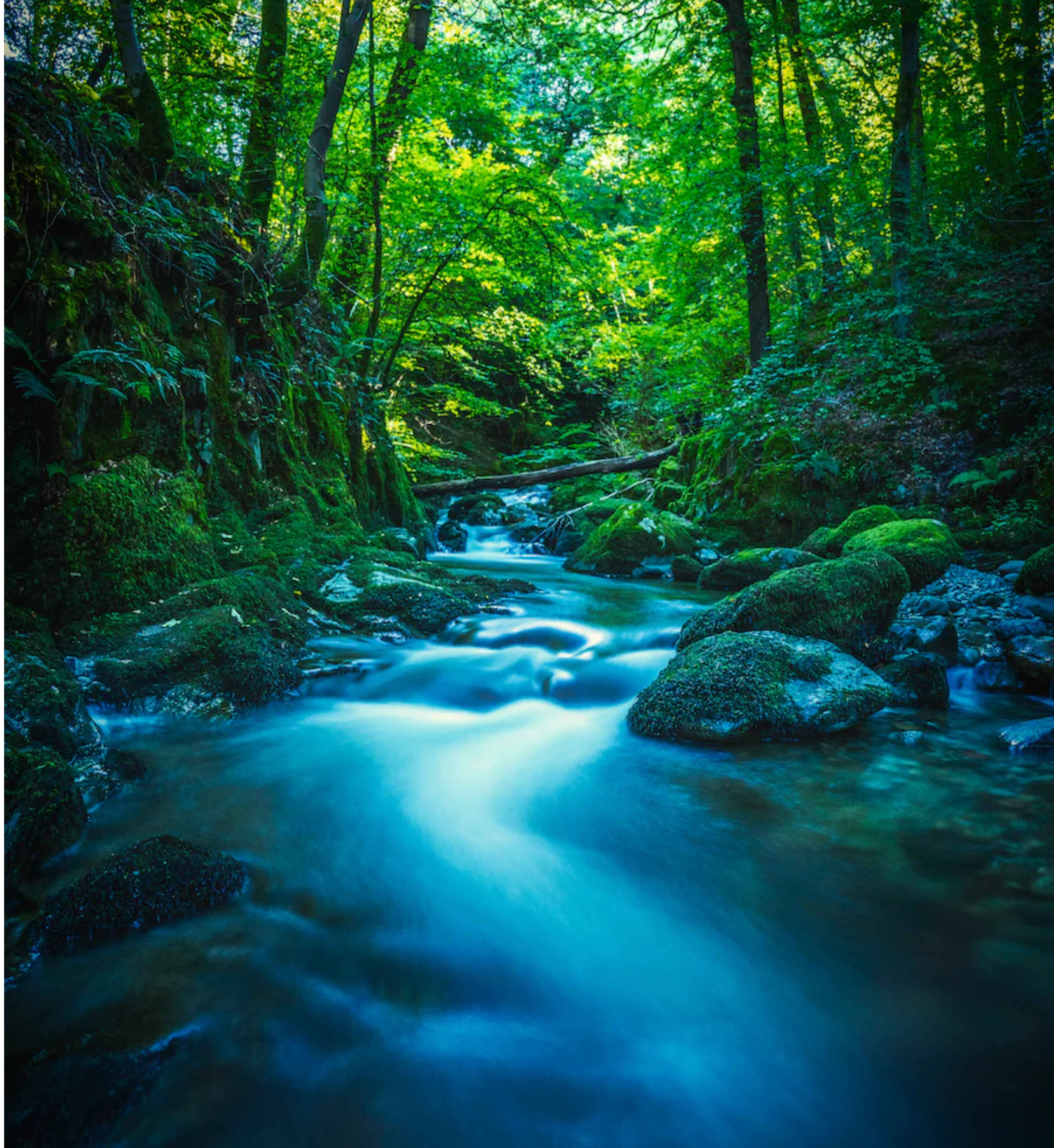
Learning from data streams: A gentle introduction

Reference: *Machine Learning for Data Streams: with Practical Examples in MOA, Albert Bifet, Ricard Gavaldà, Geoff Holmes, Bernhard Pfahringer, The MIT Press*

Learning from data streams: A gentle introduction

Executive Summary

- Big Data
- Tools: Open-Source Revolution
- Challenges in Big Data
- Real-Time Analytics



Credits: Jonny Gios - Unsplash

Big Data

Gartner* summarizes this in his definition of big data in 2012 as “high volume, velocity and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.”

Daryl C. Plummer, Kurt Potter, Richard T. Matlus, Jacqueline Heng, Rolf Jester, Ed Thompson, Adam Sarner, Esteban Kolsky, French Caldwell, John Bace, Neil MacDonald, Brian Gammage, Michael A. Silver, Leslie Fiering, Monica Basso, Ken Dulaney, David Mitchell Smith, Bob Hafner, Mark Fabbi, and Michael A. Bell. Gartner’s top predictions for it orga- nizations and users, 2007 and beyond.

Big Data

Business

Customer personalization and churn detection (customers moving from one company to a rival one)

Health

people's medical records and genomics data, to monitor and improve their health

Applications of big data should allow people to have better services and better customer experiences, and also be healthier

Technology

Reducing processing time from hours to seconds

Smart Cities

Cities focused on sustainable economic development and high quality of life, with wise management of natural resources.

Big Data: real example

UN GLOBAL PULSE

Global Pulse* uses big data to improve life in developing countries

1. Researching innovative methods and techniques for analyzing real-time digital data to detect early emerging vulnerabilities

2. Assembling a free and open-source technology toolkit for analyzing real-time data and sharing hypotheses

3. Establishing an integrated, global network of Pulse Labs, to pilot the approach at the country level

*United Nations Global Pulse. Harnessing big data for development and humanitarian action. http://www.unglobalpulse.org, accessed May 21st, 2017.

Big Data: real examples

Shell

Real-time machine learning pipeline able to detect whether people are smoking

Health

Real-time machine learning pipeline able to detect heart rate changes

Social network

Real-time machine learning pipeline able to detect fake news or bad content

Big Data: Open source tools revolution

hadoop

Spark

APACHE STORM™

Flink

TensorFlow

Big Data: Challenges in Big Data

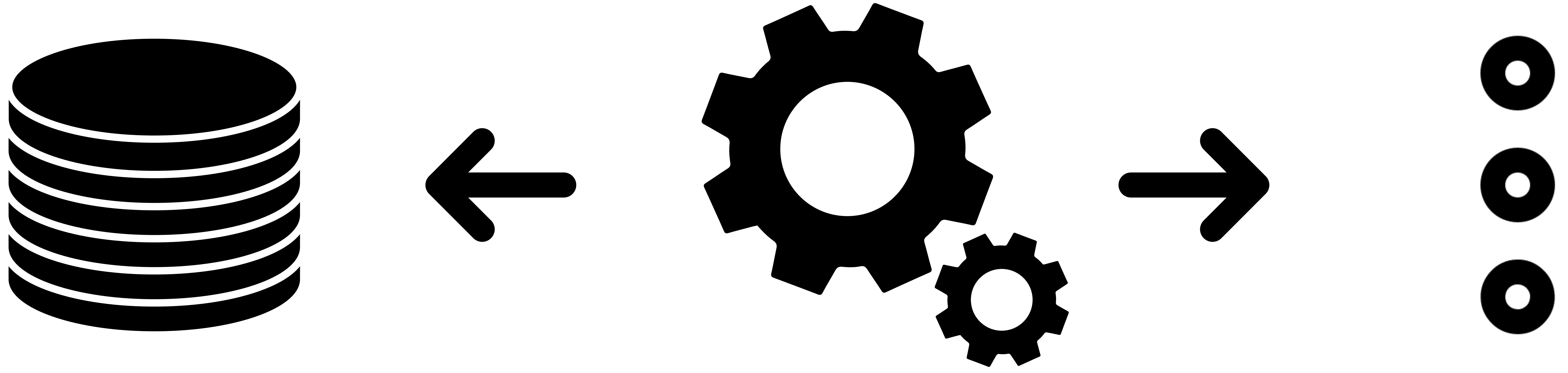
There are many challenges for the future in big data management and analytics, arising from the very nature of data: large, diverse, and evolving*

**Vivekanand Gopalkrishnan, David Steier,Harvey Lewis,and James Guszcza. Big data, big business: Bridging the gap. In Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine 2012). Beijing, China, August 12–12, 2012, pages 7–11. ACM, 2012.*

Some of the challenges that researchers and practitioners will have to deal with in the years to come are:

Big Data: Challenges in Big Data Analytics architecture

It is not clear yet how an optimal architecture of an analytics system should be built to deal with historical data and with real-time data at the same time



Challenges in Big Data

Analytics architecture

A first proposal was the Lambda architecture of Nathan Marz*. The Lambda architecture solves the problem of computing arbitrary functions on arbitrary data in real time by decomposing the problem into three layers: the batch layer, the serving layer, and the speed layer.

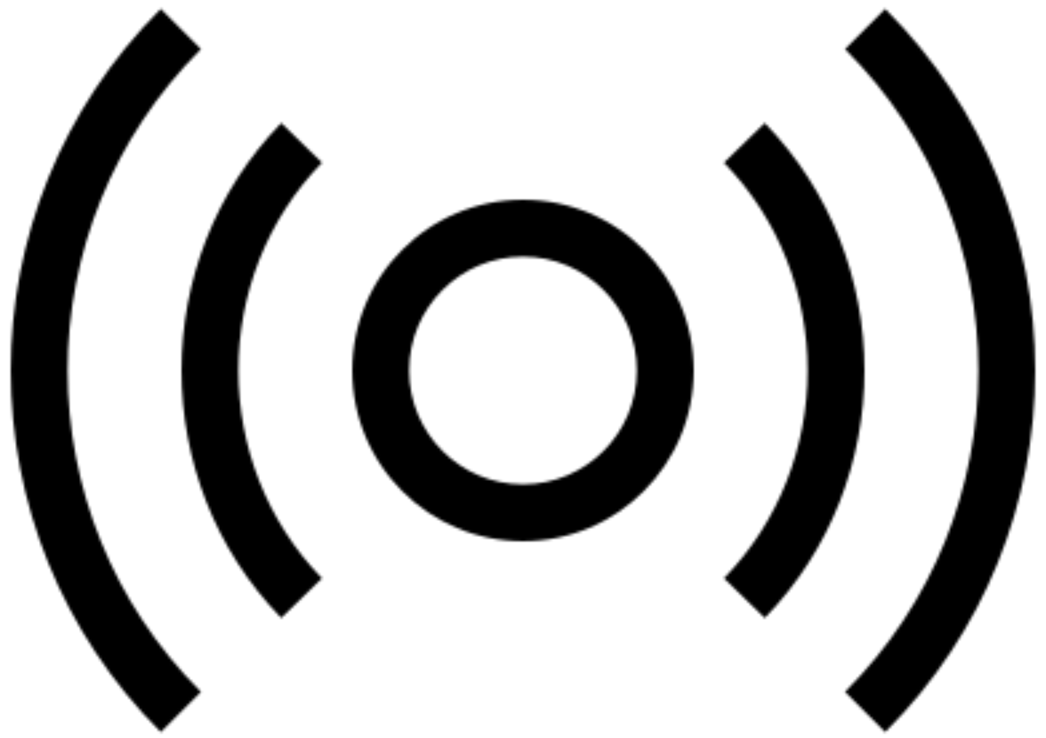
Nathan Marz and James Warren. BigData: Principles and best practices of scalable real-time data systems. Manning Publications, 2013.

Challenges in Big Data Analytics architecture

1. The Batch Layer

This layer receives data through the **master** dataset in an append-only format from **different sources**. The batch layer processes big data sets in intervals to create batch views that will be stored by the serving layer. The data in this layer is immutable. Immutability and receiving data in append-only format is what makes the Lambda architecture fault tolerant and prevents data loss.
The batch layer does not use incremental algorithms rather it uses re-computation algorithms. This layer produces complete data because the machine learning algorithms are able to train models since the batch layer takes more time to process large datasets

Challenges in Big Data Analytics architecture

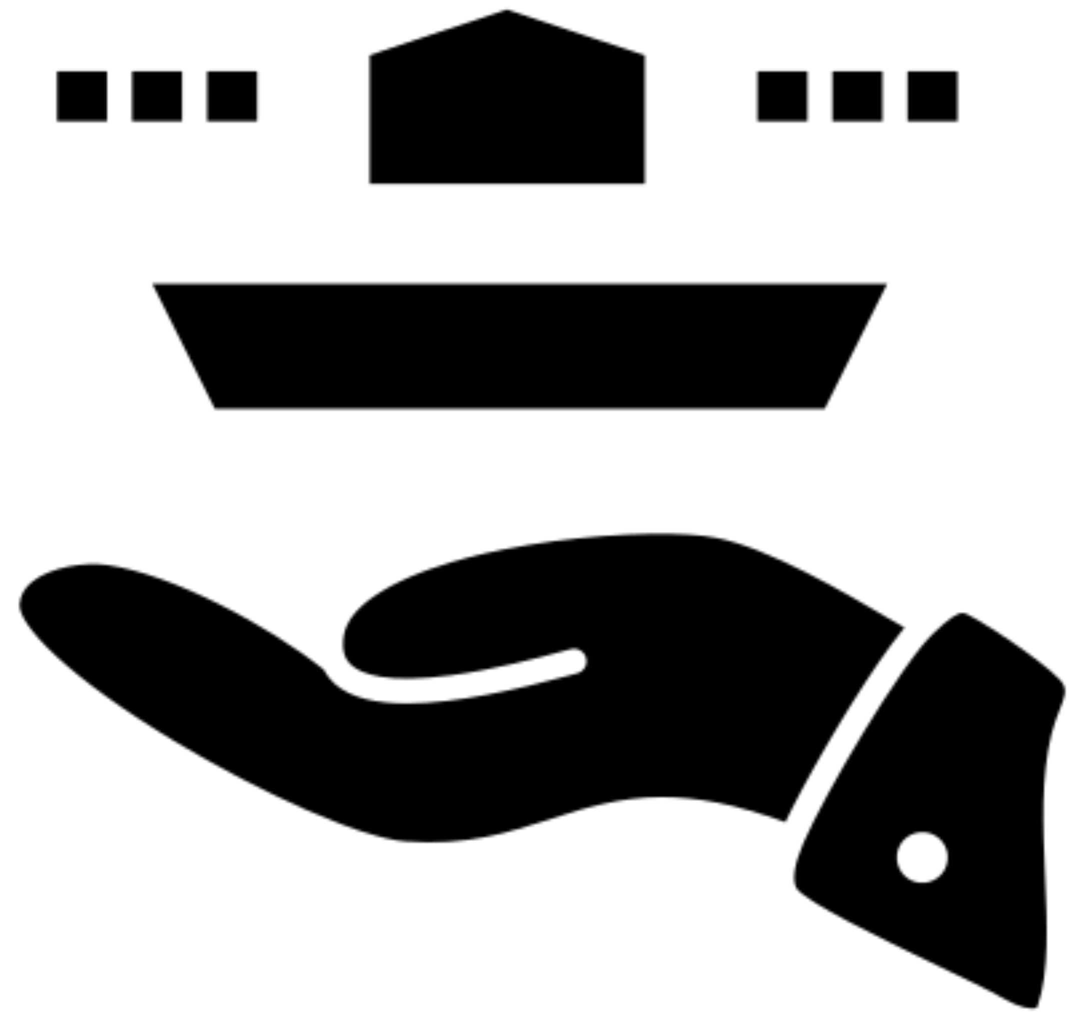


2. The Speed or Streaming Layer

The speed layer processes data using data streaming processes and tools such as Apache Kafka; its goal is to deliver data in real-time. It favors low-latency over throughput. The speed layer focuses on filling the gaps left by the batch layer. This layer uses complex incremental algorithms and computation.

Challenges in Big Data Analytics architecture

3. The Serving Layer
This layer queues batch views that have been prepared by the batch layer and then indexes them. The serving layer's goal is to make the data queryable in a very short period of time. The server layer stores the output and merges the batch layer output with the speed layer output.



Challenges in Big Data

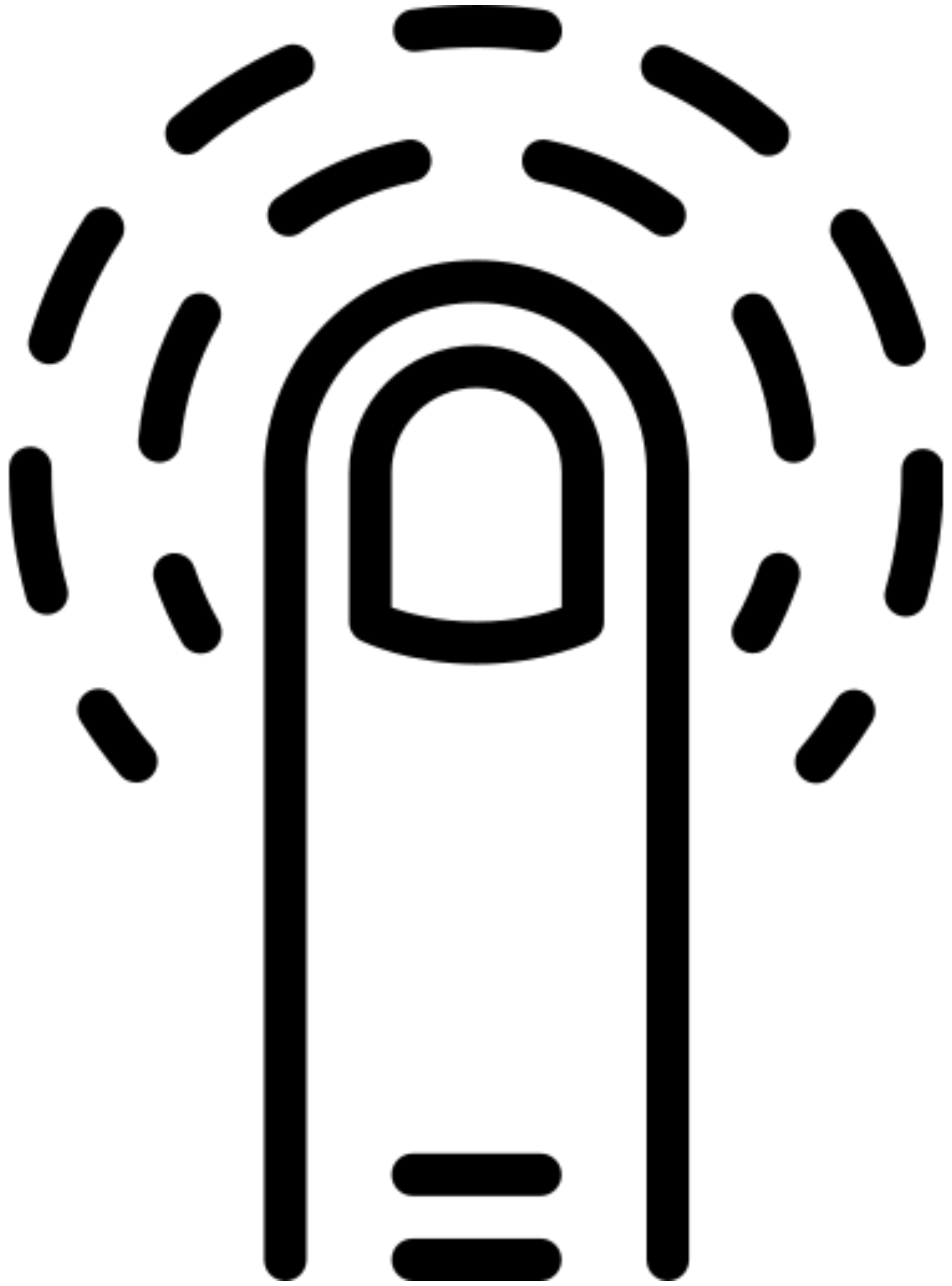
Evaluation

It is important to achieve significant statistical results, and not be fooled by randomness. If the "multiple hypothesis problem" is not properly cared for, it is easy to go wrong with huge datasets and thousands of questions to answer at once*

*B. Efron. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction.* *Institute of Mathematical Statistics Monographs.* Cambridge University Press, 2010

Challenges in Big Data

Evaluation

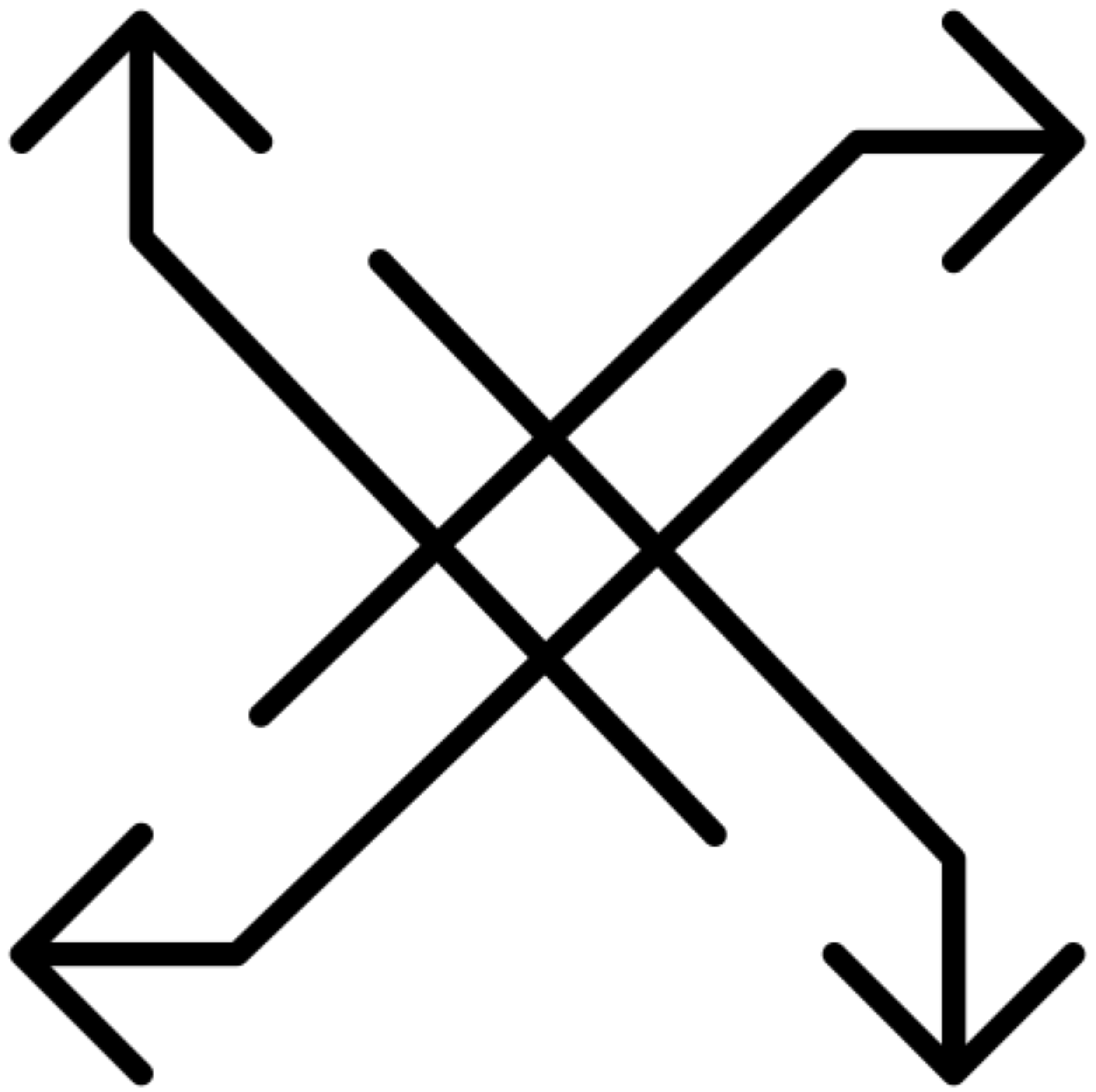


Is important to avoid the trap of focusing only on technical measures such as error or speed instead of on eventual real-world impact*

**Kiri Wagstaff. Machine learning that matters. In Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012, 2012.*

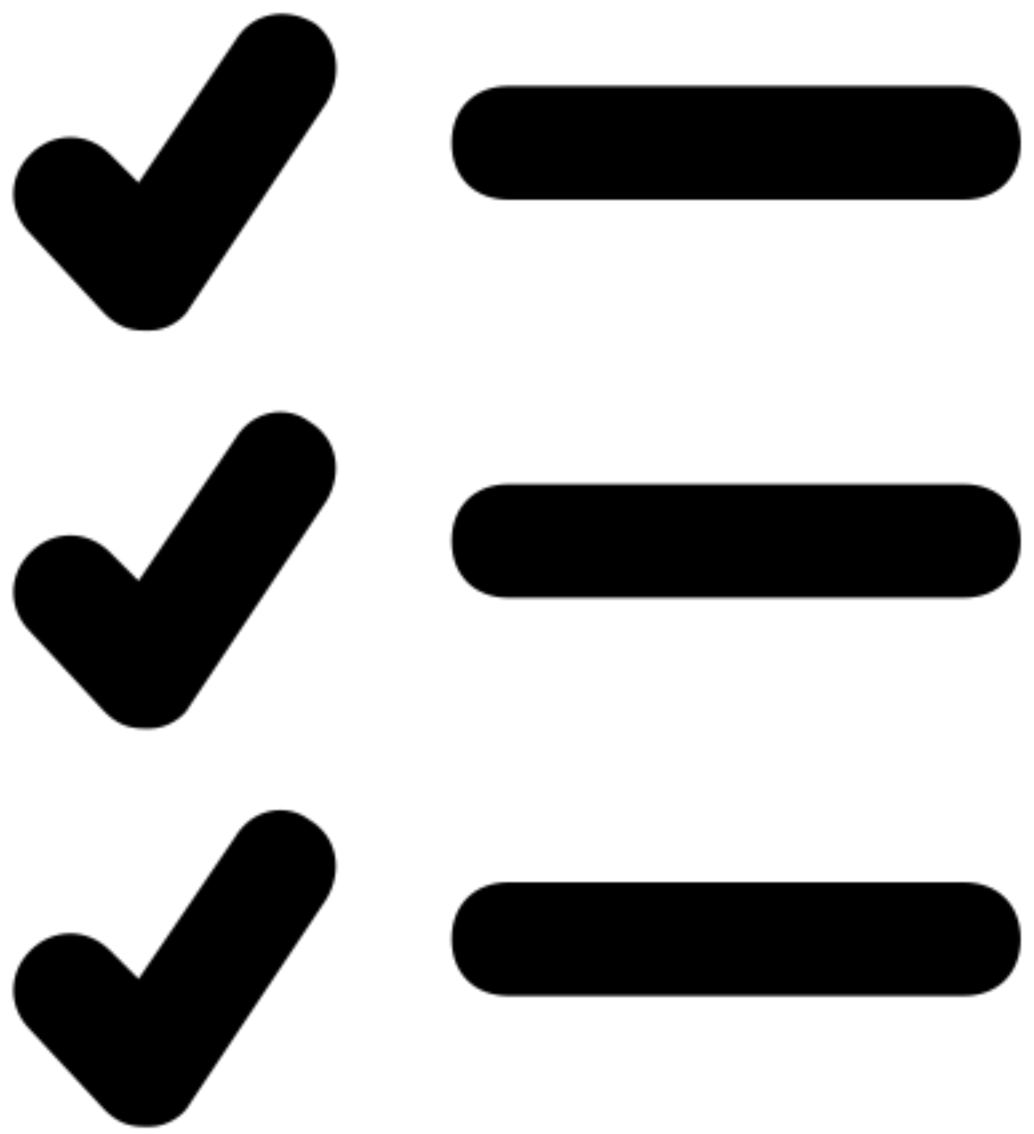
Challenges in Big Data

Distributed mining



Many data mining techniques are not trivial to parallelize. To have distributed versions of some methods, substantial research is needed with both practical experiments and theoretical analysis

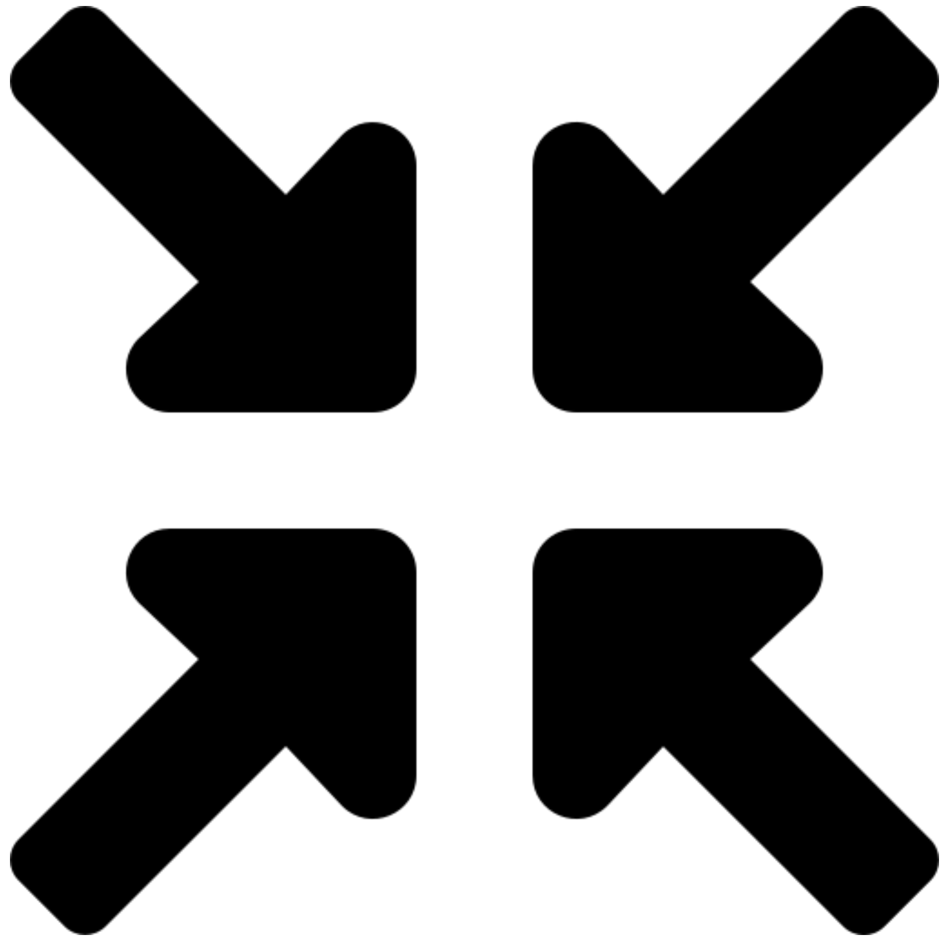
Challenges in Big Data
Time evolving data



Data may be evolving overtime, so it is important that the big data mining techniques are able to adapt to, and in some cases explicitly detect, change*

Joa˜o Gama. Knowledge Discovery from Data Streams. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press, 2010.

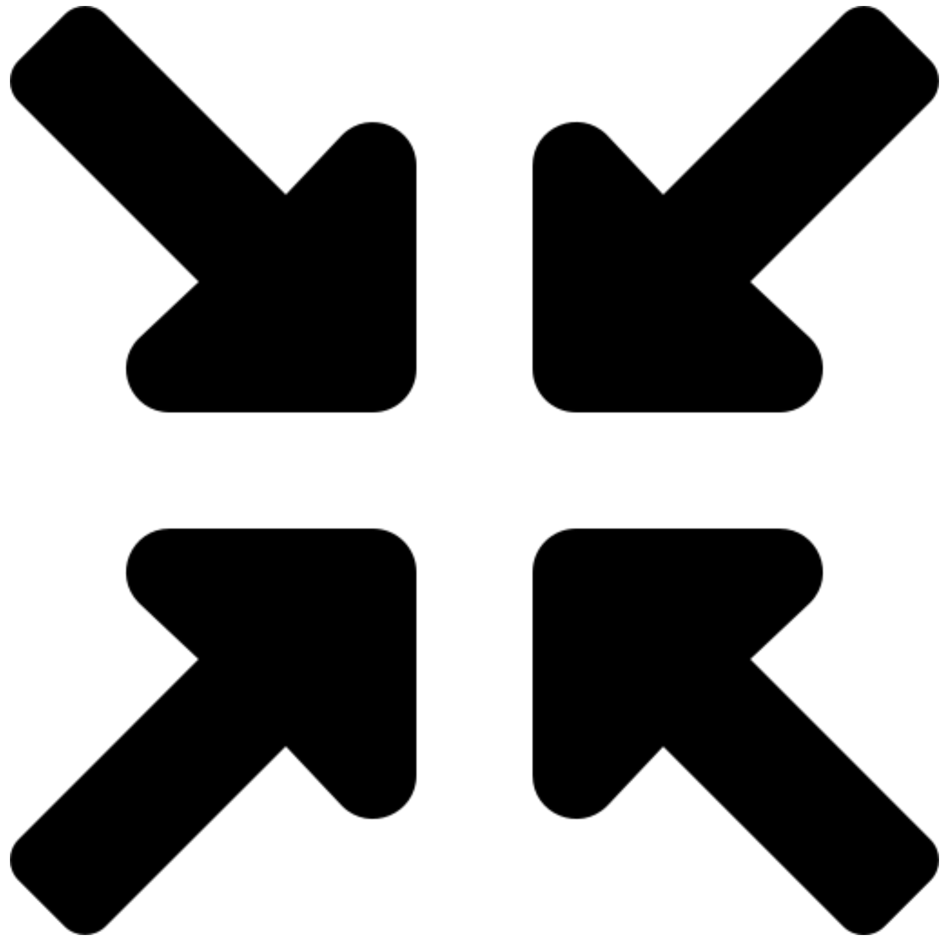
Big Data: Challenges in Big Data Compression



When dealing with big data, the quantity of space needed to store it is very relevant. There are two main approaches:

- **compression**, where we lose no information
- **sampling**, where we choose data that we deem representative

Big Data: Challenges in Big Data Compression



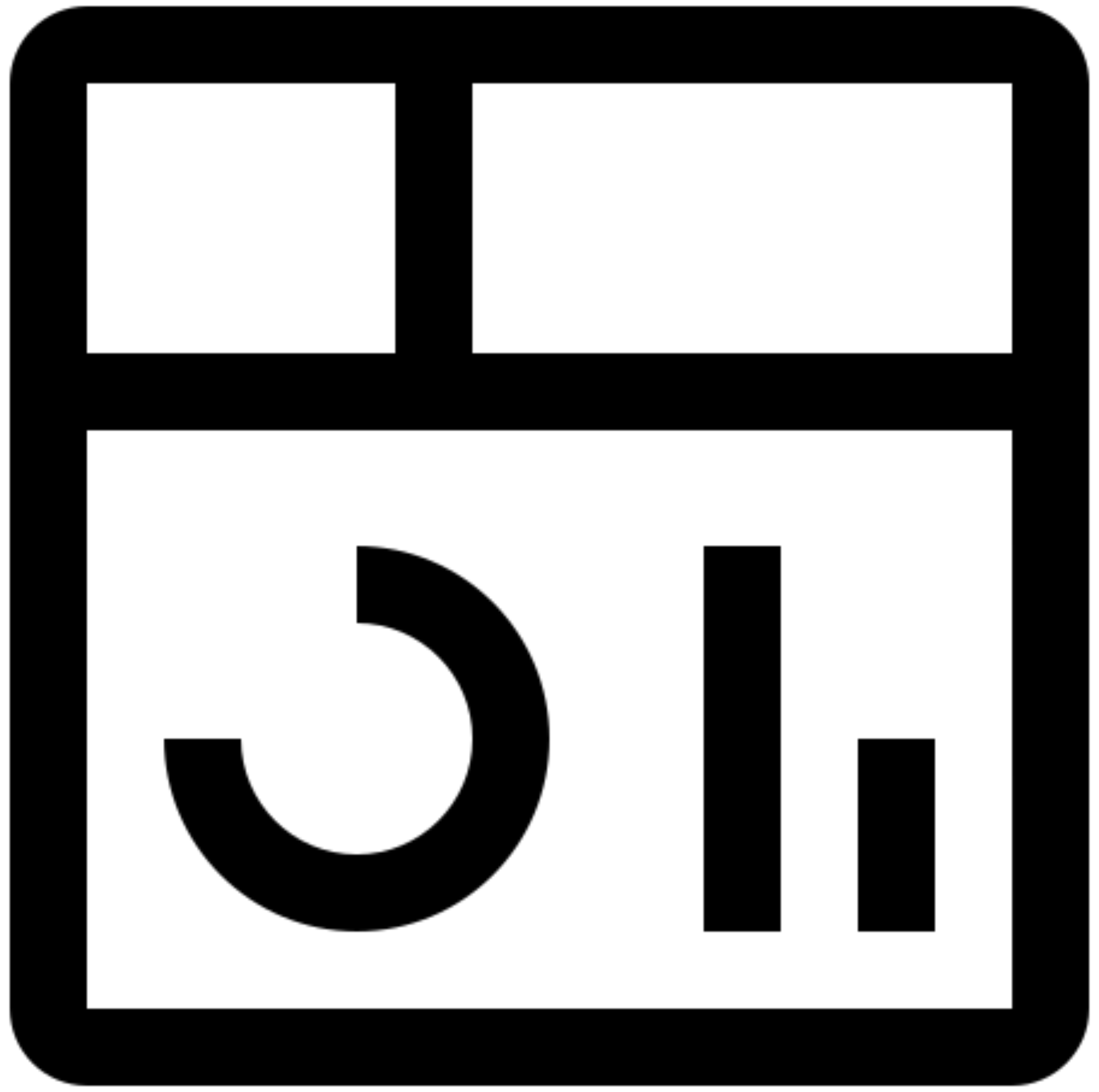
Using compression, we will use more time and less space, so we can consider it as a transformation from time to space

Using sampling, we are losing information, but the gains in space may be in orders of magnitude

For example Feldman et al* use **coresets** to reduce the complexity of big data problems; a coreset is a small subset of the data that provably approximates the original data for a given problem

Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013, pages 1434–1453, 2013

Big Data: Challenges in Big Data Visualization

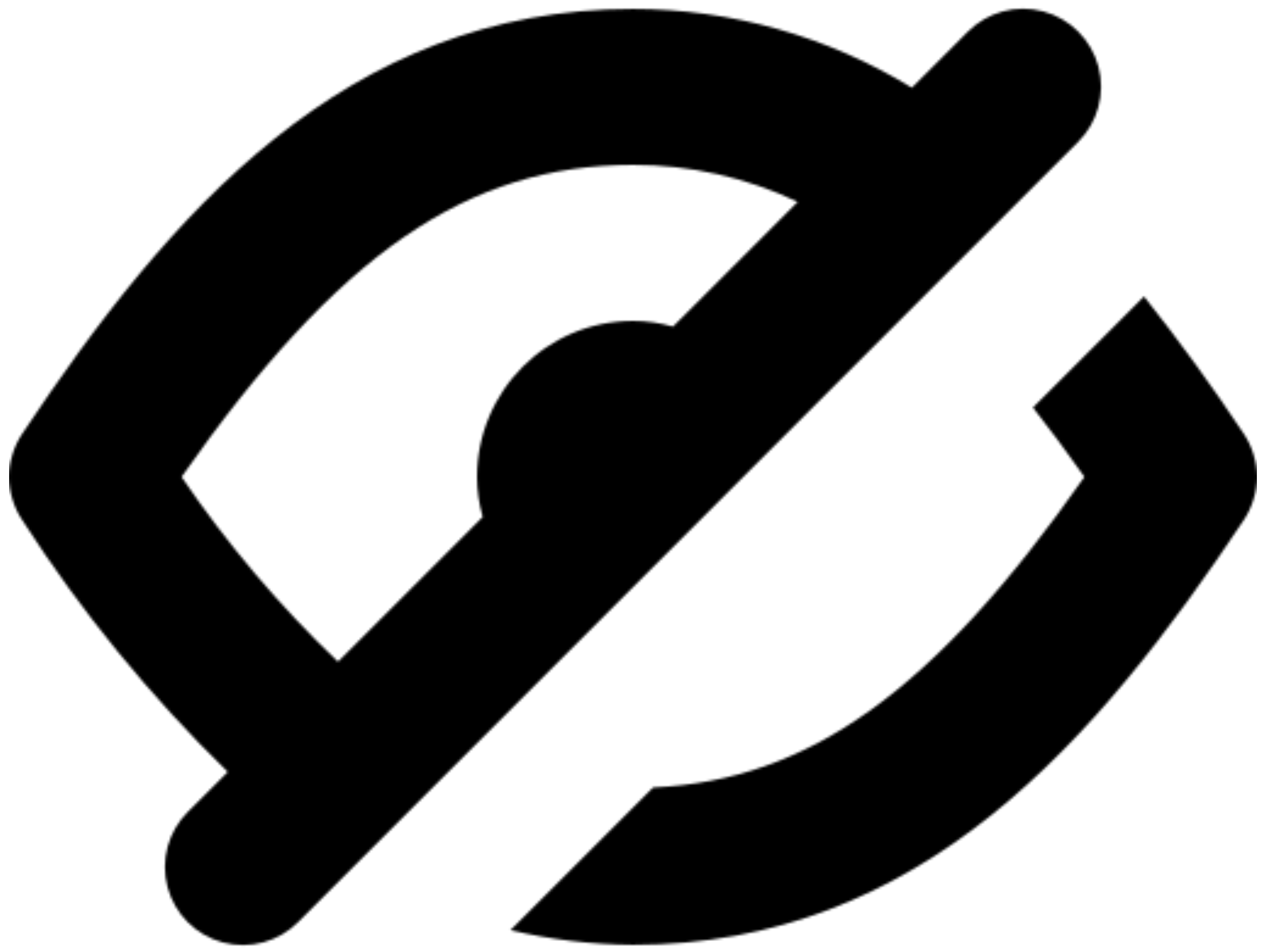


A main issue in big data analysis is how to visualize the results. Presenting information from large amounts of data in a way that is understandable to humans is quite a challenge. It requires new techniques and frameworks to tell stories, such as those covered in the book ***The Human Face of Big Data****

**R.Smolanand J.Erwitt. The Human Face of BigData. Sterling Publishing Company Incorporated, 2012*

Big Data: Challenges in Big Data

Hidden big data



Large quantities of useful data are in fact useless because they are untagged, file-based, and unstructured. The 2012 IDC study on big data* explained that, in **2012, 23% (643 exabytes)** of the digital universe would be useful for big data if tagged and analyzed.

**John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, December 2012*

Big Data: Challenges in Big Data

Hidden big data

However, at that time **only 3%** of the potentially useful data was tagged, and even less was analyzed. The figures have probably gotten worse in recent years. The **Open Data** and **Semantic Web** movements have emerged, in part, to make us aware and improve on this situation.

Real-Time Analytics

One particular case of the big data scenario is real-time analytics.
It is important for organizations not only to obtain answers to queries immediately, but to do so according to the data that has just arrived

Real-Time Analytics

Data streams: Definition

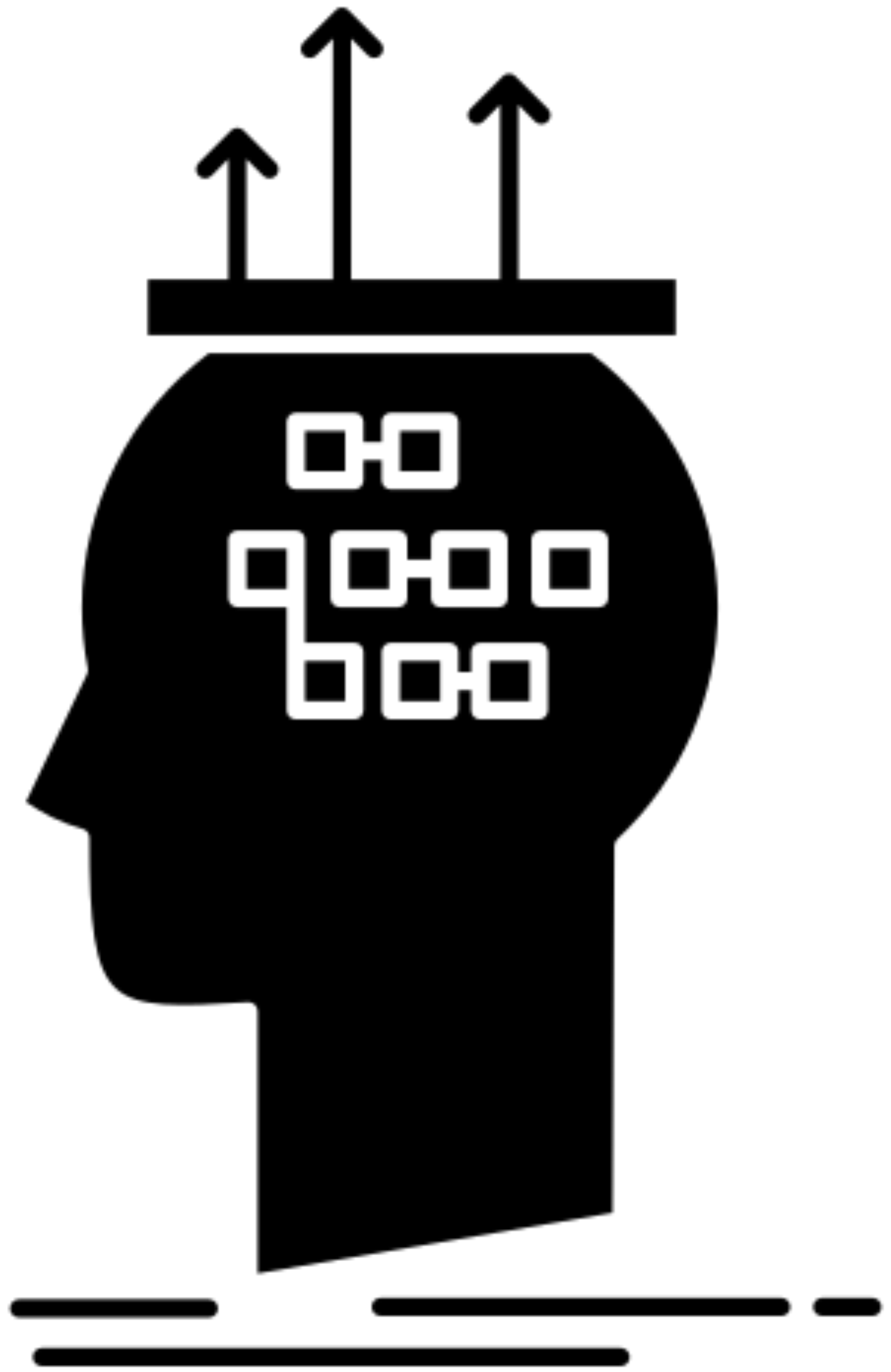
Data streams are an **algorithmic abstraction** to support real-time analytics. They are sequences of items, **possibly infinite**, each item having a **timestamp**, and so a **temporal order**. Data items arrive one by one, and we would like to build and maintain models, such as patterns or predictors, of these items in real time.

Real-Time Analytics

Data streams

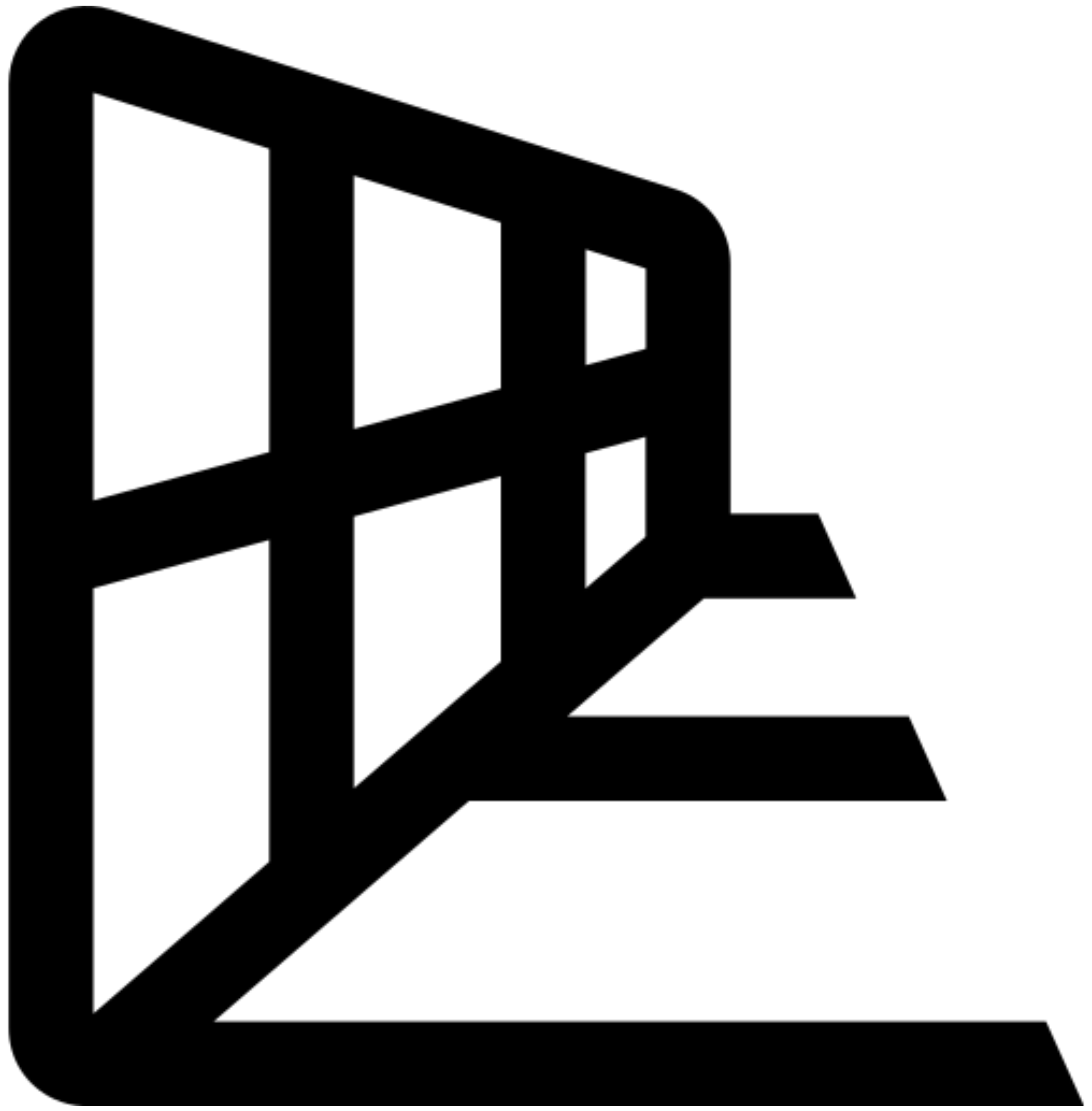
There are two main algorithmic challenges when dealing with streaming data:

- **The stream is large and fast**, and we need to extract information in real time from it. That means that usually we need to accept approximate solutions in order to use less time and memory
- **The data may be evolving**, so our models have to adapt when there are changes in the data.



Real-Time Analytics

Data streams: the dimensions



Accuracy, time, and memory are the three main resource dimensions of the stream mining process: we are interested in methods that obtain the maximum accuracy with minimum time and low total memory

Real-Time Analytics
Applications

Sensor data and the Internet of Things: Every day, more sensors are used in industry to monitor processes, and to improve their quality. Cities are starting to implement huge networks of sensors to monitor the mobility of people and to check the health of bridges and roads, traffic in cities, people's vital constants, and so on

Telecommunication data: Telecommunication companies have large quantities of phone call data. Nowadays, mobile calls and mobile phone locations are huge sources of data to be processed, often in real-time

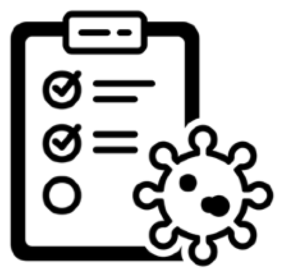
Real-Time Analytics
Applications

Social media: The users of social websites such as Facebook, Twitter, LinkedIn, and Instagram continuously produce data about their interactions and contributions. Topic and community discovery and sentiment analysis are but two of the real-time analysis problems that arise

Marketing and e-commerce: Sales businesses are collecting in real time large quantities of transactions that can be analyzed for value. Detecting fraud in electronic transactions is essential

Real-Time Analytics

Applications



Health care: Hospitals collect large amounts of time-sensitive data when caring for patients, for example, monitoring patient vital signs such as blood pressure, heart rate, and temperature. Telemedicine will also monitor patients when they are home, perhaps including data about their daily activity with separate sensors. Also, the system could have results of lab tests, pathology reports, X-rays, and digital imaging. Some of this data could be used in real time to provide warnings of changes in patient conditions

Epidemics and disasters: Data from streams originating in the Internet can be used to detect epidemics and natural disasters, and can be combined with official statistics from official centers for disease and disaster control and prevention

Real-Time Analytics

Applications

Computer security: Computer systems have to be protected from theft and damage to their hardware, software and information, as well as from disruption or misdirection of the services they provide, in particular, insider threat detection and intrusion detection

Electricity demand prediction: Providers need to know sometime in advance how much power their customers will be requesting. The figures change with time of day, time of year, geography, weather, state of the economy, customer habits, and many other factors, making it a complex prediction problem on massive, distributed data

Unlocking Data Insights - Introduction to Data-Centric AI

Big Data Stream Mining

Machine Learning for Data Streams: with Practical Examples in MOA, Albert Bifet, Ricard Gavaldà, Geoff Holmes, Bernhard Pfahringer, The MIT Press

Big Data Stream Mining

Executive Summary

- Algorithms


Credits: dalle2.gallery

Algorithms

The main algorithms in data stream mining are classification, regression, clustering, and frequent pattern mining

Algorithms
Offline setting

- We are in a **classification setting** when we need to assign a label from a set of nominal labels to each item, as a function of the other features of the item. A classifier can be trained as long as the correct label for (many of) the examples is available at a later time

- **Regression** is a prediction task similar to classification, with the difference that the label to predict is a numeric value instead of a nominal one. An example of regression is predicting the value of a stock in the stock market tomorrow

- When examples are not labeled, one interesting task is to group them in homogeneous clusters. **Clustering** can be used, for example, to obtain user profiles in a website. It is an example of an *unsupervised* learning task.

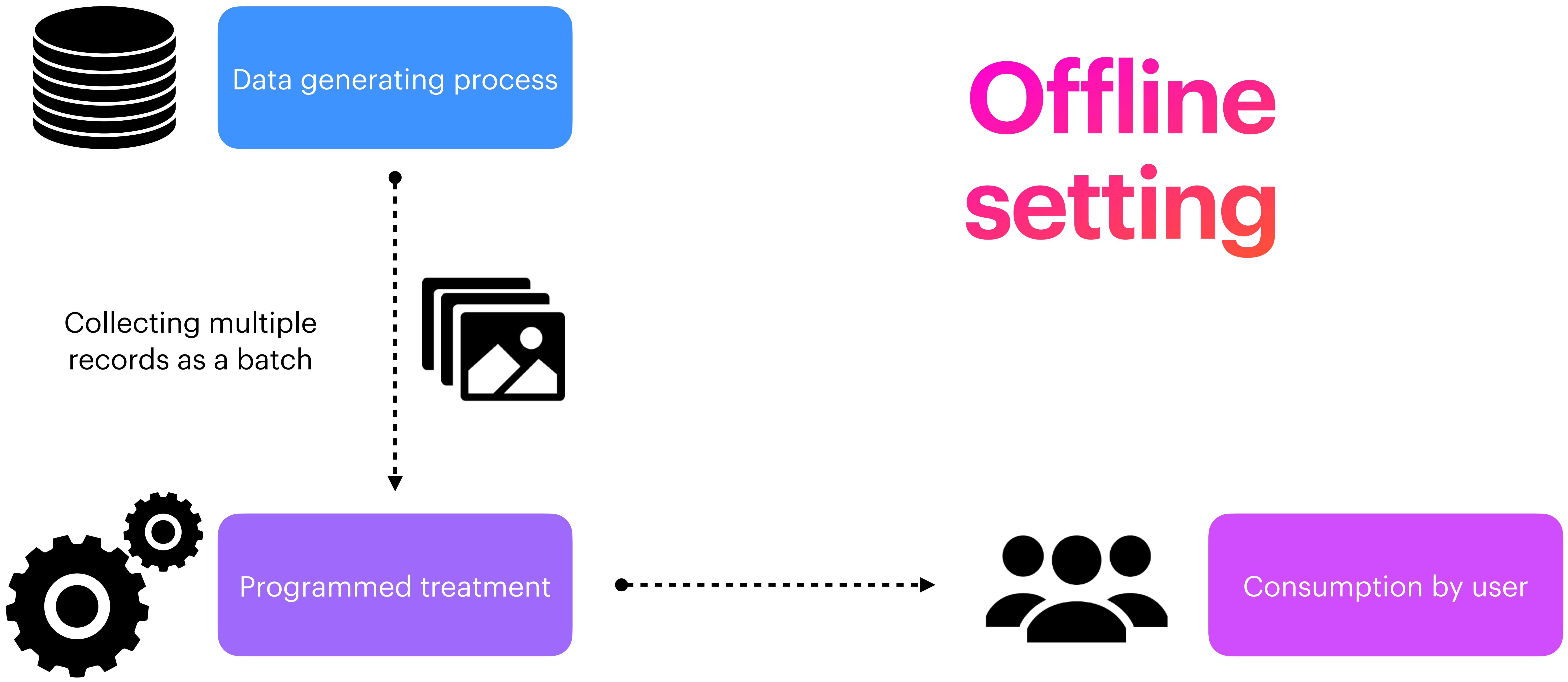
- **Frequent pattern mining** looks for the most relevant patterns within the examples. For instance, in a sales supermarket dataset, it is possible to know what items are bought together and obtain association rules, as for example: *Most times customers buy cheese, they also buy wine*.

Algorithms
But ... in the online setting

The **most significant requirements** for a stream mining algorithm are the same for predictors, clusterers, and frequent pattern miners:

- Process an instance at a time, and inspect it (at most) once
- Use a limited amount of time to process each instance
- Use a limited amount of memory
- Be ready to give an answer (prediction, clustering, patterns) at any time
- Adapt to temporal changes

Classification



Classification

Online setting

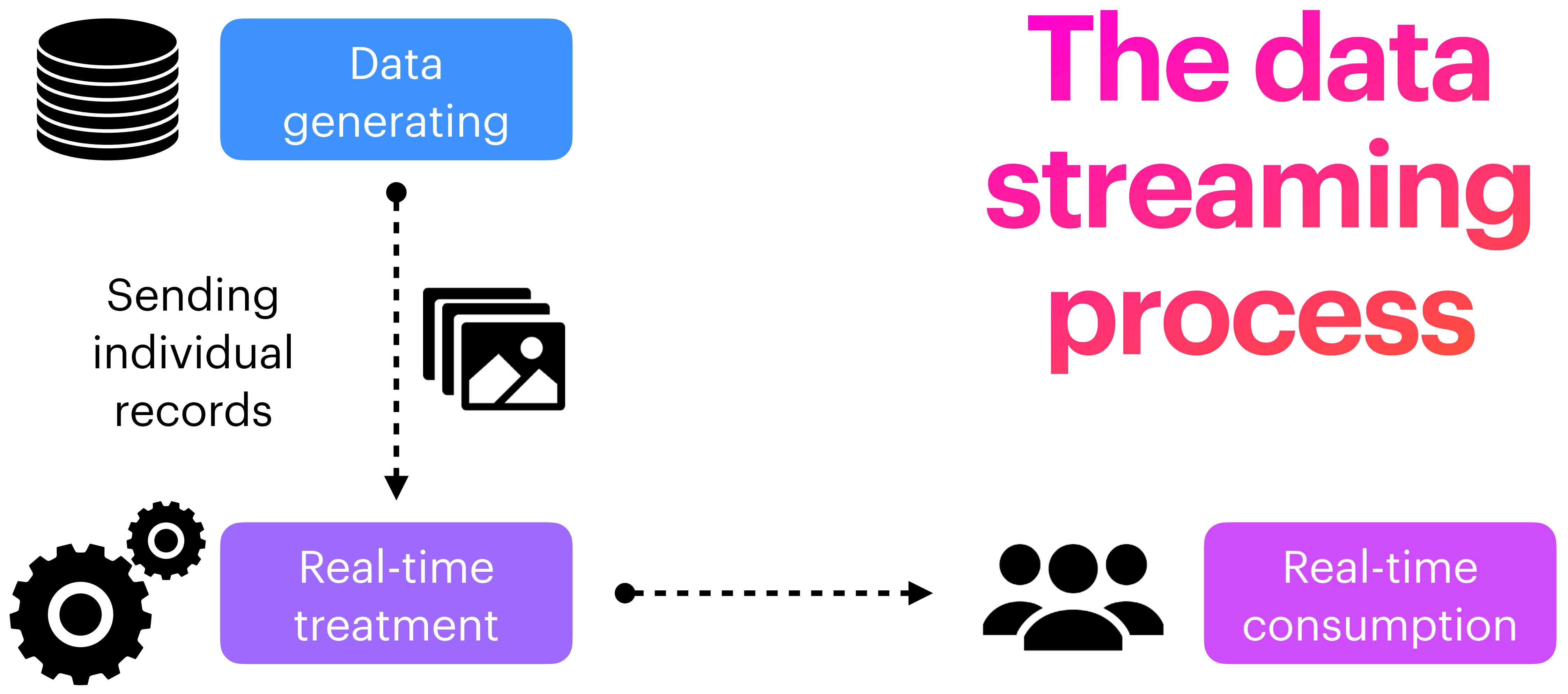
In the online setting, and in particular in streaming, this separation between training, evaluating, and testing is far less clear-cut, and is interleaved

Classification

Online setting

Generally speaking, a stream mining classifier is ready to do either one of the following at any moment:

1. Receive an unlabeled example and make a prediction for it on the basis of its current model
2. Receive the label for an example seen in the past, and use it for adjusting the model, that is, for training



Classification

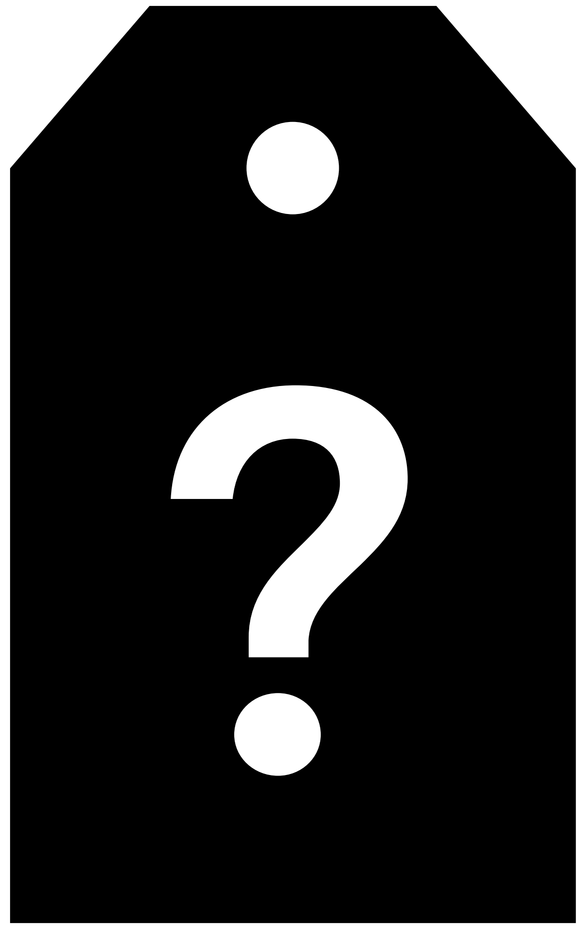
Online setting: Customer purchase

For example, an online shop may want to predict, for each arriving customer, whether the customer will or will not buy a particular product (prediction).

When the customer session ends, say, minutes later, the system gets the “label” indicating whether indeed the customer bought the product or not, and this feedback can be used to tune the predictor

Classification

Online setting: Fraud detection



In other cases, the label may never be known

If trying to detect fraudulent transactions in order to block them, transactions predicted to be fraudulent are not executed, so their true labels are never known

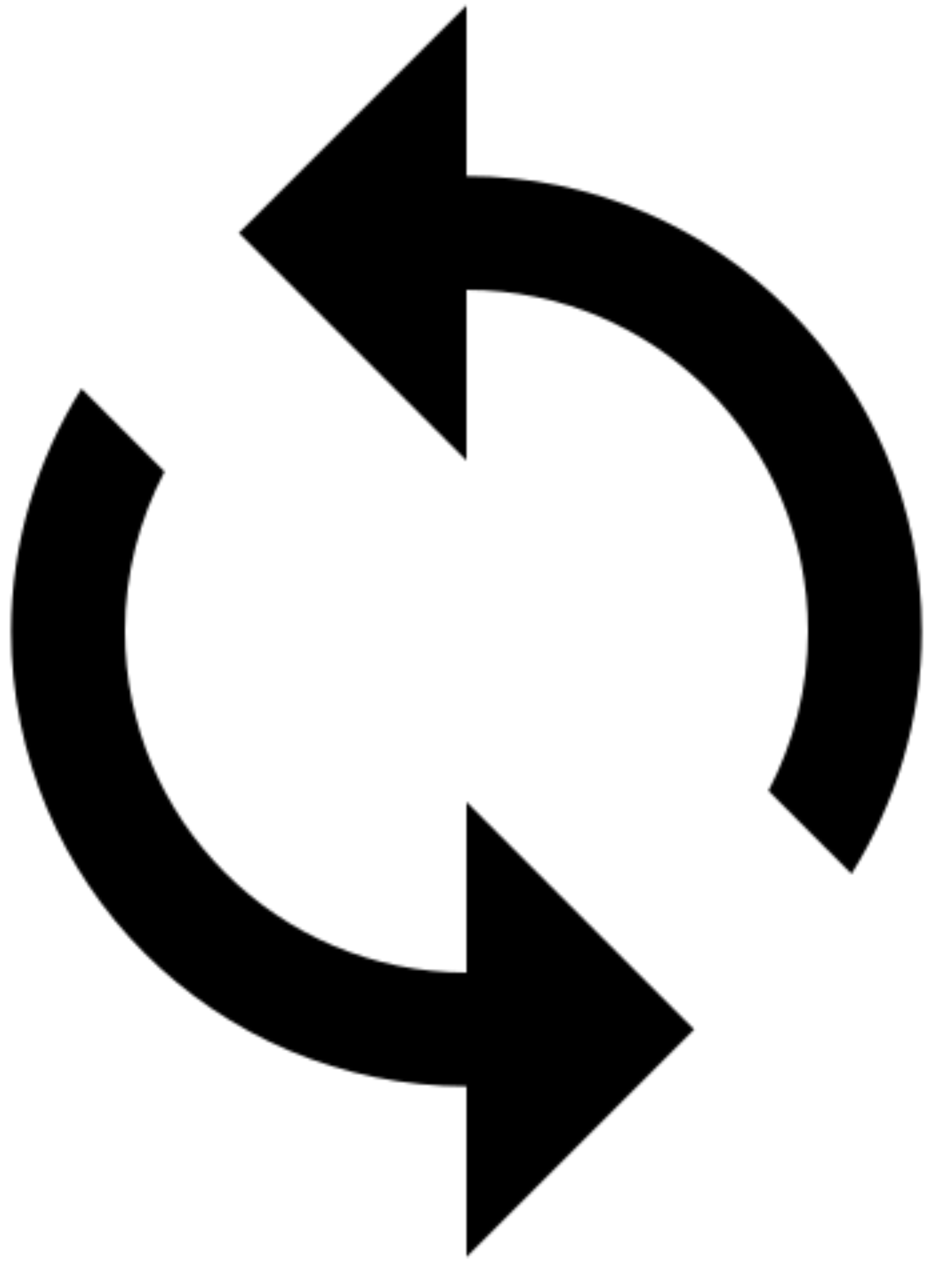
Classification

Accuracy: *How many of the unlabeled instances eventually receive their correct label?* Clearly, the fewer labels received, the harder the prediction task.

Memory: *How long should we wait for an instance label to arrive, before we drop the instance?* Efficiently managing the buffer of instances waiting for their labels is a very delicate implementation problem when dealing with massive, high-speed streams.

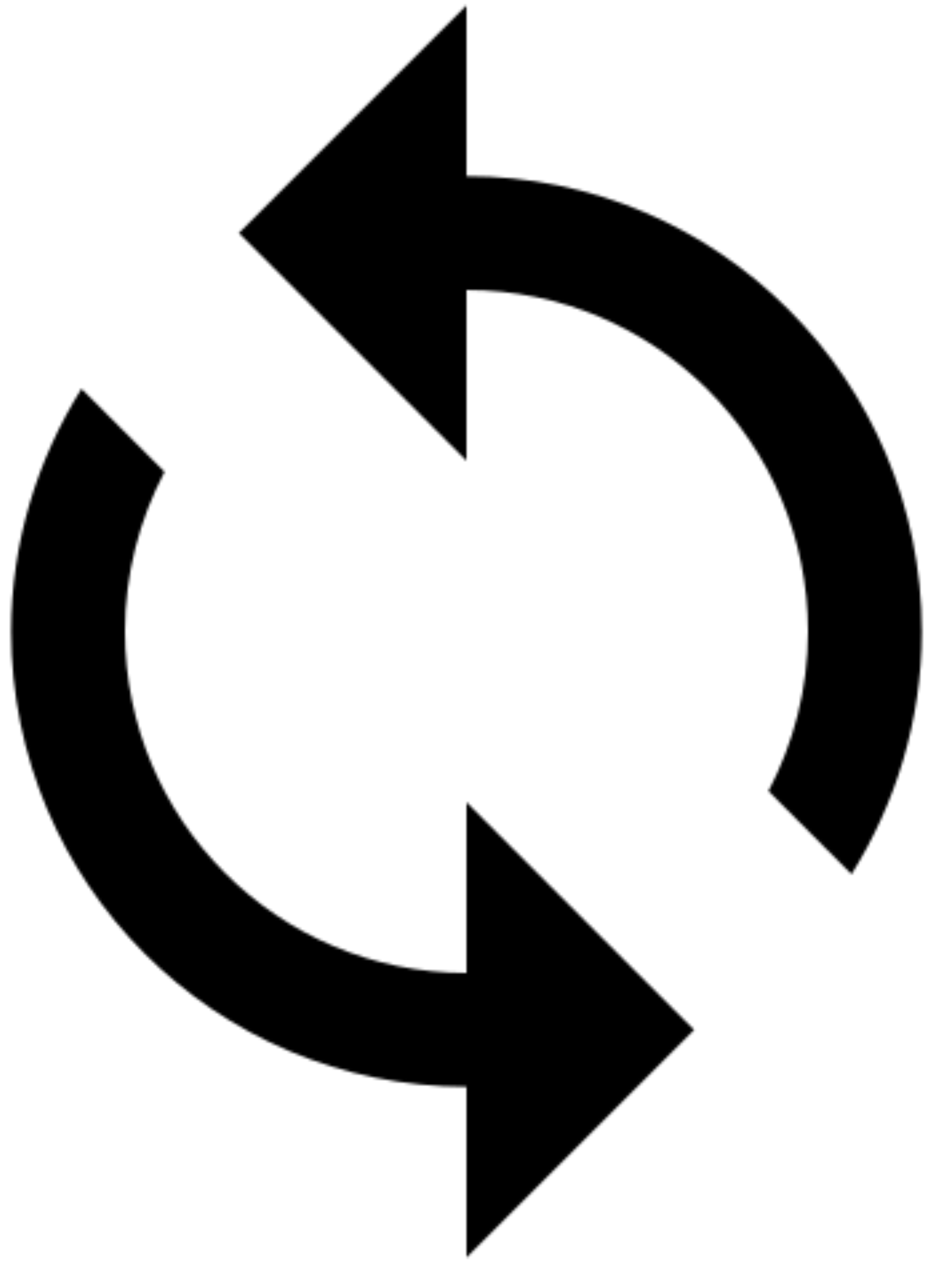
Training strategies: *Should we use all labeled instances for training?* If in fact many labels are available, perhaps there is a diminishing return in accuracy for the increased computational cost of training on all instances.

Classification



A large part of the research in stream classification deals with a simplified cycle of training/prediction: we assume that we get the true label of *every* unlabeled instance, and that furthermore we get it *immediately* after making the prediction and before the next instance arrives.

Classification



Get an unlabeled instance x

Make a prediction $\hat{y} = f(x)$ for x's label, where f is the current model

Get the true label y for x

Use the pair (x, y) to update the model f, and the pair $(\hat{y}, y)$ to update the metrics

Proceed to the next instance

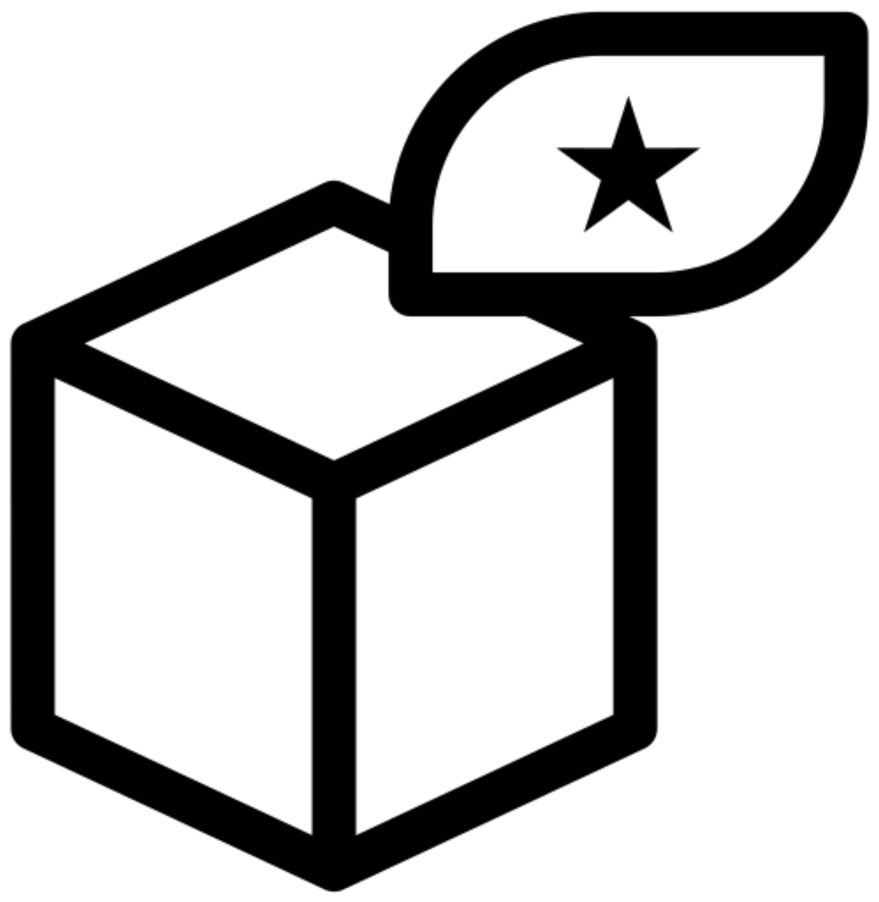
Classification

This model is rightly criticized by practitioners as too simple, because it ignores the very real problem of *delayed* and *missing* label feedback. It is however quite useful for comparing learning algorithms in a clean way, provided we have access to, or can simulate, a stream for which we have all labels.

Classification

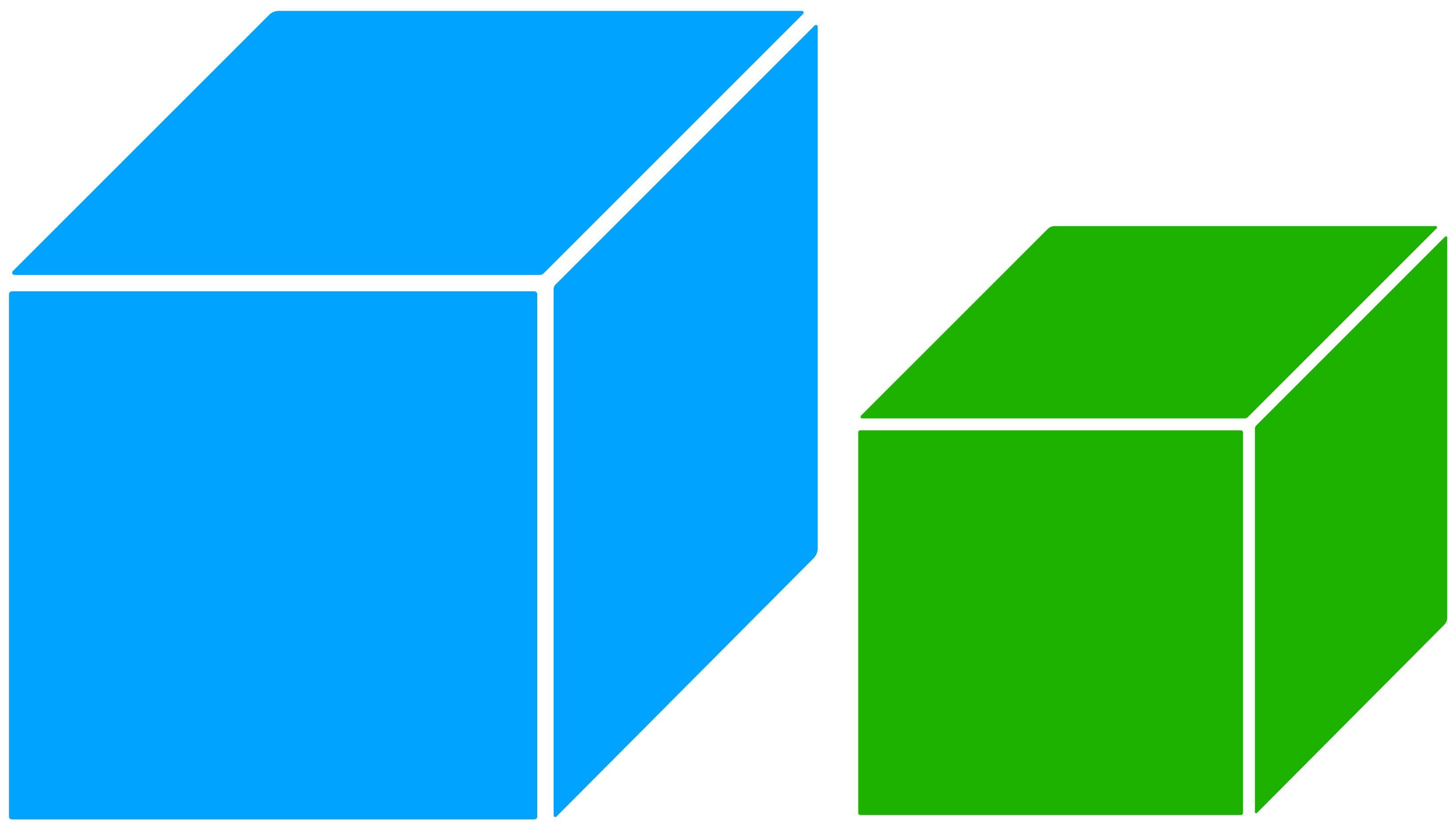
Classifier Evaluation in Data Streams

Given this cycle, it is reasonable to ask: How do we evaluate the performance of a classification algorithm?



Classification

Classifier Evaluation in Data Streams



In traditional batch learning, evaluation is typically performed by randomly splitting the data into **training and testing sets (holdout)**; if data is limited, **cross-validation** (creating several models and averaging results across several random partitions in training and test data) is preferred.

Classification

Classifier Evaluation in Data Streams

- In the stream setting, (effectively) unlimited data tends to make cross-validation too expensive computationally, and less necessary anyway. But it poses **new challenges**

- The main one is to build an **accurate picture of accuracy over time**. One solution involves taking snapshots at different times during the induction of a model to see how the model accuracy varies

Classification

Classifier Evaluation in Data Streams

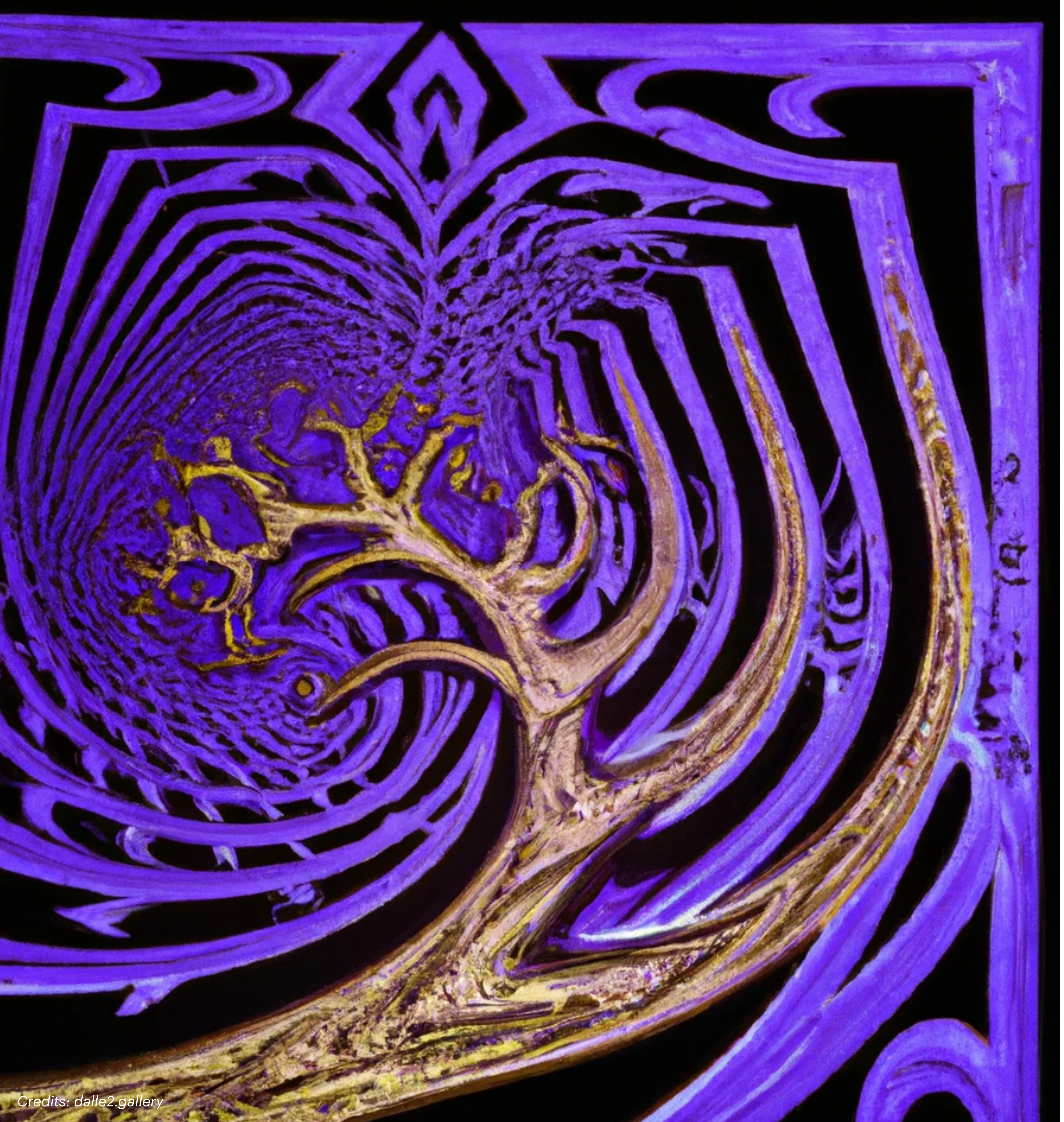
Interleaved test-then-train or prequential: Each individual example is used to test the model before it is used for training, and from this the accuracy can be incrementally updated.

When the evaluation is intentionally performed in this order, the model is always being tested on instances it has not seen. This scheme has the advantage that no holdout set is needed for testing, making maximum use of the available data

Classification

Classifier Evaluation in Data Streams

- It also ensures a smooth plot of accuracy over time, as each individual example will become less and less significant to the overall average.
- In test-then-train evaluation, all examples seen so far are taken into account to compute accuracy, while in prequential, only those in a sliding window of the most recent ones are.
- As data stream classification is a relatively new field, such evaluation practices are not nearly as well researched and established as they are in the traditional batch setting.



Classification

Decision Tree

- **Traditional decision trees** scan the entire dataset to discover the best attribute to form the initial split of the data.
- Once this is found, the data is split by the value of the chosen attribute, and the algorithm is applied recursively to the resulting datasets, to build subtrees.
- Recursion is applied until some stopping criterion is met.

This approach cannot be adopted directly in the stream setting, as we cannot afford the resource cost (time and memory) of storing instances and repeatedly scanning them.

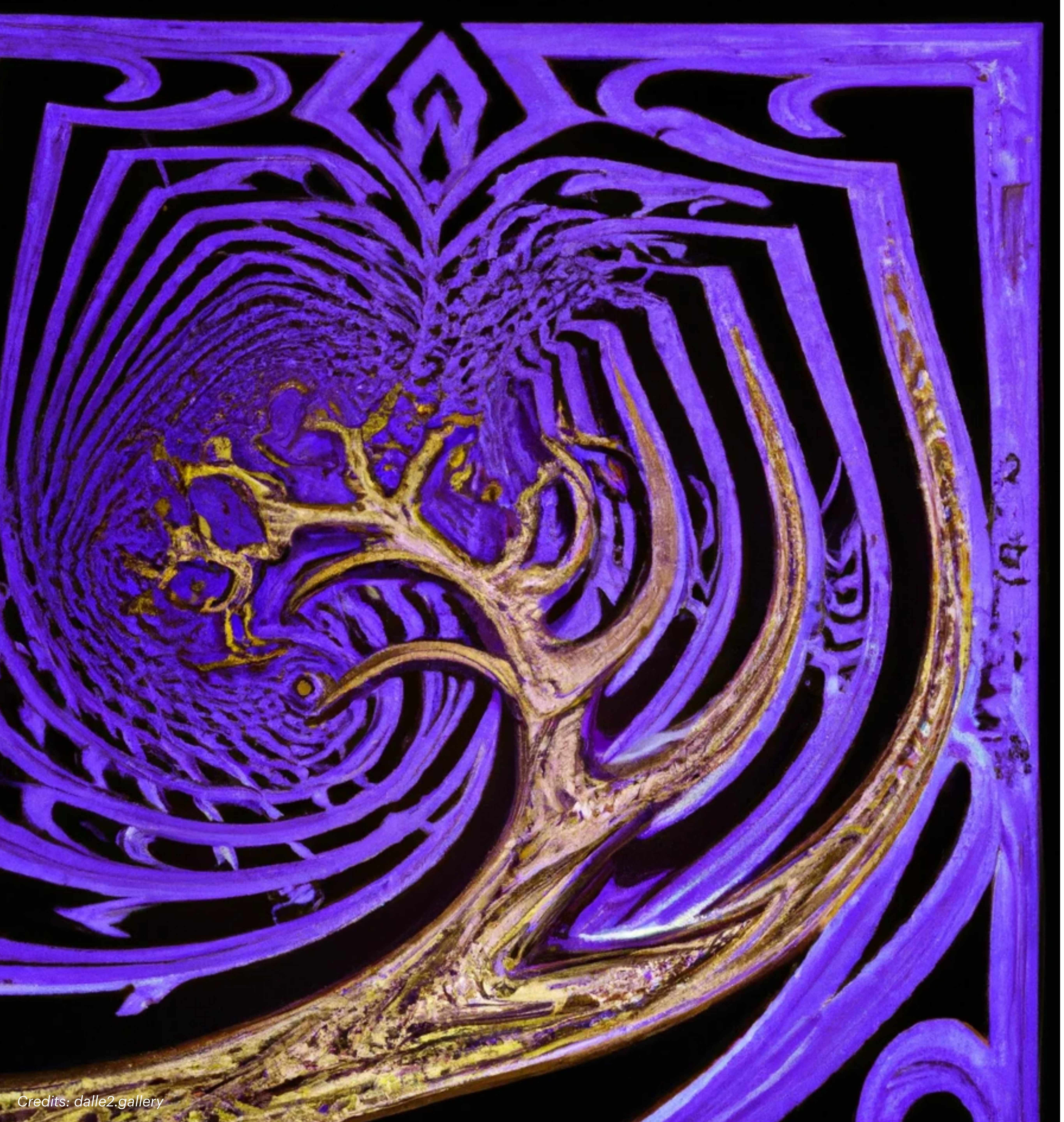
Classification

Decision Tree

Decision tree learners build a tree structure from training examples to predict class labels of unseen examples

In stream mining, the state-of-the art decision tree classifier is the ***Hoeffding tree***, due to Domingos and Hulten*, and its variations

**Pedro M.Domingos and Geoff Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 – July 1, 2001, pages 106–113, 2001.*



Classification

Decision Tree

- **The Hoeffding tree** is based on the idea that, instead of looking at previous (stored) instances to decide what splits to do in the trees, we can wait to receive enough instances and make split decisions when they can be made confidently.
- The **main advantage** of this approach is that it is not necessary **to store instances.** Instead, sufficient statistics are kept in order to make splitting decisions.
- The sufficient statistics make it easy to **incorporate Naive Bayes models** into the leaves of the tree.

The *Hoeffding adaptive tree** is an extension of the Hoeffding tree that is able to create and replace new branches when the data stream is evolving and the class label distribution or instance distribution is changing.

**Albert Bifet and Ricard Gavalda`. Adaptive learning from evolving data streams. In Advances in Intelligent Data Analysis VIII, 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31 – September 2, 2009. Proceedings, pages 249–260, 2009*

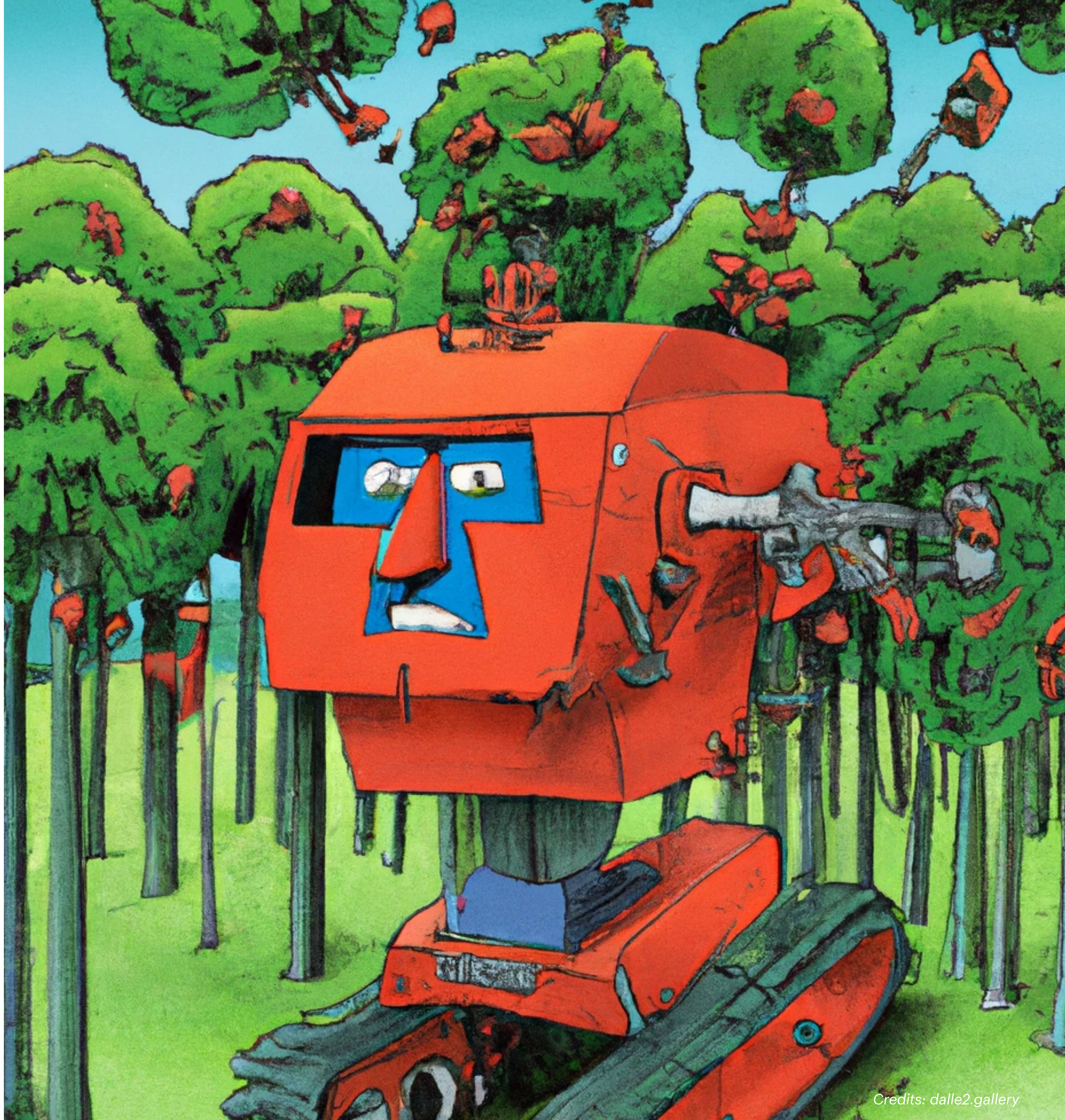
Classification
Ensambles

Ensambles are sets of classifiers that, when combined, can predict better than any of them individually

Bagging is an ensemble method that
(1) uses as input for each run of the classifier builder a subset obtained by sampling with repetition of the original input data stream
(2) uses majority voting of the classifiers as a prediction strategy

Classification
Ensambles

The *ADWIN bagging* method [38], implemented as **OzaBagAdwin** in MOA, is an extension of bagging that it is able to create and replace new classifiers when the data stream is evolving and the class label distribution is changing



Credits: dalle2.gallery

Regression

As in classification, the goal in a regression task is to learn a model that predicts the value of a label attribute for instances where the label is not (yet) known.
Several classification algorithms have natural counterparts for regression, including lazy learning and decision trees.

Unlocking Data Insights - Introduction to Data-Centric AI

Dealing with Change

Machine Learning for Data Streams: with Practical Examples in MOA, Albert Bifet, Ricard Gavaldà, Geoff Holmes, Bernhard Pfahringer, The MIT Press

Dealing with Change

Executive Summary

- Notion of Change in Streams
- Estimators
- Change Detection

Credits: Jonny Gios - Unsplash

Notion of Change in Stream

Let us first discuss the notion of change in streams with respect to notions in other paradigms, as well as some nuances that appear when carefully defining change over time

Notion of Change in Stream

First, nonstationary distributions of data may also appear in batch data analysis. Data in batch datasets may also be timestamped and vary statistically over time. Algorithms may take this possibility into account when drawing conclusions from the data, but otherwise can perform several passes and examine data from before and after any given recorded time

Notion of Change in Stream

In streaming, we cannot explicitly store all past data to detect or quantify change, and certainly we cannot use data from the future to make decisions in the present

Notion of Change in Stream

There is some similarity to the vast field of time series analysis, where data also consists of a sequence of timestamped items. In time series analysis, however, the analysis process is often assumed to be offline, with batch data, and without the requirements for low memory and low processing time per item inherent to streams.

In contrast, most of the work in streaming does not necessarily assume that change occurs in predictable ways, or has trends. **Change may be arbitrary**. The task is to build models describing how the world behaves right now, given what we are observing right now.

Notion of Change in Stream

What do we mean exactly when we say that a data stream changes or evolves?

It cannot mean that the items we observe today are not exactly the same as those that we observed yesterday. A more reasonable notion is that statistical properties of the data change more than what can be attributed to chance fluctuations

Notion of Change in Stream

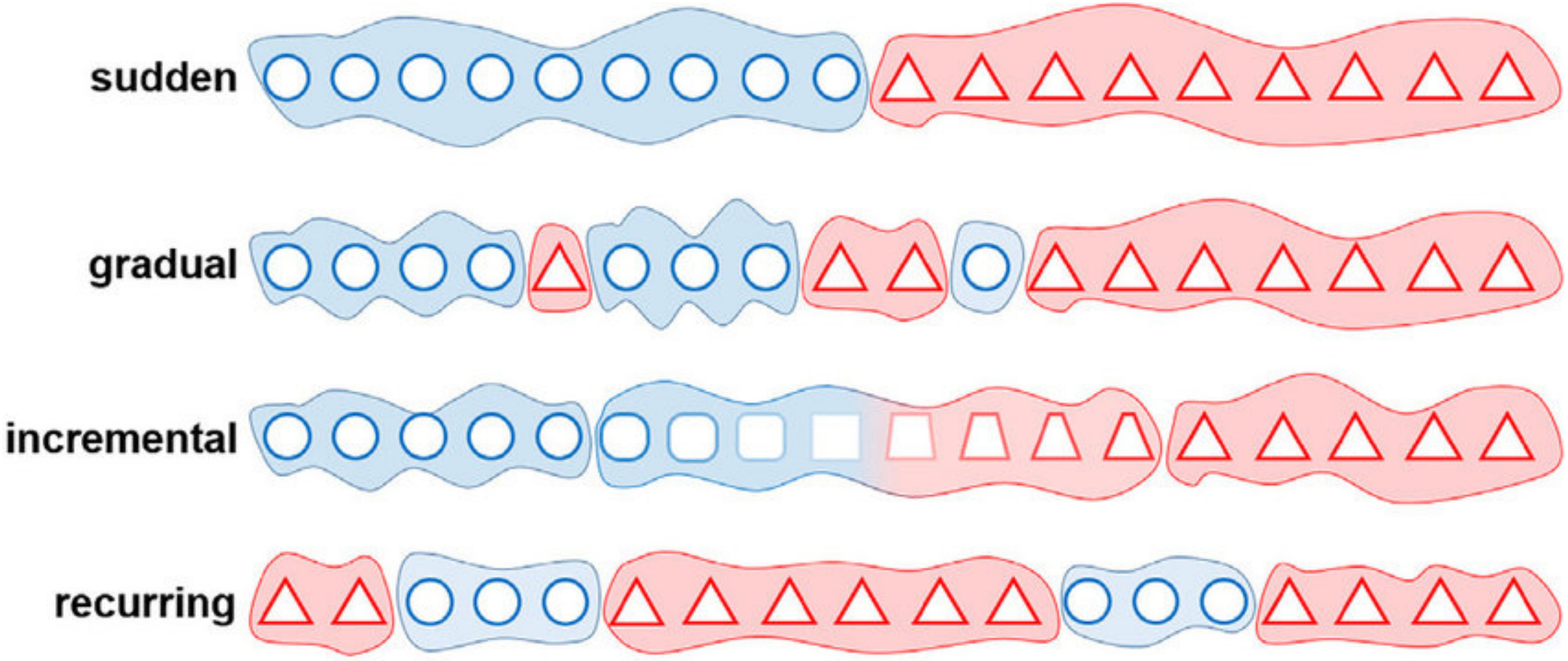
To make this idea precise, it helps to assume that the data is in fact the result of a random process that at each time generates an item according to a probability distribution that is used at that exact time, and that may or may not be the same that is used at any other given time

- There is no change when this underlying generating distribution remains stationary
- **Change occurs whenever it varies from one time step to the next**

Notion of Change in Stream

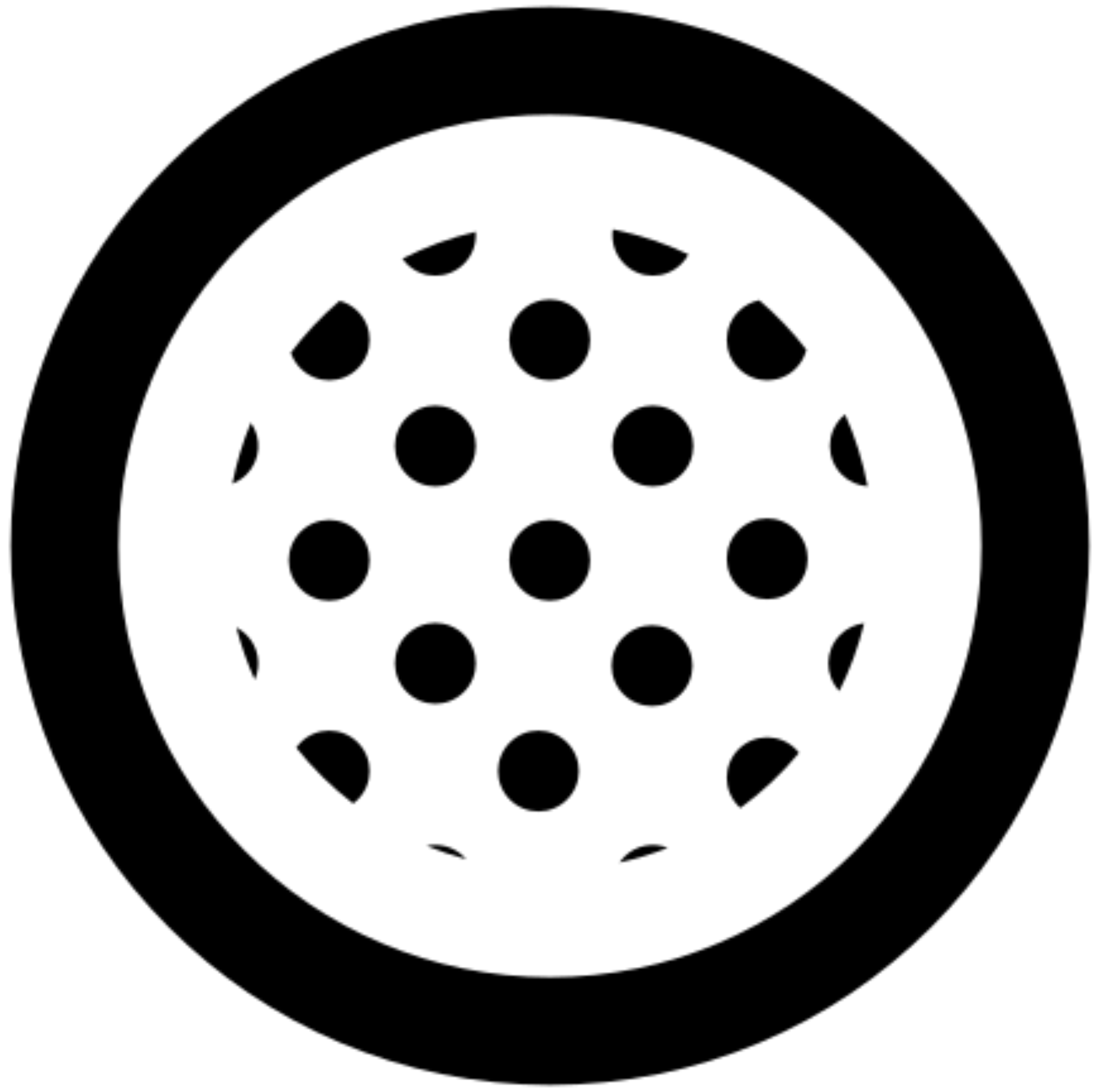
Although changes in the item distribution may be arbitrary, it helps to name a few generic types, which are not exclusive within a stream. The naming is unfortunately not consistent throughout the literature. In fact, change in general is often called ***concept drift*** in the literature

Notion of Change in Stream



Krawczyk, Bartosz & Cano, Alberto. (2018). Online Ensemble Learning with Abstaining Classifiers for Drifting and Noisy Data Streams. Applied Soft Computing. 68. 677-692. 10.1016/j.asoc.2017.12.008.

Notion of Change in Stream



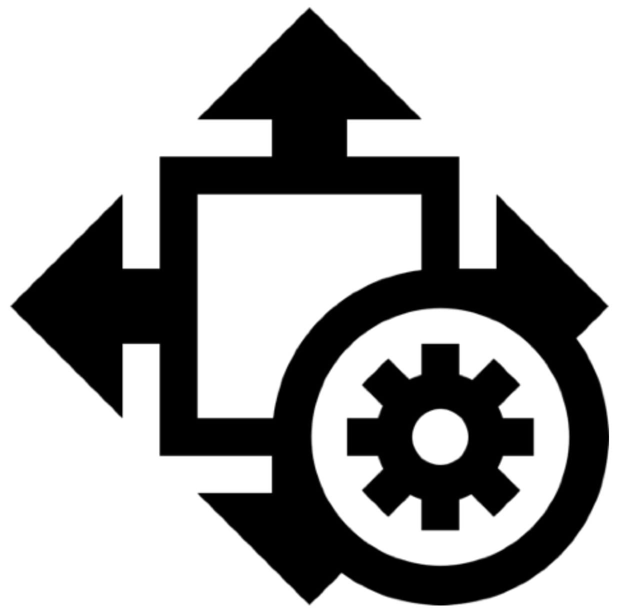
We should also **distinguish the notions of outliers and noise from that of distribution change.** Distinguishing true change from transient outliers and from persistent noise is one of the challenges in data stream mining and learning.

Requirements:

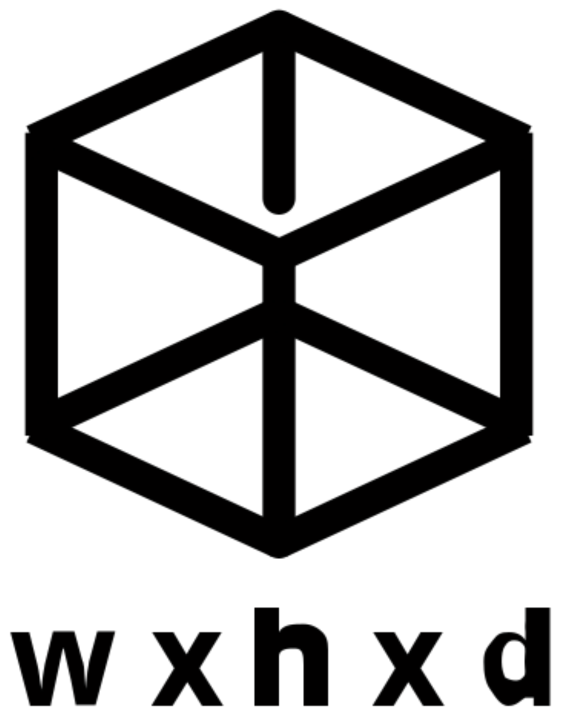
- Detect change in the stream (and adapt the models, if needed) as soon as possible
- At the same time, be robust to noise and outliers
- Operate in less than instance arrival time

Notion of Change in Stream

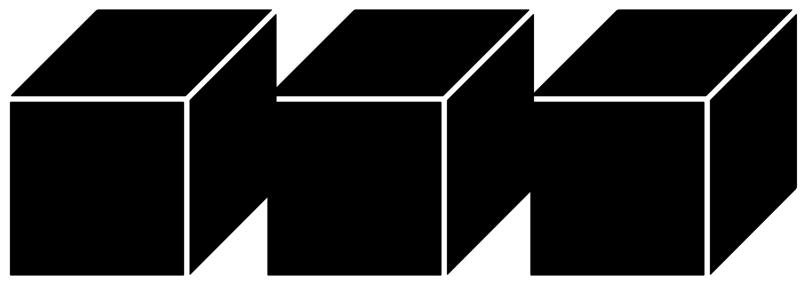
Change management strategies can be roughly grouped into three families, or a combination thereof



Adaptive estimators



Create models that are adapted or rebuilt



Ensemble methods

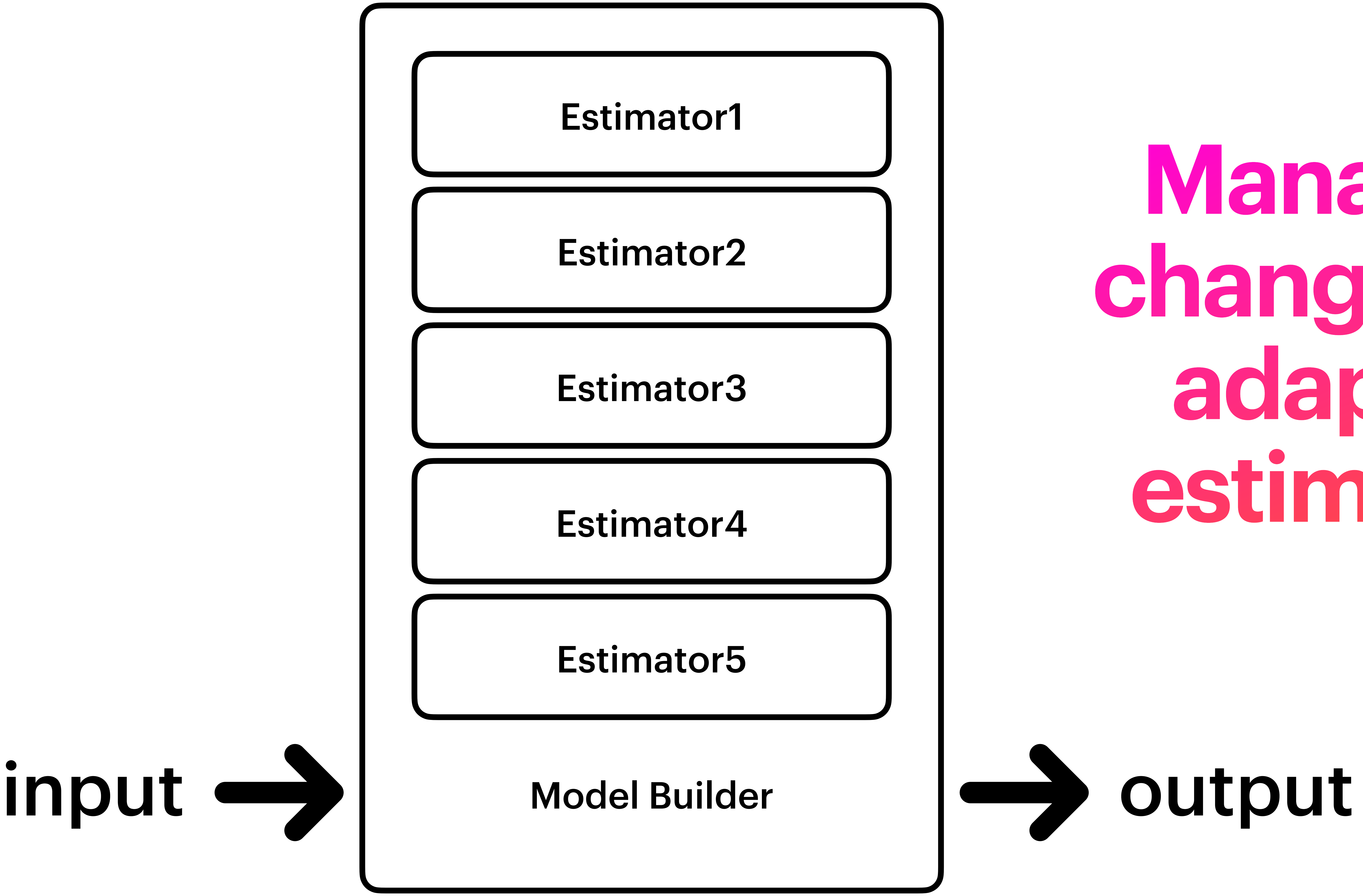
Notion of Change in Stream

The first strategy relies on the fact that many model builders work by monitoring a set of statistics from the stream and then combining them into a model. These statistics may be counts, absolute or conditional probabilities, correlations between attributes, or frequencies of certain patterns, among others.

Examples of such algorithms are Naive Bayes, which keeps counts of co-occurrences of attribute values and class values, and the perceptron algorithm, which updates weights taking into account agreement between attributes and the outcome to be predicted.

This strategy works by having a dynamic estimator for each relevant statistic in a way that reflects its current value, and letting the model builder feed on those estimators.

Notion of Change in Stream

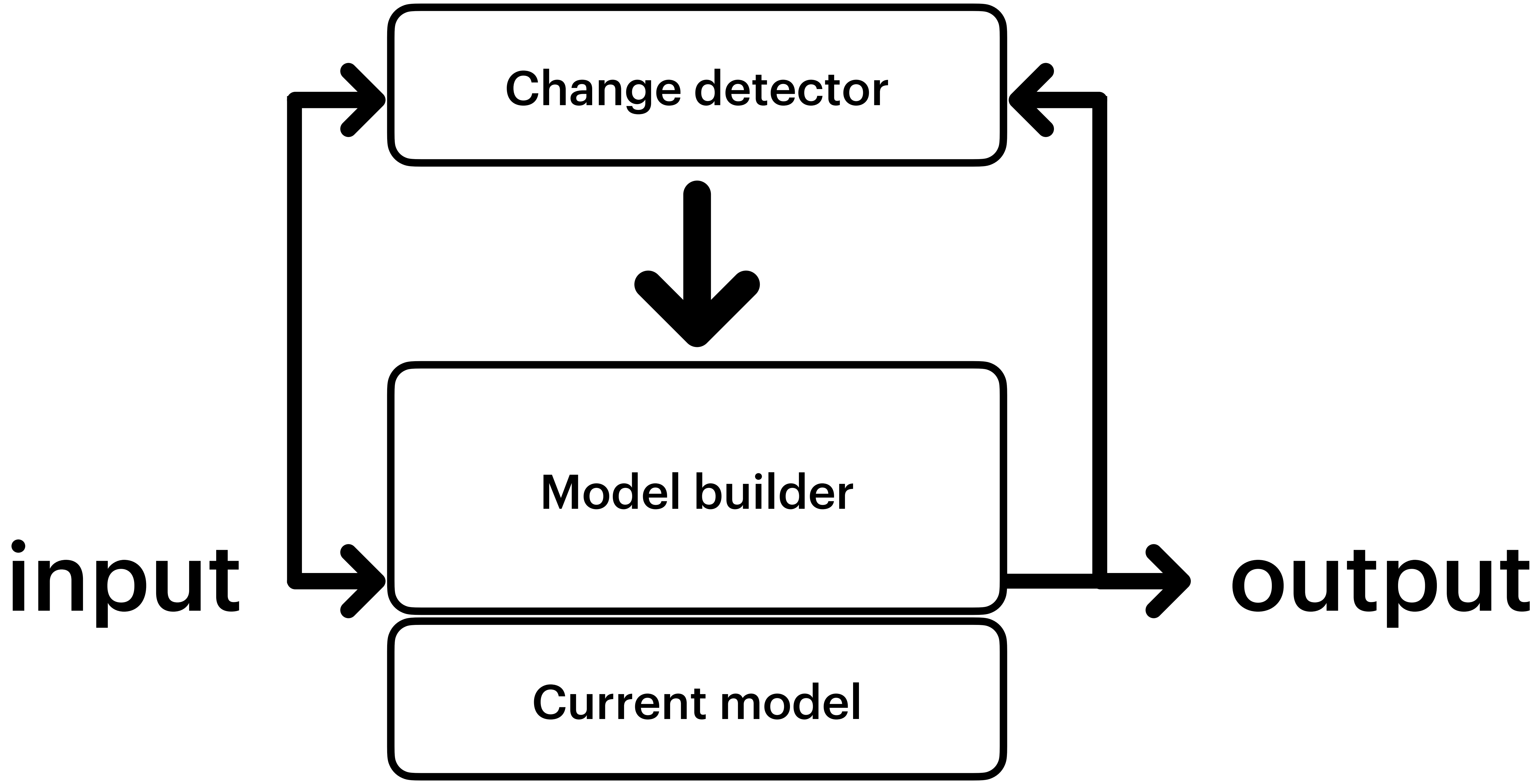


Managing change with adaptive estimators

Notion of Change in Stream

In the second strategy, one or more change detection algorithms run in parallel with the main model-building algorithm. When significant change in the stream is detected, they activate a revision algorithm

Notion of Change in Stream

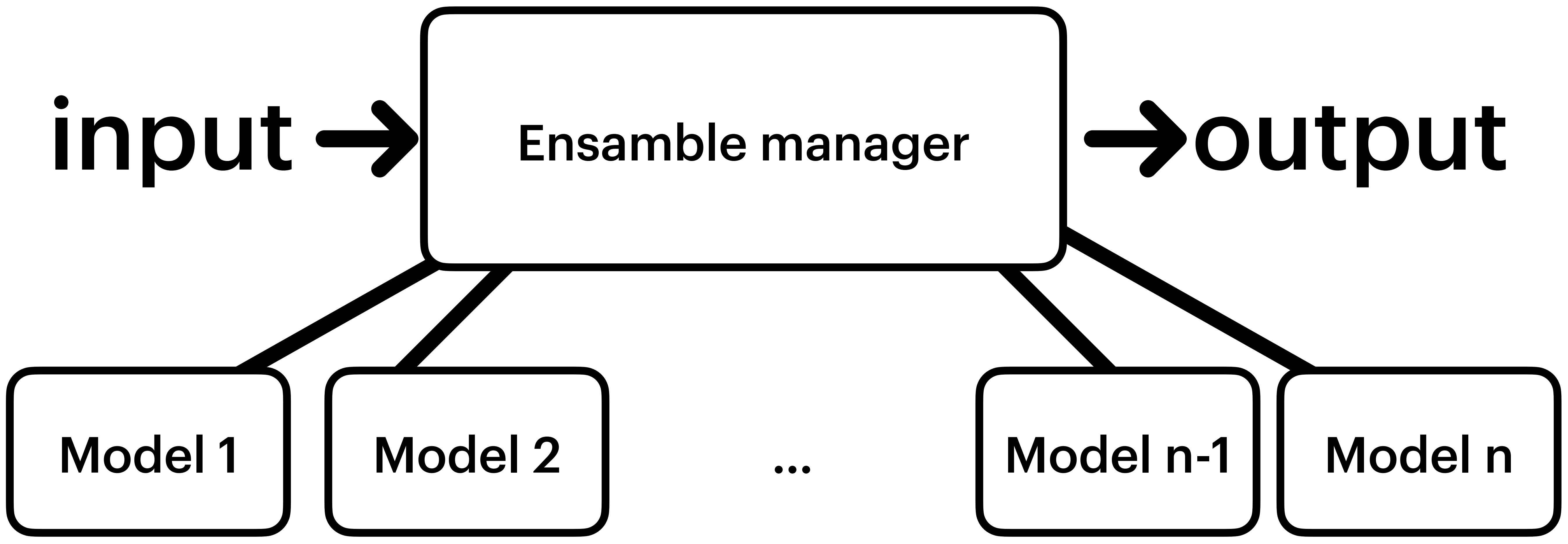


Managing change with explicit change detectors for model revision

Notion of Change in Stream

The third strategy is based on the idea of an *ensemble*, used to build complex classifiers out of simpler ones. A single or several model-building algorithms are called at different times, perhaps on different subsets of the data stream. An ensemble manager algorithm contains rules for creating, erasing, and revising the models in its ensemble, as well as for combining the predictions of the models into a single prediction.

Notion of Change in Stream



Managing change with model ensembles

Nice Tools

https://colab.research.google.com/drive/1tcFIuYKfnI1pHlkbp0-dgCmymBVrKCNE?usp=sharing