Stream Processing and Real-Time Data Systems: A Comparative Analysis for Big Data Applications

Daniele Cecca
Dr. Adriano Solidoro
Data-Driven Organizations and Management
2 December 2024

Contents

Daniele Cecca

Dr. Adriano Solidoro

Data-Driven Organizations and Management

2 December 2024

Stream Processing and Real-Time Data Systems: A Comparative Analysis for Big Data Applications

1. Introduction

Today, we have access to a vast amount of data streamed from various sources. Managing this data is challenging, not only due to its volume but also because of the velocity at which it is transmitted. It's not only important how we collect this data, but also how we process it in order to perform inferences that assist in the decision-making process.

As previously mentioned, we encounter different types of data. For this reason, this work will focus primarily on time series data, which is often confused with streaming data.

In the first part, I will explain what data and big data are, how to evaluate them, and the different types of problems we might face, with particular attention to the concepts of streaming data and time series.

In the second part, I will discuss the systems used to manage this data. I will compare different systems and approaches for managing these data, analyzing the differences and relationships between streaming and real-time data.

In the third part, I will explore the different types of models used to perform inference and the problems related to each method, delving deeper into some models from a mathematical perspective.

Thus, this paper aims to answer three main questions:

- What is data?
- How should data be managed?
- How can we make inferences from data?

Through this, I will provide an overview of the key components of a data project.

2. First Part

2.1 Definition of Big Data

Today, we no longer simply talk about ”data”; instead, we frequently encounter the term ”Big Data.” Big Data is characterized by the five V’s:

- Volume
- Velocity
- Variety
- Value
- Veracity

However, defining the quality of data solely based on these macro-level properties is insufficient. To evaluate and ensure data quality, it is necessary to focus on more detailed, microscopic aspects. Data quality refers to the degree to which data meets the needs and requirements of a business. High-quality data is essential for making informed decisions.

Four critical dimensions are commonly used to measure data quality:

- **Accuracy:** Refers to the correctness and reliability of the data. Accurate data ensures that insights and decisions derived from it are valid. Inaccurate data can lead to incorrect conclusions or actions.
- **Consistency:** Indicates that data remains uniform across different sources or systems, even when accessed at different times. Inconsistent data can result in discrepancies and confusion.
- **Completeness:** Refers to the extent to which all required data is available. Missing or incomplete data can undermine the quality of analysis and predictions.
- **Timeliness:** Refers to the freshness and relevance of data. Data must be available when needed and up-to-date to support decision-making. Delayed data can lead to missed opportunities or outdated insights.

However, as recent studies suggest(Ridzuan and Wan Zainon), these metrics are no longer sufficient for measuring data quality comprehensively. Many researchers propose new dimensions to address evolving needs. As a result, various categories have been developed to provide organizations with guidelines for selecting the appropriate dimensions to measure data quality effectively. Organizations must consider the specific use cases for their data and choose suitable data quality dimensions to ensure high-quality outcomes.

Data Quality Categories	**Data Quality Dimension**	**Definition**
Accessibility	Accessibility	Ease of access and retrieve the data
	Availability	Data is accessible and ready to use
	Security	Protect data from unauthorized access
Contextual	Appropriate Amount of Data	Quantity of data aligned with intended purpose
	Currency	Freshness of the data
	Relevance	Data is directly applicable and meaningful
	Validity	Accuracy of data in representing the intended concept
	Value Added	Additional insight derived from data
	Volatility	Frequency of data being updated
	Completeness	Data contains all necessary elements
	Timeliness	Data is usable within desired timeframe
Intrinsic	Correctness	Correspond to the real-world
	Redundancy	No duplicated data within the dataset
	Reputation	Reflecting level of confidence of the sources
	Trust	Degree of confidence on data
	Accuracy	Correctness in representing real-world
Representational	Interpretability	Ease of understanding the meaning of data
	Readability	Readability of data presentation
	Understandability	User can understand the meaning of the data representation
	Consistent	Uniform data across different resources

Table 1: Big data categories

2.2 Problems with Big Data

As previously mentioned, big data is complex to manage due to its diverse characteristics at both the micro and macro levels. For this reason, as highlighted in (Almeida et al.), we can encounter various challenges, which can be grouped into three main categories based on their nature:

- Statistical Problems
- Computational Problems
- Visualization Problems

2.2.1 Statistical Problems

Big data can introduce several statistical challenges, such as algorithm instability, noise accumulation, spurious correlations, incidental endogeneity, and measurement errors. These issues can compromise the validity of future inferences, leading to unreliable insights.

2.2.2 Computational Problems

As the name suggests, computational challenges are related to resource constraints in terms of time and space. These issues can result in high latency, low throughput, and limited scalability. Additionally, computational problems can affect model development. For example, in deep learning (DL), managing very large datasets often leads to architectures with an excessive number of parameters, making it prohibitively expensive to maintain and perform inference.

2.2.3 Visualization Problems

Visualization challenges arise primarily due to the high dimensionality of big data. This makes it difficult to represent data effectively in a visual format, hindering our ability to interpret and understand the data from a visual perspective.

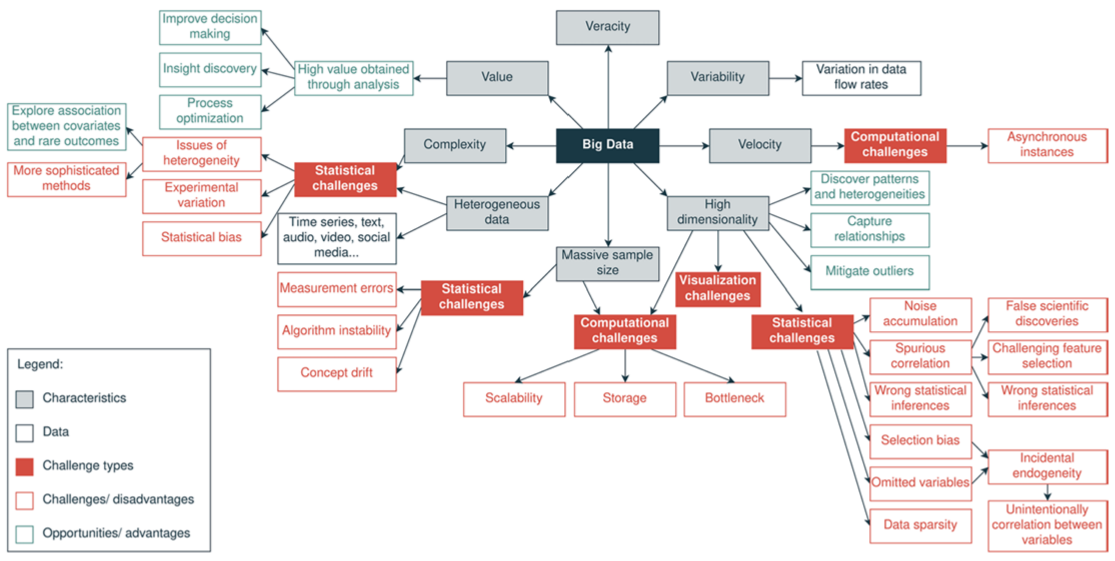


Figure 1: Big data overview

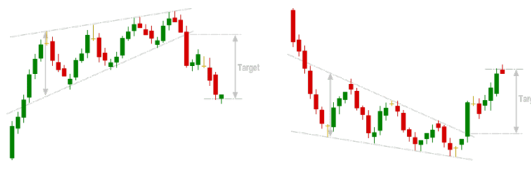
2.3 Time Series

So far, we have understood the concept of big data. For the following analysis, I want to focus on a specific type of data: time series. Often, people use the term ”time series” to refer to streaming data, but this is not entirely accurate.

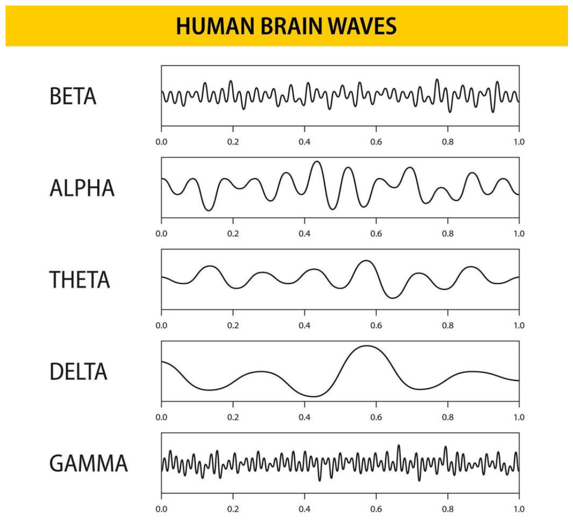
Time series data shares similarities with streaming data, as data arriving from streams often has a temporal component and a sequential order. However, not all data from streams qualify as time series, even if they have an associated timestamp.

A time series represents the evolution of one or more variables over time. It is a stochastic process indexed by time, which makes statistical properties significant. When we have only one variable, it is referred to as a univariate time series. When there are multiple variables, it is called a multivariate time series. Typically, the term ”time series” is used when dealing with univariate data.

Time series data is utilized across many domains, from finance, such as in trading systems, to healthcare, such as in the analysis of ECG (electrocardiogram) or EEG (electroencephalogram) data.



(a) Trading system



(b) EEG

Figure 2: Two images side by side

3. Second Part

3.1 Stream Processing Engines

Now that we understand the concept of big data, we need to examine the processes that occur before the inference phase. Big data applications are typically divided into five key components:

- **Data Sources:** Information originating from users, devices, or activity logs.
- **Data Storage:** Mechanisms such as databases or data warehouses used to store data.
- **Data Processing:** Frameworks that process raw data into usable formats.
- **Messaging Platform:** Systems responsible for transmitting data between components.
- **Presentation/Output:** Formats in which processed data is delivered, such as web or mobile applications or technical reports.

The messaging platform facilitates communication between modules, while the processing module often employs streaming frameworks to perform ETL (Extract, Transform, Load) operations.

3.1.1 Data Processing Overview

Data processing can be categorized based on the type of processing, the processing model, the windowing mechanism, and state retention.

Types of Processing

- **Batch Processing:** Processes bounded data streams with a defined beginning and end.
- **Stream Processing:** Processes unbounded data streams with no predetermined end. Data is processed continuously as it arrives.
- **Event-Based Processing:** Triggers alerts or actions when specific conditions are met.

Data Processing Models

- **At Most Once:** No guarantee that data is processed or persisted. Risk of data loss in case of failure. Prioritizes low latency over reliability.

- **At Least Once:** Ensures every data point is processed or persisted at least once. May result in duplicate processing or persistence.
- **Exactly Once:** Guarantees that data is processed or persisted only once.

Window Mechanisms for Stream Processing

- **Single-Pass:** Processes each new sample only once.
- **Sliding Window:** Uses a fixed-size window that moves over the data stream.
- **Tumbling Window:** Employs non-overlapping, sliding windows.
- **Session Window:** Similar to tumbling windows but allows gaps between sessions.
- **Landmark Window:** Grows a window starting from a specified point, updated periodically.
- **Damped Window:** Applies a fading mechanism, where recent data has greater weight, and older data gradually loses significance.

Stream-Based Processing Methods

- **Stateless Processing:** Does not retain information across processing steps.
- **Stateful Processing:** Retains state across processing steps.

Examples of stream processing platforms include Apache Spark, Apache Flink, Apache Storm, Apache Heron, Apache Samza, and Amazon Kinesis. However, the architectural details of these platforms are outside the scope of this work.

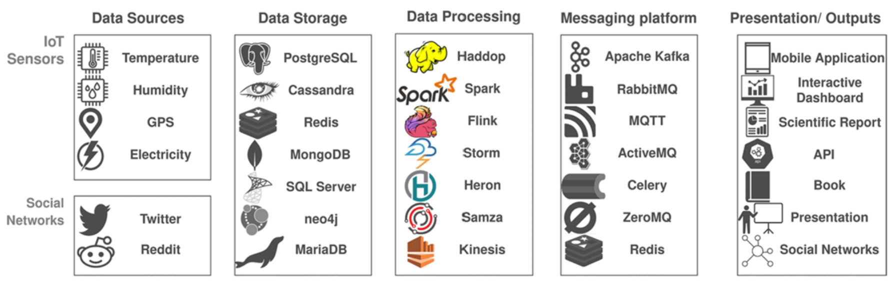


Figure 3: Data App components

3.2 Relation Between Real-Time and Streaming

Processing frameworks exhibit varying properties, making it challenging to select one without understanding their differences. Therefore, it is essential to choose the framework that best suits the specific use case, as suggests this article(Wang). To meet the real requirements of a business, we must clearly distinguish between real-time and streaming and understand their relationship. These terms are often used interchangeably in discussions, despite having distinct meanings.

Real-Time

Real-time systems must guarantee responses within specific time constraints, often referred to as deadlines. If real-time processing is not completed within the defined deadline relative to an event, it is considered a failure. Deadlines must be met regardless of system load.

There are typically two levels of real-time processing:

- **Real-Time:** Response times ranging from sub-milliseconds to seconds.
- **Near Real-Time:** Response times ranging from sub-seconds to hours.

Streaming

Streaming data refers to continuously generated data from various sources. Such data is processed incrementally using streaming techniques without requiring access to the entire dataset. Streaming is commonly associated with big data, where data is generated at high speed by multiple sources.

The distinction between real-time and streaming is subtle, particularly in scenarios like micro-batching.

In summary:

- Real-time focuses on practical constraints, emphasizing the maximum allowable response time to prevent system failures.
- Streaming data describes the continuous ingestion of data without strict requirements on response time.

A useful way to conceptualize the relationship is:

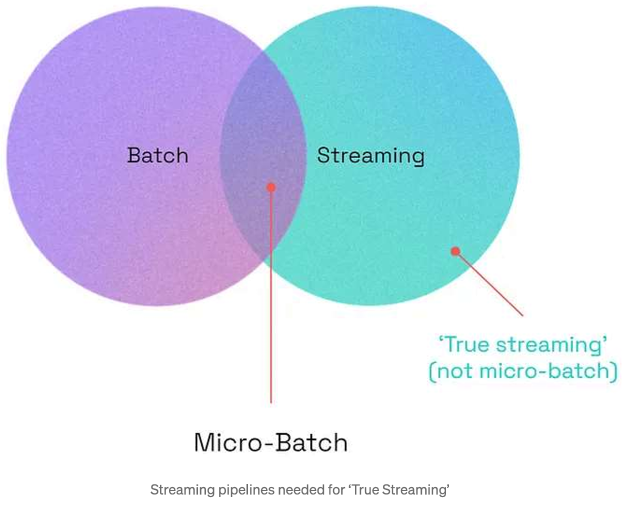


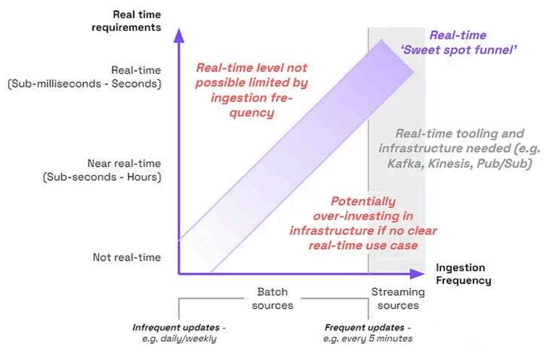
Figure 4: batch-microBatch-streaming

- **Streaming data is necessary but not sufficient for a real-time system.**

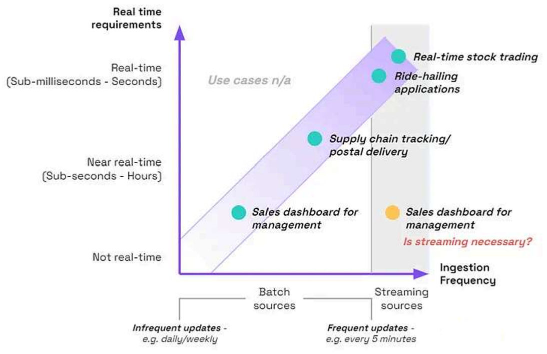
This insufficiency arises because a system may process data continuously using streaming pipelines but still have a latency longer than the response time required to classify it as real-time.

Conceptualizing the Relationship

The relationship between real-time and streaming can be visualized as a funnel:


Real time requirements
Real-time (Sub-milliseconds - Seconds)
Near real-time (Sub-seconds - Hours)
Not real-time
Real-time level not possible limited by ingestion frequency
Real-time 'Sweet spot funnel'
Real-time tooling and infrastructure needed (e.g. Kafka, Kinesis, Pub/Sub)
Potentially over-investing in infrastructure if no clear real-time use case
Ingestion Frequency
Batch sources
Streaming sources
Infrequent updates - e.g. daily/weekly
Frequent updates - e.g. every 5 minutes

(a) Real-time VS Ingestion Frequency


Real time requirements
Real-time (Sub-milliseconds - Seconds)
Near real-time (Sub-seconds - Hours)
Not real-time
Use cases n/a
Real-time stock trading
Ride-hailing applications
Supply chain tracking/ postal delivery
Sales dashboard for management
Sales dashboard for management
Is streaming necessary?
Ingestion Frequency
Batch sources
Streaming sources
Infrequent updates - e.g. daily/weekly
Frequent updates - e.g. every 5 minutes

(b) Real case scenarios

Figure 5: Two images side by side

- The area above the funnel represents scenarios that are technically impossible because batch ingestion frequencies cannot support true real-time processing.

- The area below the funnel indicates a potential over-investment in data infrastructure. Investing in streaming architecture without a clear real-time need can significantly increase complexity and resource requirements, often unnecessarily.

When designing a system, the opportunity cost must be carefully considered. Even if a model performs better in a real-time scenario, a batch environment might still be preferable depending on the trade-off between complexity and added value.

Example: Trade-Offs in System Design

A study (Kwieciński et al.) compared the performance of a recommendation system model in batch and real-time environments:

- **Model Not Retrained, User Recommendations Not Affected**
- **Model Not Retrained, User Recommendations Affected**

The results demonstrated that the outcomes in both scenarios were comparable. For instance, the study measured converted users—users who clicked on at least one recommended job ad and then applied for that job. Variant B showed a 10.14% higher percentage of converted users, suggesting that even without real-time capabilities, the performance was satisfactory.

Variant	**Users**	**Converted Users**	**% Converted Users**
A (batch)	92,835	3,731	4.02%
B (real-time)	92,194	4,081	4.43%
Difference	-0.69%	9.38%	10.14%

Table 2: Results

Takeaway

When deciding between real-time and batch systems, businesses must weigh the trade-off between complexity and added value. While real-time systems may offer better performance in specific scenarios, batch processing often provides sufficient results with lower complexity and resource requirements.

4. Third Part

To make a well-informed decision, it is not enough to simply observe the data; we must also perform inference by using forecasting models. Forecasting can be categorized based on the time horizon into near future, medium future, or distant future.

Forecasting models can generally be divided into four types:

- Historical
- Statistical
- Machine Learning-Based
- Deep Learning-Based

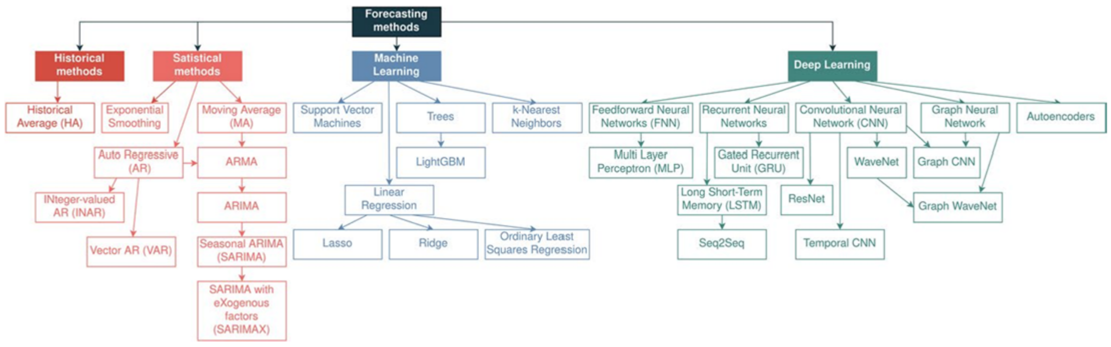


Figure 6: forecasting methods

Below, I present some of the most popular models from each type, explained from a mathematical perspective:

4.1 1. Historical Average (HA)

Explanation: The Historical Average(Institute) method involves calculating the average value of past data over a specific period and using this average to predict future values. It assumes that past trends will continue in a similar manner. This method is simple and works well when data exhibits little to no fluctuation.

Example: The data below provides the average historical returns of an index over a 5-year period. The data used is for educational purposes only and does not depict real-time historical data.

- December 31, 2015: 28%
- December 31, 2016: 18.7%

- December 31, 2017: 19.9%
- December 31, 2018: 23.1%
- December 31, 2019: 29.7%

Using a simple mean computation, the average historical return can be found by summing up all the returns and dividing the sum by the number of years (or periods). The calculation can be longer and more tedious, depending on the number of time periods used – e.g., 5 years or more.

Hence, the average historical return for the index based on the data provided above is calculated as:

$$\begin{aligned}
\text{Average Historical Return(s)} &= \frac{\text{Sum of Returns}}{\text{Number of Periods}} \\
&= \frac{28\% + 18.7\% + 19.9\% + 23.1\% + 29.7\%}{5} \\
&= \frac{119.4\%}{5} \\
&= 23.9\%
\end{aligned}$$

4.2 2. Autoregressive Integrated Moving Average (ARIMA)

Explanation: An ARIMA model(students of PhD in Data Science Batch 2023 at the Asian Institute of Management) is a set of models that explains a time series using its own previous values given by the lags (**AutoRegressive**) and lagged errors (**Moving Average**), while considering stationarity corrected by differencing (opposite of **Integration**). In other words, ARIMA assumes that the time series is described by autocorrelations in the data rather than trends and seasonality.

In this context, we define trends and seasonality as the following:

- **Trend:** A time series has a trend if there is an overlying long-term increase or decrease in the data, which is not necessarily linear.
- **Seasonality:** A time series has seasonality when it is affected by seasonal factors such as the time of the year or the day of the week. The seasonality of data is apparent as there is a fixed frequency of pattern occurrence.

. Thus, it combines three components:

- **AR: Autoregression.** A model that uses the dependent relationship between an observation and some number of lagged observations.
- **I: Integrated.** The use of differencing of raw observations (e.g., subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- **MA: Moving Average.** A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of **ARIMA(p, d, q)** where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used:

- p: The number of lag observations included in the model, also called the *lag order* (deals with the window of X_t).
- d: The number of times that the raw observations are differenced, also called the *degree of differencing* (deals with the order of differencing of X_t).
- q: The size of the moving average window, also called the *order of moving average* (deals with residuals).

Given this, the general case of **ARIMA(p, d, q)** can be written as:

$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$$

Or in words:

Predicted X_t = Constant + Linear combination of Lags of X (up to p lags) + Linear Combination of Lagged forecast errors (up to q lags). Provided that the time-series is already differenced (up to d terms) to ensure stationarity.

Special Case	**ARIMA Model**
White noise	ARIMA(0,0,0)
Random walk	ARIMA(0,1,0) with no constant
Random walk with drift	ARIMA(0,1,0) with a constant
Autoregression	ARIMA(p,0,0)
Moving average	ARIMA(0,0,q)

Table 3: Special Cases of ARIMA

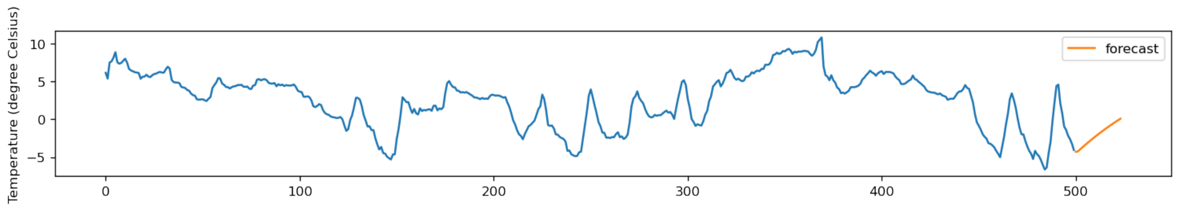


Figure 7: Forecasting ARIMA

4.3 3. Support Vector Machine (SVM)

Explanation: SVM(MathWorks) is a supervised machine learning algorithm used for both classification and regression tasks. It works by finding the optimal hyperplane in a higher-dimensional space that separates data points into different classes or predicts continuous values. SVM is effective in high-dimensional spaces.

Digging deeper into the mathematical details, support vector machines fall under a class of machine learning algorithms called kernel methods where the features can be transformed using a kernel function. Kernel functions map the data to a different, often higher dimensional space with the expectation that the classes are easier to separate after this transformation, potentially simplifying a complex non-linear decision boundaries to linear ones in the higher dimensional, mapped feature space. In this process, the data doesn't have to be explicitly transformed, which would be computationally expensive. This is commonly known as the kernel trick.

Type of SVM	**Mercer Kernel**	**Description**
Gaussian or Radial Basis Function (RBF)	$K(x_1, x_2) = \exp\left(-\frac{\|x_1-x_2\|^2}{2\sigma^2}\right)$	One class learning. σ is the width of the kernel.
Linear	$K(x_1, x_2) = x_1^\top x_2$	Two class learning.
Polynomial	$K(x_1, x_2) = (x_1^\top x_2 + 1)^\rho$	ρ is the order of the polynomial.
Sigmoid	$K(x_1, x_2) = \tanh\left(\beta_0 x_1^\top x_2 + \beta_1\right)$	It is a Mercer kernel for certain β_0 and β_1 values only.

Table 4: Types of SVM and Their Mercer Kernels

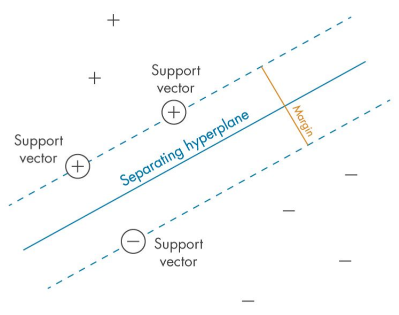


Figure 8: SVM

4.4 4. Recurrent Neural Network (RNN)

Explanation: A deep learning approach for modeling sequential data is **Recurrent Neural Networks (RNNs)**(Goodfellow et al.). They are called recurrent because they perform the same task for every element of the sequence, with the output at each step depending on the previous computation.

The structure of an RNN can be divided into the following components:

1. Input

The input sequence is defined as:

$$\mathbf{x}(t) = \{x(1), x(2), \ldots, x(\tau)\}, \quad \text{where } t = 1, \ldots, \tau$$

At each time step t, the input $x(t)$ is fed into the network.

2. Hidden State

The hidden state $\mathbf{h}(t)$ represents the network's "memory" at time t. It is computed based on the current input $\mathbf{x}(t)$ and the hidden state from the previous time step $\mathbf{h}(t-1)$:

$$\mathbf{h}(t) = a\big(U\mathbf{x}(t) + V\mathbf{h}(t-1)\big)$$

where:

- a is the activation function (e.g., tanh or ReLU).

- U is the weight matrix for input-to-hidden connections.
- V is the weight matrix for hidden-to-hidden connections.

3. Weights

The RNN utilizes the following shared weight matrices across all time steps:

- U: Input-to-hidden connection weights.
- V: Hidden-to-hidden recurrent connection weights.
- W: Hidden-to-output connection weights.

4. Output

The output of the network at time step t, denoted as $\mathbf{o}(t)$, Often o(t) is subject to activation functions, especially when the network contains further layers downstream

5. Training Process

Similar to other neural networks, RNNs are trained using two main steps:

1. **Forward pass**: Computes the outputs and intermediate hidden states for all time steps.
2. **Backward pass (Backpropagation Through Time, BPTT)**: Computes the gradients of the loss function with respect to the network's weights and updates them using an optimization algorithm (e.g., gradient descent).

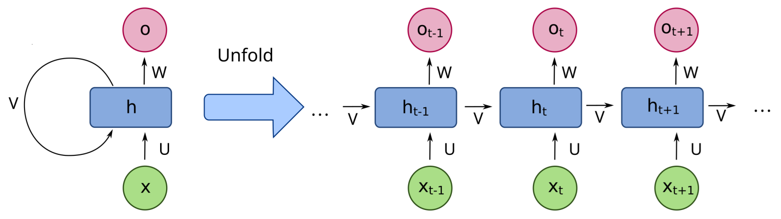


Figure 9: RNN

4.5 Comparing Approaches

Although deep learning-based models have gained significant popularity due to the prevalence of big data, each approach has its own advantages and disadvantages.

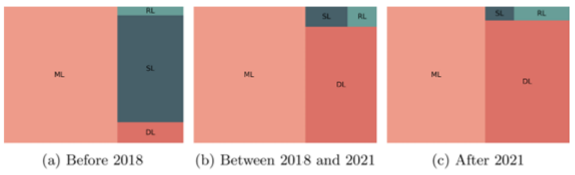


Figure 10: Evolution methods popularity

4.5.1 Statistical Methods:

Advantages:

- Explainable and interpretable.
- Robust for short-term predictions.
- Perform well in contexts where short-term accuracy is key.

Disadvantages:

- Generally unsuitable for long-term forecasting.

4.5.2 Deep Learning Methods:

Advantages:

- Powerful for capturing complex patterns and long-term dependencies.
- Effective with large datasets.

Disadvantages:

- Lack of interpretability.
- Higher computational cost and complexity.

The choice of model depends on the specific context and requirements of the task. For instance, while statistical methods may outperform deep learning models in short-term contexts due to their robustness and simplicity, deep learning models might excel in tasks requiring long-term predictions or when working with large, complex datasets.

5. Conclusion

In today's world, the availability of vast amounts of data presents both opportunities and challenges. While data in its raw form may hold limited value for organizations and society, its true potential lies in its transformation into actionable knowledge. By leveraging robust data management systems, processing frameworks, and forecasting models, businesses can extract meaningful insights to improve decision-making processes.

This paper explored three fundamental questions:

- **What is data?** Through an understanding of data and its quality dimensions, we can evaluate its relevance and utility.
- **How should data be managed?** With a comparison of batch and real-time systems and a detailed analysis of processing frameworks, we underscored the importance of tailoring infrastructure to specific business needs.
- **How can inferences be made?** A review of forecasting models demonstrated the varied approaches—ranging from statistical methods to advanced deep learning techniques—that can be employed depending on the time horizon and context.

The choice of methods and systems must always consider trade-offs such as complexity, cost, and added value. While cutting-edge deep learning models are popular in the era of big data, simpler methods like statistical techniques often provide better interpretability and robustness in specific scenarios.

Ultimately, the value of data lies not in its volume but in the insights it offers. By choosing appropriate tools and approaches, organizations can navigate the complexities of big data and harness its power for meaningful advancements.

Works Cited

Almeida, Ana, et al. “Time series big data: a survey on data stream frameworks, analysis, and algorithms.” *Journal of Big Data*, vol. 10, 2023, Open Access under a Creative Commons Attribution 4.0 International License, p. 83. https://doi.org/10.1186/s40537-023-00760-1.

Goodfellow, Ian, et al. *Deep Learning.* Chapter on Recurrent Neural Networks, MIT P, 2016, www.deeplearningbook.org/contents/rnn.html.

Institute, Corporate Finance. “Historical Returns in Capital Markets,” 2024, corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/historical-returns/.

Kwieciński, Robert, et al. “Comparison of Real-Time and Batch Job Recommendations.” *IEEE Access*, vol. 11, Mar. 2023, Received 29 January 2023, accepted 16 February 2023, published 27 February 2023, pp. 20553–59. https://doi.org/10.1109/ACCESS.2023.3249356.

MathWorks. “Support Vector Machine (SVM) Overview,” 2024, nl.mathworks.com/discovery/support-vector-machine.html.

Ridzuan, Fakhitah, and Wan Mohd Nazmee Wan Zainon. “A Review on Data Quality Dimensions for Big Data.” *Procedia Computer Science*, vol. 234, Mar. 2024, Presented at the Seventh Information Systems International Conference (ISICO 2023), pp. 341–48. https://doi.org/10.1016/j.procs.2024.03.008.

Students of PhD in Data Science Batch 2023 at the Asian Institute of Management. “AutoRegressive Integrated Moving Average (ARIMA),” 2023, phdinds-aim.github.io/time_series_handbook/01_AutoRegressiveIntegratedMovingAverage/01_AutoRegressiveIntegratedMovingAverage.html.

Wang, Richard. “Streaming data vs. real-time data: What’s the difference?” *Medium*, 2023, Accessed via Medium. medium.com/@richardwang.