UNIT 2

MINING DATA STREAMS

Data Stream is a continuous, fast-changing, and ordered chain of data transmitted at a very high speed. It is an ordered sequence of information for a specific interval.

The sender's data is transferred from the sender's side and immediately shows in data streaming at the receiver's side. Streaming does not mean downloading the data or storing the information on storage devices.

A data stream is an existing, continuous, ordered (implicitly by entrance time or explicitly by timestamp) chain of items. It is unfeasible to control the order in which units arrive, nor it is feasible to locally capture stream in its entirety.

It is enormous volumes of data, items arrive at a high rate.

Sources of Data Stream

There are so many sources of the data stream, and a few widely used sources are listed below:

- Internet traffic
- Sensors data
- Real-time ATM transaction
- Live event data
- Call records
- Satellite data
- Audio listening
- Watching videos
- Real-time surveillance systems
- Online transactions

Types of Data Streams :

A data stream is a(possibly unchained) sequence of tuples. Each tuple comprised of a set of attributes, similar to a row in a database table.

1. Transactional data stream –
It is a log interconnection between entities

- Credit card – purchases by consumers from producer
- Telecommunications – phone calls by callers to the dialed parties
- Web – accesses by clients of information at servers

2. Measurement data streams –

- Sensor Networks – a physical natural phenomenon, road traffic
- IP Network – traffic at router interfaces
- Earth climate – temperature, humidity level at weather stations

Examples of Stream Sources-

1. **Sensor Data –**
   In navigation systems, sensor data is used. Imagine a temperature sensor floating about in the ocean, sending back to the base station a reading of the surface temperature each hour.
2. **Image Data –**
   Satellites frequently send down-to-earth streams containing many terabytes of images per day. Surveillance cameras generate images with lower resolution than satellites, but there can be numerous of them, each producing a stream of images at a break of 1 second each.
3. **Internet and Web Traffic –**
   A bobbing node in the center of the internet receives streams of IP packets from many inputs and paths them to its outputs. Websites receive streams of heterogeneous types. For example, Google receives a hundred million search queries per day.

Characteristics of Data Streams :

1. Large volumes of continuous data, possibly infinite.
2. Steady changing and requires a fast, real-time response.
3. Data stream captures nicely our data processing needs of today.
4. Random access is expensive and a single scan algorithm
5. Store only the summary of the data seen so far.
6. Maximum stream data are at a pretty low level or multidimensional in creation, needs multilevel and multidimensional treatment.

Applications of Data Streams :

1. Fraud perception
2. Real-time goods dealing
3. Consumer enterprise
4. Observing and describing on inside IT systems

Advantages of Data Streams :

- This data is helpful in upgrading sales
- Help in recognizing the fallacy
- Helps in minimizing costs
- It provides details to react swiftly to risk

Disadvantages of Data Streams :

- Lack of security of data in the cloud
- Hold cloud donor subordination
- Off-premises warehouse of details introduces the probable for disconnection

DATA STREAM MODEL AND ARCHITECTURE

Handle the never-ending stream of events natively

Batch processing tools need to gather batches of data and integrate the batches to gain a meaningful conclusion. By reducing the overhead delays associated with batching events, organizations can gain instant insights from huge amounts of stream data.

Real-time data analytics and insights

Stream processing processes and analyzes data in real-time to provide up-to-the-minute data analytics and insights. This is very beneficial to companies that need real-time tracking and streaming data analytics on their processes.

It also comes in handy in other scenarios such as detection of fraud and data breaches and machine performance analysis

Simplified data scalability

Batch processing systems may be overwhelmed by growing volumes of data, necessitating the addition of other resources, or a complete redesign of the architecture. On the other hand, modern streaming data architectures are hyper-scalable, with a single stream processing architecture capable of processing gigabytes of data per second.

Detecting patterns in time-series data

Detection of patterns in time-series data, such as analyzing trends in website traffic statistics, requires data to be continuously collected, processed, and analyzed. This process is considerably more complex in batch processing as it divides data into batches, which may result in certain occurrences being split across different batches.

Increased ROI

The ability to collect, analyze and act on real-time data gives organizations a competitive edge in their respective marketplaces. Real-time analytics makes organizations more responsive to customer needs, market trends, and business opportunities.

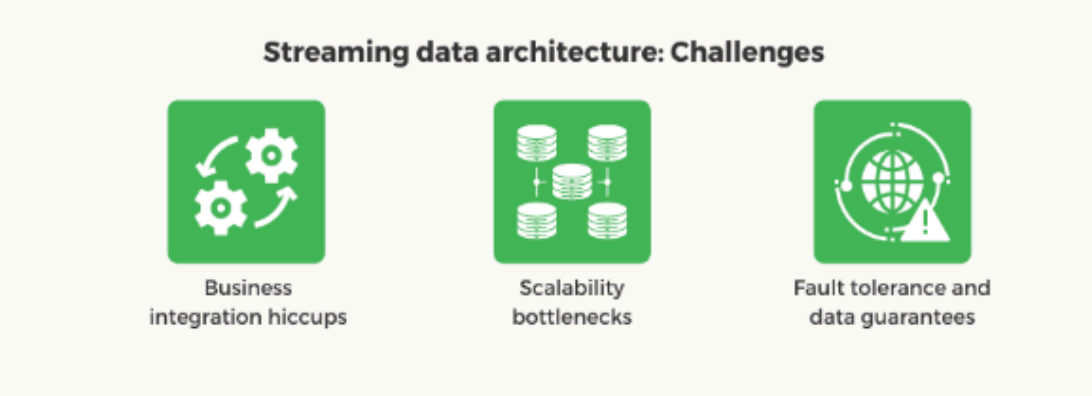
Improved customer satisfaction

Organizations rely on customer feedback to gauge what they are doing right and what they can improve on. Organizations that respond to customer complaints and act on them promptly generally have a good reputation.

Losses reduction

In addition to supporting customer retention, stream processing can prevent losses as well by providing warnings of impending issues such as financial downturns, data breaches, system outages, and other issues that negatively affect business outcomes. With real-time information, a business can mitigate, or even prevent the impact of these events.

DATA STREAM ARCHITECTURE:



Business Integration hiccups

Most organizations have many lines of business and applications teams, each working concurrently on its own mission and challenges. For the most part, this works fairly seamlessly for a while until various teams need to integrate and manipulate real-time event data streams.

Organizations can federate the events by multiple integration points so that the actions of one or more teams don't inadvertently disrupt the entire system.

Scalability bottlenecks

As an organization grows, so do its datasets. When the current system is unable to handle the growing datasets, operations become a major problem. For example, backups take much longer and consume a significant number of resources. Similarly, rebuilding indexes, reorganizing historical data, and defragmenting storage becomes more time-consuming and resource-intensive operations.

To solve this, organizations can check the production environment loads. By test-running the expected load of the system using past data before implementing it, they can find and fix problems.

Fault Tolerance and data guarantees

These are crucial considerations when working with stream processing or any other distributed system. Since data comes from different sources in varying volumes and formats, an organization's systems must be able to stop disruptions from any point of failure and effectively store large streams of data.

STREAM COMPUTING

Stream computing is a distributed computing paradigm that processes data as it arrives in real-time, without the need for batch processing. This approach has become increasingly popular in recent years due to its ability to solve real-world problems more efficiently and effectively.

1. Real-Time Data Processing
2. Fraud Detection in Financial Systems
3. IoT Data Analysis
4. Social Media Analytics
5. Advanced Analytics for Healthcare

1. Real-Time Data Processing

One of the most significant benefits of stream computing is its ability to process data in real-time. With traditional batch processing, data must be collected, sorted, and processed all at once, which can take considerable time and resources. By contrast, stream computing can process data as it arrives, allowing businesses to quickly respond to changing market conditions or customer needs.

For example, financial institutions can use stream computing to analyze real-time market data and make instantaneous trading decisions based on changing market conditions. Similarly, e-commerce companies can use stream computing to analyze customer behavior in real-time, enabling them to personalize their marketing campaigns and improve customer engagement.

2. Fraud Detection in Financial Systems

Fraud detection is a critical problem in the financial industry, where false transactions can have significant consequences for both customers and businesses. Stream computing can help address this problem by analyzing large volumes of transaction data in real-time and flagging any suspicious activity for further investigation.

3. IoT Data Analysis

The Internet of Things (IoT) has revolutionized the way we interact with technology, but managing the vast amounts of data generated by IoT devices can be challenging. Stream computing can help solve this problem by processing data from multiple IoT devices in real-time, providing insights into device performance, user behavior, and overall system efficiency.

For example, manufacturers can use stream computing to monitor production equipment in real-time, detecting potential issues before they cause downtime or costly repairs. Similarly, logistics companies can use stream computing to optimize shipping routes and reduce transportation costs by analyzing real-time traffic data and weather conditions.

4. Social Media Analytics

Social media platforms generate vast amounts of data every day, making it difficult for businesses to analyze and derive insights from this information effectively. Stream computing can help solve this problem by processing social media data in real-time, providing businesses with a comprehensive understanding of their customers' preferences, opinions, and behavior patterns.

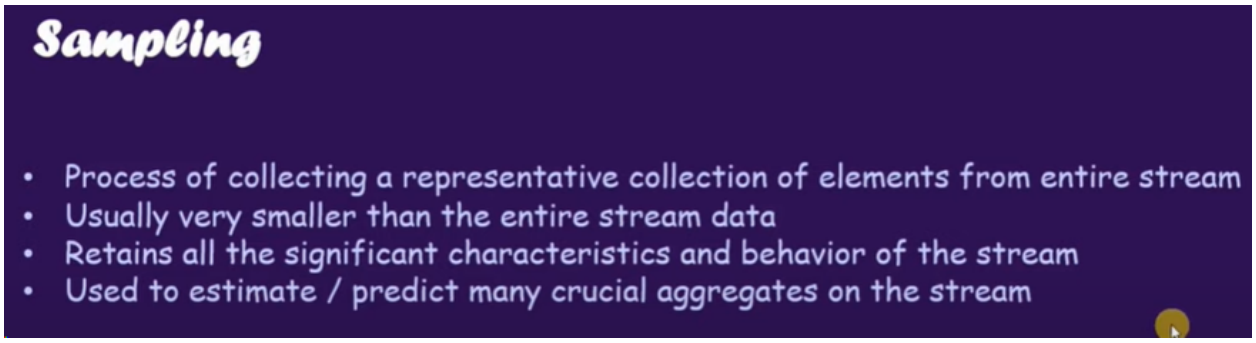
By using natural language processing (NLP) algorithms and sentiment analysis techniques, stream computing can extract valuable insights from social media data, helping businesses better understand their target audience and develop more effective marketing strategies.

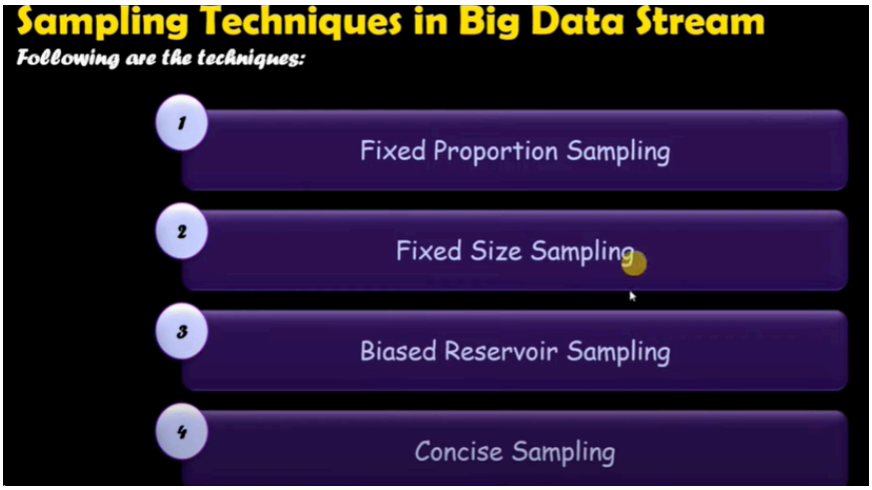
5. Advanced Analytics for Healthcare

Healthcare is a critical industry that relies heavily on accurate data analysis to improve patient outcomes and reduce healthcare costs. Stream computing can help address this problem by processing large volumes of medical data in real-time, providing healthcare providers with actionable insights into patient health status and treatment effectiveness.

For example, hospitals can use stream computing to monitor patients' vital signs in real-time, detecting potential health issues before they become serious problems. Similarly, insurance companies can use stream computing to analyze claims data in real-time, identifying fraudulent claims and reducing their overall costs.

SAMPLING FROM A DATA STREAM





Since we can not store the entire stream, one obvious approach is to store a sample

• Two different problems:
- (1) Sample a fixed proportion of elements in the stream (say 1 in 10)
- (2) Maintain a random sample of fixed size over a potentially infinite stream

Sampling a Fixed Proportion Problem 1: Sampling fixed proportion Scenario: Search engine query stream -Stream of tuples: (user, query, time)

- Answer questions such as: How often did a user run the same query in a single days

- Have space to store 1/10th of query stream

Naïve solution:

- Generate a random integer in [0..9] for each query Play 00- Store the query if the integer is 0, otherwise discard

Solution: Sample Users

Solution:

• Pick 1/10th of users and take all their searches in the sample

• Use a hash function that hashes the user name or user id uniformly into 10 buckets

Maintaining a fixed-size sample

• Problem 2: Fixed-size sample

Suppose we need to maintain a random sample S of size exactly s tuples

-E.g., main memory size constraint

Why? Don't know length of stream in advance Suppose at time n we have seen n items

- Each item is in the sample S with equal prob. s/n

How to think about the problem: say s = 2 Stream: a xcy zkcdeg...

At n= 5, each of the first 5 tuples is included in the sample S with equal prob.

At n= 7, each of the first 7 tuples is included in the sample S with equal prob.

Impractical solution would be to store all the n tuples seen so far and out of them pick s at random.

Sliding Windows

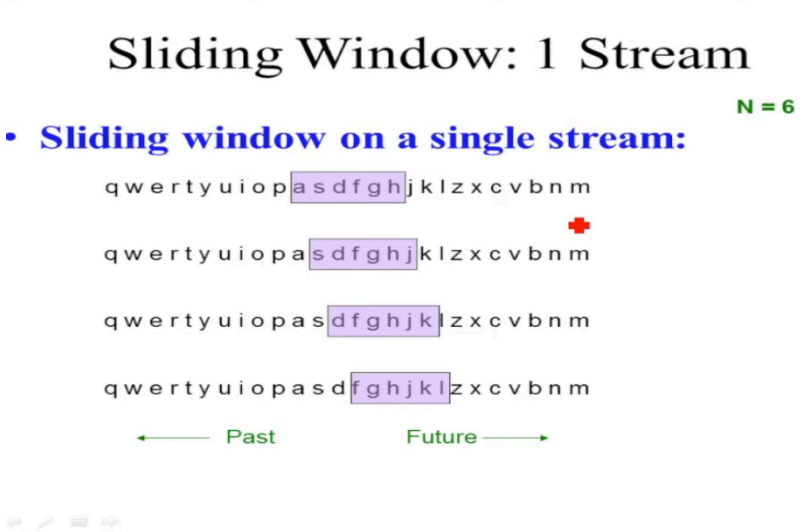
A useful model of stream processing is that queries are about a window of length N- the N most recent elements received

• Interesting case: N is so large that the data cannot be stored in memory, or even on disk - Or, there are so many streams that windows for all cannot be stored

• Amazon example:

- For every product X we keep 0/1 stream of whether that product was sold in the n-th transaction

- We want answer queries, how many times have we sold X in the last k sales



FILTERING

Data filtering is the process of examining a dataset to exclude, rearrange, or apportion data according to certain criteria. For example, data filtering may involve finding out the total number of sales per quarter and excluding records from last month. IT professionals often use data filtering to complete their responsibilities and assist others within their organization in data examination.

A set S containing m values – A whitelist of a billion non-spam email addresses

• Memory with n bits. – Say 1 GB memory

• Goal: Construct a data structure that can efficient check whether a new element is in S

– Returns TRUE with probability 1, when element is in S

– Returns FALSE with high probability (1-ε), when element is not in S

Bloom Filter

Consider a set of hash functions {h1 , h2 , .., hk }, hi [1, n]◊: S

Initialization:

• Set all n bits in the memory to 0.

Insert a new element ‘a’:

• Compute h1 (a), h2 (a), …, hk (a). Set the corresponding bits to 1.

Check whether an element ‘a’ is in S:

• Compute h1 (a), h2 (a), …, hk (a).

If all the bits are 1, return TRUE. Else, return FALSE

Analysis

If a is in S:

• If h1 (a), h2 (a), …, hk (a) are all set to 1.

• Therefore, Bloom filter returns TRUE with probability 1

If a not in S:

• Bloom filter returns TRUE if each hi(a) is 1 due to some other element

Pr[bit j is 1 after m insertions] = 1 – Pr[bit j is 0 after m insertions]

= 1 – Pr[bit j was not set by k x m hash functions]

= 1 – (1 – 1/n)km

Pr[Bloom filter returns TRUE] = {1 – (1 – 1/n)km} k } ≈ (1 – e -km/n) k

Example

• Suppose there are m = 109 emails in the white list.

• Suppose memory size of 1 GB (8 x 109 bits)

k = 1

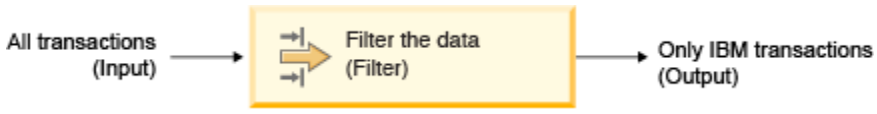
• Pr[Bloom filter returns TRUE | a not in S] = 1 – e -m/n = 1 – e -1/8 = 0.1175

k = 2

• Pr[Bloom filter returns TRUE | a not in S] = (1 – e -2m/n) 2 = (1 – e -1/4) 2 ≈ 0.0493

FILTERING LARGE DATA SETS

In this example, the stream processing application needs to filter the stock transaction data for IBM® transaction records. You use the Filter operator to extract relevant information from potentially large volumes of data. As shown, the input for the Filter operator is all the transactions; the output is only the IBM transactions.



In SPL terminology, a *tuple* is a unit of data for an operator. In this example, each transaction record is a tuple. A *stream* is a sequence of tuples. In this example, "All transactions" is an input stream, and "Only IBM transactions" is an output stream. An *operator* transforms an input stream into an output stream. In this example, the operator filters the data by processing each tuple from the input stream and submitting the tuple to the output stream only if it is an IBM transaction.

Suppose that each transaction record (that is, a tuple) contains the following four fields. Each field in a record corresponds to an *attribute* in a tuple.

Field	Type	Name	Description
1	string	ticker	ticker name
2	string	date	transaction date
3	string	time	transaction time
4	decimal	price	trading price

Table 1. Attributes for a transaction record

In SPL code, the transaction record can be represented by the following TransactionRecord type:

```
type
  TransactionRecord = rstring ticker,
                      rstring date,
                      rstring time,
                      decimal64 price;
```

where rstring is a sequence of raw bytes that supports string processing when the character encoding is known, and decimal64 is the IEEE 754 decimal 64-bit floating point number.

The SPL code for the Filter operator is shown. To read the code, you say that the output stream is produced by operating on the input stream. In this case, you say that IBMTransactions is produced by filtering AllTransactions.

```
  stream<TransactionRecord> IBMTransactions = Filter(AllTransactions) {
    param
      filter : ticker == "IBM";
  }
```

In general, the Filter operator receives tuples from an input stream and submits a tuple to the output stream only if the tuple satisfies the criteria that are specified by the filter parameter.

In this example, the Filter operator performs the following steps:

1. Receives a tuple from the input stream (AllTransactions).
2. If the value of the ticker attribute is IBM, it submits the tuple to the output stream (IBMTransactions).
3. Repeats Steps 1 to 2 until all the tuples from the input stream are processed.

The Filter operator requires that the type of the output stream is the same as the type of the input stream. The type of the output stream is specified by the *tupleType* in the "stream<*tupleType*> *OutputStream* = Filter(*InputStream*)" declaration. In this example, the type of the output and input streams is TransactionRecord.

COUNTING DISTINCT ELEMENTS

INPUT:
• A stream S of elements from a domain D
– A stream of logins to a website
– A stream of URLs browsed by a user
• Memory with n bits

OUTPUT
• An estimate of the number of distinct elements in the stream
– Number of distinct users logging in to the website
– Number of distinct URLs browsed by the user
Consider a hash function h:D□{0,1}L which uniformly hashes elements in the stream to L bit values
IDEA: The more distinct elements in S, the more distinct hash values are observed.
Define: Tail0 (h(x)) = number of trailing consecutive 0's
– Tail0 (101001) = 0
– Tail0 (101010) = 1
– Tail0 (001100) = 2
– Tail0 (101000) = 3
– Tail0 (000000) = 6 (=L)

COUNTING ONENESS IN A WINDOWS:

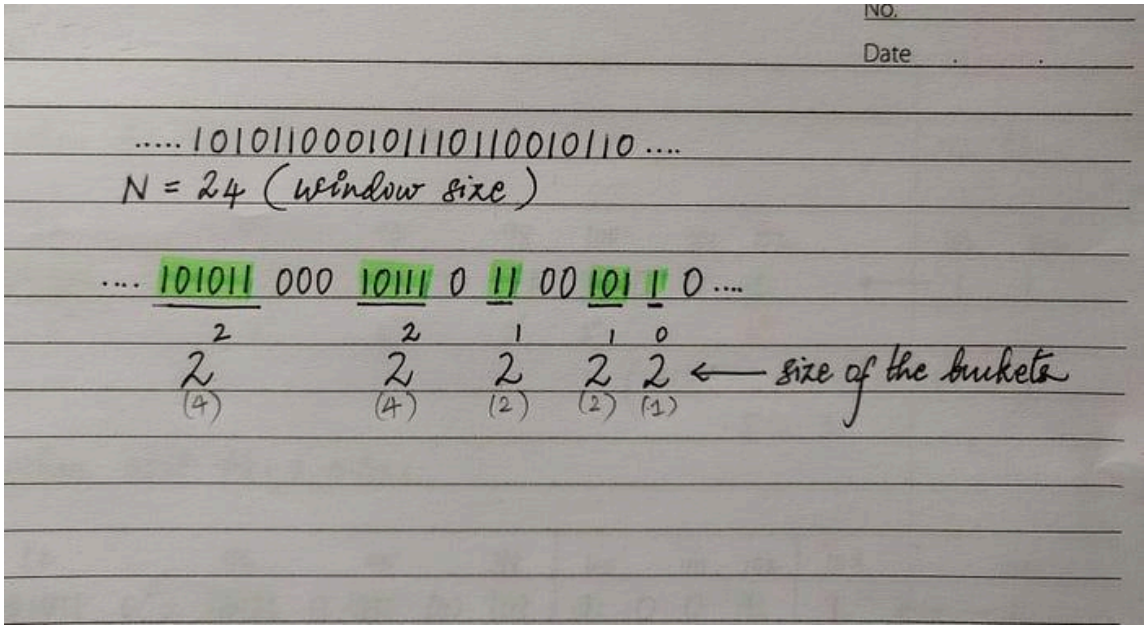
DGIM algorithm (*Datar-Gionis-Indyk-Motwani Algorithm)*

Designed to find the number 1's in a data set. This algorithm uses $O(\log^2 N)$ bits to represent a window of N bit, allows to estimate the number of 1's in the window with and error of no more than 50%. So this algorithm gives a 50% precise answer.

In DGIM algorithm, each bit that arrives has a timestamp, for the position at which it arrives. if the first bit has a timestamp 1, the second bit has a timestamp 2 and so on.. the positions are recognized with the window size N (the window sizes are usually taken as a multiple of 2).The windows are divided into buckets consisting of 1's and 0's.

RULES FOR FORMING THE BUCKETS:

1. The right side of the bucket should always start with 1. (if it starts with a 0,it is to be neglected) E.g. · 1001011 → a bucket of size 4 ,having four 1's and starting with 1 on it's right end.
2. Every bucket should have at least one 1, else no bucket can be formed.
3. All buckets should be in powers of 2.
4. The buckets cannot decrease in size as we move to the left. (move in increasing order towards left)
5. Estimating the number of 1's and counting the buckets in the given data stream.



Decaying windows:

- The decaying window algorithm not only tracks the most recurring elements in an incoming data stream, but also discounts any random spikes or spam requests that might have boosted an element's frequency.
- In a decaying window, you assign a score or weight to every element of the incoming data stream. Further, you need to calculate the aggregate sum for each distinct element by adding all the weights assigned to that element. The element with the highest total score is listed as trending or the most popular.
    - Assign each element with a weight/score.
    - Calculate aggregate sum for each distinct element by adding all the weights assigned to that element.

Advantages of Decaying Window Algorithm :-

- Sudden spikes or spam data is taken care.
- New element is given more weight by this mechanism, to achieve right trending output.

RTAP

• Refers to finding meaningful patterns in data at the actual time of receiving

• Real-Time Analytics Platform (RTAP) analyses the data, correlates, and predicts the outcomes in the real time.

It applies logic and mathematics to data to provide insights for making better decisions quickly. For some use cases, real time simply means the analytics is completed within a few seconds or minutes after the arrival of new data and thus supporting instant decision making.

Manages and processes data and helps timely decision-making

- Helps to develop dynamic analysis applications • Leads to evolution of business intelligence
- The platforms connect the data sources for better analytics and visualization
- Real-time analytics allows businesses to react without delay. They can seize opportunities or prevent problems before they happen.

Benefits:

1. **Tracking customer data**. See the latest time-sensitive customer data and craft an immediate response. Real-time analytics reveals when and why your customers behave as they do, and how to optimize their satisfaction.

2. **Cost efficiencies.** Real-time analytics can help improve profitability by saving money across the organization in areas like hiring and retention, employee engagement, and of course, reducing the workload of the IT department.

3. **Faster response time**. A sudden market fluctuation can mean big opportunities for businesses. Real-time analytics can ensure that you get ahead of situations that might cost money or conversely, could be a big money-maker.

4. **Real-time testing**. With immediate answers at their fingertips, businesses can forecast with confidence and optimize their data to find the best options. Split-testing or A/B testing can be carried out with ease to make big decisions clearer.

RTAP Applications

1. Fraud detection systems for online transactions
2. Log analysis for understanding usage pattern
3. Click analysis for online recommendations
4. Social Media Analytics
5. Push notifications to the customers for location-based advertisements for retail
6. Action for emergency services such as fires and accidents in an industry
7. Any abnormal measurements require immediate reaction in healthcare monitoring
8. Targeting individual customers in retail outlets with promotions and incentives, while the customers are in the store and next to the merchandise.

What Is Real-Time Sentiment Analysis?

Real-time sentiment analysis is a technique that uses machine learning (ML) so it can identify subjective opinions expressed in live social and online feeds, and derive sentiment from text for audience insights. Applications of real-time sentiment analysis are predominantly in the areas of monitoring brand mentions for market intelligence, tracking keyword mentions for research purposes, or to deter cyberbullying, check negative comments, and such.

Why Do We Need Real-Time Sentiment Analysis?

Real-time sentiment analysis has several applications for brand and customer analysis. These include the following.

1. Live social feeds from video platforms like Instagram or Facebook

2. Real-time sentiment analysis of text feeds from platforms such as Twitter. This is immensely helpful in prompt addressing of negative or wrongful social mentions as well as threat detection in cyberbullying.
3. Live monitoring of Influencer live streams.
4. Live video streams of interviews, news broadcasts, seminars, panel discussions, speaker events, and lectures.
5. Live audio streams such as in virtual meetings on Zoom or Skype, or at product support call centers for customer feedback analysis.
6. Live monitoring of product review platforms for brand mentions.
7. Up-to-date scanning of news websites for relevant news through keywords and hashtags along with the sentiment in the news.

STOCK MARKET PREDICTION

The stock market is the collection of markets where stocks and other securities are bought and sold by investors. Publicly traded companies offer shares of ownership to the public, and those shares can be bought and sold on the stock market. Investors can make money by buying shares of a company at a low price and selling them at a higher price. The stock market is a key component of the global economy, providing businesses with funding for growth and expansion. It is also a popular way for individuals to invest and grow their wealth over time.

Importance of Stock Market

Importance	Description
Capital Formation	It provides a source of capital for companies to raise funds for growth and expansion.
Investment Opportunities	Investors can potentially grow their wealth over time by investing in the stock market.
Economic Indicators	The stock market can indicate the overall health of the economy.
Job Creation	Publicly traded companies often create jobs and contribute to the economy's growth.
Corporate Governance	Shareholders can hold companies accountable for their actions and decision-making processes.
Risk Management	Investors can use the stock market to manage their investment risk by diversifying their portfolio.
Market Efficiency	The stock market helps allocate resources efficiently by directing investments to companies with promising prospects.

Big data can be used to predict stock market trends by leveraging data from a variety of sources to identify patterns and correlations. By analyzing the data, it can be determined which stocks are likely to outperform or underperform in the market. This data can then be used to make informed decisions about which stocks to buy, sell or hold.

What Types of Data Can Be Used?

When using big data to predict stock market trends, a variety of data sources can be used. This includes data from financial news and reports, market analysis, macroeconomic data, company financials, and social media.