University of Camerino

School of Science and Technology

Master degree in Computer Science (Class LM-18)

OPENSEARCH

An analysis of the tool's key features

Group members

Sofia Scattolini

Email: sofia.scattolini@studenti.unicam.it

Ismael Liuzzi

Email: ismael.liuzzi@studenti.unicam.it

Unicam Supervisor

Dr. Massimo Callisto De Donato

A.Y. 2024/2025

Abstract

This report explores **OpenSearch**, an open source search, analytics, and observability platform originally derived from Elasticsearch.

This study explores its key capabilities, focusing on four main aspects: **Observability**, the practice of monitoring and analyzing complex distributed systems to ensure reliability and performance; its integration with **OpenTelemetry**, a standardized framework for collecting and processing telemetry data; the **features and usability** of OpenSearch for search and analytics; and its support for **AI/ML applications**.

The research methodology is based primarily on a comprehensive review of the official documentation, analyzing the architecture, features, and integration potential of OpenSearch. The report discusses the theoretical foundations of observability within OpenSearch, the principles behind OpenTelemetry, and the potential for leveraging OpenSearch in AI/ML workflows.

Contents

1 Introduction **1**
- 1.1 What is OpenSearch? . . . 1
  - 1.1.1 Key Features . . . 1
  - 1.1.2 Brief history . . . 2
- 1.2 Objectives of the Study . . . 2
- 1.3 Structure of the Report . . . 3

2 OpenSearch Architecture and Setup **5**
- 2.1 How OpenSearch works: architecture and components . . . 5
  - 2.1.1 Clusters . . . 5
  - 2.1.2 Nodes . . . 5
  - 2.1.3 Indices and Documents . . . 5
  - 2.1.4 Shards . . . 6
- 2.2 Getting started . . . 6

3 Observability **9**
- 3.1 Understanding Observability . . . 9
- 3.2 Observability fundamentals . . . 10
- 3.3 Benefits of Observability . . . 10

4 OpenTelemetry: A Standard for Observability **13**
- 4.1 Introduction to OpenTelemetry . . . 13
- 4.2 How does it work? . . . 14
- 4.3 What is Otel used for? . . . 15

5 The Role of AI and ML in OpenSearch **17**
- 5.1 Machine Learning Features . . . 17
  - 5.1.1 Key ML Features in OpenSearch . . . 17
- 5.2 AI Capabilities . . . 18
  - 5.2.1 Key AI Features in OpenSearch . . . 18
- 5.3 ML Frameworks and Integration with OpenSearch . . . 18
  - 5.3.1 Native ML Support in OpenSearch . . . 18
  - 5.3.2 Integration with External ML Frameworks . . . 18

6 Conclusions **19**

List of Figures

1.1 OpenSearch Logo 1
1.2 From ElasticSearch to OpenSearch 2

2.1 OpenSearch Response 7
2.2 OpenSearch Dashboard Home 8

3.1 Benefits of Observability 11

4.1 Otel Flow 14

1. Introduction

In today's digital world, a large amount of data is created every second. The challenge is to search, analyze, and understand this data to gain useful insights. OpenSearch, a key part of the ELK stack (including OpenSearch, Logstash, and OpenSearch Dashboard), has become a powerful search and analytics engine known for its speed, scalability, and flexibility. It supports everything from full-text search to real-time data analysis, handling large amounts of information and providing essential support to businesses.

1.1 What is OpenSearch?

Figura 1.1: OpenSearch Logo

OpenSearch is an open source, distributed search and analytics engine. Developed as an open source, community driven project, OpenSearch helps businesses store, search and analyze massive amounts of data at scale. [1]

One of its key components is OpenSearch Dashboards, an integrated visualization tool that allows users to explore and analyze their data in real time. OpenSearch also provides built-in support for observability, security, and machine learning, making it a comprehensive solution for search and analytics workloads.

1.1.1 Key Features

OpenSearch offers a variety of features that facilitate the indexing, search, and management of large datasets quickly and in near real-time.

Some of the key features include:

- **Integrated search engine**: OpenSearch has built-in features such as full text querying, autocomplete, and recommendations based on relationships within the data which helps users find what they are looking for more effectively.
- **Analytics and machine learning**: You can use OpenSearch in multiple analytics solutions such as events analytics, trace analytics, and machine learning, which uses algorithms such as anomaly detection and data clustering.

- **Built-in security**: OpenSearch supports secure authentication and encryption protocols out of the box and includes a security plugin that provides more fine-grained security configurations.
- **Monitoring**: OpenSearch includes a number of monitoring and alerting features which allow you to detect security threats in real time.
- **Observability**: You can use OpenSearch to create observability applications through the OpenSearch Dashboard. You can also use it to schedule, export and share reports.

1.1.2 Brief history

Figura 1.2: From ElasticSearch to OpenSearch

Elasticsearch, originally developed by Elastic NV, became one of the most widely used open source search and analytics engines, distributed under the Apache 2.0 license. However, in early 2021, Elastic NV announced a licensing change, transitioning Elasticsearch and Kibana to a dual SSPL (Server Side Public License) and Elastic License, effectively restricting their open-source nature.

In response, Amazon Web Services (AWS), along with other contributors, decided to fork the last Apache 2.0 licensed version of Elasticsearch (7.10) and continue its development under the fully open-source Apache 2.0 license. This led to the creation of OpenSearch, a truly open, community-driven search and analytics suite designed to provide a viable alternative to Elasticsearch.

1.2 Objectives of the Study

This study aims to provide an in-depth exploration of OpenSearch, focusing on four key areas:

1. **Observability**: Understanding how OpenSearch facilitates log management, metric collection, and distributed tracing for monitoring complex systems.
2. **Integration with OpenTelemetry**: Examining OpenSearch's compatibility with OpenTelemetry for collecting, processing, and analyzing telemetry data.
3. **Feature Set and Usability**: Evaluating OpenSearch's capabilities as a search and analytics engine, including data indexing, querying, and visualization.
4. **AI/ML Support**: Investigating OpenSearch's built-in machine learning functionalities and their real-world applications.

1.3 Structure of the Report

This report is structured as follows:

- *Chapter 2*: Introduces OpenSearch's core architecture and provides a deployment guide.
- *Chapter 3*: Observability in OpenSearch is discussed, including log management, metrics collection, and tracing support.
- *Chapter 4*: Explores OpenTelemetry integration, highlighting its role in enhancing OpenSearch observability.
- *Chapter 5*: Investigates the AI/ML capabilities of OpenSearch and evaluates their real-world applications.
- *Chapter 6*: Presents conclusions, challenges, and recommendations for future work.

Following this structured approach, this report aims to offer a comprehensive and practical evaluation of OpenSearch's capabilities and its relevance in modern data-driven environments.

2. OpenSearch Architecture and Setup

2.1 How OpenSearch works: architecture and components

OpenSearch is built on a **cluster-based architecture**, ensuring high availability and fault tolerance. It consists of several core components that work together to manage data storage, indexing, and search operations. These components are conceptually similar to those found in Elasticsearch.

2.1.1 Clusters

An OpenSearch cluster is a collection of one or more nodes that work together to store, search, and analyze data. Clusters provide redundancy and load balancing, ensuring that the failure of a single node does not compromise the system's overall performance. Each cluster has a unique identifier, known as the **cluster name**, which is used for managing and connecting to the cluster.

2.1.2 Nodes

Nodes are individual instances of OpenSearch that make up a cluster. Each node stores data and participates in indexing and search processes. OpenSearch nodes can have different roles:

- **Master nodes**: Manage cluster state and metadata, including index creation, deletion, and node coordination.
- **Data nodes**: Store indexed data and handle query execution, including full-text searches and aggregations.
- **Coordinator nodes** (formerly client nodes): Distribute search and indexing requests across the cluster but do not store data themselves. They improve load balancing in large deployments.

2.1.3 Indices and Documents

An **index** in OpenSearch is a logical namespace that groups related documents, similar to a database in a relational database management system (RDBMS). Each document within an index is structured in JSON format.

A **document** represents a basic unit of data storage. For example, in an e-Commerce application, a single product listing can be stored as a document.

2.1.4 Shards

Shards are the fundamental storage units in OpenSearch. Each index is divided into smaller partitions called **shards**, which are distributed across multiple nodes to enhance performance and scalability.

There are two types of shards:

1. **Primary shards**: Store the original data and handle the write operations.
2. **Replica shards**: Duplicates of primary shards, which provide redundancy and improve query performance by balancing the load.

By default, OpenSearch creates `5` primary shards per index, but this setting can be customized. The system automatically distributes shards across nodes to maintain high availability in case of failure.

2.2 Getting started

To start using OpenSearch, it is recommended to deploy it in a local or cloud-based environment. One of the most efficient ways to install OpenSearch is via Docker Compose, which simplifies deployment by managing OpenSearch and its dependencies in isolated containers.

Follow these steps to setup OpenSearch using Docker Compose:

1. **Install Docker and Docker Compose**: Ensure that both Docker and Docker Compose are installed on your system. Docker can be downloaded from the official website: Docker Get Started. If Docker Compose is not already installed, follow the instructions here: Docker Compose Install Guide.
2. **Obtain the Docker-Compose File**: Download or clone the `docker-compose.yml` file, which defines the necessary services to run OpenSearch. This file is available in the OpenSearch GitHub repository and official documentation.
3. **Understand the Configuration**: The `docker-compose.yml` file typically defines a minimal OpenSearch cluster with two nodes:
   - `opensearch-node1`: Handles indexing and search operations.
   - `opensearch-node2`: Works alongside node1 to provide redundancy and scalability.

   This setup ensures:
   - `Fault tolerance`: If one node fails, the other continues operations.
   - `Scalability`: Workloads are distributed across multiple nodes, improving performance.
   - `Advanced search capabilities`: Supports full-text search, log analysis, and machine learning-based anomaly detection.

4. **Launch OpenSearch**: Navigate to the directory containing the `docker-compose.yml` file and execute the following command to start the cluster in detached mode:

   ```
   docker-compose up -d
   ```

   This will download the necessary Docker images and launch OpenSearch along with OpenSearch Dashboards.

   By default:

   - OpenSearch is accessible at `http://localhost:9200`
   - OpenSearch Dashboards is accessible at `http://localhost:5601`

5. **Verify OpenSearch is Running**: To check if OpenSearch is active, visit `http://localhost:9200` in a browser or run:

   ```
   curl -X GET "http://localhost:9200"
   ```

   A successful response should return JSON output similar to:


```
{
    "name": "opensearch-node1",
    "cluster_name": "opensearch-cluster",
    "cluster_uuid": "8qHNNTCSRlWke8As-EOl8w",
    "version": {
        "distribution": "opensearch",
        "number": "2.18.0",
        "build_type": "tar",
        "build_hash": "99a9a81da366173b0c2b963b26ea92e15ef34547",
        "build_date": "2024-10-31T19:08:04.231254959Z",
        "build_snapshot": false,
        "lucene_version": "9.12.0",
        "minimum_wire_compatibility_version": "7.10.0",
        "minimum_index_compatibility_version": "7.0.0"
    },
    "tagline": "The OpenSearch Project: https://opensearch.org/"
}
```


Figura 2.1: OpenSearch Response

6. **Exploring OpenSearch Dashboard**: After successfully launching OpenSearch, you can access the OpenSearch Dashboard, a graphical interface that allows you to interact with indexed data, create visualizations, and monitor system performance.

   To open OpenSearch Dashboard, navigate to: `http://localhost:5601`

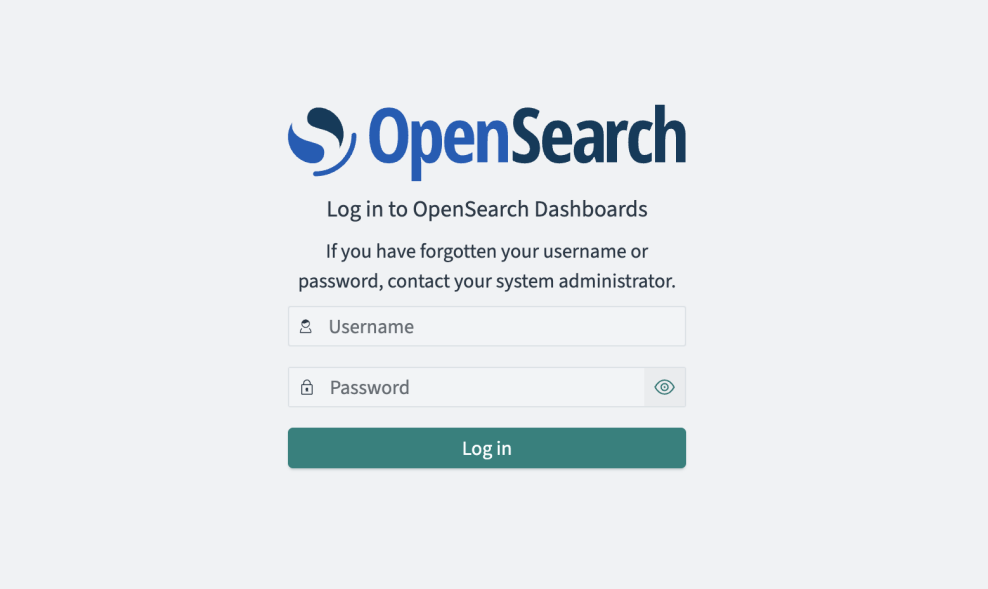

Figura 2.2: OpenSearch Dashboard Home

   Key features include:

   - Discover - Search and filter indexed data interactively.
   - Dashboards - Create interactive data visualizations for better analysis.
   - Index Management - View and configure indices, mappings, and storage settings.
   - Dev Tools - Execute OpenSearch queries using Query DSL.
   - Alerting and Security - Set up real-time alerts and manage security roles.

7. **Stopping the OpenSearch Cluster**: To stop the services, use the following command:

   ```
   docker-compose down
   ```

   This command will shut down and remove all containers, networks, and volumes associated with the OpenSearch deployment.

Upon completion of these steps, OpenSearch and OpenSearch Dashboard will be fully operational.

3. Observability

In modern software systems, understanding how applications behave in real time is crucial to ensure performance, reliability, and fast issue resolution.

This is where **Observability** comes in. Observability refers to the ability to monitor and diagnose systems and applications in real time, allowing engineers to understand the internal state of a system by collecting data from logs, metrics, and traces.

This chapter explores the concept of observability, its core principles, and its importance in modern distributed systems.

3.1 Understanding Observability

Observability is the ability of a system to make its internal state understandable, allowing developers, IT administrators and DevOps teams to detect, diagnose, and resolve issues affecting the performance, scalability, or availability of software and infrastructure based on the data it generates.

It consists of three main pillars:

- **Logs**: are detailed records of events that occur within an application. They help in debugging, understanding specific failures, and determining what happened at a given point in time.
- **Metrics**: are a specific kind of telemetry data and provide numerical data on system performance, such as CPU usage, memory consumption, response times, error rates, etc. providing quantifiable data about the performance and behavior of systems and applications. The importance of supporting metrics structured schema lies in the fact that it enables better analysis and understanding of system behavior. They represent a snapshot of the current state for a set of data. Metrics are distinct from logs or events, which focus on records or information about individual events.
- **Traces**: follow a request as it moves through different parts of a system, helping identify bottlenecks and latency issues. Tracing is particularly important in microservices architectures, where a single user request may involve multiple services.

Moreover, observability is closely linked to the concept of "data quality." In Big Data, the integrity and reliability of data are paramount. Observability frameworks can help ensure that data is collected accurately and consistently, which is essential for generating trustworthy insights. [2]

This is particularly important in sectors such as finance and public health, where decisions based on flawed data can have significant repercussions [3]. By providing mechanisms to monitor data quality, observability contributes to the overall effectiveness of Big Data initiatives.

3.2 Observability fundamentals

Monitoring and observability are distinct concepts that depend on each other.

- Traditional IT monitoring is an action you perform to increase the observability of your system.
- Observability is a property of that system, like functionality or testability.

Specifically, monitoring is the act of observing a system's performance over time. Monitoring tools collect and analyze system data and translate it into actionable insights. Monitoring technologies can tell you if a system is up or down or if there is a problem with application performance.

Observability measures how well the software system's internal states can be inferred from knowledge of its external outputs. It uses the data and insights that monitoring produces to provide a holistic understanding of your system, including its health and performance. The observability of your system, then, depends partly on how well your monitoring metrics can interpret your system's performance indicators.

Another important difference is that monitoring requires you to know what's important to monitor in advance.

Observability lets you determine what's important by watching how the system performs over time and asking relevant questions about it. [4]

3.3 Benefits of Observability

Observability allows DevOps developers to understand an application's internal state at any given time and have access to more accurate information about system faults in distributed production environments.

A few key benefits include [4]:

- **Better visibility**: Sprawling distributed systems often make it hard for developers to know what services are in production, whether application performance is strong, who owns a certain service or what the system looked like before the most recent deployment. Observability gives them real-time visibility into production systems that can help remove these impediments.
- **Better alerting**: Observability helps developers discover and fix problems faster, providing deeper visibility that allows them to quickly determine what has changed in the system, debug or fix the issues and determine what, if any, problems those changes have caused.

- **Better workflow**: Observability allows developers to see a request's end-to-end journey, along with relevant contextualized data about a particular issue, which in turn streamlines the investigation and debugging processes for an application, optimizing its performance.
- **Less time in meetings**: Historically, developers would have to track down information through third-party companies and apps to find out who was responsible for a particular service or what the system looked like days or weeks before the most-recent deployment. With effective observability, this information is readily available.
- **Accelerated developer velocity** Observability makes monitoring and troubleshooting more efficient, removing the main friction point for developers. The result is increased speed of delivery and more time for engineering staff to come up with innovative ideas to meet the needs of the business and its customers.

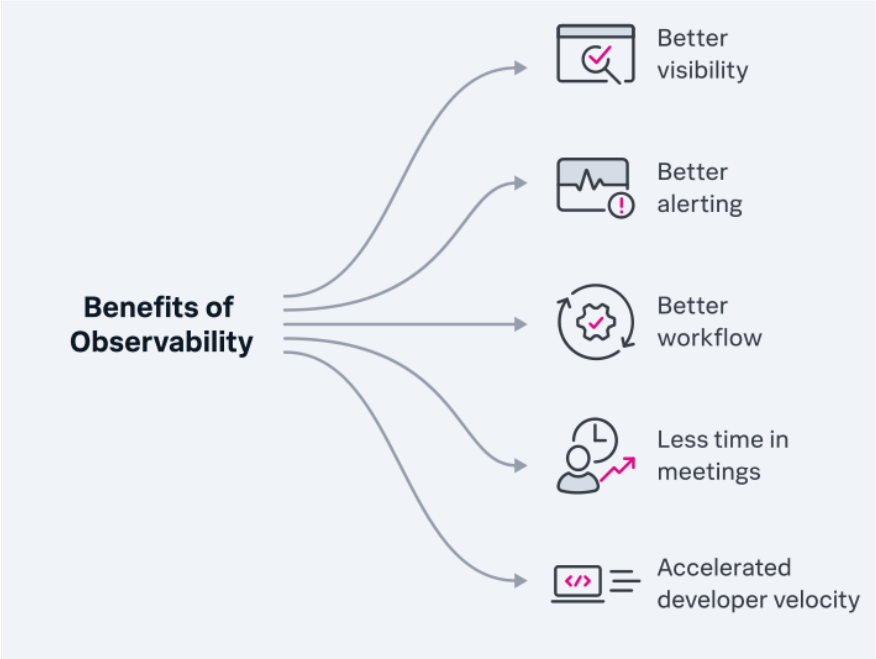

Figura 3.1: Benefits of Observability

4. OpenTelemetry: A Standard for Observability

OpenTelemetry (OTel) is an open-source observability framework designed for collecting, processing, and exporting telemetry data from distributed systems. It provides a standardized approach to instrumenting applications, ensuring comprehensive and consistent observability across platforms.

4.1 Introduction to OpenTelemetry

OpenTelemetry is a key part of observability since it allows you to collect traces, metrics, and logs from your systems to help you understand why issues may be happening. This is crucial, especially in complex microservices and cloud-native systems, where human beings can't track every single transaction traversing your architecture. However, it is not a back-end solution like Prometheus or Jaeger. It's open source and vendor-neutral, with an active and thriving community. [5]

So OpenTelemetry is an open-source, vendor-neutral framework designed for the collection of telemetry data from distributed systems. It standardizes the collection, processing, and export of logs, metrics, and traces, enabling organizations to have a unified and consistent observability pipeline across different platforms and technologies. The integration of OpenTelemetry with OpenSearch enhances the system's observability by providing structured and standardized data, which allows for better insights and decision-making.

By adopting OpenTelemetry, organizations can:

- Standardize observability data.
- Minimize vendor lock-in.
- Gain deep insights to optimize performance and enhance digital experiences.

One of the primary offerings of OpenTelemetry is its ability to handle the high volume and variety of data generated in Big Data environments. As organizations increasingly rely on diverse data sources, including IoT devices, cloud services, and traditional databases, OpenTelemetry provides a unified approach to instrumentation that simplifies the collection of telemetry data across these varied systems.[6] This capability is crucial because the effectiveness of Big Data analytics often hinges on the quality and comprehensiveness of the data collected. High-quality data, characterized by volume, variety, and veracity, can lead to better decision-making and improved operational efficiency.

Moreover, OpenTelemetry enhances the veracity of data collected by ensuring that the telemetry data is accurate and reliable. This is particularly important in Big Data contexts where the integrity of data can significantly impact the insights derived from analytics. For instance, in healthcare, accurate telemetry data can lead to better patient outcomes by enabling more precise analytics for clinical decision support and disease management.[7] The framework's ability to provide context around the data, such as its source and the conditions under which it was collected, further enriches the analytical capabilities of organizations.

4.2 How does it work?

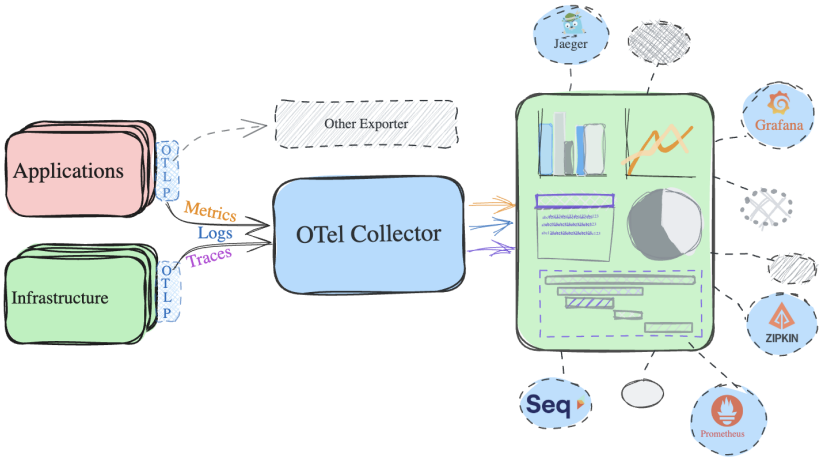

Figura 4.1: Otel Flow

OpenTelemetry consists of several components that work together to collect, process, and export telemetry data:

- **Language SDKs**: OpenTelemetry provides language SDKs that enable you to use the OpenTelemetry API to generate telemetry data with a certain programming language and export this data to a specific back-end.
- **Automatic Instrumentation**: This is the process of embedding OpenTelemetry SDKs into applications and services to collect telemetry data. These SDKs automatically collect and generate logs, metrics, and traces from the code.
- **Exporters**: OpenTelemetry includes exporters that allow telemetry data to be sent to various destinations, including OpenSearch. These exporters are responsible for transmitting logs, metrics, and traces in a format that OpenSearch can ingest and process. An exporter works by decoupling the instrumentation from your backend configuration, making it easier to change backends without changing the instrumentation you added to the code to extract the data.

- **Collectors**: OpenTelemetry Collectors act as intermediaries that receive, process, and export telemetry data to back-end systems. They can be configured to perform tasks such as data aggregation, filtering, and transformation before sending the data to OpenSearch or other observability platforms.

By collecting telemetry data across all components of an application, OpenTelemetry provides a comprehensive view of system performance, allowing teams to identify bottlenecks, failures, and optimize resource utilization.

4.3 What is Otel used for?

The main purpose of OpenTelemetry is to provide a consistent, standardized way to expose and collect application monitoring and telemetry data.

To be clear, you certainly don't need to use OpenTelemetry to monitor applications. You could instrument your own code to tell your applications how to generate telemetry data and expose it to your monitoring tools.

However, with OpenTelemetry, you get a set of instrumentation libraries that tell applications how to expose data, with minimal coding needed. Your developers can simply use the OpenTelemetry libraries, rather than writing their own code.

In addition, any monitoring and observability tool that supports OpenTelemetry can collect data from any application that uses OpenTelemetry instrumentation. This means you can switch monitoring and observability tools (or use multiple tools at once) without having to change the way your applications generate and expose data.

5. The Role of AI and ML in OpenSearch

Artificial Intelligence (AI) and Machine Learning (ML) capabilities have become essential to enhance data analysis, detect anomalies, and drive predictive insights. OpenSearch, with its integrated AI and ML features, provides organizations with powerful tools to perform advanced data analysis and automate decision-making processes. These capabilities make it an even more valuable platform for handling large datasets, improving operational intelligence, and uncovering hidden insights from vast and complex data sources.

This chapter explores OpenSearch's built-in AI/ML capabilities, with a focus on anomaly detection, predictive analytics, and machine learning models. We will examine how these features are integrated into the platform and how they can be used to drive business value.

5.1 Machine Learning Features

OpenSearch includes a ML framework called ML Commons, which allows users to develop, train, and deploy ML models directly within the platform. This eliminates the need for external ML tools and provides a streamlined approach to data-driven decision-making.

5.1.1 Key ML Features in OpenSearch

1. **Anomaly Detection**: helps identify unusual patterns in data. This is useful for security monitoring, fraud detection, and system performance analysis. OpenSearch can automatically detect irregularities and alert users.

2. **Forecasting**: enables users to predict future trends based on historical data. OpenSearch applies machine learning algorithms to analyze time-series data, allowing businesses to anticipate demand fluctuations, optimize resource allocation, and manage risks effectively. This feature is especially useful in industries like finance, supply chain management, and infrastructure planning.

3. **Clustering and Classification**: supports clustering and classification algorithms that group data based on shared characteristics. Clustering helps identify hidden patterns in large datasets, while classification assigns predefined labels to data points. These techniques are widely used in recommendation systems, customer segmentation, and content categorization.

5.2 AI Capabilities

Beyond traditional machine learning, OpenSearch integrates AI techniques to enhance search intelligence, improve data interpretation, and enable advanced automation. These AI-driven features enhance the way users interact with and retrieve information.

5.2.1 Key AI Features in OpenSearch

1. **Natural Language Processing (NLP)**: enables OpenSearch to understand and process human language efficiently. NLP enhances search capabilities by interpreting queries in a more natural way, improving accuracy in retrieving relevant documents. AI-driven NLP models can also analyze sentiment, extract key phrases, and summarize text data, making it valuable for customer feedback analysis and content management.
2. **Search Relevance and Ranking**: AI-powered ranking algorithms optimize search result relevance, ensuring that users receive the most pertinent information based on their queries. OpenSearch uses deep learning models to refine ranking strategies, taking into account user behavior, context, and semantic meaning, leading to a more personalized and accurate search experience.
3. **Recommendation Systems**: use AI models to suggest relevant content or actions to users based on their behavior and historical data. This feature is commonly applied in e-commerce, media platforms, and enterprise applications to enhance user engagement and content discovery.

5.3 ML Frameworks and Integration with OpenSearch

OpenSearch provides native support for machine learning tasks and integrates with popular machine learning frameworks, such as TensorFlow, PyTorch, and Scikit-Learn. These integrations allow data scientists and developers to leverage the full power of OpenSearch's data processing capabilities while using familiar ML tools for training and deploying models.

5.3.1 Native ML Support in OpenSearch

OpenSearch includes built-in machine learning modules that enable users to perform tasks such as anomaly detection, classification, regression, and clustering directly within the platform. These modules offer a user-friendly interface for managing and deploying models, as well as tools for monitoring and evaluating model performance.

5.3.2 Integration with External ML Frameworks

For more advanced or custom machine learning workflows, OpenSearch integrates seamlessly with popular ML frameworks. By exporting data from OpenSearch into external tools such as TensorFlow or Scikit-Learn, users can take advantage of specialized algorithms, model training capabilities, and additional features that may not be natively available in OpenSearch. Once the models are trained, they can be re-imported into OpenSearch for real-time analysis and prediction.

6. Conclusions

This study has provided a comprehensive and detailed analysis of OpenSearch, focusing on its key features, observability capabilities, integration with OpenTelemetry, and its support for AI/ML applications. Through extensive documentation review and theoretical exploration, we have highlighted how OpenSearch functions as a powerful and versatile platform for managing complex search, analytics, and observability needs across various use cases.

OpenSearch stands out due to its robust search and analytics capabilities, making it well suited for organizations that require efficient, large-scale data processing and retrieval. Its flexible architecture allows for seamless scalability, ensuring that it can accommodate both small and enterprise-level applications. The observability features of the platform, in particular, are highly beneficial for monitoring distributed systems, providing deep insights into the health, performance, and potential bottlenecks of the system. Integrationh OpenTelemetry enhances these capabilities by offering stanstandardizeds-platform telemetry data collection, makinmaking to gain visibility across various parts of a system and enabling renablinge monitoring and troubleshooting.

Furthermore, OpenSearch's support for AI/ML functionalities adds significant value to the platform. The inclusion of machine learning models within OpenSearch facilitates predictive analytics and anomaly detection, which can be crucial for a wide range of applications, from business intelligence to cybersecurity. These AI/ML tools allow for the processing of large datasets and provide users with advanced capabilities that would otherwise require additional third-party software or complex setups.

In conclusion, OpenSearch offers a strong foundation for building sophisticated search, analytics, and observability solutions. Its open-source nature and flexible design make it an attractive option for developers and organizations looking to implement scalable, data-driven applications. The combination of search, machine learning, and observability tools on the platform positions it as a highly valuable resource for businesses and teams that want to improve their data management and operational efficiency. OpenSearch, with its ongoing development and growing community support, has the potential to become an even more powerful tool in the coming years, supporting a wider range of use cases and industries.

Bibliography

[1] Instaclustr. What is opensearch? an overview, 01 2024. URL https://instaclustr.medium.com/what-is-opensearch-an-overview-c8530bd0f9de.

[2] Janet G. Baseman, Debra Revere, and Ian Painter. Big data in the era of health information exchanges: Challenges and opportunities for public health. *Informatics*, 4(4), 2017. ISSN 2227-9709. doi: 10.3390/informatics4040039. URL https://www.mdpi.com/2227-9709/4/4/39.

[3] Angappa Gunasekaran, Thanos Papadopoulos, Rameshwar Dubey, Samuel Fosso Wamba, Stephen J. Childe, Benjamin Hazen, and Shahriar Akter. Big data and predictive analytics for supply chain and organizational performance. *Journal of Business Research*, 70:308–317, 2017. ISSN 0148-2963. doi: https://doi.org/10.1016/j.jbusres.2016.08.004. URL https://www.sciencedirect.com/science/article/pii/S014829631630491X.

[4] Stephen Watts. What is observability? an introduction, 2023. URL https://www.splunk.com/en_us/blog/learn/observability.html.

[5] Giulia Di Pietro. Introduction to opentelemetry, 2022. URL https://isitobservable.io/open-telemetry/traces/a-short-guide-to-opentelemetry.

[6] Francesco Cappa, Raffaele Oriani, Enzo Peruffo, and Ian McCarthy. Big data for creating and capturing value in the digitalized environment: Unpacking the effects of volume, variety, and veracity on firm performance*. *Journal of Product Innovation Management*, 38(1):49–67, 2021. doi: https://doi.org/10.1111/jpim.12545. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/jpim.12545.

[7] Yichuan Wang, LeeAnn Kung, and Terry Anthony Byrd. Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological Forecasting and Social Change*, 126:3–13, 2018. ISSN 0040-1625. doi: https://doi.org/10.1016/j.techfore.2015.12.019. URL https://www.sciencedirect.com/science/article/pii/S0040162516000500.