International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

Big Data Analytics Using Artificial Intelligence: A Review Of Platforms, Learning Techniques And Applications

Kharade Shruti Sharad[1]; Prof. M. C. Kolse[2]; Prof. S. K. Said[3]
Department Of Artificial Intelligence And Data Science Jaihind College Of Engineering Pune, Maharashtra, India

Abstract—Organizations in healthcare, banking, education, retail and government generate data in volumes, varieties and speeds that traditional database tools cannot process effectively. Big Data Analytics using Artificial Intelligence (AI) combines scalable data platforms with machine learning and deep learning to identify patterns, predict future trends and generate actionable insights from this data. This paper reviews the field from both the systems and the learning perspective. It explains the characteristics of big data, the evolution of processing frameworks from MapReduce and the Hadoop distributed file system to in-memory engines such as Apache Spark, stream processors such as Apache Kafka and Flink, and the lakehouse model. It then examines how machine learning and deep learning are scaled through distributed libraries and frameworks, and how analytics serves applications such as disease prediction, fraud detection, recommendation systems and learning analytics. A five-layer architecture comprising data sources, ingestion, storage, processing and AI, and visualization is proposed, followed by a research gap analysis, objectives, a comparison of frameworks and a discussion of challenges including data quality, privacy, cost and the hidden maintenance burden of machine-learning systems. The review concludes that AI-driven big data analytics converts raw data into timely, meaningful information for better decision-making.

Index Terms—Big data analytics, artificial intelligence, machine learning, deep learning, Apache Spark, Hadoop, stream processing, data lakehouse

I. Introduction

Every online purchase, sensor reading, bank transaction and hospital visit generates data. Gandomi and Haider [1] characterize big data by its volume, velocity and variety, to which veracity and value are often added. Data at this scale can reveal customer preferences, early signs of disease or patterns of fraud, but only if it can be stored, processed and analysed efficiently.Traditional relational databases and single-machine analysis tools were not designed for petabytes of mixed structured and unstructured data arriving continuously. At the same time, the most powerful AI methods, especially deep learning, become more accurate as they see more data [2]. Big data platforms and AI therefore reinforce each other: platforms make large-scale learning feasible, and AI extracts value that simple queries cannot.

This paper reviews Big Data Analytics using AI, covering platforms, learning techniques and applications. It proposes a reference architecture, identifies research gaps and discusses the challenges of deploying AI on big data.

II. Background and Fundamental Concepts

2.1 Characteristics of Big Data

Volume refers to the scale of data, velocity to the speed at which it arrives, variety to the mix of tables, text, images, logs and sensor streams, veracity to its uncertainty and quality, and value to the benefit it can deliver. Chen and Zhang [3] surveyed the techniques and technologies required to handle these properties.

2.2 Types of Analytics

Analytics can be descriptive (what happened), diagnostic (why it happened), predictive (what will happen) and prescriptive (what should be done). AI is central to the last two. Fig. 1 classifies the AI techniques used in big data analytics.

International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

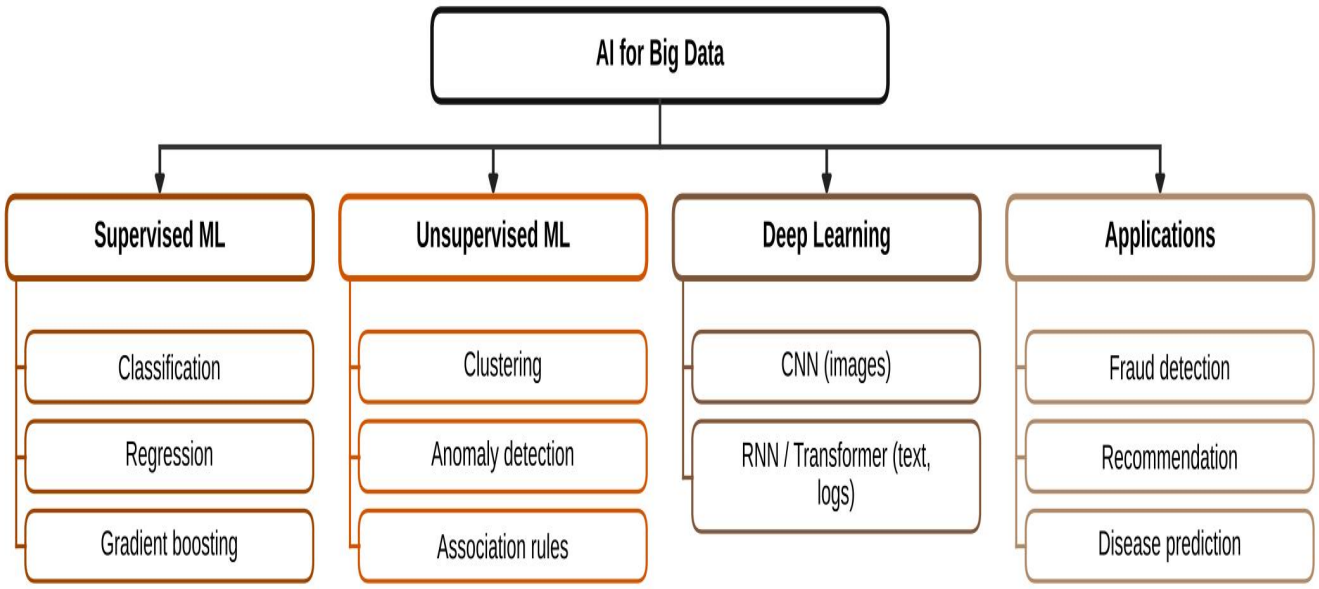


Fig. 1. AI techniques used in big data analytics.

III. Literature Survey

Dean and Ghemawat [4] introduced MapReduce, a programming model that splits a computation into map and reduce functions executed in parallel across thousands of commodity machines, with automatic handling of failures. The Hadoop Distributed File System [5] provided an open-source storage layer that replicates large files across clusters, making MapReduce widely available.

MapReduce writes intermediate results to disk, which makes iterative algorithms such as machine learning slow. Zaharia *et al.* [6] proposed resilient distributed datasets, an in-memory abstraction that can recompute lost data from its lineage, and Apache Spark [7] built on it to unify batch processing, SQL, streaming, graph processing and machine learning in one engine. MLlib [8] provides distributed implementations of common algorithms on Spark.

For data in motion, Kafka [9] offered a distributed, durable log for high-throughput event streams, and Apache Flink [10] provided a single engine for stream and batch processing with exactly-once guarantees. Armbrust *et al.* [11] proposed the lakehouse, combining the low-cost storage of data lakes with the management features of data warehouses.

Scaling deep learning required new systems. Dean *et al.* [12] trained very large networks across many machines using model and data parallelism, and Ray [13] offered a general framework for distributed AI workloads such as hyper-parameter tuning and reinforcement learning. Najafabadi *et al.* [2] reviewed how deep learning extracts abstract representations from big data and identified challenges with streaming and high-dimensional data. Sculley *et al.* [14] warned that ML systems accumulate hidden technical debt through data dependencies and pipeline complexity. Table I summarizes these works.

Ref.	Authors	Year	Technique Used	Merits	Demerits
[4]	Dean and Ghemawat	2008	MapReduce programming model	Fault-tolerant parallel processing on clusters	Slow for iterative ML
[5]	Shvachko et al.	2010	Hadoop Distributed File System	Reliable storage of very large files	Not suited to small files or low latency
[9]	Kreps et al.	2011	Kafka distributed log	High-throughput event streaming	Messaging only; no processing
[6]	Zaharia et al.	2012	Resilient distributed datasets	In-memory, fault-tolerant computation	Memory-hungry
[10]	Carbone et al.	2015	Apache Flink	Unified stream and batch with low latency	Smaller ML ecosystem
[2]	Najafabadi et al.	2015	Review of deep learning for big data	Identified representation-learning benefits	Pre-dates transformers
[14]	Sculley et al.	2015	Analysis of ML technical debt	Exposed hidden maintenance costs	No quantitative benchmark
[7]	Zaharia et al.	2016	Apache Spark unified engine	Batch, SQL, streaming and ML in one system	Tuning complexity
[11]	Armbrust et al.	2021	Lakehouse architecture	Warehouse features on low-cost lake storage	Relatively new ecosystem

TABLE I Summary of Literature Survey

IV. Research Gap

The following gaps are identified:

1. Data quality and integration across heterogeneous sources remain major obstacles, consuming most project effort.

2. Real-time AI on streaming data is less mature than batch analytics.

3. Privacy-preserving analytics at scale is still costly and complex.

4. Explainability of predictions from large, complex models is limited for business users.

5. Smaller organizations lack affordable, easy-to-use integrated platforms and skilled personnel.

V. Objectives

The objectives of this review are:

• To explain the characteristics of big data and the types of analytics.

• To review big data processing platforms and their evolution.

• To study how machine learning and deep learning are applied and scaled on big data.

• To propose a layered architecture for AI-driven big data analytics.

• To discuss applications, challenges and future directions.

International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

VI. Proposed System Architecture

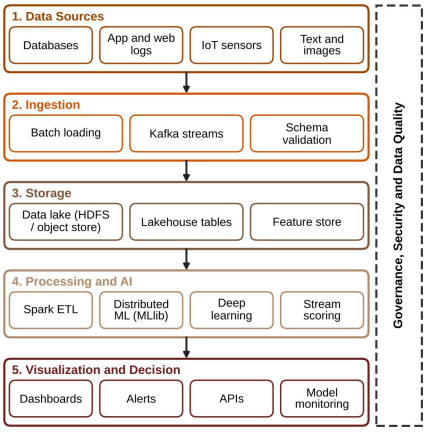


Fig. 2 presents a five-layer architecture that supports both batch and real-time analytics.

Fig. 2. Proposed five-layer architecture for AI-driven big data analytics.

6.1 Data Source Layer

Sources include transactional databases, web and mobile application logs, IoT sensors, social media, documents and images.

6.2 Ingestion Layer

Batch data is loaded on a schedule, while streaming data is published to a distributed log such as Kafka. Connectors validate schemas and record data lineage.

6.3 Storage Layer

Raw data lands in a data lake on distributed or cloud object storage. Cleaned and structured data is organized in a lakehouse with table formats that support transactions and versioning.

6.4 Processing and AI Layer

Spark performs large-scale cleaning, transformation and feature engineering. Machine-learning models such as gradient-boosted trees [15] and deep networks are trained on distributed clusters, and stream processors apply models to events in real time.

6.5 Visualization and Decision Layer

Dashboards, reports, alerts and application interfaces deliver predictions and insights to users. A model registry and monitoring service track model versions and performance.

VII. Methodology and Working

The analytics workflow is shown in Fig. 3. Distributed processing speed-up is bounded by the fraction of work that must run serially; according to Amdahl's law, with a parallel fraction p and N workers,

International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 5
Sep-Oct- 2026, PP 01-7

$$Speedup = 1 / ((1 - p) + p / N) \quad (1)$$

which shows why minimizing serial steps such as data shuffles is important. Predictive models are evaluated with metrics suited to the task, such as root-mean-square error for regression,

$$RMSE = \sqrt{((1/n) \Sigma (y_i - \hat{y}_i)^2)} \quad (2)$$

and precision, recall and AUC for classification.

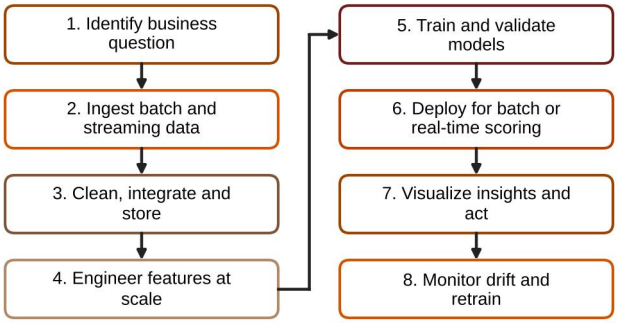


Fig. 3. Big data analytics workflow using AI.

After deployment, models are monitored for data drift, and pipelines are automated so that retraining uses fresh data without manual effort.

VIII. Comparative Analysis

Table II compares major processing frameworks. Hadoop MapReduce is reliable for large batch jobs but slow for iterative work; Spark is much faster in memory and supports ML; Flink excels at low-latency streaming; and Ray is designed for distributed AI workloads.

Framework	Processing	Latency	AI Support
Hadoop MapReduce	Batch (disk)	High	Limited
Apache Spark	Batch + micro-batch	Low–Medium	MLlib, integrations
Apache Flink	True streaming	Very low	Moderate
Apache Kafka	Event log	Very low	Via consumers
Ray	Distributed tasks	Low	Strong (DL, RL, tuning)

TABLE II Comparison of Big Data Processing Frameworks

Applications span many sectors. In healthcare, analytics predicts disease risk and hospital demand; in banking, it detects fraudulent transactions; in retail, recommendation engines personalize offers; and in education, learning analytics identifies students who need support.

IX. Challenges and Future Scope

Table III summarizes the main challenges. Future directions include AutoML that lowers the skill barrier, federated analytics across organizations, integration of large language models for natural-language querying of data, energy-efficient computing, and edge analytics that processes IoT data close to its source.

Challenge	Mitigation
Poor data quality	Validation rules, profiling, data contracts
Privacy and security	Access control, encryption, anonymization
High cost	Cloud elasticity; storage tiering
Skill shortage	AutoML and managed services
ML technical debt	MLOps pipelines, monitoring, versioning

TABLE III Challenges and Mitigation

X. Conclusion

Big Data Analytics using AI combines scalable platforms with intelligent algorithms to turn massive, fast and diverse data into useful knowledge. The reviewed literature traces a clear evolution from disk-based MapReduce to in-memory engines, stream processors and lakehouses, alongside distributed frameworks that scale machine learning and deep learning. These technologies enable faster decisions, higher accuracy and reduced human effort in healthcare, banking, education and business. Their success depends on data quality, privacy protection, explainability and disciplined engineering of ML pipelines.

10.1 Acknowledgment

The authors express their sincere gratitude to the Department of Artificial Intelligence and Data Science, Jaihind College of Engineering, Pune, for providing the facilities, guidance and encouragement needed to carry out this work. The authors also thank the faculty members and colleagues whose valuable suggestions helped improve the quality of this review.

References

1. A. Gandomi and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics,” Int. J. Inf. Manage., vol. 35, no. 2, pp. 137–144, 2015.
2. M. M. Najafabadi et al., “Deep learning applications and challenges in big data analytics,” J. Big Data, vol. 2, Art. no. 1, 2015.
3. C. L. P. Chen and C.-Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data,” Inf. Sci., vol. 275, pp. 314–347, 2014.
4. J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.
5. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop distributed file system,” in Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST), 2010, pp. 1–10.
6. M. Zaharia et al., “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in Proc. 9th USENIX NSDI, 2012, pp. 15–28.
7. M. Zaharia et al., “Apache Spark: A unified engine for big data processing,” Commun. ACM, vol. 59, no. 11, pp. 56–65, 2016.
8. X. Meng et al., “MLlib: Machine learning in Apache Spark,” J. Mach. Learn. Res., vol. 17, no. 34, pp. 1–7, 2016.

9. J. Kreps, N. Narkhede, and J. Rao, “Kafka: A distributed messaging system for log processing,” in Proc. NetDB Workshop, 2011.
10. P. Carbone et al., “Apache Flink: Stream and batch processing in a single engine,” Bull. IEEE Comput. Soc. Tech. Committee Data Eng., vol. 38, no. 4, pp. 28–38, 2015.
11. M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, “Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics,” in Proc. Conf. Innov. Data Syst. Res. (CIDR), 2021.
12. J. Dean et al., “Large scale distributed deep networks,” in Proc. NeurIPS, 2012, pp. 1223–1231.
13. P. Moritz et al., “Ray: A distributed framework for emerging AI applications,” in Proc. 13th USENIX OSDI, 2018, pp. 561–577.
14. D. Sculley et al., “Hidden technical debt in machine learning systems,” in Proc. NeurIPS, 2015, pp. 2503–2511.
15. T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2016, pp. 785–794.