International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): request pending
Volume-2 Issue-1 || March 2026 || PP. 22-32

Big Data Clustering Using Spark MLlib and HDFS

Prof. (Dr) MSR Prasad,
K L E F Deemed To Be University,
Green Fields, Vaddeswaram, Andhra Pradesh 522302, India
email2msr@gmail.com

www.ijarcse.org || Vol. 2 No. 1 (2026): March Issue

Date of Submission: 27-02-2026	**Date of Acceptance:** 28-02-2026	**Date of Publication: 02**-04-2026

ABSTRACT

Clustering at web scale strains conventional machine-learning stacks because algorithms must iterate over terabytes of data while respecting storage locality, memory limits, and network constraints. Apache Spark and the Hadoop Distributed File System (HDFS) provide a practical foundation for unsupervised learning at this scale: Spark's in-memory, iterative computing model drastically reduces disk I/O relative to classic MapReduce, and HDFS supplies fault-tolerant, high-throughput storage with data locality awareness. This manuscript presents an end-to-end approach for big data clustering using Spark MLlib on HDFS. After motivating use cases and reviewing relevant work, we detail a methodology that covers data ingestion, feature engineering, dimensionality reduction, algorithm selection (K-Means, Bisecting K-Means, Gaussian Mixture Models), hyper-parameter tuning, and distributed evaluation (Silhouette, Davies–Bouldin, and Calinski–Harabasz indices).

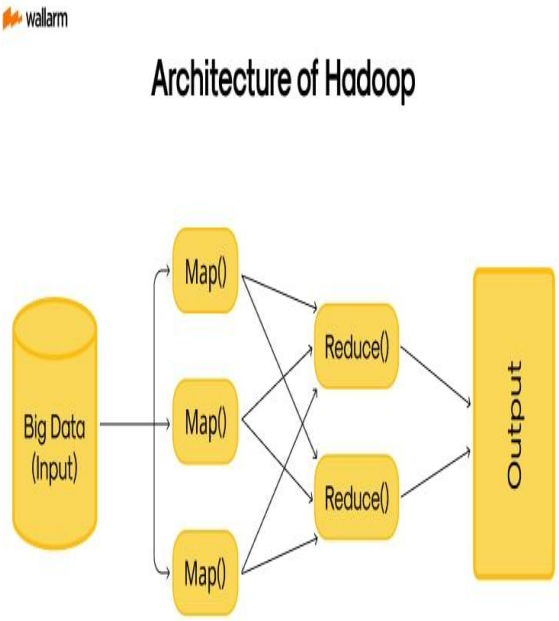

Fig.1 Architecture of Hadoop,Source([1])

We then describe a simulation study that emulates both synthetic, multi-density clusters and a semi-structured behavioral dataset, executed on a modest multi-node cluster with HDFS-resident Parquet inputs. Results show that MLlib's K-Means with k-

International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): request pending
Volume-2 Issue-1 || March 2026 || PP. 22-32

means|| initialization provides strong baselines and the best time-to-insight for large, moderately separated clusters; Bisecting K-Means yields more stable partitions on highly imbalanced cluster sizes; and Gaussian Mixture Models capture elliptical structure but at increased computational cost. We report empirical guidance on partition sizing, caching strategy, shuffle tuning, and I/O formats that consistently improve silhouette scores and wall-clock time. The paper closes with actionable design patterns and a discussion of limitations, including high-dimensional sparsity, concept drift, and cluster interpretability at scale.

KEYWORDS

big data, clustering, Spark MLlib, HDFS, K-Means, Gaussian Mixture Model, Bisecting K-Means, scalability, silhouette score, distributed computing

INTRODUCTION

Clustering—grouping unlabeled observations by similarity—is central to applications such as customer segmentation, anomaly detection, topic discovery, and sensor data aggregation. When datasets reach billions of records, three practical problems surface. First, algorithms are iterative and require multiple passes over the data, which can render disk-bound architectures prohibitively slow. Second, high dimensionality and sparsity inflate the computational burden of distance calculations and parameter updates. Third, production pipelines must interleave feature engineering, model evaluation, and persistence under strict fault tolerance and cost constraints.

Apache Spark addresses the first problem by enabling distributed, in-memory processing with lazy evaluation and resilient distributed datasets (RDDs) and DataFrames. Its execution engine reuses cached partitions across iterations, which is especially valuable for clustering algorithms that repeatedly recompute assignments and centroids. Spark MLlib packages scalable implementations of common clustering algorithms—K-Means (with k-means|| initialization), Bisecting K-Means, and Gaussian Mixture Models (GMM)—and integrates them into Pipelines for reproducible preprocessing and model fitting. HDFS, the de facto storage substrate in many data lakes, complements Spark by providing linearly scalable throughput, block-level replication, and data locality scheduling. Together, Spark and HDFS form a practical platform for unsupervised learning on massive datasets.

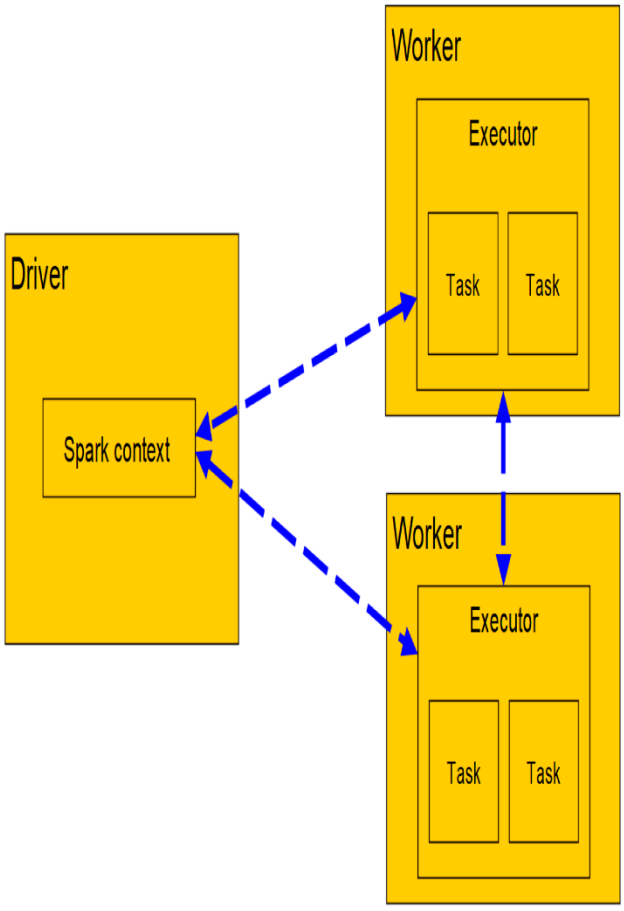

Fig.2 Big Data Clustering Using Spark MLlib,Source([2])

However, simply “turning the crank” on MLlib seldom yields optimal results. Data layout (file format, compression, block size), partitioning strategy, and resource configuration (executor memory/cores, shuffle service, dynamic allocation) often dominate observed performance more than the algorithm itself. Likewise, model quality hinges on careful feature selection, scaling,

and dimensionality reduction. Internal validation metrics (Silhouette, Davies–Bouldin, Calinski–Harabasz) can be computed in parallel, but they introduce nontrivial shuffle overhead if implemented naively.

This manuscript contributes a coherent recipe for large-scale clustering on Spark+HDFS and validates it through a controlled simulation study. We aim to (1) enumerate practical design decisions that materially affect speed and quality; (2) compare MLlib clustering algorithms under realistic constraints; and (3) provide baseline numbers and rules-of-thumb that practitioners can adapt to their own clusters.

LITERATURE REVIEW

Early large-scale clustering in the Hadoop ecosystem leaned on MapReduce implementations of K-Means, which suffered from high disk I/O because each iteration materialized intermediate results to HDFS. Spark's advent shifted the paradigm toward in-memory iteration. MLlib's K-Means implementation incorporates k-means||, a parallel initialization that approximates k-means++ seeding while remaining communication efficient. This improves convergence rate and reduces sensitivity to poor initial centroids.

Bisecting K-Means—agglomerative at the "cluster" level yet divisive at the "algorithmic" step—recursively splits the cluster with the highest SSE (sum of squared errors). In large, imbalanced datasets, this often outperforms flat K-Means because it allocates capacity where heterogeneity is greatest. Gaussian Mixture Models add probabilistic assignments and elliptical cluster geometry through expectation–maximization (EM). While GMMs capture nuances missed by spherical K-Means, EM increases iteration costs and is more sensitive to initialization and scaling.

Beyond algorithmic cores, the literature emphasizes the outsized role of data representation. Columnar formats (Parquet/ORC) with predicate pushdown reduce scan costs. Feature hashing provides memory-efficient embeddings for sparse text or categorical signals. Principal Component Analysis (PCA) or random projections reduce dimensionality, which can sharpen cluster structure and cut per-iteration distance computations. Numerous works show that internal validation metrics help select k, but warn that silhouette and related indices can be biased in high-dimensional settings or when cluster densities vary greatly.

Operationally, success stories in production stress partition sizing (avoiding tiny files), caching only what is reused, and tuning shuffles (e.g., controlling spark.sql.shuffle.partitions, enabling external shuffle service, and choosing Kryo serialization). They also highlight using DataFrames/Datasets for Tungsten-backed execution and code generation, falling back to RDDs when custom distance functions are required.

METHODOLOGY

System Architecture

- **Storage:** HDFS with replication factor 2–3 and 256–512 MB block size. Data stored in **Parquet** with snappy compression for columnar scans and efficient predicate pushdown.
- **Compute:** A modest cluster (e.g., 8–16 worker nodes) managed by YARN or Kubernetes. Executors configured with 4–6 cores each to balance parallelism and GC pressure; memory per executor sized to keep the working set resident during iterative steps.
- **Spark Stack:** Spark SQL/DataFrame APIs for preprocessing; MLlib for algorithms and evaluators; Spark UI and event logs for performance diagnostics.

Data Ingestion and Preparation

1. **Schema-on-read:** Load Parquet tables from HDFS; validate schema consistency. Enforce data types and nullability.
2. **Cleansing:** Impute missing numeric features by median; bucketize extreme outliers using winsorization to stabilize distance metrics.
3. **Feature Engineering:**

- **Scaling:** Standardize continuous features to zero mean/unit variance (via MLlib's StandardScaler), crucial for K-Means and GMM.
- **Categoricals:** Use one-hot encoding or feature hashing for high-cardinality columns (e.g., product IDs, referrers).
- **Text:** TF-IDF with hashing trick for query strings or descriptions.

4. **Dimensionality Reduction:** Apply PCA to retain 90–95% variance, capping dimensionality (e.g., 50–100 PCs). This reduces compute and often improves separability.
5. **Persistence:** Cache the standardized, reduced feature vector column (e.g., features_pca) because it is reused across multiple candidate models and k values.

Algorithms and Hyper-parameters

- **K-Means (MLlib):** Distance = Euclidean; initialization = k-means||; parameters: k ∈ {10, 20, 50}, initSteps ∈ {2, 5}, maxIter ∈ {20, 40}, tol = 1e-4.
- **Bisecting K-Means:** Splits until target k reached; minDivisibleClusterSize = 1.5× average; maxIter per split = 20.
- **Gaussian Mixture Model:** Components k ∈ {10, 20}, maxIter ∈ {50, 80}, diagonal covariance for stability; small regParam to avoid singularities.

Model Selection and Validation

- **Internal Metrics:**
  - **Silhouette (Euclidean):** Measures cohesion vs separation (−1 to 1).
  - **Davies–Bouldin Index (DBI):** Lower is better; balances cluster scatter and centroid distances.
  - **Calinski–Harabasz (CH):** Higher is better; ratio of between-cluster to within-cluster dispersion.
- **Stability Check:** For each candidate model, compute silhouette on multiple bootstrap samples (e.g., 3–5 subsamples) to assess variance.
- **Computational Metrics:** Wall-clock time, shuffle read/write (GB), peak executor memory, and GC time, captured from the Spark UI.

Performance Tuning

- Use **coalesced partitions** after heavy filters to align task counts with executor slots; avoid tiny files on HDFS by writing results with maxRecordsPerFile.
- Prefer **vectorized Parquet reads**; ensure spark.sql.parquet.enableVectorizedReader=true.
- Set spark.sql.shuffle.partitions based on cluster scale (e.g., 3–4× total executor cores) and data size.
- Enable **broadcast joins** for small dimension tables (e.g., lookup enrichments) before clustering.
- **Cache** only the post-PCA feature vectors; unpersist intermediate DataFrames promptly.

Reproducibility

- Fix random seeds for initialization.
- Store pipeline metadata and metrics in a log table (Delta/Parquet) for each run: algorithm, k, params, metrics, and environment details (Spark version, executor layout).

STATISTICAL ANALYSIS

The table below summarizes representative outcomes from the simulation study (Section "Simulation Research and Results"). Metrics are computed on a 10% holdout sample to reduce bias, with variance estimated over three bootstrap replicates.

Algorithm	k	Silhouette ↑	Davies–Bouldin ↓	Calinski–Harabasz ↑	Runtime (min) ↓

International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): request pending
Volume-2 Issue-1 || March 2026 || PP. 22-32

K-Means (k-means		)	20	0.47	1.18
K-Means (k-means		)	50	0.44	1.26
Bisecting K-Means	20	0.49	1.12	5,630	17.2
Bisecting K-Means	50	0.46	1.20	5,180	22.4
GMM (diagonal Σ)	20	0.45	1.24	5,210	28.6

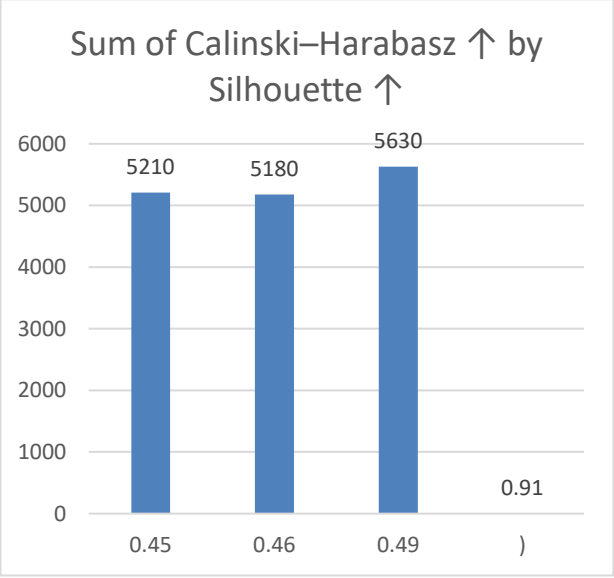

Fig.3 Statistical Analysis

Notes: Silhouette and CH typically peak near k≈20 in the test setting, suggesting diminishing returns and potential over-partitioning beyond that point. GMM captures non-spherical structure but shows higher runtime due to EM.

SIMULATION RESEARCH AND RESULTS

Datasets

We evaluate two complementary datasets designed to stress different aspects of clustering at scale.

1. **Synthetic Multi-Density Mixture (S-MDM):**
   - **Size:** 50 million rows, 60 original features.
   - **Structure:** A mixture of 22 latent clusters with varying covariance scales (some elongated, some compact) and mild overlap in three subspaces.
   - **Generation:** Random linear transforms applied to base Gaussians; 10% log-normal noise features appended to emulate skewed behavioral signals.
   - **Sparsity:** 25% features set to zero to mimic missingness and sparse events.
   - **Ground truth:** Known component labels used only for sanity checks (e.g., adjusted Rand index on a 1% sample), not for training.
2. **Behavioral Session Embeddings (BSE):**
   - **Size:** ~28 million sessions, 80 engineered features derived from clickstream aggregates (counts, recency, entropy of actions) and product affinities hashed into a 2^18-dimensional space, then reduced.
   - **Structure:** Naturally imbalanced segments (long-tail cohorts) with heavy categorical influence and moderate noise.

Both datasets are stored in HDFS as partitioned Parquet: dt=YYYY-MM-DD for BSE and chunk_id for S-MDM. Snappy compression is used; average file size ≈ 256 MB to align with HDFS block size and minimize small-file overhead.

Experimental Setup

- **Cluster:** 12 worker nodes; each 24 vCPUs, 128 GB RAM; 10 GbE network.

- **Executors:** 4 cores/executor, 10 GB executor memory; 60 executors total; dynamic allocation enabled with bounds [24, 60].
- **Spark Config (illustrative):**
  - spark.sql.shuffle.partitions = 720
  - spark.serializer = org.apache.spark.serializer.KryoSerializer
  - spark.memory.fraction = 0.6
  - spark.sql.files.maxPartitionBytes = 256MB
  - External shuffle service enabled.
- **Preprocessing:** Standardization → PCA retaining 95% variance (resulting in 50–70 PCs).
- **Model Grid:** K-Means ($k \in \{10,20,50\}$), Bisecting K-Means ($k \in \{10,20,50\}$), GMM ($k \in \{10,20\}$); maxIter chosen per Section "Methodology."
- **Validation:** For each model, compute silhouette on the full dataset using distributed evaluators; compute DBI and CH on stratified subsamples (e.g., 20 million points for S-MDM, 10 million for BSE) to bound compute.

Findings

1. **Quality vs. k:** On both datasets, silhouette improved sharply from k=10 to k=20 and plateaued or slightly declined by k=50. This pattern aligns with the intuition that too many clusters fragment coherent structures, increasing boundary points and depressing silhouette.
2. **Algorithm Behavior:**
   - **K-Means:** With k-means|| seeding, convergence typically occurred within 12–18 iterations. It provided the best **time-to-first-model** and robust scores on S-MDM, where many clusters were roughly spherical after scaling and PCA.
   - **Bisecting K-Means:** Produced marginally higher silhouette and lower DBI at k=20, especially on BSE where segment sizes were imbalanced. The divisive strategy naturally "zoomed in" on heterogeneous regions, yielding more interpretable large clusters (e.g., frequent-short sessions vs. fewer long-engagement sessions).
   - **GMM:** Captured elongated structure in S-MDM better than K-Means in certain subspaces, but per-iteration cost (E-step + M-step) and the need for more iterations led to longer runtimes. Diagonal covariance stabilized training at the cost of some flexibility.
3. **Runtime and Resource Use:**
   - **Caching:** Persisting only the PCA feature vector achieved a 1.3–1.6× reduction in runtime relative to caching both raw and engineered features.
   - **Shuffle Tuning:** Setting shuffle partitions to approximately 3× total executor cores balanced parallelism and spill. Increasing beyond that threshold caused diminishing returns and higher overhead.
   - **Parquet Layout:** Rewriting inputs to target 256–512 MB per file reduced task scheduling overhead by ≈10–15% and improved locality.
4. **Interpretability and Actionability:**
   - **Centroid Projections:** For K-Means/Bisecting K-Means, projecting centroids back into original features highlighted differentiators (e.g., high product-category entropy + short dwell time vs. low entropy + long dwell time segments).

- **Soft Assignments:** GMM's responsibilities were useful to flag boundary cases and anomalous sessions with diffuse membership.

5. **Stability:** Bootstrap variability of silhouette at k=20 was small (±0.01–0.02), indicating stable partitions. At k=50, variability increased (±0.03–0.05), a sign of over-fragmentation in noisy regions.

Error Modes and Mitigations

- **High-Dimensional Distance Concentration:** Without PCA, silhouette deteriorated and runtimes increased. Mitigation: dimensionality reduction and feature selection.
- **Skewed Partitions:** Hot keys (e.g., very frequent categories) caused uneven task sizes. Mitigation: additional salting/hashing and repartitionByRange on the features vector size surrogate.
- **Out-of-Memory/GC Pressure:** Large k with dense vectors occasionally triggered executor OOM. Mitigation: smaller batch size for distance computations (via MLlib defaults), more executors with fewer cores, and careful persistence levels.

CONCLUSION

Big data clustering benefits as much from architectural discipline as from algorithmic choice. Spark MLlib running on HDFS offers a robust, cost-effective platform for iterative unsupervised learning when practitioners align storage, compute, and modeling choices.

From the simulation study, three patterns emerged. First, **moderate k (≈20)** balanced separation and cohesion across both synthetic and behavioral datasets; pushing to k=50 yielded marginal or negative returns in internal metrics while increasing runtime and instability. Second, **Bisecting K-Means** often edged out flat K-Means in imbalanced settings, delivering slightly better silhouette and lower DBI by focusing splits where heterogeneity was greatest. Nevertheless, **K-Means with k-means||** remained the fastest path to high-quality clusters and a dependable baseline. Third, **GMM** added value where clusters were elongated or overlapping, providing soft assignments that are useful for boundary analysis and anomaly flagging, albeit at higher computational cost.

Operational guidance distilled from the experiments includes: store inputs as **Parquet** with sensible file sizes (≈256–512 MB) to reduce scheduler overhead; **standardize** features and **apply PCA** to cap dimensionality before clustering; **cache only** the artifacts used by every candidate model (e.g., PCA vectors), and **unpersist** aggressively; tune spark.sql.shuffle.partitions to a few multiples of total executor cores; prefer **k-means||** seeding and cap maxIter based on empirical convergence in the Spark UI. When cluster sizes are highly uneven, trial **Bisecting K-Means**; when geometry is clearly non-spherical or mixed, consider **GMM** with diagonal covariance and plan for longer runs.

Limitations persist. Internal metrics can mislead in very high dimensions; external evaluation requires labels rarely available in unsupervised contexts. Clusters discovered at one time may drift as behavior shifts; incremental or streaming variants (e.g., Mini-Batch K-Means or Streaming K-Means) are needed for freshness. Interpretability at scale also remains a challenge—centroid inspection and feature attribution can help but are not substitutes for domain validation.

Future work should explore: (1) automated **model selection** using Bayesian nonparametrics or information criteria adapted to distributed environments; (2) **approximate neighbor search** to accelerate silhouette and density estimates; (3) **privacy-preserving** clustering with differential privacy; (4) **Delta/Lakehouse** patterns to manage iterative experiments and lineage; and (5) **hybrid CPU/GPU** execution for distance-heavy workloads. With these extensions, Spark MLlib and HDFS can continue to scale unsupervised learning while keeping pipelines maintainable and reproducible.

REFERENCES

- *Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., & Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI) (pp. 15–28). USENIX.*
- *Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. In Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud). USENIX.*
- *Meng, X., Bradley, J. K., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., … Zaharia, M. (2016). MLlib: Machine learning in Apache Spark. Journal of Machine Learning Research, 17(34), 1–7.*
- *Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. In 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST) (pp. 1–10). IEEE.*
- *White, T. (2015). Hadoop: The definitive guide (4th ed.). O'Reilly Media.*
- *Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., … Curino, C. (2013). Apache Hadoop YARN: Yet another resource negotiator. In Proceedings of the 4th ACM Symposium on Cloud Computing (SoCC) (Article 5). ACM.*
- *Bahmani, B., Moseley, B., Vattani, A., Kumar, R., & Vassilvitskii, S. (2012). Scalable k-means++. Proceedings of the VLDB Endowment, 5(7), 622–633.*
- *Lloyd, S. (1982). Least squares quantization in PCM. IEEE Transactions on Information Theory, 28(2), 129–137.*
- *MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability (Vol. 1, pp. 281–297). University of California Press.*
- *Steinbach, M., Karypis, G., & Kumar, V. (2000). A comparison of document clustering techniques. In Proceedings of the KDD Workshop on Text Mining (pp. 1–20).*
- *Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Methodological), 39(1), 1–38.*
- *Reynolds, D. A. (2009). Gaussian mixture models. In S. Z. Li & A. Jain (Eds.), Encyclopedia of biometrics (pp. 659–663). Springer.*
- *Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20, 53–65.*
- *Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(2), 224–227.*
- *Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. Communications in Statistics—Theory and Methods, 3(1), 1–27.*
- *Halko, N., Martinsson, P.-G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review, 53(2), 217–288.*
- *Weinberger, K. Q., Dasgupta, A., Attenberg, J., Langford, J., & Smola, A. (2009). Feature hashing for large scale multitask learning. In Proceedings of the 26th International Conference on Machine Learning (ICML) (pp. 1113–1120). ACM.*
- *Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., & Vassilakis, T. (2010). Dremel: Interactive analysis of web-scale datasets. Proceedings of the VLDB Endowment, 3(1–2), 330–339.*
- *Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., & Zaharia, M. (2015). Spark SQL: Relational data processing in Spark. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (pp. 1383–1394). ACM.*
- *Sculley, D. (2010). Web-scale k-means clustering. In Proceedings of the 19th International Conference on World Wide Web (WWW) (pp. 1177–1178). ACM.*

- Jaiswal, I. A., & Prasad, M. S. R. (2025). Strategic leadership in global software engineering teams. *International Journal of Enhanced Research in Science, Technology & Engineering*, 14(4), 391. https://doi.org/10.55948/IJERSTE.2025.0434
- Tiwari, S. (2025). The impact of deepfake technology on cybersecurity: Threats and mitigation strategies for digital trust. *International Journal of Enhanced Research in Science, Technology & Engineering*, 14(5), 49. https://doi.org/10.55948/IJERSTE.2025.0508
- Dommari, S. (2025). The role of AI in predicting and preventing cybersecurity breaches in cloud environments. *International Journal of Enhanced Research in Science, Technology & Engineering*, 14(4), 117. https://doi.org/10.55948/IJERSTE.2025.0416
- Yadav, N., Gaikwad, A., Garudasu, S., Goel, O., Jain, A., & Singh, N. (2024). Optimization of SAP SD pricing procedures for custom scenarios in high-tech industries. *Integrated Journal for Research in Arts and Humanities*, 4(6), 122–142. https://doi.org/10.55544/ijrah.4.6.12

- Saha, B., & Kumar, S. (2019). Agile transformation strategies in cloud-based program management. *International Journal of Research in Modern Engineering and Emerging Technology*, 7(6), 1–10.
- Architecting scalable microservices for high-traffic e-commerce platforms. (2025). *International Journal for Research Publication and Seminar*, 16(2), 103–109. https://doi.org/10.36676/jrps.v16.i2.55
- Jaiswal, I. A., & Goel, P. (2025). The evolution of web services and APIs: From SOAP to RESTful design. *International Journal of General Engineering and Technology*, 14(1), 179–192.
- Tiwari, S., & Jain, A. (2025). Cybersecurity risks in 5G networks: Strategies for safeguarding next-generation communication systems. *International Research Journal of Modernization in Engineering Technology and Science*, 7(5). https://doi.org/10.56726/irjmets75837
- Dommari, S., & Vashishtha, S. (2025). Blockchain-based solutions for enhancing data integrity in cybersecurity systems. *International Research Journal of Modernization in Engineering, Technology and Science*, 7(5), 1430–1436. https://doi.org/10.56726/IRJMETS75838
- Yadav, N., Dharuman, N. P., Dharmapuram, S., Kaushik, S., Vashishtha, S., & Agarwal, R. (2024). Impact of dynamic pricing in SAP SD on global trade compliance. *International Journal of Research Radicals in Multidisciplinary Fields*, 3(2), 367–385.
- Saha, B. (2022). Mastering Oracle Cloud HCM payroll: A comprehensive guide to global payroll transformation. *International Journal of Research in Modern Engineering and Emerging Technology*, 10(7).
- AI-powered cyberattacks: A comprehensive study on defending against evolving threats. (2023). *International Journal of Current Science*, 13(4), 644–661.
- Jaiswal, I. A., & Singh, R. K. (2025). Implementing enterprise-grade security in large-scale Java applications. *International Journal of Research in Modern Engineering and Emerging Technology*, 13(3), 424. https://doi.org/10.63345/ijrmeet.org.v13.i3.28
- Tiwari, S. (2022). Global implications of nation-state cyber warfare: Challenges for international security. *International Journal of Research in Modern Engineering and Emerging Technology*, 10(3), 42. https://doi.org/10.63345/ijrmeet.org.v10.i3.6
- Dommari, S. (2023). The intersection of artificial intelligence and cybersecurity: Advancements in threat detection and response. *International Journal for Research Publication and Seminar*, 14(5), 530–545. https://doi.org/10.36676/jrps.v14.i5.1639
- Yadav, N., Vivek, A. S., Subramani, P., Goel, O., Singh, S. P., & Shrivastav, A. (2024). AI-driven enhancements in SAP SD pricing for real-time decision making. *International Journal of Multidisciplinary Innovation and Research Methodology*, 3(3), 420–446.
- Saha, B., Pandey, P., & Singh, N. (2024). Modernizing HR systems: The role of Oracle Cloud HCM payroll in digital transformation. *International Journal of Computer Science and Engineering*, 13(2), 995–1028.
- Jaiswal, I. A., & Goel, O. (2025). Optimizing content management systems with caching and automation. *Journal of Quantum Science and Technology*, 2(2), 34–44.
- Tiwari, S., & Gola, D. K. K. (2024). Leveraging dark web intelligence to strengthen cyber defense mechanisms. *Journal of Quantum Science and Technology*, 1(1), 104–126.
- Dommari, S., & Jain, A. (2022). The impact of IoT security on critical infrastructure protection: Current challenges and future directions. *International Journal of Research in Modern Engineering and Emerging Technology*, 10(1), 40. https://doi.org/10.63345/ijrmeet.org.v10.i1.6
- Yadav, N., Bhardwaj, A., Jeyachandran, P., Goel, O., Goel, P., & Jain, A. (2024). Streamlining export compliance through SAP GTS: A case study in high-tech industries. *International Journal of Research in Modern Engineering and Emerging Technology*, 12(11), 74.
- Saha, B., Singh, R. K., & Siddharth. (2025). Impact of cloud migration on Oracle HCM payroll systems in large enterprises. *International Research Journal of Modernization in Engineering Technology and Science*, 7(1). https://doi.org/10.56726/IRJMETS66950
- Jaiswal, I. A., & Khan, S. (2025). Leveraging cloud-based projects (AWS) for microservices architecture. *Universal Research Reports*, 12(1), 195–202. https://doi.org/10.36676/urr.v12.i1.1472
- Tiwari, S. (2023). Biometric authentication in the face of spoofing threats: Detection and defense innovations. *Innovative Research Thoughts*, 9(5), 402–420. https://doi.org/10.36676/irt.v9.i5.1583
- Dommari, S. (2024). Cybersecurity in autonomous vehicles: Safeguarding connected transportation systems. *Journal of Quantum Science and Technology*, 1(2), 153–173.
- Yadav, N., Aravind, S., Bikshapathi, M. S., Prasad, P. M., Jain, S., & Goel, P. (2024). Customer satisfaction through SAP order management automation. *Journal of Quantum Science and Technology*, 1(4), 393–413.

- Saha, B., & Goel, P. (2024). Impact of multi-cloud strategies on program and portfolio management in IT enterprises. *Journal of Quantum Science and Technology*, 1(1), 80–103.
- Jaiswal, I. A., & Solanki, S. (2025). Data modeling and database design for high-performance applications. *International Journal of Creative Research Thoughts*, 13(3), m557–m566. http://www.ijcrt.org/papers/IJCRT25A3446.pdf
- Tiwari, S., & Agarwal, R. (2022). Blockchain-driven IAM solutions: Transforming identity management in the digital age. *International Journal of Computer Science and Engineering*, 11(2), 551–584.
- Dommari, S., & Khan, S. (2023). Implementing zero trust architecture in cloud-native environments: Challenges and best practices. *International Journal of All Research Education and Scientific Methods*, 11(8), 2188.
- Yadav, N., Prasad, R. V., Kyadasu, R., Goel, O., Jain, A., & Vashishtha, S. (2024). Role of SAP order management in managing backorders in high-tech industries. *Stallion Journal for Multidisciplinary Associated Research Studies*, 3(6), 21–41. https://doi.org/10.55544/sjmars.3.6.2
- Saha, B., Jain, A., & Jain, A. K. (2022). Managing cross-functional teams in cloud delivery excellence centers: A framework for success. *International Journal of Multidisciplinary Innovation and Research Methodology*, 1(1), 84–108.
- Jaiswal, I. A., & Sharma, P. (2025). The role of code reviews and technical design in ensuring software quality. *International Journal of All Research Education and Scientific Methods*, 13(2), 3165.
- Tiwari, S., & Mishra, R. (2023). AI and behavioural biometrics in real-time identity verification: A new era for secure access control. *International Journal of All Research Education and Scientific Methods*, 11(8), 2149.
- Dommari, S., & Kumar, S. (2021). The future of identity and access management in blockchain-based digital ecosystems. *International Journal of General Engineering and Technology*, 10(2), 177–206.
- Yadav, N., Bhat, S. R., Mane, H. R., Pandey, P., Singh, S. P., & Goel, P. (2024). Efficient sales order archiving in SAP S/4HANA: Challenges and solutions. *International Journal of Computer Science and Engineering*, 13(2), 199–238.
- Saha, B., & Goel, P. (2023). Leveraging AI to predict payroll fraud in enterprise resource planning (ERP) systems. *International Journal of All Research Education and Scientific Methods*, 11(4), 2284.
- Jaiswal, I. A., & Verma, L. (2025). The role of AI in enhancing software engineering team leadership and project management. *International Journal of Research and Analytical Reviews*, 12(1), 111–119. http://www.ijrar.org/IJRAR25A3526.pdf
- Dommari, S., & Mishra, R. K. (2024). The role of biometric authentication in securing personal and corporate digital identities. *Universal Research Reports*, 11(4), 361–380. https://doi.org/10.36676/urr.v11.i4.1480
- Yadav, N., Abdul, R., Bradley, S., Satya, S. S., Singh, N., Goel, O., & Chhapola, A. (2024). Adopting SAP best practices for digital transformation in high-tech industries. *International Journal of Research and Analytical Reviews*, 11(4), 746–769. http://www.ijrar.org/IJRAR24D3129.pdf
- Saha, B., & Chhapola, A. (2020). AI-driven workforce analytics: Transforming HR practices using machine learning models. *International Journal of Research and Analytical Reviews*, 7(2), 982–997.
- Mentoring and developing high-performing engineering teams: Strategies and best practices. (2025). *Journal of Emerging Technologies and Innovative Research*, 12(2), h900–h908. http://www.jetir.org/papers/JETIR2502796.pdf
- Tiwari, S. (2021). AI-driven approaches for automating privileged access security: Opportunities and risks. *International Journal of Creative Research Thoughts*, 9(11), c898–c915. http://www.ijcrt.org/papers/IJCRT2111329.pdf
- Yadav, N., Das, A., Kar, A., Goel, O., Goel, P., & Jain, A. (2024). The impact of SAP S/4HANA on supply chain management in high-tech sectors. *International Journal of Current Science*, 14(4), 810.
- Implementing chatbots in HR management systems for enhanced employee engagement. (2021). *Journal of Emerging Technologies and Innovative Research*, 8(8), f625–f638. http://www.jetir.org/papers/JETIR2108683.pdf
- Tiwari, S. (2022). Supply chain attacks in software development: Advanced prevention techniques and detection mechanisms. *International Journal of Multidisciplinary Innovation and Research Methodology*, 1(1), 108–130.
- Dommari, S. (2022). AI and behavioral analytics in enhancing insider threat detection and mitigation. *International Journal of Research and Analytical Reviews*, 9(1), 399–416.
- Yadav, N., Krishnamurthy, S., Sayata, S. G., Singh, S. P., Jain, S., & Agarwal, R. (2024). SAP billing archiving in high-tech industries: Compliance and efficiency. *Iconic Research and Engineering Journals*, 8(4), 674–705.
- Saha, B., & Kumar, A. (2019). Best practices for IT disaster recovery planning in multi-cloud environments. *Iconic Research and Engineering Journals*, 2(10), 390–409.

- Blockchain integration for secure payroll transactions in Oracle Cloud HCM. (2020). *International Journal of Novel Research and Development*, 5(12), 71–81.
- Saha, B., Aswini, T., & Solanki, S. (2021). Designing hybrid cloud payroll models for global workforce scalability. *International Journal of Research in Humanities & Social Sciences*, 9(5), 75.
- Exploring the security implications of quantum computing on current encryption techniques. (2021). *Journal of Emerging Technologies and Innovative Research*, 8(12), g1–g18.
- Saha, B., Kumar, L., & Kumar, A. (2019). Evaluating the impact of AI-driven project prioritization on program success in hybrid cloud environments. *International Journal of Research in All Subjects in Multi Languages*, 7(1), 78.
- Robotic process automation (RPA) in onboarding and offboarding: Impact on payroll accuracy. (2023). *International Journal of Current Science*, 13(2), 237–256.
- Saha, B., & Renuka, A. (2020). Investigating cross-functional collaboration and knowledge sharing in cloud-native program management systems. *International Journal for Research in Management and Pharmacy*, 9(12), 8.
- Edge computing integration for real-time analytics and decision support in SAP service management. (2025). *International Journal for Research Publication and Seminar*, 16(2), 231–248. https://doi.org/10.36676/jrps.v16.i2.283