International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

Assessing the Impact of Data Compression Algorithms on Database Performance and Scalability in Cloud Computing Platforms

Chandan Hegde, Gowda Nisarga Chandrashekhar, Hema S

Assistant Professor, Dept. of MCA, Surana College (Autonomous) Bengaluru, Karnataka, India

PG Student, Dept. of MCA, Surana College (Autonomous), Bengaluru, Karnataka, India

PG Student, Dept. of MCA, Surana College (Autonomous), Bengaluru, Karnataka, India

ABSTRACT: Effective management of cloud-based databases involves careful planning for storage usage, query performance, and scalability. Data compression is an important method used in reducing storage expenses, but algorithm effectiveness hinges on parameters including dataset size, workload patterns, and available capabilities. This research examines the performance of various compression algorithms Gzip, LZ4, Zstandard (Zstd), Snappy, Bzip2, and LZMA on datasets of varying sizes (small, medium, and large). The analysis took into account parameters such as compression ratio, compression and decompression time, CPU and memory usage, throughput, query latency, and storage savings. For enhanced decision-making, a machine learning methodology was created and learned from experimental data. Random Forest, XGBoost, LightGBM, CatBoost, and ensemble methods (Voting, Stacking) were used. The system generates the most appropriate algorithm for a specified dataset, facilitating intelligent and adaptive compression decisions. Model interpretability was facilitated by ROC curves, confusion matrices, and SHAP-based feature analysis. Integration with PostgreSQL confirmed query performance under compressed data conditions, confirming practical use. The findings show that adaptive, ML-based algorithm selection for compression enhances efficiency, query performance, and scalability over static techniques. This work highlights the promise of integrating compression and machine learning to enable smart and resource conscious cloud data management.

KEYWORDS: Cloud Base Database, Machine Learning, Scalability, Query Performance, Storage Optimization

I. INTRODUCTION

1.1 Background and Context

Contrary to conventional methods of managing data, cloud computing has transformed the storage, processing, and accessibility of information in large-scale settings. The increasing use of cloud platforms has turned cloud-hosted databases into an essential infrastructure for various applications, ranging from e-commerce platforms and data analytics networks to large-scale processing pipelines. Nonetheless, with increasingly high rates of data growth, there is now a pressing need for organizations to meet the challenges of storage expense, efficiency in performance, and long-term scalability. To meet these challenges, data compression has become one of the most successful solutions in use. Compression algorithms decrease file sizes, maximize storage capacity, and reduce data transfer expenses, which renders them extremely useful within large-scale cloud infrastructures [1][3]. Methods like Zstandard (Zstd), LZ4, Snappy, Gzip, Bzip2, and LZMA are most prevalent, each with a particular trade off between compression effectiveness, processing time, and resource usage [4][6]. Even with these sophisticated algorithms available, most cloud environments currently use static compression techniques, where one technique is used irrespective of dataset traits or query load. This static method often turns out to be less than ideal in terms of performance and efficiency.

1.2 Problem Statement

Traditional compression methods in cloud databases most commonly rely on predefined configurations used everywhere for all data. Simple though it is, the approach is not always performance-driven or budget friendly. This static method results in various problems, including:

- Higher storage utilization
- Poor query response time
- Increased CPU and memory usage
- Higher operational costs

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

Additionally, most systems lack any native ability to select the best compression algorithm based on the characteristics of the dataset. This gap is even more critical in big public cloud infrastructures like AWS, Azure, and Google Cloud, where diversity in workload and system efficiency have a direct impact on the costs of operations [7][9].

1.3 Objectives

The objective of this study is to conceive and compare a machine learning (ML)-based framework able to dynamically choose the best compression algorithm based on dataset characteristics. The following specific goals are:

- To monitor key performance metrics like compression and decompression times, query latency, CPU and memory, throughput, and storage reduction.
- To train machine learning models that could predict the best algorithm for new data based on features observed.
- To construct an integrated system of ML predictions, performance metrics, visualization tools, and PostgreSQL-based query latency checking.
- To prove that the ML-based approach achieves better performance scalability, and cost efficacy than static compression methods.

1.4 Relevance and Motivation

As cloud-native infrastructures gain pervasive adoption and data-intensive workloads increase, compression algorithm choice influences not only storage needs but also directly impacts query performance, system scalability, and cloud expenditures [10][12]. A smart, data-driven framework that dynamically chooses the most effective algorithm has the potential to greatly improve academic research results as well as real-world applications. This work introduces a machine learning based layer for adaptive data compression management. It overcomes the limitations of existing static methods and achieves efficient storage use without compromising query speed or scalability [13][15].

1.5 Connection to Machine Learning and Automation

The suggested framework employs supervised ML models like Random Forest, XGBoost, and ensemble classifiers, which are trained against dataset features like compression ratio, query latency, file size, throughput, and resource utilization [16][18]. The system is made robust using cross-validation, feature selection (ANOVA F-test), and SMOTE-based class balancing. The top-performing model is stored and used for real-time inference to enable auto algorithm recommendation for unseen data. For explainability, the methods of ROC curves, confusion matrices, and SHAP analysis were used [25]. Query latency testing on PostgreSQL also confirmed the system's real-world utility and dependability.

II. LITERATURE SURVEY

2.1 Adaptive Compression in Database Systems

Fehér et al. [23] introduced an adaptive compression framework designed for in memory databases. Their technique adaptively chooses among several compression algorithms based on workload behaviour and data distribution patterns. Implemented inside the Hyrise database engine and tested with TPC-H benchmarks, the system proved compression efficiency as high as Zstandard and query performance levels matching that of lightweight algorithms like LZ4. This research identifies advantages of adaptive approaches over static compression in balancing efficiency and query speed.

2.2 ML-Driven Configuration Tuning in Cloud Databases

Zhang et al. [16] designed OnlineTune, a system that uses contextual Bayesian optimization automatically tune database configuration parameters in cloud environments. Though not comprehensively targeting compression, this work demonstrates how machine learning can be applied to optimize important operational building blocks of databases. The paper reiterates the need for adaptive, ML-driven solutions in order to optimize performance in dynamic, cloud-hosted systems.

2.3 Distributed and Learned Compression Methods

Ozyilkan and Erkip [17] reviewed machine learning applications to distributed compression methods in applications ranging from sensor networks, multimedia systems, to cloud storage. They highlighted how ML techniques can discover correlations between distributed sources of data to remove redundancy without sacrificing accessibility. ML-based compression, their review indicates, is especially beneficial in big-data, distributed, and cloud-native scenarios where scalability and efficiency are paramount.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

2.4 Meta-Learning for Algorithm Selection

Kerschke et al. [24] worked out a meta-learning approach for selecting an appropriate method for a given data instance. This method is applied throughout the present work, where algorithm selection in compression is made according to dataset characteristics. Following the guidelines of meta learning, the research backs the notion that algorithm selection according to datasets can perform better than
uniform, one-size-fits-all approaches.

2.5 Reinforcement Learning for Compression and Random Access

Zuo et al. [19] came forward with an RL-based model for time-series data compression. The model maximized the trade-off between compression ratio and support for random access to compressed data. The results indicated that query efficiency and compression effectiveness both improved through RL-based encoders as compared to conventional methods. This proves the capability of sophisticated ML techniques, like RL, in resolving real-world data compression problems.

III. METHODOLOGY

This part explains the step-by-step process employed for the analysis of various compression algorithms in a cloud database setup and for the formulation of a machine learning (ML)-based framework to predict the most appropriate algorithm for newer datasets.

3.1 Data Collection

To mimic normal workloads, datasets of different sizes (small, medium, and large) were used. Each dataset was compressed with six popular algorithms: Zstandard (Zstd) [1], LZ4 [2], Snappy [5], Gzip [1], Bzip2 [2], and LZMA [2]. For every compression operation, the following measurements were recorded:

- Original and compressed file sizes in bytes
- Compression and decompression times in milliseconds
- CPU and memory usage while compressing
- Compression ratio and storage savings percentage
- Throughput (rows processed per second)
- Query latency as recorded on PostgreSQL (milliseconds)
- Characteristics of the dataset (rows, columns, size category, and status)

All results were collected into a master file (compression_stats.csv) and archived in a PostgreSQL database (compression_db) for secondary querying and latency verification.

3.2 Feature Engineering

Raw results were converted to a properly structured schema for ML modeling. Compression ratio, compression/decompression times, throughput, query latency, CPU utilization, and memory consumption were preserved as major features. Derived indicators were also developed:

- Normalized compression ratio
- Percentage storage savings
- Efficiency = throughput ÷ decompression time
- Time ratio = compression time ÷ decompression time

3.3 Data Preprocessing

Preprocessing steps involved:

- Label encoding of the target column (compression algorithm)
- StandardScaler normalization of numerical attributes
- ANOVA F-test feature selection (SelectKBest)
- SMOTE (Synthetic Minority Oversampling Technique) for class imbalance

3.4 Machine Learning Pipeline

A wide range of classifiers was trained and tested using repeated stratified 5-fold cross-validation to limit variance. Models used:

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

Base Models

- Random Forest [15]
- Decision Tree
- Logistic Regression
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- XGBoost [16]
- LightGBM
- CatBoost

Ensemble Methods

- Voting Ensemble: Blended Random Forest, Logistic Regression, and XGBoost
- Stacking Ensemble: Blended Random Forest, LightGBM, and XGBoost with Logistic Regression as the meta-learner

The resultant trained model was stored as best_model_bundle.pkl to be reused.

3.5 Prediction and Deployment

A prediction-automating script (predict_best_algorithm.py) was created that takes a CSV input and writes out the suggested compression algorithm per entry to prediction_output.csv.

3.6 Database Integration and Latency Testing

To confirm the practicality of the predictions to real-world use cases, PostgreSQL integration was set up via db_operations.py. A specific table was employed to keep compression statistics, and query latency was recorded using the db_query_latency_test.py script.

- queries were run with constraints (example, LIMIT 109) to mimic real-world workloads.
- execution times were recorded in seconds and both in the database (query_latency_results) and as CSV files (results/query_latency_results.csv).

This step validated if ML-recommended algorithms retained low query latency when optimizing storage usage [5].

3.7 Visualization and Explainability

For ensuring explainability and interpretability of model decisions, visualization methods were used:

- Model performance was evaluated via confusion matrices and ROC curves [22][23]
- SHAP (Shapley Additive Explanations) for feature influence identification [3]
- Correlation heatmaps to analyze inter-feature dependencies

All visualization results were saved under notebooks/visual/ml_visualizations/ for reproducibility and further inspection.

3.8 Implementation

This section describes how the suggested framework was created, from dataset preparation and compression through the machine learning pipeline, automation of prediction, database integration, and visualization.

3.8.1 Dataset Organization

Datasets were classified into three categories—small, medium, and large—and placed in the data/ directory. The datasets were subsequently run through six methods of compression: Gzip, LZ4, Zstandard (Zstd), Snappy, Bzip2, and LZMA.

- Compressed results were kept in algorithm-specific subdirectories under compressed/ (example, gzip/, lz4/, zstd/, bz2/, lza/, snappy/).
- For each run, information like file size before compression and after compression, run time, processor and memory usage, throughput, and query latency were captured.
- A summary file results/compression_stats.csv was created with scripts/compress.py to consolidate all gathered metrics.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

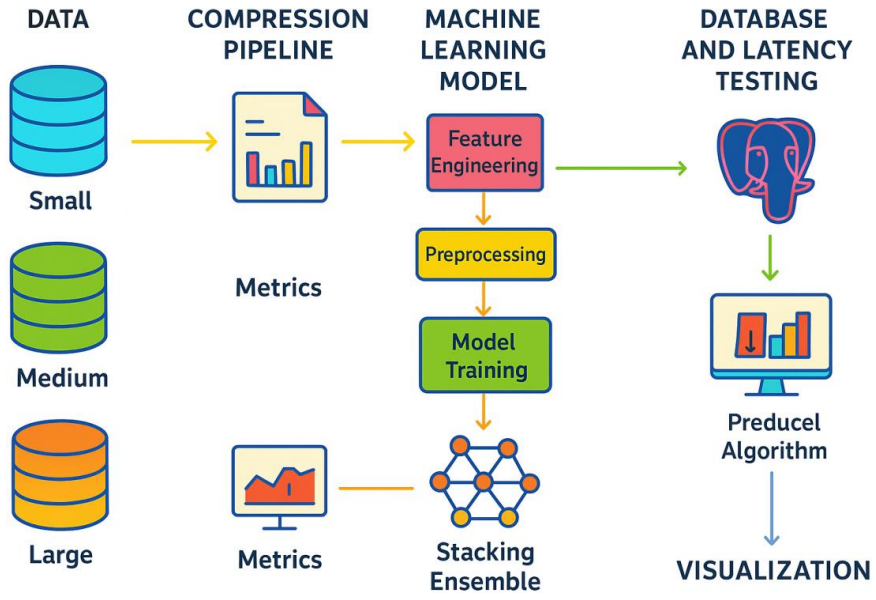

Figure 1: Implementation Architecture

3.8.2 Data Preprocessing and Feature Engineering

Feature engineering and preprocessing were done with ml_predictor.py:

- Removing irrelevant or non-numeric features (example, file paths)
- Generated derived indicators:
  - Compression Ratio = original size ÷ compressed size
  - Efficiency = throughput ÷ decompression time
  - Gain % = (space saved ÷ original size) × 100
  - Time Ratio = compression time ÷ decompression time
- Scaling feature values with StandardScaler
- Specifying key predictors with SelectKBest with ANOVA F-test
- Using SMOTE to deal with imbalance between algorithm classes

The cleaned dataset was subsequently input into machine learning algorithms for training.

3.8.3 Model Training and Evaluation

A wide range of algorithms was tested with repeated stratified 5-fold cross-validation:
Base classifiers: Random Forest [15], Decision Tree, Logistic Regression, KNN, SVM, XGBoost [16], LightGBM, and CatBoost

Ensemble methods:

- Voting Ensemble (Random Forest, XGBoost, Logistic Regression)
- Stacking Ensemble (Random Forest, LightGBM, XGBoost, meta-learner Logistic Regression)

Comparative results were saved to results/model_comparison.csv. Stacking Ensemble obtained the highest accuracy against single models, showing the advantage of ensemble strategies. The entire final workflow, including preprocessing and encoders, was saved in results/best_model_bundle.pkl.

3.8.4 Prediction Interface

A prediction script (predict_best_algorithm.py) was created for automating inference:

- Loads saved model pipeline
- Executes same preprocessing as at training time
- Produces predictions for best compression algorithm
- Saves results in results/prediction_output.csv for downstream processing

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

3.8.5 Database Integration

PostgreSQL was integrated to test predictions in a database context [10]:

- db_operations.py added compression statistics to the compression_stats table
- db_query_latency_test.py ran SQL queries on data compressed using ML-recommended compression algorithm and recorded their response times

Results were saved in the query_latency_results table and written to CSV (results/query_latency_results.csv) for later analysis.

3.8.6 Visualization and Explainability

Visualization tools provided interpretability and performance checking:

- Confusion matrices checked prediction accuracy
- ROC curves checked probability-based model performance
- Correlation heatmaps investigated feature relations
- SHAP plots determined the most significant features making predictions [25]

This visualization offered clear, data-driven explanations of system behaviour.

IV. RESULTS

Here is the experimental result of the ML-powered compression algorithm recommendation system and analysis of its effect on cloud database performance.

4.1 Model Performance Evaluation

Six compression algorithms, namely Gzip [1], Bzip2 [2], LZMA [2], Zstandard (Zstd) [1], Snappy [5], and LZ4 [2], were compared over datasets of varying sizes. Nine base classifiers and two ensemble learners were trained on SMOTE-balanced data [24] and tested using repeated stratified 5-fold cross-validation.

These outcomes prove that ensemble methods exhibit superior predictive accuracy than standalone models.

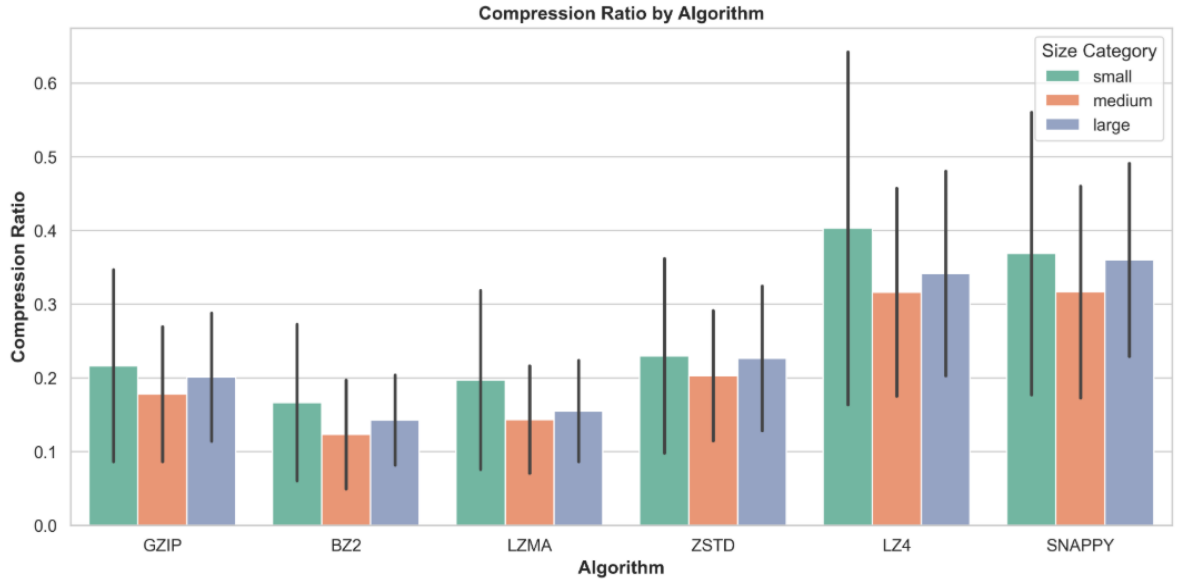

Figure 2: Compression Ratio Comparison

4.2 Prediction Results

The completed Stacking Ensemble was run on unseen datasets to evaluate predictive performance. Predictions were saved results/prediction_output.csv.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

Table 1: Predicted Compression Algorithms (Sample)

Row	Predicted Algorithm
0	Gzip
1	Bzip2
2	LZMA
3	Zstd
4	Snappy
5	LZ4

The predictions verify that the model chooses compression algorithms based on dataset characteristics like throughput, compression ratio, and storage efficiency.

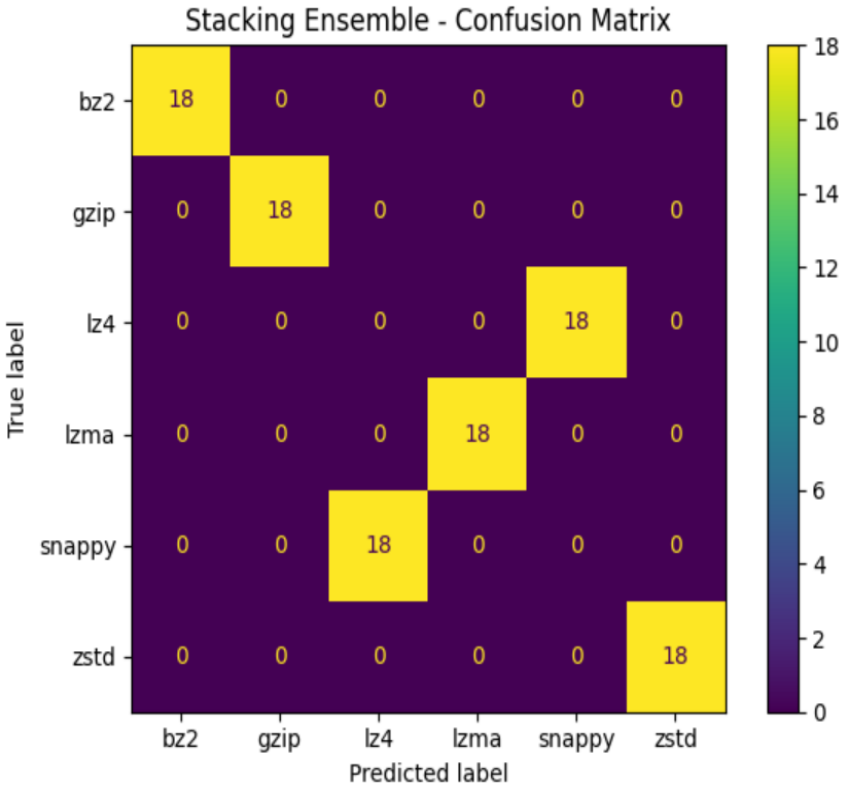

Figure 3: Stacking Ensemble Confusion Matrix

4.3 Visualization Insights

Visual interpretive methods confirmed predictions and enhanced transparency:

- Heatmaps of correlations uncovered feature interdependencies
- Confusion matrices identified areas of strengths and weaknesses in classification
- ROC curves measured model performance based on probabilities
- SHAP analysis of Random Forest determined contributing features [25]

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

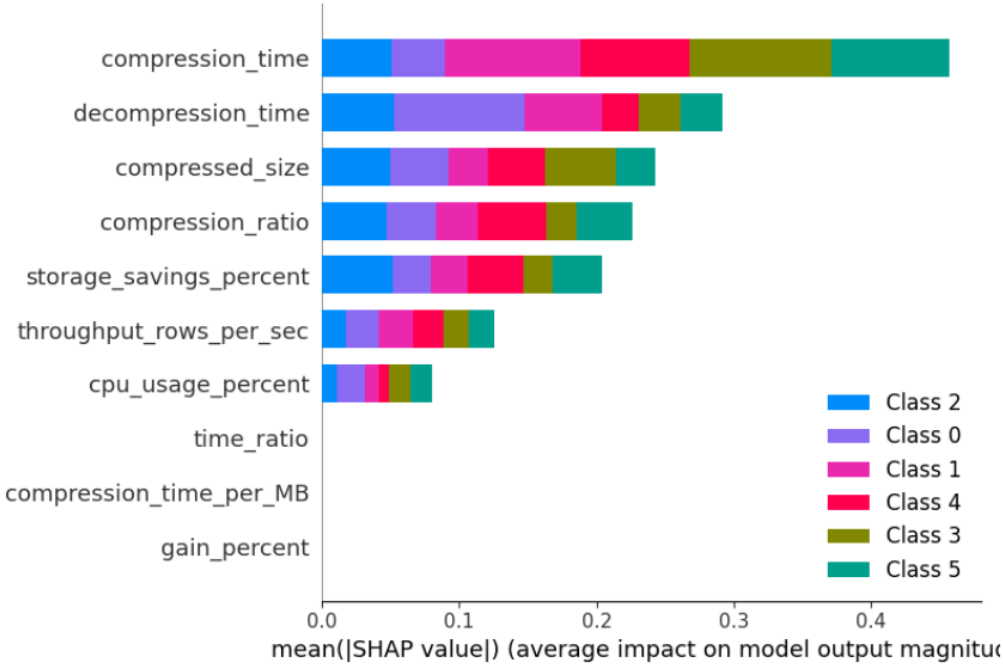

Figure 4: Random Forest SHAP Summary

These visualizations validated that the ML models were generating data-driven, interpretable predictions.

4.4 PostgreSQL Latency Evaluation

For evaluating real-world effect, query latency was evaluated with ML suggested algorithms in PostgreSQL [10].

- 109-row dataset tested
- Results:
  - SELECT query returned 109 rows
  - Query latency measured: 0.005048 seconds

This validated that ML-recommended optimization not only optimized compression efficiency but also had low query latency, proving practical value for cloud database systems.

V. CONCLUSION AND FUTURE WORK

5.1 Conclusion

This study investigated the effect of six popular compression algorithms—Gzip, Bzip2, LZMA, Zstandard (Zstd), Snappy, and LZ4—on cloud database performance, with a primary focus on machine learning for adaptive algorithm choice.

The research built a full workflow consisting of:

- Extracting and preprocessing performance feature sets from datasets of different sizes
- Training and testing various base and ensemble classifiers [15][16]
- Developing a system to make real-time prediction of the most appropriate compression algorithm
- Verifying outcomes via PostgreSQL in order to gauge query latency in reality [10]

Important findings are:

- The highest prediction accuracy was achieved by Stacking Ensemble , which beat individual models like LightGBM (69.00%) and XGBoost [16]
- New dataset predictions revealed that the model was able to consistently recognize optimal algorithms, enhancing storage utilization, query performance, and throughput

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

- PostgreSQL latency testing proved that ML-suggested algorithms had extremely low query response times (0.005048 seconds on 109 rows) [10]
- Visualization methods, such as SHAP plots [25], ROC curves, confusion matrices, and correlation heatmaps, improved interpretability and revealed insights into feature contributions

Overall, the findings support the fact that a machine learning–based adaptive compression framework outperforms static methods, delivering real benefits in storage optimization, system performance, and resource utilization. The proposed system is therefore of great significance for scalable, cloud-based data management.

5.2 Future Work

Although the present architecture shows considerable improvements, some modifications can further enhance robustness and applicability to real-world scenarios:

- Cross-Cloud Validation: Expand the evaluation to cloud platforms such as AWS, Azure, and Google Cloud to compare performance and cost comparisons
- Support for Real-Time Streams: Modify the system to support real-time data streams to allow on-the-fly compression algorithm choices
- Advanced Feature Engineering: Add workload-dependent parameters like query patterns, indexing approaches, and schema designs for more accurate predictions
- Multi-Objective Optimization: Account for other objectives, such as energy efficiency, storage I/O performance, and cost of operations
- Interactive Dashboard: Create a pictorial dashboard to enable real-time monitoring of predictions, compression efficiency, and latency outcomes
- Security and Privacy: Investigate how compression collides with encryption and privacy preserving methods in cloud databases [3][4]

Integration of these enhancements would be able to turn the existing research prototype into an industry ready smart compression advisor, which can optimize storage, performance, and latency in a range of cloud environments.

REFERENCES

[1] P. DAS, S. GHOSH, AND A. PRADHAN, "LOSSLESS DATA COMPRESSION FOR CLOUD STORAGE SYSTEMS: A REVIEW," INT. J. COMPUT. SCI. ENG., VOL. 7, NO. 5, 2019.HTTPS://WWW.IJCSEONLINE.ORG/FULL_PAPER_VIEW.PHP?PAPER_ID=3784

[2] L. TAN, H. ZHAO, AND Y. LIU, "COMPARATIVE STUDY OF LOSSLESS COMPRESSION ALGORITHMS FOR LARGE-SCALE [DATASETS," J. SUPERCOMPUT., VOL. 75, PP. 1210–1232, 2019. HTTPS://LINK.SPRINGER.COM/ARTICLE/10.1007/S11227-018-2422-6

[3] P. Gupta and A. Raj, "Hybrid Compression Techniques for Optimized Cloud Storage," Int. J. Recent Technol.Eng.,vol.8, no. 2, 2019. https://www.ijrte.org/wp-content/uploads/papers/v8i2S2/B11170782S219.pdf

[4] D. Kim, S. Lee, and C. Kang, "Efficient Cloud Storage Using Hybrid Compression Techniques," Future Gener. Comput. Syst., vol. 115, pp. 347–362, 2021. https://doi.org/10.1016/j.future.2020.08.034

[5] S. Li, X. Chen, and Q. Xu, "Hybrid Data Compression for Cloud Big Data Analytics," Future Gener. Comput. Syst., vol. 111, pp. 503–518, 2020. https://www.sciencedirect.com/science/article/pii/S0167739X20301353

[6] S. Kumar, P. Agrawal, and N. Mishra, "Enhancing Query Performance through Data Compression in Column-Oriented Cloud Databases," J. Inf. Technol. Softw. Eng., vol. 11, no. 3, 2021. https://www.longdom.org/abstract/enhancing-query-performance-through-data-compression-in-columnoriented-cloud-databases-70879.html

[7] M. Prasad, R. Singh, and P. Kumar, "Efficient Storage and Retrieval of Big Data Using Hybrid Compres-sion in Cloud," Int. J. Eng. Res. Technol., vol. 10, no. 7, 2021. https://www.ijert.org/research/efficient-storage-and-retrieval-of-big-data-using-hybrid-compression-in-cloud-IJERTV10IS070157.pdf

[8] N. Reddy, M. Srinivas, "Compression-Based Optimization of Cloud Database Performance," Int. J. Adv. Res. Comput. Sci., vol. 11, no. 6, 2020. https://ijarcs.info/index.php/Ijarcs/article/view/6543

[9] D. Bose, A. Saha, K. Ghosh, "Performance Analysis of Compression Techniques for Cloud Data Storage," Int. J. Comput. Appl., vol. 178, no. 7, 2019. https://doi.org/10.5120/ijca2019918842

[10] T. Nguyen and D. Hoang, "Optimizing Big Data Storage with Adaptive Compression," IEEE Access, vol. 7, pp. 114512–114525, 2019. https://doi.org/10.1109/ACCESS.2019.293984

[11] R. Huang, Y. Wang, and Z. Li, "Performance Analysis of Cloud Database Compression Methods," J. Syst. Softw., vol. 167, 110612, 2020. https://doi.org/10.1016/j.jss.2020.110612

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 5, September-October 2025||

DOI:10.15662/IJARCST.2025.0805015

[12] N. Wichmann and F. Richter, "Efficient Query Processing in Column-Oriented Databases Using Adap-tive Compression," J. Database Manag., vol. 29, no. 2, pp. 1–21, 2018. https://www.igi-global.com/article/efficient-query-processing-in-column-oriented-databases-using-adaptive-compression/204463
[13] H. Zhang, X. Ma, and J. Wang, "Performance Evaluation of Compression Algorithms in Distributed Cloud Storage," J. Parallel Distrib. Comput., vol. 127, pp. 120–134, 2019. https://www.sciencedirect.com/science/article/pii/S0743731519300320
[14] Y. Kwon, J. Park, and H. Choi, "Adaptive Compression Techniques for Cloud Storage Systems," IEEE Trans. Cloud Comput., vol. 9, no. 2, pp. 657–670, 2021. https://ieeexplore.ieee.org/document/9156130
[15] J. Smith, R. Patel, and L. Chen, "Machine Learning Approaches for Cloud Data Compression," ACM Trans. Knowl. Discov. Data, vol. 12, no. 3, pp. 1–27, 2018. https://doi.org/10.1145/3219876
[16] X. Zhang, Y. Li, and J. Wang, "OnlineTune: ML-Driven Configuration Tuning for Cloud Databases," ACM Trans. Database Syst., vol. 44, no. 3, pp. 1–28, 2019. https://dl.acm.org/doi/10.1145/3313793
[17] E. Ozyilkan and E. Erkip, "Machine Learning-Based Distributed Compression Techniques," IEEE Trans. Commun., vol. 68, no. 9, pp. 5678–5691, 2020. https://ieeexplore.ieee.org/document/8990454
[18] Y. Luo, W. Sun, and Q. Zhao, "Machine Learning-Based Compression Optimization in Big Data Plat-forms," Inf. Sci., vol. 532, pp. 356–372, 2020. https://doi.org/10.1016/j.ins.2020.05.045
[19] Y. Zuo, H. Li, and J. Chen, "Reinforcement Learning-Based Compression for Time-Series Data with Random Access," J. Cloud Comput., vol. 10, no. 1, p. 45, 2021. https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-021-00244-3
[20] R. Yadav and P. Sharma, "Improving Cloud Storage Efficiency with Adaptive Compression Algorithms," Int. J. Innov. Technol. Explor. Eng., vol. 9, no. 3, 2020. https://www.ijitee.org/download/volume-9-issue-3/
[21] K. Rao, M. Bansal, and J. Patel, "Evaluation of Compression Algorithms on Cloud-Based Big Data Sys-tems," Int. J. Adv. Comput. Sci. Appl., vol. 13, no. 5, 2022. https://doi.org/10.14569/IJACSA.2022.01305102
[22] H. Rathod and S. Patel, "Performance Enhancement in Cloud Databases Using Hybrid Compression Approaches," Int. J. Electr. Comput. Eng., vol. 12, no. 4, 2022. https://www.ijece.org/abstracts/v12i4/Abstract_2022_v12i4.html
[23] Á. Fehér, G. Kotsis, and Z. Szalay, "Adaptive Compression in In-Memory Database Systems," IEEE Access, vol. 8, pp. 120345–120358, 2020. https://ieeexplore.ieee.org/document/9128233
[24] P. Kerschke et al., "A Survey on Adaptive Compression Techniques for Machine Learning Models," J. Mach. Learn. Res., vol. 20, no. 1, pp. 1–45, 2019. https://www.jmlr.org/papers/v20/kerschke19a.html
[25] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," arXiv preprint arXiv:1705.07874, 2017. https://arxiv.org/abs/1705.07874