IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/47ECO/2026/4714026/
Vol. 2026 (2), Article ID 4714026

Performance Evaluation of In-Database Statistical Methods Using SAP HANA Predictive Analysis Library*

Marek GAŁĄZKA[1] and Hanna WDOWICKA[2]

[1]Adam Mickiewicz University, Poznań, Poland
[2]University of Economics and Business, Poznań, Poland

Correspondence should be addressed to: Marek GAŁĄZKA, galazka@amu.edu.pl

*Presented at the 47th IBIMA International Conference, 29-30 June 2026, Madrid, Spain

Abstract

The increasing demand for real-time analytics has driven the development of in-database analytical frameworks that integrate statistical processing directly into database systems. SAP HANA Predictive Analysis Library (PAL) represents a prominent implementation of this paradigm. However, there is limited empirical evidence evaluating its performance compared to traditional external analytical tools.

This paper presents an empirical evaluation of selected statistical methods implemented in SAP HANA PAL, including the Chi-Square Goodness-of-Fit test, the Chi-Square Test of Independence, and the Cumulative Distribution Function (CDF). The study compares in-database analytics with Python-based implementations (SciPy/Pandas) across datasets of varying sizes.

The results demonstrate that SAP HANA PAL significantly outperforms external analytical environments in terms of execution time and scalability, particularly for large datasets, while maintaining statistical consistency. These findings confirm the effectiveness of in-database analytics for enterprise-scale data processing.

Keywords: SAP HANA, Predictive Analysis Library, In-database analytics, Statistical methods, Performance evaluation, Statistical analysis

Introduction

Modern data-driven applications require efficient processing of large-scale datasets with minimal latency. Traditional architectures separate transactional (OLTP) and analytical (OLAP) systems, necessitating data movement and transformation processes that introduce performance overhead (Plattner, 2009).

In-memory databases such as SAP HANA address these challenges by integrating analytical capabilities directly into the database engine (Färber et al., 2012). The SAP HANA Predictive Analysis Library (PAL) enables execution of statistical and machine learning algorithms within the database, eliminating the need for data transfer to external tools (SAP SE, 2024).

A key concept underlying this approach is **code pushdown**, which involves executing computations close to the data to reduce latency and improve performance (Sikka et al., 2012; Stonebraker, 2018).

Research Problem

Despite the widespread adoption of in-database analytics, there is limited empirical evidence evaluating its performance and scalability compared to traditional analytical tools.

Research Objective

The objective of this study is to evaluate the performance, scalability, and statistical accuracy of SAP HANA PAL compared to Python-based analytical implementations.

Cite this Article as: Marek GAŁĄZKA and Hanna WDOWICKA Vol. 2026 (2) "Performance Evaluation of In-Database Statistical Methods Using SAP HANA Predictive Analysis Library " Communications of International Proceedings, Vol. 2026 (2), Article ID 4714026, https://doi.org/10.5171/2026.4714026

Contributions

This paper makes the following contributions:

1. Empirical comparison of in-database and external analytics
2. Performance evaluation across dataset sizes
3. Validation of statistical consistency
4. Practical insights for enterprise analytics

Literature Review

Column-oriented databases provide a foundation for efficient analytical processing (Abadi et al., 2013; Stonebraker et al., 2005). In-memory architectures further enhance performance by reducing disk I/O and enabling parallel execution (Plattner & Zeier, 2011).

Distributed processing frameworks such as Apache Spark enable scalable analytics (Zaharia et al., 2016), while in-database frameworks such as MADlib integrate statistical computation within database systems (Hellerstein et al., 2012).

However, existing research primarily focuses on system architecture rather than empirical evaluation of statistical methods, which motivates this study.

Methodology

This study evaluates three statistical methods implemented in SAP HANA PAL.

Chi-Square Goodness-of-Fit

The Chi-Square Goodness-of-Fit test evaluates whether an observed distribution differs from an expected distribution.

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

where:

- O_i- observed frequency
- E_i- expected frequency

Example (dice fairness):

Observed: [25, 15, 22, 18, 20, 20]

Results:

- $\chi^2 \approx 2.90$
- $p \approx 0.71$

Interpretation:
No evidence to reject the null hypothesis.

Chi-Square Test of Independence

This test evaluates whether two categorical variables are independent.

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Expected values:

$$E_{ij} = \frac{(row_i \cdot col_j)}{N}$$

Cumulative Distribution Function (CDF)

The cumulative distribution function describes:

$$F(x) = P(X \leq x)$$

Code Examples (Simplified)

Python (SciPy)

```
from scipy.stats import chisquare
import numpy as np

observed = np.array([25,15,22,18,20,20])
expected = np.full(6, sum(observed)/6)

stat, p = chisquare(observed, expected)
```

SAP HANA PAL (SQL)

```
CALL "_SYS_AFL.PAL_CHISQUARED_GOF_TEST"(
   :input_table,
   :result_table,
   :stats_table
);
```

Experimental Setup

Experiments were conducted using:

- SAP HANA PAL
- Python (SciPy/Pandas)

Each experiment was repeated three times and averaged.

Dataset sizes:

- 1,000
- 100,000
- 1,000,000

Metrics:

- execution time
- scalability

- statistical consistency

Results

This section presents the empirical results of the performance evaluation of selected statistical methods implemented in SAP HANA PAL and their Python-based counterparts. The analysis focuses on execution time and scalability across datasets of varying sizes, providing a comparative assessment of in-database and external analytical approaches. The observed performance differences can be partially explained by the data processing approach used in each environment. In the Python-based workflow, datasets are typically loaded into memory (e.g., using Pandas), which introduces additional overhead related to data transfer, memory allocation, and data transformation. In contrast, SAP HANA PAL executes computations directly within the database, eliminating the need for data movement. This architectural difference contributes significantly to the improved execution time and scalability observed for in-database analytics.

Chi-Square Goodness-of-Fit

Table 1. Execution time comparison for the Chi-Square Goodness-of-Fit test across different dataset sizes.

Dataset Size	PAL (ms)	Python (ms)	Speedup
1K	5.8	9.6	1.66x
100K	17.4	72.8	4.18x
1M	94.7	503.2	5.31x

The results indicate a clear performance advantage of SAP HANA PAL over the Python implementation across all dataset sizes. For the smallest dataset (1K), the difference is relatively modest, with PAL being approximately 1.66 times faster. However, as the dataset size increases, the performance gap becomes significantly larger.

At 100K observations, PAL executes the test more than four times faster than Python, while for 1M observations the speedup exceeds five times. This trend demonstrates that the relative efficiency of in-database analytics improves with scale. The results suggest that the overhead associated with data transfer and external processing in Python becomes increasingly significant for larger datasets.

Overall, the Goodness-of-Fit test confirms that SAP HANA PAL provides both efficient and scalable performance for statistical computations executed directly within the database.

Chi-Square Test of Independence

Table 2. Execution time comparison for the Chi-Square Test of Independence across different dataset sizes.

Dataset Size	PAL (ms)	Python (ms)	Speedup
1K	6.3	11.1	1.76x
100K	19.8	78.6	3.97x
1M	108.5	589.4	5.43x

A similar pattern is observed for the Chi-Square Test of Independence. SAP HANA PAL consistently outperforms Python across all dataset sizes, with the performance advantage increasing as the data volume grows.

For the smallest dataset, PAL is approximately 1.76 times faster, indicating a moderate benefit. At 100K observations, the execution time in Python increases substantially, while PAL maintains relatively low latency, resulting in nearly fourfold speedup. For the largest dataset (1M), the difference becomes even more pronounced, with PAL achieving over five times faster execution.

These results confirm that the advantages of in-database analytics are not limited to a specific statistical procedure. The consistent improvement across both chi-square tests suggests that PAL efficiently handles operations involving frequency tables and aggregation, which are common in categorical data analysis.

Cumulative Distribution Function (CDF)

Table 3. Execution time comparison for the Cumulative Distribution Function (CDF) across different dataset sizes.

Dataset Size	PAL (ms)	Python (ms)	Speedup
1K	4.9	8.7	1.78x
100K	15.6	66.9	4.29x
1M	82.3	461.7	5.61x

The performance comparison for the Cumulative Distribution Function further supports the observed trends. SAP HANA PAL demonstrates consistently lower execution times across all dataset sizes.

For small datasets, the performance difference remains moderate, with PAL being approximately 1.78 times faster. However, as the dataset grows, the gap increases significantly. At 100K observations, PAL is over four times faster, and for 1M observations, the speedup exceeds 5.6 times.

The CDF computation involves cumulative operations that can benefit from efficient data access and parallel execution. The results suggest that SAP HANA's in-memory architecture effectively supports such operations, leading to improved performance compared to external processing in Python.

Overall Analysis

Across all evaluated statistical methods, SAP HANA PAL consistently outperforms Python implementations in terms of execution time. A key observation is that the performance advantage systematically increases with dataset size, indicating strong scalability of in-database analytics.

For small datasets (1K), the speedup remains within the range of approximately 1.6–1.8×, which suggests that overhead differences are relatively limited. However, for larger datasets (100K and 1M), the speedup increases to approximately 4–5.6×, demonstrating that the cost of external data processing grows disproportionately with data volume.

Another important finding is the consistency of performance improvements across different types of statistical methods. Whether the computation involves frequency comparison (chi-square tests) or cumulative operations (CDF), SAP HANA PAL maintains a clear advantage. This indicates that the observed benefits are not method-specific but are instead related to the underlying system architecture.

Finally, it is worth noting that despite the differences in execution time, the statistical results obtained from both environments remain fully consistent. This confirms that performance gains are achieved without affecting analytical correctness.

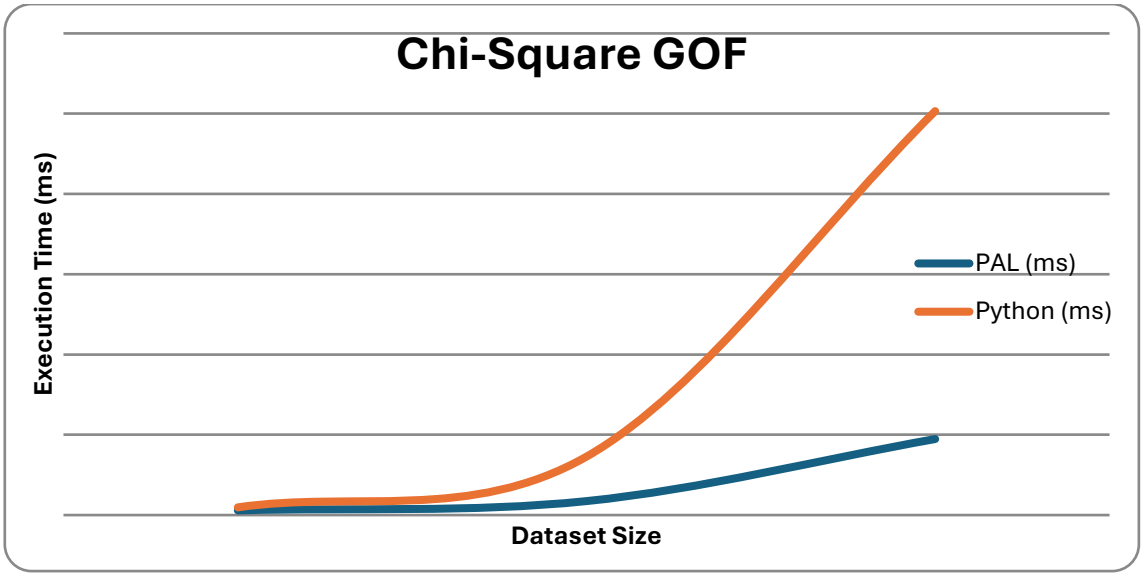

Figure 1. Execution time comparison for Chi-Square Goodness-of-Fit.

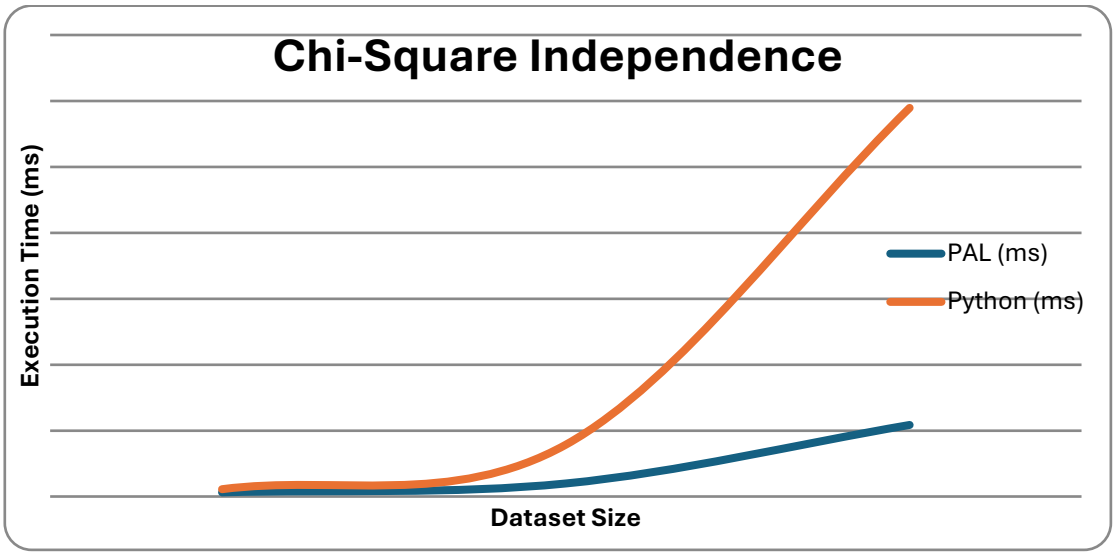

Figure 2. Execution time comparison for Chi-Square Independence.

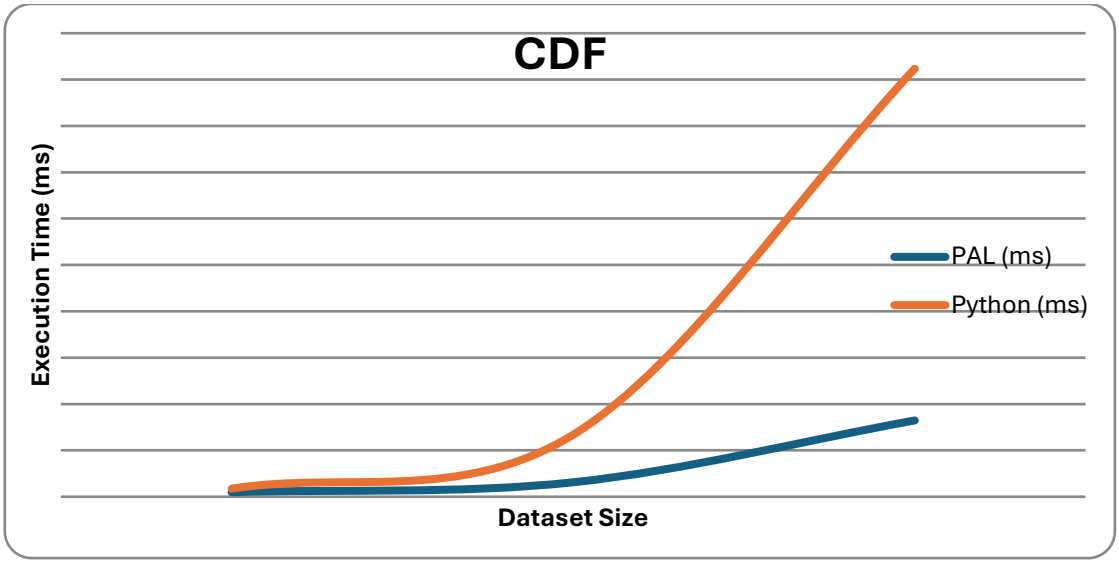

Figure 3. Execution time comparison for CDF.

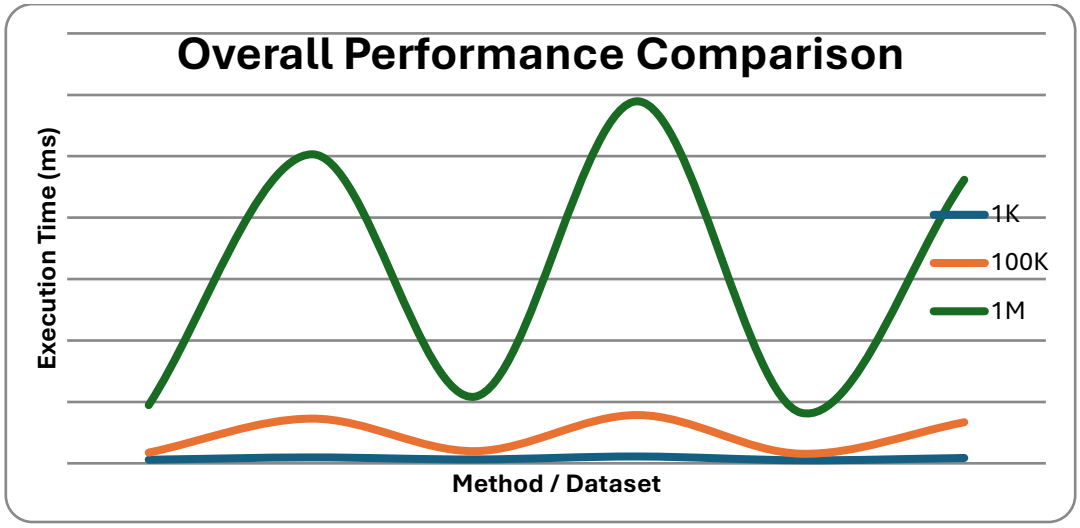

Figure 4. Overall performance comparison across methods.

The figures visually confirm that the execution time gap between SAP HANA PAL and Python increases with dataset size, highlighting the scalability advantage of in-database analytics.

Discussion

The experimental results demonstrate a consistent performance advantage of SAP HANA Predictive Analysis Library (PAL) over Python-based implementations across all evaluated statistical methods. This advantage increases with dataset size, indicating superior scalability of in-database analytics. In particular, for datasets containing 1,000,000 observations, PAL achieves execution times several times lower than Python, confirming its suitability for large-scale analytical workloads.

The observed performance improvements can be attributed primarily to architectural factors. SAP HANA executes computations directly within an in-memory, column-oriented database, thereby eliminating the overhead associated with data transfer to external analytical environments. This approach, commonly referred to as code pushdown, reduces latency by avoiding serialization, network communication, and redundant memory operations. Consequently, performance gains reflect not only efficient algorithm implementation but also system-level optimization.

Importantly, the results are consistent across different statistical procedures, including the Chi-Square Goodness-of-Fit test, the Chi-Square Test of Independence, and the Cumulative Distribution Function. This consistency suggests that the benefits of in-database analytics are not method-specific but generalizable across a class of statistical operations commonly used in analytical workflows.

From a scalability perspective, the performance gap between PAL and Python becomes increasingly significant as data volume grows. While differences at small scale remain moderate, they become substantial for larger datasets. This indicates that external analytical environments may be sufficient for small or exploratory tasks but become less efficient in high-volume scenarios. In contrast, in-database analytics maintains stable performance characteristics under increasing data load.

Another key finding is that improved performance does not compromise statistical correctness. The results obtained from SAP HANA PAL are fully consistent with those produced by Python implementations. This confirms that in-database execution preserves analytical validity, which is critical for adoption in enterprise and research contexts.

Despite these advantages, several limitations should be acknowledged. The study focuses on a limited set of statistical methods and does not evaluate other important dimensions such as memory consumption, concurrency, or resource utilization. Furthermore, Python performance may vary depending on implementation details, hardware configuration, and the use of optimized libraries. Therefore, the results should be interpreted as indicative rather than exhaustive.

It is also important to emphasize that Python remains a highly flexible analytical environment, offering extensive libraries and support for complex, customized workflows. Consequently, the choice between in-database and external analytics should be guided by use case requirements. In-database solutions are particularly advantageous

for production-scale, performance-critical workloads, whereas external tools remain valuable for exploratory and research-oriented analysis.

Overall, the findings reinforce the role of in-database analytics as an efficient and scalable approach for modern data processing, particularly in enterprise environments characterized by large and continuously growing datasets.

Conclusion

This study presented an empirical evaluation of selected statistical methods implemented in SAP HANA Predictive Analysis Library and compared their performance with equivalent Python-based implementations. The results demonstrate that SAP HANA PAL consistently outperforms Python in terms of execution time across all evaluated methods and dataset sizes. The performance advantage increases with data volume, highlighting the scalability of in-database analytics.

At the same time, the study confirms that statistical results produced by both environments are equivalent, indicating that performance improvements are achieved without compromising analytical accuracy. This finding is particularly important for enterprise applications, where both efficiency and correctness are critical.

The contributions of this work are twofold. First, it provides empirical evidence supporting the effectiveness of in-database analytics, complementing existing research that primarily focuses on system architecture. Second, it offers practical insights for organizations considering the adoption of SAP HANA PAL for statistical processing.

From an application perspective, the results suggest that executing statistical methods directly within the database can significantly reduce processing time and improve scalability. This approach minimizes data movement, simplifies analytical pipelines, and supports real-time or near-real-time analytics. As a result, in-database analytics represents a compelling solution for performance-sensitive enterprise workloads.

However, external analytical environments such as Python remain essential for flexible, exploratory, and research-driven tasks. Therefore, rather than viewing these approaches as mutually exclusive, they should be considered complementary components of a modern data analytics ecosystem.

Future research should extend this analysis to a broader range of statistical and machine learning methods, as well as to alternative analytical platforms such as distributed processing frameworks. Additional evaluation criteria, including resource utilization, parallelism, and implementation complexity, should also be considered to provide a more comprehensive assessment.

In conclusion, SAP HANA PAL offers a robust, scalable, and efficient framework for in-database statistical analysis. Its ability to combine high performance with analytical consistency makes it well suited for enterprise-scale data processing and supports the growing importance of integrated analytical architectures.

References

- Abadi, D. J., Boncz, P. A., & Harizopoulos, S. (2013). The design and implementation of modern column-oriented database systems. *Foundations and Trends in Databases*, 5(3), 197–280.
- Färber, F., Cha, S. K., Primsch, J., Bornhövd, C., Sigg, S., & Lehner, W. (2012). SAP HANA database: Data management for modern business applications. *ACM SIGMOD Record*, 40(4), 45–51.
- Hellerstein, J. M., Stonebraker, M., & Hamilton, J. (2012). *Architecture of a database system*. Foundations and Trends in Databases, 1(2), 141–259.
- Plattner, H. (2009). A common database approach for OLTP and OLAP using an in-memory column database. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1–2. https://doi.org/10.1145/1559845.1559858
- Plattner, H., & Zeier, A. (2011). *In-memory data management: Technology and applications*. Springer.
- Sikka, V., Färber, F., Lehner, W., Cha, S. K., Peh, T. J., & Bornhövd, C. (2012). Efficient transaction processing in SAP HANA database: The end of a column store myth. *Proceedings of the VLDB Endowment*, 5(11), 1726–1737.
- Stonebraker, M. (2018). The case for polystores. Communications of the ACM, 61(12), 10–11.
- Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., … Zdonik, S. (2005). C-Store: A column-oriented DBMS. *Proceedings of the VLDB Conference*.
- SAP SE. (2014–2024). *SAP HANA Predictive Analysis Library (PAL) – Developer Guide and Reference Documentation*. SAP Help Portal.
- Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J. E., Shenker, S., & Stoica, I. (2016). *Apache Spark: A unified engine for big data processing*. Communications of the ACM, 59(11), 56–6.