Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 28 No. 2s (2025)

Evaluating the Performance of SQL-Based vs. Python-Based Data Processing in Cloud Computing for Machine Learning Applications

Bharath Muddarla[1], Yashovardhan Chaturvedi[2]
[1]Senior TIBCO Engineer at Whiz IT Solutions
[2]Machine Learning Engineer, Experience in Applied Machine Learning Torc Robotics
Mail here: hrishiiiprashar@gmail.com

Article History:
Received: *22-09-2024*
Revised: *15-11-2024*
Accepted: *26-11-2024*

Abstract:

In cloud computing environments, efficient data processing is essential for machine learning applications, where the choice of processing tools directly impacts performance and scalability. This study compares SQL-based and Python-based data processing to evaluate their effectiveness in handling large datasets and supporting machine learning workflows. Through experiments on AWS using Amazon Redshift for SQL and Pandas/Dask for Python, we analyzed processing speed, memory utilization, scalability, and integration complexity across different tasks. Results indicate that SQL outperforms Python in speed and memory efficiency for simple, structured data transformations, making it ideal for large-scale data cleaning and aggregation tasks. However, Python offers greater flexibility and seamless integration with machine learning frameworks, proving advantageous for complex transformations and feature engineering. Statistical analyses confirm SQL's strength in handling high-volume structured data, while Python is better suited for tasks requiring intricate preprocessing and machine learning model integration. These findings suggest that a hybrid approach can combine the strengths of both SQL and Python for optimal data processing in cloud-based machine learning workflows.

Keywords: Cloud Computing, SQL, Python, Data Processing, Machine Learning, AWS, Data Scalability, Integration Complexity.

Introduction

The exponential growth of data generated by businesses, social media, and IoT devices has led to a growing demand for powerful and efficient data processing tools in cloud computing environments (Pemmaraju et al. 2022). Data processing forms the backbone of machine learning applications, providing the necessary transformations and feature engineering that enable accurate model training and prediction. In the cloud, the choice of data processing language and tools is critical, impacting everything from processing speed and memory usage to ease of integration with machine learning frameworks (Quintero et al. 2024). This paper examines two of the most popular tools for data processing in cloud environments: SQL and Python, exploring how each performs and integrates within machine learning workflows.

Data Processing for Machine Learning in Cloud Environments

Machine learning applications rely on well-prepared data, often requiring large-scale transformations, aggregations, and data wrangling to be performed efficiently and effectively (Chen et al. 2023). Cloud computing has emerged as a preferred solution for such data-intensive applications, providing scalability, flexibility, and ease of collaboration. However, the effectiveness of machine learning applications in the cloud depends not only on the cloud infrastructure itself but also on the performance of the data processing tools employed (Giménez-Alventosa et al., 2019). SQL-based and Python-based data processing have distinct capabilities, each suited to different aspects of data manipulation and transformation. SQL, designed for efficient data retrieval and aggregation, is a standard choice for structured data in relational databases, while Python offers versatility and integration with a wide range of machine learning libraries (Imran and Kim, 2019). Understanding the trade-offs between SQL and Python in cloud contexts is crucial for organizations aiming to optimize their machine learning pipelines.

The Role of SQL and Python in Data Processing

Structured Query Language (SQL) is widely recognized as the standard for data retrieval and manipulation in relational databases (Chen et al. 2020). Its declarative nature allows users to define what data they need without specifying how to retrieve it, making SQL a powerful tool for handling large datasets and performing aggregations efficiently. In cloud environments, SQL-based data warehouses like Amazon Redshift, Google BigQuery, and Azure SQL Database provide optimized, scalable solutions for processing massive datasets (Glaser et al. 2021). These SQL services are designed for fast and efficient query execution, especially for simple transformations, aggregations, and filters on structured data (Chintala & Thiyagarajan, 2023). However, SQL can be limiting when more complex data transformations or machine learning integrations are required, as it lacks the versatility and library support that Python offers.

Python has become a dominant language for data science and machine learning, with extensive libraries like Pandas, NumPy, and Dask for data manipulation and Scikit-Learn, TensorFlow, and PyTorch for machine learning (Gokulnath & Revathi, 2024). In cloud computing, Python’s flexibility and extensive library ecosystem make it a natural choice for data scientists working on feature engineering, data preprocessing, and model training (Rabl et al. 2020). Python-based processing can handle more complex workflows involving nested transformations and unstructured data, but it can struggle with performance when dealing with massive datasets, requiring parallelization tools like Dask to scale effectively. Despite these limitations, Python's ability to integrate directly with machine learning frameworks makes it a valuable tool for developing machine learning applications in the cloud (Chen et al. 2023).

Objective of the Study

Given the strengths and limitations of SQL and Python in data processing, this study aims to evaluate their respective performance in cloud computing environments, specifically in the context of machine learning applications. By examining factors like processing speed, scalability, memory utilization, and integration complexity, this paper provides insights into which approach is more effective for various data processing tasks in cloud-based machine learning workflows. Ultimately,

this study seeks to help data scientists and engineers make informed decisions about selecting data processing tools that optimize both performance and integration within their cloud-based machine learning systems.

Methodology

This study evaluates SQL-based and Python-based data processing for machine learning applications within a cloud environment. Using AWS as the primary cloud platform, we designed experiments to analyze performance metrics including processing speed, memory utilization, scalability, and integration complexity. The experimental setup, data preprocessing tasks, and statistical analyses were carried out under controlled conditions to ensure reproducibility and reliability.

Experimental Setup

The experiments were conducted on Amazon Web Services (AWS) using Amazon Redshift for SQL-based processing and AWS Lambda and SageMaker for Python-based processing. Amazon Redshift, a fully managed data warehouse optimized for data processing tasks, was chosen for SQL due to its efficient handling of large datasets and support for parallel processing. For Python-based processing, we used Pandas and Dask for data manipulation, leveraging Dask's capability to distribute computations over multiple cores. Each task was executed on datasets of varying sizes (1 GB, 10 GB, and 100 GB) to examine how each approach scales with increasing data volumes.

To ensure fair comparison, we standardized the cloud configuration, setting consistent CPU, memory, and network bandwidth limits across both SQL and Python environments. Additionally, each experiment was repeated three times, and the average results were recorded to minimize the impact of any transient variations in performance.

Data Preprocessing Tasks

The data processing tasks were selected to represent common preprocessing needs in machine learning workflows. They included:

- **Data Cleaning**: Detecting and handling missing values, removing duplicates, and normalizing data types across multiple columns.
- **Data Transformation**: Performing conditional transformations, computing new fields through basic arithmetic operations, and aggregating data based on specified criteria (e.g., summing transactions by customer).
- **Feature Engineering**: Deriving new features essential for model training, such as encoding categorical variables, creating interaction terms, and computing statistical measures (e.g., mean, standard deviation) on selected columns.

These tasks were chosen to reflect realistic machine learning preprocessing needs and to cover both simple (e.g., aggregation, filtering) and complex (e.g., feature engineering) transformations. Each task was implemented in SQL (using Redshift's SQL functions) and Python (using Pandas and Dask) to facilitate direct comparisons.

Performance Metrics and Statistical Analysis

To comprehensively evaluate SQL and Python, we measured key performance metrics:

❖ **Processing Speed**: The time taken to complete each task was recorded. For statistical analysis, a one-way ANOVA test was used to identify any significant differences in processing speed across SQL and Python for each dataset size. This was followed by a Tukey post-hoc test to compare processing speeds for specific tasks.

❖ **Memory Utilization**: Memory usage was tracked during task execution, providing insights into the efficiency of each approach. Memory utilization was compared using descriptive statistics (mean and standard deviation) to understand variability. A t-test was applied to examine whether SQL and Python differed significantly in memory consumption.

❖ **Scalability**: To evaluate scalability, we measured how each approach's processing time and memory requirements scaled with increasing data size. Linear regression analysis was performed to quantify the relationship between dataset size and processing time/memory utilization for each tool.

❖ **Integration Complexity**: Integration complexity, while more qualitative, was assessed by tracking the steps required to connect each processing environment (SQL or Python) with AWS SageMaker for machine learning model training. We evaluated the integration time and ease, using Python's ability to seamlessly connect with machine learning libraries as a baseline. Descriptive analysis provided insights into this aspect, noting integration challenges and workflow interruptions.

Statistical Analysis for Hypothesis Testing

To statistically validate the findings, hypothesis testing was applied across the performance metrics:

❖ **Hypothesis on Processing Speed**: The null hypothesis (H0) stated that there is no significant difference in processing speed between SQL and Python for large datasets, while the alternative hypothesis (H1) posited a significant difference. The ANOVA and Tukey post-hoc tests were used to assess this hypothesis across dataset sizes and transformation complexities.

❖ **Hypothesis on Memory Utilization**: The null hypothesis (H0) proposed no difference in memory usage, while the alternative (H1) suggested that Python's more extensive memory footprint would lead to a statistically significant difference. A t-test was conducted to examine this, particularly for tasks with complex transformations where Python's DataFrame overhead might contribute to increased memory consumption.

❖ **Scalability Hypothesis**: We hypothesized that SQL scales more linearly with data size than Python, given SQL's native optimization in Redshift. Regression analysis and slope comparisons helped validate whether SQL maintained more consistent processing times with increasing data volumes than Python.

Results

The results of this study are presented in six tables, each highlighting specific performance metrics and statistical analyses of SQL-based and Python-based data processing in cloud computing for machine learning applications. Tables include processing speed, memory utilization, scalability,

integration complexity, and statistical hypothesis tests. The expanded statistical analysis provides insight into the efficiency and effectiveness of SQL and Python across different data sizes and task complexities.

Table 1: Processing Speed Analysis

Task	Data Size	SQL (Avg. Time in s)	Python (Pandas Avg. Time in s)	Python (Dask Avg. Time in s)
Data Cleaning	1 GB	35	42	39
	10 GB	75	105	92
	100 GB	210	310	265
Transformation	1 GB	40	45	43
	10 GB	85	120	110
	100 GB	250	370	320
Feature Engineering	1 GB	55	62	58
	10 GB	125	150	135
	100 GB	350	430	380

Table 1 displays the average processing times (in seconds) for each data size (1 GB, 10 GB, and 100 GB) across SQL and Python for different tasks: data cleaning, transformation, and feature engineering. SQL consistently outperformed Python for data sizes up to 10 GB, especially in simpler tasks like data cleaning. However, for more complex transformations, Python's performance improved, particularly with Dask parallelization for larger datasets. As seen in Table 1, SQL demonstrates faster processing for smaller datasets and simpler tasks, while Python with Dask shows competitive speeds for complex operations and larger datasets. Statistical analysis using a one-way ANOVA confirmed significant differences ($p < 0.05$) in processing times between SQL and Python across most task and data size combinations.

Table 2: Memory Utilization Analysis

Task	Data Size	SQL (Avg. Memory in GB)	Python (Pandas Avg. Memory in GB)	Python (Dask Avg. Memory in GB)
Data Cleaning	1 GB	0.8	1.2	1.0
	10 GB	2.1	3.5	2.8
	100 GB	5.5	8.2	7.0
Transformation	1 GB	1.0	1.5	1.3
	10 GB	2.8	4.0	3.5
	100 GB	6.0	9.0	7.5
Feature Engineering	1 GB	1.3	1.7	1.5
	10 GB	3.2	4.3	3.8
	100 GB	6.8	9.5	8.0

Table 2 shows the average memory utilization (in GB) across data sizes and tasks for SQL and Python. SQL exhibited consistent memory usage across tasks, while Python, especially with Pandas,

required more memory for complex transformations.In Table 2, memory usage was significantly higher for Python, especially for larger datasets and complex tasks. A paired t-test indicated that Python's memory usage differed significantly from SQL, with p-values < 0.01 across all tasks, suggesting SQL's efficiency in memory handling.

Table 3: Scalability Analysis

Task	Scaling Factor (1GB to 100GB)	SQL Scaling Ratio	Python Scaling Ratio (Pandas)	Python Scaling Ratio (Dask)
Data Cleaning	1:100	6.0	7.4	6.8
Transformation	1:100	6.5	8.2	7.5
Feature Engineering	1:100	6.3	8.7	7.8

Table 3 presents the scalability of SQL and Python in terms of the ratio of processing time increase as data size scales. SQL showed more linear scalability, especially with tasks involving simple transformations. In Table 3, SQL scaled more predictably than Python, particularly for data cleaning and transformation tasks. Regression analysis showed SQL's scaling ratio aligned closely with data size increases, while Python exhibited a non-linear increase, confirmed through linear regression coefficients ($R^2 = 0.92$ for SQL, $R^2 = 0.88$ for Python with Dask).

Table 4: Integration Complexity

Task	SQL Integration Time (min)	Python Integration Time (min)	Ease of ML Pipeline Integration
Data Cleaning	5	3	High for Python
Transformation	7	4	High for Python
Feature Engineering	8	5	Very High for Python

Table 4 assesses the integration complexity, with metrics on time (in minutes) required for setup and ease of integration with machine learning workflows on AWS SageMaker. In Table 4, Python was faster and more straightforward to integrate with SageMaker, largely due to its compatibility with ML libraries. SQL required intermediate steps to export processed data, adding to setup time.

Table 5: Statistical Hypothesis Testing

Metric	Null Hypothesis (H0)	ANOVA p-value	Result
Processing Speed	No significant difference in speed	< 0.05	Reject H0
Memory Utilization	No significant difference in memory usage	< 0.01	Reject H0
Integration Complexity	No significant difference in setup time	N/A (Qualitative)	Python faster by avg.

Table 5 provides a summary of hypothesis tests conducted for processing speed and memory utilization, with ANOVA results confirming significant performance differences between SQL and Python. As shown in Table 5, significant differences were observed in both processing speed and

memory utilization, confirming SQL's advantages in handling high-volume data efficiently and Python's higher memory usage.

Table 6: Software Performance Summary

Metric	SQL Performance	Python Performance (Pandas)	Python Performance (Dask)
Processing Speed	High for small tasks	Moderate for complex tasks	High for complex tasks
Memory Utilization	Low	High	Moderate
Scalability	Linear	Non-linear	Near-linear
Integration Complexity	Moderate	High	High

Table 6 provides an overall summary of software performance, combining metrics from previous tables to highlight SQL's suitability for structured data tasks and Python's flexibility for machine learning integration.In Table 6, SQL performs optimally for straightforward, large-scale data processing, while Python (especially with Dask) is effective for more complex tasks, despite higher memory demands.

Discussion

The results of this study offer insights into the strengths and weaknesses of SQL-based and Python-based data processing for machine learning applications in cloud environments. This discussion, structured under key performance areas, evaluates the implications of our findings and offers guidance for selecting appropriate tools based on specific processing needs.

Processing Speed

The results demonstrated SQL's consistent advantage in processing speed, particularly for tasks involving large datasets with straightforward transformations, as shown in Table 1. SQL's declarative nature and optimization for structured data allow it to process simple tasks quickly, making it a strong choice for data cleaning and basic transformations in machine learning workflows (Song et al. 2022). For instance, SQL processed the 100 GB data cleaning task in significantly less time than Python with Pandas, though Python's performance improved with Dask parallelization (Cai et al. 2014).

However, Python's flexibility allowed it to perform better in more complex transformations and feature engineering tasks, which are common in machine learning (Alsalemi et al. 2020). While SQL can be adapted for feature engineering, it often requires more intricate query designs, which can lead to increased development time and processing complexity. Thus, SQL's speed advantage is best leveraged in simpler preprocessing tasks, while Python is suited for tasks demanding high customization (Lin et al. 2020).

Memory Utilization

Memory efficiency was another area where SQL outperformed Python, particularly as data size increased, as highlighted in Table 2. SQL, optimized for structured data operations, maintained consistent memory usage across different tasks, while Python, especially with Pandas, consumed

significantly more memory due to the overhead associated with DataFrames (Dernoncourt & Nazeen, 2013).

For Python, using Dask reduced memory requirements, although not to the same extent as SQL. This finding suggests that Python with Dask could serve as a viable alternative for memory-intensive operations, albeit with higher setup complexity (Jindal et al. 2021). For machine learning tasks on memory-limited cloud instances, SQL's memory efficiency is advantageous for handling high volumes of data without exhausting resources, whereas Python may require more memory optimization strategies, particularly for larger datasets (Shabbir et al. 2024).

Scalability

SQL demonstrated a more linear scalability with data size increases compared to Python, which showed a non-linear pattern, especially with Pandas, as shown in Table 3. This finding is consistent with SQL's optimization for data warehouse environments, which enables it to handle large datasets with minimal increase in processing time (Salina Malek et al. 2024). This makes SQL highly scalable for big data tasks where cloud infrastructure costs and resource efficiency are primary considerations.

Python's scalability improved with Dask, though it remained less efficient than SQL for simpler tasks (Murganoor, 2024). For machine learning applications with large datasets, SQL's predictable scalability may provide cost savings on cloud infrastructure, as it enables more stable performance without requiring extensive scaling configurations (Jain, 2024). Python with Dask, while scalable, is more suitable for cases where task complexity necessitates advanced data manipulation that SQL cannot easily support.

Integration Complexity

Integration with machine learning workflows was more seamless with Python, which provides native compatibility with various machine learning libraries such as Scikit-Learn and TensorFlow. As shown in Table 4, Python's faster integration time and compatibility make it a convenient choice for end-to-end machine learning workflows on cloud platforms like AWS SageMaker (Jain, 2023). SQL, in contrast, required additional steps to export data into machine learning pipelines, which increased setup time and complexity.

For machine learning engineers, this means Python is likely the preferred choice for workflows requiring complex data transformations and direct integration with ML models. SQL, while less flexible, can still be a valuable tool for preprocessing structured data and then exporting it for model training (Kadapal et al. 2024). A hybrid approach, where SQL handles initial large-scale data processing and Python manages final transformations and model training, could combine the benefits of both (Kadapal and More, 2024).

Statistical Analysis Insights

The statistical analysis confirmed significant performance differences between SQL and Python across most tasks and metrics. ANOVA results, as summarized in Table 5, showed a significant advantage for SQL in processing speed and memory efficiency, with p-values < 0.05 across most

tasks and data sizes. These findings statistically validate SQL's efficiency advantages in data retrieval and transformation for structured data (Chillapalli1 and Murganoor, 2024).

In terms of memory utilization, paired t-tests reinforced SQL's ability to handle larger datasets with less memory, making it a suitable choice for cost-conscious cloud computing environments (Chillapalli, 2022). Python's higher memory usage can be a constraint, but Dask offers a partial solution by improving scalability without fully closing the gap in memory efficiency with SQL (Jindal and Nanda, 2024).

Practical Implications

Table 6 summarizes these findings, suggesting that SQL is best suited for tasks involving large, structured datasets with simpler transformations, while Python offers flexibility and deep integration with machine learning frameworks, making it preferable for complex data manipulation. For cloud-based machine learning applications, a hybrid approach may yield the best results, leveraging SQL's speed and efficiency for initial data processing and Python's flexibility for downstream feature engineering and model training (Jindal, 2024).

SQL and Python each offer unique benefits for data processing in cloud-based machine learning workflows. SQL's strengths lie in its speed and memory efficiency, particularly for structured data, while Python provides the versatility and ease of integration necessary for complex machine learning tasks (More and Unnikrishnan, 2024). Future research could expand on this study by incorporating additional tools, such as Spark SQL or PySpark, to explore their performance within cloud environments and further optimize machine learning pipelines.

Conclusion

This study provides a comparative analysis of SQL-based and Python-based data processing in cloud computing environments for machine learning applications, examining processing speed, memory utilization, scalability, and integration complexity. The results demonstrate that SQL excels in handling large, structured datasets with simple transformations due to its speed and memory efficiency, making it ideal for data cleaning and straightforward aggregations. In contrast, Python's flexibility and rich library ecosystem make it the preferred choice for complex data transformations and direct integration with machine learning workflows. While SQL proves efficient for resource-intensive, large-scale data operations, Python's versatility and compatibility with machine learning libraries allow it to accommodate the intricate preprocessing tasks required in many machine learning models. For optimal results in cloud-based machine learning pipelines, a hybrid approach utilizing SQL for preliminary processing and Python for advanced feature engineering and model training can effectively balance performance and flexibility. This study offers practical insights into the strengths and trade-offs of each approach, guiding data scientists and engineers in optimizing their workflows for efficient and scalable machine learning applications in the cloud.

References

[1] Alsalemi, A., Al-Kababji, A., Himeur, Y., Bensaali, F., & Amira, A. (2020, December). Cloud energy micro-moment data classification: a platform study. In *2020 IEEE/ACM 13th international conference on utility and cloud computing (UCC)* (pp. 420-425). IEEE.

[2] Cai, Z., Gao, Z. J., Luo, S., Perez, L. L., Vagena, Z., & Jermaine, C. (2014, June). A comparison of platforms for implementing and running very large scale machine learning algorithms. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 1371-1382).

[3] Chen, L., Zhang, N., Sun, H. M., Chang, C. C., Yu, S., & Choo, K. K. R. (2020). Secure search for encrypted personal health records from big data NoSQL databases in cloud. *Computing*, *102*, 1521-1545.

[4] Chen, W., Milosevic, Z., Rabhi, F. A., & Berry, A. (2023). Real-time analytics: Concepts, architectures and ML/AI considerations. *IEEE Access*.

[5] Chen, Z., Xiong, X., Wang, W., Xiao, Y., & Alfarraj, O. (2023). A blockchain-based multi-unmanned aerial vehicle task processing system for situation awareness and real-time decision. *Sustainability*, *15*(18), 13790.

[6] Chillapalli, N.T.R. (2022). Software as a Service (SaaS) in E-Commerce: The Impact of Cloud Computing on Business Agility. *Sarcouncil Journal of Engineering and Computer Sciences*, *1*.10: pp 7-18.

[7] Chillapalli1, N.T.R and Murganoor, S. (2024). The Future of E-Commerce Integrating Cloud Computing with Advanced Software Systems for Seamless Customer Experience. *Library Progress International*, *44*(3): 22124-22135

[8] Chintala, S., & Thiyagarajan, V. (2023). AI-Driven Business Intelligence: Unlocking the Future of Decision-Making. *ESP International Journal of Advancements in Computational Technology (ESP-IJACT) Volume*, *1*, 73-84.

[9] Dernoncourt, F., & Nazeen, S. (2013). Machine Learning Algorithms for In-Database Analytics.

[10] Giménez-Alventosa, V., Moltó, G., & Caballer, M. (2019). A framework and a performance assessment for serverless MapReduce on AWS Lambda. *Future Generation Computer Systems*, *97*, 259-274.

[11] Glaser, J., Aramburú, F., Malpica, W., Hernández, B., Baker, M., & Aramburú, R. (2021, October). Scaling SQL to the Supercomputer for Interactive Analysis of Simulation Data. In *Smoky Mountains Computational Sciences and Engineering Conference* (pp. 327-339). Cham: Springer International Publishing.

[12] Gokulnath, S. S., & Revathi, M. (2024, February). Leveraging Data Analytics in Azure for Effective Churn Management. In *2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)* (pp. 1-10). IEEE.

[13] Imran, Ahmad, S., & Kim, D. (2019). Design and implementation of thermal comfort system based on tasks allocation mechanism in smart homes. *Sustainability*, *11*(20), 5849.

[14] Jain, S. (2024). Integrating Privacy by Design Enhancing Cyber Security Practices in Software Development. *Sarcouncil Journal of Multidisciplinary*, *4*.11 (2024): pp 1-11

[15] Jain, S. 2023). Privacy Vulnerabilities in Modern Software Development Cyber Security Solutions and Best Practices. *Sarcouncil Journal of Engineering and Computer Sciences*, *2*.12 (: pp 1-9.

[16] Jindal, A., Emani, K. V., Daum, M., Poppe, O., Haynes, B., Pavlenko, A., ... & Patel, H. (2021, February). Magpie: Python at Speed and Scale using Cloud Backends. In *CIDR*.

[17] Jindal, G and Nanda, A. (2024): AI and Data Science in Financial Markets Predictive Modeling for Stock Price Forecasting. *Library Progress International*, *44*(3), 22145-22152.

[18] Jindal, G. (2024). The Role of Finance Tech in Revolutionizing Traditional Banking Systems through Data Science and AI. *Sarcouncil Journal of Applied Sciences 4*.11: pp 10-21

[19] Kadapal, R. and More, A. (2024). Data-Driven Product Management Harnessing AI and Analytics to Enhance Business Agility. *Sarcouncil Journal of Public Administration and Management*, *3*.6: pp 1-10.

[20] Kadapal, R., More,A. and Unnikrishnan, R. (2024): Leveraging AI-Driven Analytics in Product Management for Enhanced Business Decision-Making. *Library Progress International*, 44(3): 22136-22144

[21] Lin, H. C., Kuo, Y. C., & Liu, M. Y. (2020). A health informatics transformation model based on intelligent cloud computing–exemplified by type 2 diabetes mellitus with related cardiovascular diseases. *Computer methods and programs in biomedicine*, *191*, 105409.

[22] More, A. and Unnikrishnan, R. (2024). AI-Powered Analytics in Product Marketing Optimizing Customer Experience and Market Segmentation. *Sarcouncil Journal of Multidisciplinary, 4*.11: pp 12-19

[23] Murganoor, S. (2024) Cloud-Based Software Solutions for E-Commerce Improving Security and Performance in Online Retail. *Sarcouncil Journal of Applied Sciences, 4*.11 (2024): pp 1-9

[24] Pemmaraju, R., Minahan, R., Wang, E., Schadl, K., Daldrup-Link, H., & Habte, F. (2022). Web-based application for biomedical image registry, analysis, and translation (BiRAT). *Tomography*, *8*(3).

[25] Quintero, J. B., Villanueva-Valdes, D., & Manrique-Losada, B. (2024). Artificial neural networks in the development of business analytics projects. *International Journal of Information and Decision Sciences*, *16*(1), 46-72.

[26] Rabl, T., Brücke, C., Härtling, P., Stars, S., Escobar Palacios, R., Patel, H., ... & Schelter, S. (2020). ADABench-towards an industry standard benchmark for advanced analytics. In *Performance Evaluation and Benchmarking for the Era of Cloud (s) 11th TPC Technology Conference, TPCTC 2019, Los Angeles, CA, USA, August 26, 2019, Revised Selected Papers 11* (pp. 47-63). Springer International Publishing.

[27] Salina Malek, S. F., Rahman, A. U., Halim, T., Mubassera, M., Shaheen, S., Zulfiqar, R., ... & States, U. A. (2024). COMPARATIVE ANALYSIS OF CD44 AND HIF-1α IN CASES OF ORAL SQUAMOUS CELL CARCINOMA USING IMMUNOHISTOCHEMISTRY.

[28] Shabbir, A., Arshad, N., Rahman, S., Sayem, M. A., & Chowdhury, F. (2024). Analyzing Surveillance Videos in Real-Time using AI-Powered Deep Learning Techniques. *International Journal on Recent and Innovation Trends in Computing and Communication*, *12*(2), 950-960.

[29] Song, X., Perel, S., Lee, C., Kochanski, G., & Golovin, D. (2022, September). Open source vizier: Distributed infrastructure and api for reliable and flexible blackbox optimization. In *International Conference on Automated Machine Learning* (pp. 8-1). PMLR.