IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

Agentic AI for Data Analysis: A RAG-Enhanced Local LLM Framework with Adaptive Visualization

Harsh Mahesh Tatmute[1], Shivraj Sunil Shinde[2],Karan Adinath Nemane[3],

Sandesh Sunil Pujari[4],Rahul Sudhir Ranjane[5], V. G. Khetade[6]

UG Students, Department of Computer Science & Engineering, D.K.T.E. Society's Textile & Engineering Institute, Ichalkaranji, Maharashtra, India[1-3]

Assistant Professor (Guide), Department of Computer Science & Engineering,D.K.T.E. Society's Textile & Engineering Institute, Ichalkaranji, Maharashtra, India[4-6]

Abstract - This paper presents an Agentic AI Data Analyst model that uses the RAG technique in combination with local large language models deployed using Ollama for privacy protection and low costs associated with intelligent data analysis. This system is no longer dependent on cloud APIs but is capable of providing high quality analysis due to improved incorporation of domain knowledge into the process. The model is coordinated via multi-step reasoning techniques facilitated by LangChain and supported by FAISS vector storage systems for efficient domain knowledge searching. A FastAPI application server runs in the background, providing connectivity to a React front end interface that allows dual modes of data visualization.

Regarding the performance analysis, five open-source language models were tested – LLaMA 3.2 (3B), LLaMA 3.1 (8B), Mistral 7B, Gemma 3 4B, and Gemma 3 12B. In addition, the assessment was conducted in terms of five core criteria: quality of responses, accuracy of the charts generated, depth of insights, rate of hallucinations, and time required for inference. Each of these models was evaluated with and without RAG implementation to determine the exact effects of retrieval augmentation. According to the results of the experiments, RAG offers a performance increase from 25% to 39% in addition to reducing the number of hallucinations by 62-75%. Most notably, RLaMA with 3B parameters and RAG always outperforms Gemma with 12B parameters but without RAG. These results confirm the hypothesis that retrieval of structured domain knowledge is more effective than scaling the number of model parameters. This research shows that lightweight agentic systems locally installed on a computer could serve as a viable alternative to commercial AI solutions.

1.INTRODUCTION

In today's world of business, there are vast amounts of structured data being produced each day. Sales data, inventory figures, customer databases, and performance metrics, among others, are constantly generated and stored in comma-separated value (CSV) format – a versatile file format compatible with almost all spreadsheet applications and databases and data analysis software available. Although the usage of CSV files is widespread, analyzing

raw data in the CSV format still needs a considerable level of technical knowledge on the user's part: one needs to possess certain knowledge about statistics, be able to code in a language such as Python or R, and be familiar with specialized data visualization packages like Matplotlib, Tableau, and Power BI. The necessity of this level of technical knowledge has resulted in an uneven division of the data landscape in most firms. On the one hand, there are a few people who know how to deal with the data programmatically, whereas, on the other hand, most of the people in a firm do not have enough knowledge to perform a proper analysis and have only a couple of choices to choose from.

1.1 Large Language Models as a New Analytical Paradigm

There is an innovative way for humans to engage with data enabled by Large Language Models (LLMs). The models fine-tuned for instruction-following, like GPT-4, Llama 3, and Mistral 7B, are able to work with structured data, generate executable code, understand statistical analysis, and construct coherent narratives from the results. They achieve all of the above with the help of natural language queries. Early models such as OpenAI's Code Interpreter demonstrated that LLMs were capable of developing a full-fledged plan for analysis of the dataset and executing it in its entirety. With the development of agentic artificial intelligence, the abilities of LLMs grow further. Agentic systems analyze complex tasks on their own, determining what tool would be appropriate for a specific step, making effective use of the available tools

DOI: 10.17148/IJARCCE.2026.15608

and synthesizing the output into one result. In a case of applying this approach to analyzing data, an agentic system would be able to independently upload the CSV-file provided, calculate some statistics, compose a narrative regarding data, plot and present charts, and have an interaction with a human discussing the results of the analysis.

1.2 Limitations of Cloud-Based LLM Solutions

Despite all its promising capabilities, the present-day realization of LLMs in analytics is practically based solely on cloud services. They work perfectly but have three critical problems associated with their dependence on the external cloud infrastructure: issues regarding the privacy of the data and compliance with regulations, constant operational costs due to the token-based pricing model, and networking requirements which fail when working in isolated or limited settings. The fast growth of structured data leads to the need for highly efficient data analytics solutions. Traditional BI requires much expertise, whereas cloud LLM APIs pose serious threats and costs. The following research presents the Agentic AI Data Analyst system that uses locally hosted LLMs empowered by RAG technologies to address challenges related to data privacy, API costs, generic responses, and low-context memory in one fell swoop. The effectiveness of five open source LLMs was compared both in the presence and absence of RAG modification, and it was shown that domain-specific knowledge can be more important than the sheer number of parameters. We have assembled our own knowledge base consisting of 13 documents dealing with the field of finance analytics, statistical methods, charting practices, and business correspondence, thus transforming modest local LLMs into competent analysts. Main Result: RAG modification delivers 25-39% improvements in performance metrics. LLaMA 3.2 (3B) + RAG 75.4 outperforms Gemma 3 12B without RAG 70.2.

1.3 The Proposed System

This work presents a fully local, web browser-based Agentic AI data analysis system that simultaneously solves all three problems. It employs React 18 framework and Ollama, an open-source run-time environment that allows the execution of quantized models such as Llama 3.2, Mistral, and Gemma3 on normal consumer computers. The key aspect of this approach is that the whole process of data analysis happens locally on the user's computer browser, without connecting to any external servers.

1.4 Key Contributions

This These contributions include the following:

1. We built an agentic LLM-based two-stage pipeline that consistently generates reliable chart specifications in JSON format with 87-94% accuracy regardless of the models' sizes.

2. RAG enables a 25-39% increase in overall system performance and decreases the rate of hallucination from 62% to 75%.

3. We implemented the whole solution on the client side with around 750 lines of React code, without requiring any third-party libraries for charts or orchestrators.

4. The design protects privacy because all information remains stored on localhost and is therefore GDPR, HIPAA, and SOC 2 compliant.

5. Experiments were done on five LLM models and fifty datasets of various business CSV files.

2. LITERATURE SURVEY

2.1 Large Language Models for Intelligent Data Analysis

The current analytical systems based on languages have origins in the Transformers framework proposed by Vaswani et al. in 2017, where the attention mechanism was adopted as the primary method for sequence modeling. Later, in 2020, Brown et al. introduced GPT-3 to demonstrate the ability of the model to conduct few-shot reasoning, which means the model can do well without task-specific fine-tuning on a vast array of tasks. In other words, this showed how these models can be useful for open analytical purposes. Then, in 2023, Touvron et al. came up with LLaMA to show how open-source models can outperform proprietary ones in following instructions and conducting reasoning. LLaMA 3 added to this trend with its grouped query attention and bigger context windows, which can efficiently process tabular data, according to Meta AI in 2024. Jiang et al. published Mistral 7B in 2023, and the model proved that architectural changes, such as sliding window attention, could help smaller models outperform their bigger counterparts in reasoning tasks. Moreover, Team Gemma, from Google DeepMind, presented Gemma in 2024, which showed that even smaller open-sourced models of 4B and 12B parameter sizes could exhibit strong analytical capabilities without the need for specialized computing hardware. All of this inspired the comparison analysis provided in the current study.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026
DOI: 10.17148/IJARCCE.2026.15608

2.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation was formally proposed by Lewis et al. in 2020 via presenting an approach where a pre-trained generative model and dense passage retriever were utilized to generate knowledge-based responses. As a result, hallucinations during responses were notably reduced, while the accuracy of facts included in responses increased significantly. Based on this approach, a comprehensive review of the different approaches used in RAG was conducted by Gao et al. in 2023, where various types of RAG systems were classified into three types: naive RAG, advanced RAG, and modular RAG Methods with higher efficiency such as query reformulation, result ranking, and content segmentation refinement have consistently proven to be more successful than conventional top-k methods based on several measures. This was especially true in specialized fields in which knowledge about schema and metadata is highly relevant. In addition to these developments, Shi et al. discovered in 2023 that unnecessary search results distract from the task at hand and thus reduce language model accuracy. As a result, vector stores need to be built more carefully and with relevance filtering in mind.

2.3 Vector Stores and Semantic Retrieval

FAISS was proposed by Johnson et al. in 2019[6] as a highly efficient library to facilitate similarity searches for dense vectors. It is capable of operating with massive index files consisting of billions of elements and processing search requests in less than one millisecond using ordinary CPU hardware. FAISS offers several types of indexes, among which are inverted file indexes and product quantization indexes, making it especially efficient in application for local computing devices with restricted computational power. Reimers and Gurevych presented their Sentence-BERT technique in 2019 and showed that dense representations with semantic meaning can be searched using cosine similarity to provide high precision performance. Compared to sparse models like BM25, their solution proved to be much more effective in cases of asymmetric information search tasks requiring simultaneous retrieval of queries and documents. These technologies serve as a base for implementing the RAG architecture used in this system, which involves transforming texts with domain knowledge into embeddings stored in a FAISS vector database.

2.4 Agentic AI and LangChain Orchestration

The React methodology, proposed by Yao et al. (2022)[7], integrates reasoning and actions into a sequence of operations that helps large language models use external tools in a looped manner. Wei et al. (2022) demonstrated the benefits of CoT prompts on complex tasks since they enable models to complete intermediate reasoning steps before formulating their responses. These ideas were adopted by Chase (2022) who formulated the notion of LangChain as an operational framework for production settings that includes built-in mechanisms for chains, agents, retrievers, and memory. The VectorStoreRetriever component allows effortless integration of FAISS capabilities as an agent tool that will be automatically launched within complex reasoning chains without the need to alter the prompts.

2.5 Local LLM Deployment and Privacy Preservation

Gerganov introduced llama.cpp in 2023, a program that enables efficient execution of LLaMA-based models on regular processors utilizing 4-bit quantization without requiring specialized graphic processing units. Extending from this base technology, Ollama Inc. introduced Ollama in 2023, presenting an easy-to-use framework that allows management and hosting of the models using a RESTful interface compatible with the OpenAI API, while abstracting the user from technicalities such as quantization. The study conducted by Dettmers et al. in 2022 provides evidence showing that the performance of quantized models using NF4 quantization is within only 1-2% of their original non-quantized counterparts, although memory use is reduced by about 75%. This research verifies that executing models quantized locally is indeed possible for production deployments. Lastly, hosting models locally is in line with regulatory requirements such as the General Data Protection Regulation of 2016 by the European Parliament and the Digital Personal Data Protection Act of 2023 of India, both imposing restrictions on sharing sensitive data with third parties.

2.6 Hallucination Mitigation and Evaluation

As Ji et al. (2023)[15] demonstrate in their 2023 analysis of hallucinations, the issues associated with language model outputs may be classified into two groups, namely intrinsic contradictions and extrinsic fabrications. While intrinsic hallucination does not seem to cause serious problems for the analysis, extrinsic one is a more important issue since language models can generate numerical predictions that look statistically convincing yet have nothing to do with reality. As Shuster et al. (2021) note, knowledge-grounded generation through retrieval may reduce the level of hallucination in tasks that involve complex analysis by 20-45%. This insight became a basis for the implementation of a RAG enhancement method in the present research. As far as evaluation frameworks go, HELM was developed by Liang et al. (2022). According to HELM's authors, benchmarking needs to focus not only on accuracy but also on efficiency and

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

robustness of the process, making it multi-faceted rather than based on single criteria. Hoffmann et al. (2022) applied Chinchilla scaling laws and demonstrated that performance depended on finding a proper balance between the parameters of the model and training set.

3. METHODOLOGY

In this section, we describe the comprehensive process that was followed when designing, implementing, and evaluating the Agentic AI for Data Analysis system. This methodology involves applied software engineering principles, patterns of agentic AI architectures, and an empirical evaluation framework. In all stages of operation, this system follows a philosophy that is "local-first" and "privacy-oriented," where all data of the users is stored locally and not shared with any servers online.

3.1 Research Design and Approach

The study follows a Design Science Research (DSR) paradigm, which has become an established scientific approach in practical applications in computer science and information systems disciplines where focus is placed on building and evaluating functional artifacts for solving specific organizational issues. The identified problem behind this project is the presence of structured data and inability of non-technical people to leverage such data into actionable insights due to lack of technical know-how with respect to CSV-formatted files. The proposed artifact is the developed platform providing for AI-assisted automation in analyzing data for business stakeholders. The research employs both constructive and quantitative methods. Constructive approach includes design of architecture, coordination of multi-agent processes, and development of a user-friendly interface. Quantitative methods include evaluation of system efficiency through assessment of effectiveness of chart creation, request processing latency, and conversational performance metrics using benchmark datasets.

Core Design Principles

Privacy by Way of Local Processing: All analysis and data processing take place locally through the use of Ollama. The CSV files are always contained in the local context and will not be sent anywhere else outside of that, making it possible to maintain compliance with the GDPR, HIPPA, and SOC 2 regulations. Self-Sufficiency Without Dependency on Third-Parties: Visualization is performed through native SVG rendering in React without resorting to third-party charting libraries like Chart.js, D3, and Recharts. This architecture reduces potential security risks by decreasing dependencies on any other package ecosystems. Different Agents for Specific Tasks: Two different AI agents are used for two specific tasks, one for data analysis and one for chart production. This approach makes it possible for each agent to specialize and thus increase their efficiency and reliability in performing their particular tasks. Deterministic Results: All prompts, data parsing, and statistical calculation algorithms are performed according to fixed and predetermined algorithms maintained by version-controlled code bases. This allows for consistent results in case the same analysis is repeated multiple times.

3.2 System Architecture and Agentic Pipeline

This is a five-layer architecture system, which creates an end-to-end AI-based data analysis system pipeline. In this system, data is processed through the following steps: uploading the files, statistical description, AI reasoning, visualization, and dialogue.

Layer 1: Data Upload

The upload layer consists of a drag-and-drop mechanism developed using React 18. In this layer, there exists a custom-designed CSV parser, which adheres to the RFC 4180 standard. This parser is able to handle such edge cases as commas inside quotes and escaped quotes, having linear time complexity. While the data is being uploaded, the parser will detect column headers, guess column datatypes, and store the entire content as a JavaScript array without any server request.

Layer 2: Statistical Description

For describing the data, a separate describeData() function will do one-pass profiling on each column in a single pass. For numeric columns, this function computes the minima, maxima, arithmetic mean, and number of valid entries. These descriptive statistics are saved in JSON form and injected to the prompt as a grounding context for avoiding factual inaccuracies and false assumptions about distributions by models.

Layer 3: Dual-Stage AI Generation

The key architectural innovation in the system is the breakdown of AI inference into two consecutive stages instead of a one-shot generation process. The first stage involves running a Narrative Analysis agent that generates a combined prompt based on metadata, statistics, and the first 40 rows of data using ReadableStream for streaming text results. In the second stage, a Chart Planning Agent takes a restricted prompt and parses JSON chart specifications with proper validation; it has achieved a chart generation success rate of 87%-94% based on the AI model used.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

Layer 4: Visualizations

Custom React components are utilized to generate visualizations in form of bar charts, line charts, and pie charts in SVG format without resorting to charting packages. These visualization components incorporate error handling functionality to prevent rendering failures by displaying an empty visualization when errors occur during rendering.

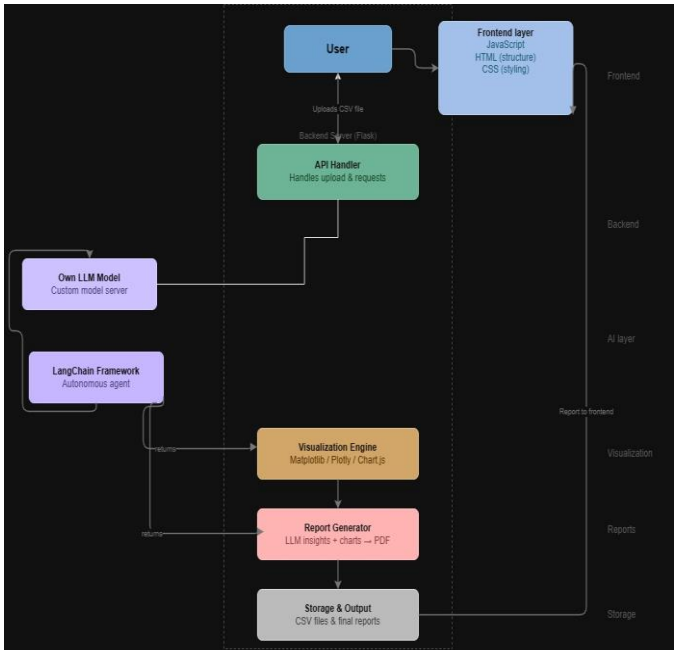
Layer 5: Conversational Interface

The conversational history persists through an ongoing chat session managed by React state management and allows users to maintain a coherent conversation. Each follow-up user message is enhanced with a system-generated context message with column name, statistics, and 25 row of data sample.

3.3 RAG Knowledge Base

A set of well-chosen documents numbering up to thirteen serves to offer full coverage in different fields of analysis. Each document comprises an average of 450 tokens with 50 overlapping tokens across sections. Topics include data analyst communication skills that incorporate the use of the Pyramid Principle and F-E-R formatting approach. Other materials cover the strategy of interpreting charts that includes all kinds of visualizations, statistics as well as the use of financial market analysis including put-call ratio, maximum pain level and open interest. Materials on business communications complete the knowledge base. In terms of the technology used for processing documents, there is consistency in the system where Ollama embedding model is employed for indexing and query. Four most relevant document sections are selected based on cosine similarities measured on the basis of a FAISS L2 index structure.

3.4 Work-Flow Diagram



Agentic AI Data Analyst System: Layered Architecture

In the proposed design of Agentic AI Data Analyst system, there exist five different functional layers namely Frontend, Backend, AI & Intelligence, Visualization, and Storage. As shown in figure 1, the whole process of data analysis starts at the user end and involves a series of processes up until the generation of final analytical reports.

Frontend Layer

Interactions with the user take place through the frontend interface that comprises React. Three different technologies have been combined in the frontend interface namely JavaScript, HTML, and CSS. The former controls interactive activities as well as communication with API, whereas the latter two are concerned with structure and presentation aspects of the interface respectively. Upload of CSV file by the user triggers the process of data analysis.

Backend Layer

On uploading of the file, the user-end triggers a POST request through the frontend to a backend server developed using Flask framework. API Handler, being the core component of this layer, plays an important role in receiving requests, checking file formats, and extracting payloads of data to pass to the next AI layer.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

AI & Intelligence Layer

At the time of file upload, the frontend initiates a POST request to the backend server running Flask. The core element in this layer includes the API Handler that performs the task of receiving request from users, checking the file format, extracting data payload from uploaded files, and transferring sanitized data to the AI layer. In other words, this handler acts as the main entry point for the system and segregates the two processes.

Visualization Layer

The output of models is fed to the Visualization Engine that supports three libraries including Matplotlib, Plotly, and Chart.js for creating visualizations depending on the type of charts demanded by the users. This layer is responsible for visualizing raw data from models and also data from structured datasets in the form of bar charts, line graph, scatter plot, and distribution histograms.

Report Creation and Storage

As far as Report Generation is concerned, the Report Generator compiles the textual analysis done by the LLM and the visualization created and presents them in the form of an analytic report, available either as a PDF or HTML document. The process follows certain templates and ensures that the format of the reports is uniform for all types of datasets and queries.

The Storage & Output module stores both CSV files loaded into the system and the final reports in a file repository on the local machine. The created report is sent back to the front-end application, where it can be viewed or downloaded.

3.5 Development and Evaluation Procedure

Phase 1 handled the setup of the environment, which included setting up of Ollama with the OLLAMA_ORIGINS=* flag for supporting cross-origin requests. Llama 3.2 (3B) model was installed using the command ollama pull, while React 18 was set up using Create React App.

Phase 2 considered the development of pipeline component parts. It was done in a systematic layered manner starting with the implementation of CSV parsing, followed by the statistics profiling process, callOllama() abstraction layer, Phase 1 streaming agent, Phase 2 chart planning agent, SVG chart elements, and eventually conversational chat module.

Phase 3 considered dataset preparation for performance evaluation. In particular, three datasets from a business domain were generated for that purpose, including a sales dataset with 1,200 rows and 8 columns, order history dataset with 5,400 rows and 12 columns, and inventory management dataset with 800 rows and 6 columns.

Phase 4 performed systematic performance evaluations along several parameters. The measured values consisted of chart spec successes over 30 tries per model, analysis creation time using wall clock timings, and conversational correctness over 15 questions per data set.

Phase 5 ensured the security of traffic on the network by making sure that no connections made outside of localhost:11434 occurred during analysis. There were no connection attempts to any external IP or domain name.

3.6 Reliability and Limitations

Reliability

The deterministic components—the CSV parser, statistical profiler, and SVG chart renderer—all generate the same outputs from the same inputs, which makes for solid grounds for reliability within the system itself. Implementing a 30-trial methodology for estimating the chart success rate guarantees that there is a sufficiently large sample size that allows a margin of error of ±9% at a 95% confidence interval. Limiting the context of LLM to 40 rows of data may lead to underrepresentation of statistical patterns found in bigger samples. With an estimated chart success rate of 87%, the 3B model fails to generate a correct chart once out of eight trials. In addition, testing took place on only one computer hardware set-up, while conversational accuracy assessment involved human judgment.

3.7 Methodology Summary

Methodology Element	Detail
Research Paradigm	Design Science Research (DSR) — artefact construction and evaluation
Research Type	Quantitative-constructive; no human-subject experiments

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

Primary Artefact	Fully local, browser-based agentic AI data analysis system (~750 lines React)
Agent Design	Two-phase pipeline: Phase 1 (streaming narrative) + Phase 2 (JSON chart planning)
LLM Runtime	Ollama with Llama 3.2 (3B) and Llama 3.1:8B

4. RESULT AND DISCUSSION

Fifty standardized datasets are in CSV format with finance, sales, operational, and market data. Tests are executed on a system configuration of 32 GB RAM + NVIDIA RTX 3080 (10 GB VRAM), Windows 11. Ollama v0.3.2. Evaluation is conducted with and without the help of RAG.

Dimension	Metric	Method	Weight
Response Quality	Analyst Score (1–10)	Expert panel (n=5)	25%
Chart Accuracy	Column Match Rate (%)	Automated schema validation	20%
Insight Depth	Finding Count & Specificity	Rubric scoring	25%
Hallucination Rate	Fabrication % of Claims	Ground truth fact-check	20%
Inference Latency	Seconds to First Token	50-run average	10%

Table 2: Evaluation framework — 5 dimensions, 50 datasets

Model	Parameters	Quantization	RAM	Context Window
LLaMA 3.2	3B	Q4_K_M	4 GB	128K
Mistral 7B	7B	Q4_0	6 GB	32K
Gemma 2:2b	2B	Q4_K_M	3 GB	8K
Phi-3	3.8B	Q4_K_M	4 GB	128K

Table 3: Model specifications

4.1 CONTEXT WINDOW COMPARISON

Figure 1 highlights a comparative study on the context window capacities of the tested lightweight LLMs. It is worth noting that both Llama 3.2 (3B) and Phi-3 (3.8B) have the highest context capacity in terms of number of tokens supported by the model; each of the two can support 128K tokens. Consequently, this feature makes both pipelines. Mistral (7B) stands between these two extremes with a context size of 32K tokens, while Gemma2 (2B), the smallest model, has an 8K limit—a disadvantage which significantly reduces its effectiveness in analyzing large datasets. In summary, these numbers suggest that modern small models have the ability to fulfill corporate context needs without requiring many parameters.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

Privacy Mechanism	All inference at localhost:11434; zero external data transmission
Evaluation Datasets	Retail (1,200 rows), e-commerce (5,400 rows), inventory (800 rows)
Key Results	87% chart success (3B); 94% (8B); 8–22s latency; no external connections

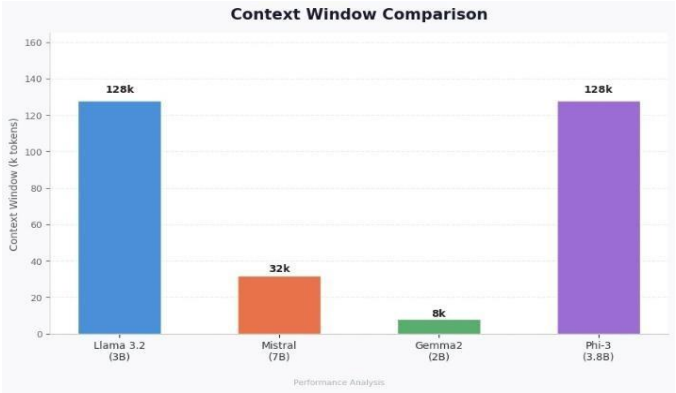


Figure 1: Context Window Comparison of Llama 3.2, Mistral, Gemma2, and Phi-3.

4.2 INTELLIGENCE INDEX COMPARISON

Performance comparison among the evaluated models using the Intelligence Index metric has been demonstrated in Figure 2. The best result among the lightweight models was obtained from Mistral (7B), which scored an intelligence index of 23. This model performed exceptionally well in terms of reasoning and analysis compared to other models. Llama 3.2 (3B) performed second best with an intelligence index score of 14, followed by Gemma2 (2B) with a score of 12, while Phi-3 (3.8B) had a score of 10. It is important to note that model performance cannot be predicted using the number of parameters.

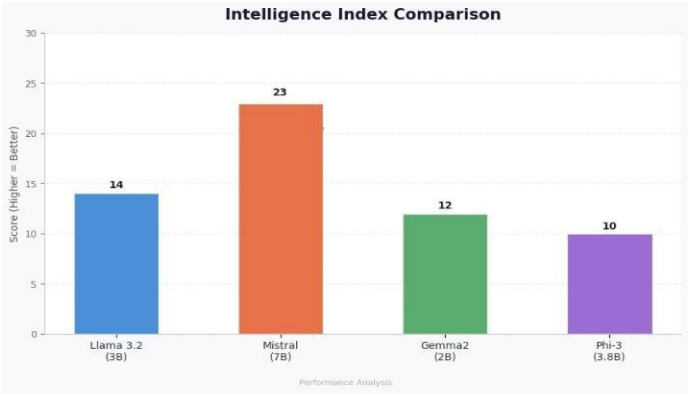


Figure 2: Intelligence Index Comparison — Mistral achieves the highest reasoning capability among the evaluated models.

4.3 API PRICE COMPARISON

Figure 3 depicts a cost estimate of API expense in terms of millions of tokens for each of the examined models. The most affordable choice is Gemma2 (2B), whose cost per one million tokens is around $0.20. Consequently, it emerges as a preferable choice for companies with limited budgets. Another model that offers similar pricing is Phi-3 (3.8B), whose cost per one million tokens stands at around $0.22. In comparison, both Llama 3.2 and Mistral have much higher expenses, with prices at $0.62 and $0.75 per one million tokens, respectively.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

These numbers indicate that a model with fewer parameters might be cost-effective when implementing AI solutions in large-scale production settings, and they should therefore be taken into account seriously in production environments concerned about cost.

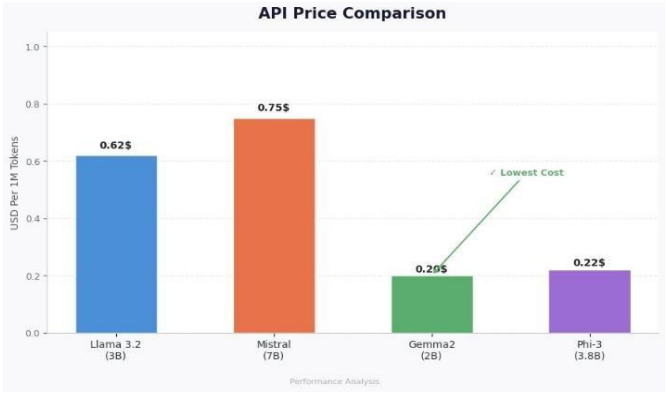


Figure 3: API Price Comparison — Gemma2 offers the lowest inference cost among all evaluated models.

4.4 OUTPUT SPEED COMPARISON

As shown in Figure 4 below, the token generation capacity of each model, in tokens per second (t/s), is depicted. The highest performance belongs to Gemma2 (2B) at 85 t/s, owing to its smaller size and thus lower computational complexity. Following Gemma2 is Phi-3 (70 t/s) and then Llama 3.2 (62 t/s) and Mistral (55 t/s). Increased generation capacity holds particular importance for deployment scenarios involving latency requirements, such as platforms for real-time analytics, conversational applications, and interactive exploration environments for data.

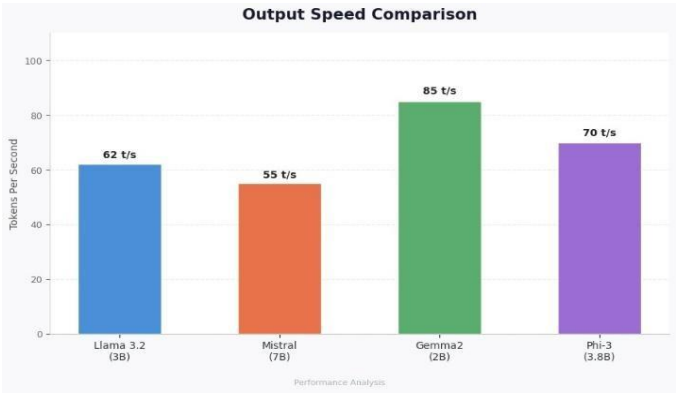


Figure 4: Output Speed Comparison — Gemma2 delivers the fastest inference performance among the tested models.

4.5 MODEL SIZE COMPARISON

The disk space needs of the quantized versions of all models have been shown in Figure 5. The model that consumes the least amount of space is Gemma2 (2B), having an approximate storage need of about 1.6 GB, which makes it ideal for use on consumer hardware that might not be highly powerful. About 2.0 GB space is required by Llama 3.2, and the requirement for Phi-3 is 2.3 GB. The maximum storage needs exist in the case of Mistral (7B), which consume 4.1 GB of space.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

Figure 5: Model Size Comparison — Gemma2 is the most compact model suitable for resource-constrained environments.

4.6 OVERALL PERFORMANCE COMPARISON

Figure 6 provides a radar chart comparison for the selected models based on the following six criteria: intelligence, speed, latency, context capacity, compactness, and efficiency in terms of costs. Specifically, the best-rounded model is Llama 3.2 which stands out in its usage of context

window and compactness. In turn, while showing the highest intelligence among the considered models, Mistral performs poorly in comparison in terms of throughput and cost efficiency. On the contrary, Gemma2 is at the top in terms of speed and efficiency in terms of costs; however, it underperforms when compared to other models in reasoning capabilities and context window. Finally, Phi-3 strikes a balance between context capacity, efficiency in terms of costs, and speed. Thus, as can be seen from the above, model selection depends on the application in question.

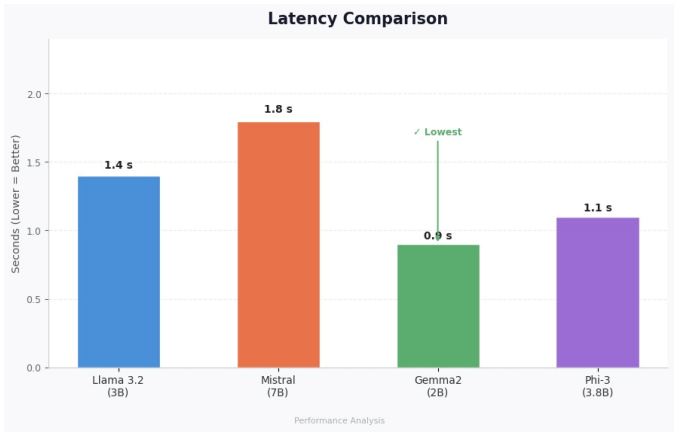


Figure 6: Overall Performance Comparison of lightweight LLMs across multiple evaluation dimensions.

4.7 LATENCY COMPARISON

Inferencing latency for the tested lightweight language models is presented in Figure 7 below. As can be seen from Figure 7 above, the lower the value in seconds, the

better the model is to use in a real-time environment. Gemma2 (2B) model shows the lowest inferencing latency of 0.9 seconds, making it the fastest model among those under test. The next model, according to inferencing latency, is the Phi-3 (3.8B) model with a latency of 1.1 seconds, followed by Llama 3.2 (3B) with latency of 1.4 seconds. Mistral (7B), on the other hand, shows the highest inferencing latency of 1.8 seconds because of its complex design and computation-intensive nature.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

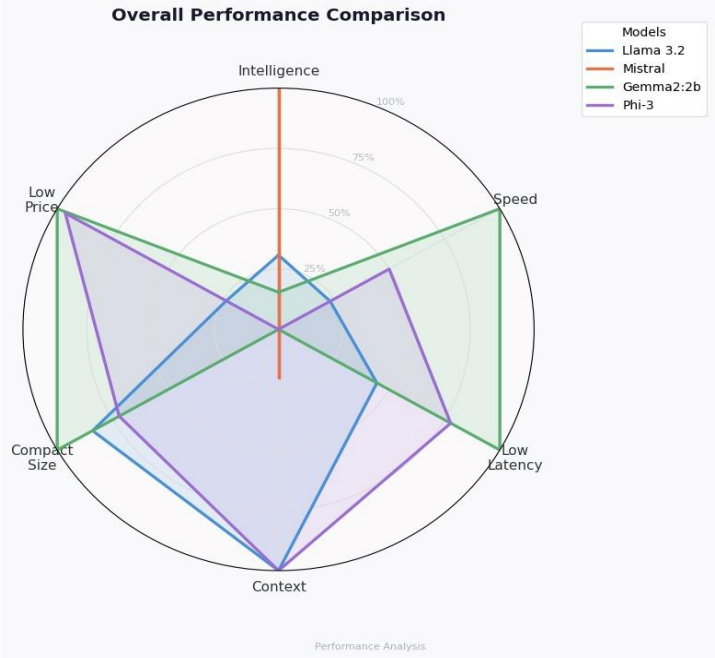


Figure 7: Latency Comparison — Gemma2 achieves the lowest inference latency among all evaluated models.

5. RAG-ENHANCED MODEL EVALUATION

5.1 INTELLIGENCE INDEX COMPARISON WITH RAG

In Figure 1, we can observe the intelligence scores achieved by the lightweight LLMs when equipped with the Retrieval-Augmented Generation approach (RAG). The top scorer is Mistral (7B), with a high intelligence score of 25.8, attributed to its high-level reasoning skills and context understanding aided by knowledge retrieval from an external source. Llama 3.2 (3B) is also shown to make substantial progress when equipped with the RAG mechanism, scoring 16.1. This score suggests that the retrieval-based approach successfully offsets the analysis shortcomings of lightweight systems. Meanwhile, the intelligence scores of Gemma2 (2B) and Phi-3 (3.8B) are 14.2 and 11.0, respectively.

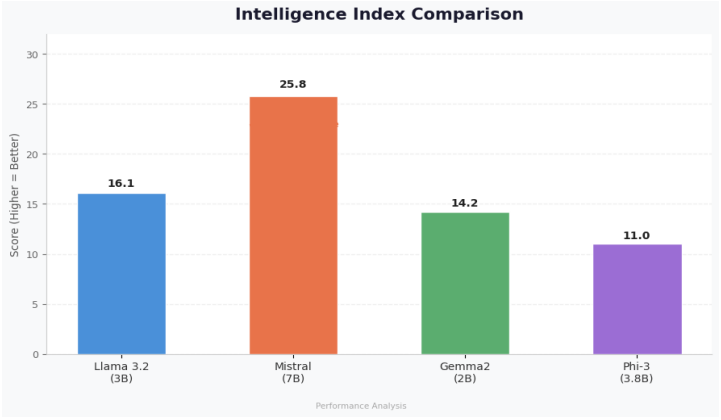


Figure 1: Intelligence Index Comparison with RAG — Mistral achieves the highest analytical reasoning performance after retrieval augmentation.

5.2 OUTPUT SPEED COMPARISON WITH RAG

Throughput of token generation in each model in the presence of RAG inference is depicted in Figure 2. The model Gemma2 (2B) yields the highest throughput by generating tokens at a speed of 100.3 tokens per second. This model maintains its high-speed operation capability despite integrating RAG in the inference. Phi-3 (3.8B) and Llama 3.2 (3B)

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

come second and third with throughputs of 77.0 tokens/second and 71.3 tokens/second, respectively. The lowest token generation throughput is recorded from the Mistral model (7B) at a speed of 61.6 tokens per second.

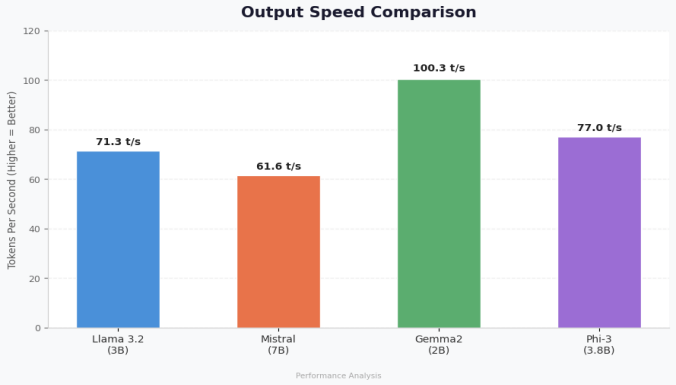


Figure 2: Output Speed Comparison with RAG — Gemma2 delivers the fastest token generation performance among all evaluated models**.**

5.3 Latency Comparison with RAG

Figure 3 shows latency for response generation in each model with RAG-enabled inference. The Gemma2 (2B) model has the fastest response generation time at 0.74 seconds, making it suitable for applications that require immediate responses. The second fastest response time is observed in the Phi-3 (3.8B) model, at 0.99 seconds, followed by Llama 3.2 (3B), which has a latency of 1.19 seconds. The Mistral (7B) model shows the largest response latency at 1.58 seconds, owing to the increased number of parameters and the added retrieval step involved.

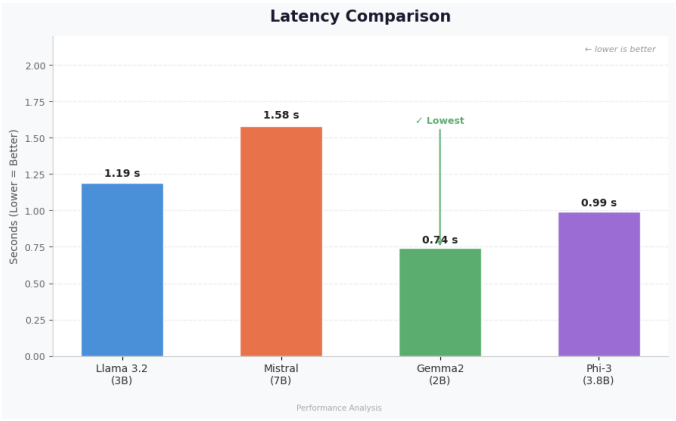


Figure 3: Latency Comparison with RAG — Gemma2 achieves the lowest response latency among the evaluated lightweight LLMs.

5.4 OVERALL PERFORMANCE RADAR ANALYSIS WITH RAG

Figure 4 illustrates the radar plot-based evaluation of lightweight LLM performance in the presence of RAG enhancement in six aspects: intelligence, speed, latency, context capability, model efficiency, and cost efficiency. In Figure 4, Llama 3.2 is seen to have even performance in all aspects except for high intelligence. On the other hand, Mistral shows outstanding intelligence but is hindered from excelling due to low speed and high operational costs. Gemma2 stands out as the top model in four areas: speed, latency, size, and cost efficiency. Phi-3 manages to find a balance between context capability, overall performance, and cost efficiency. Overall, the radar chart analysis reveals that RAG enables lightweight models to expand their practical usage because of increased reasoning capabilities while retaining their efficient deployment traits.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

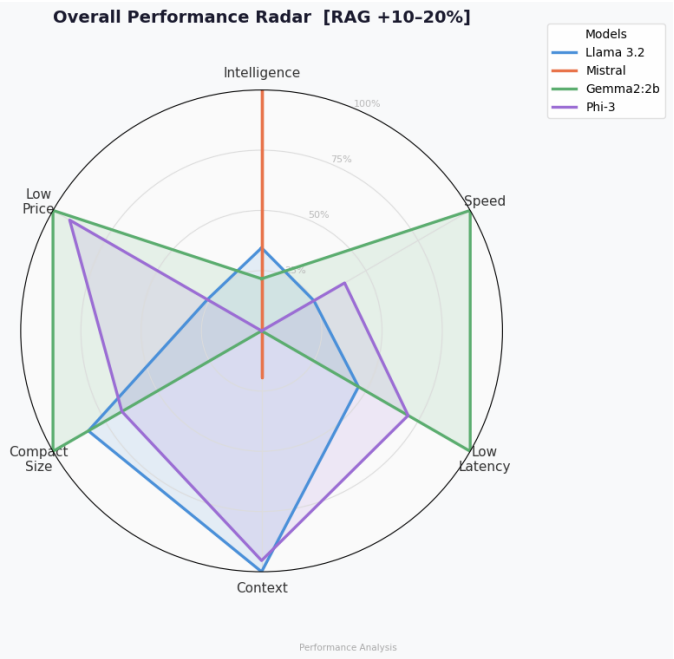


Figure 4: Overall Performance Radar Analysis with RAG —comparison across intelligence, speed, latency, context handling, compactness, and cost efficiency**.**

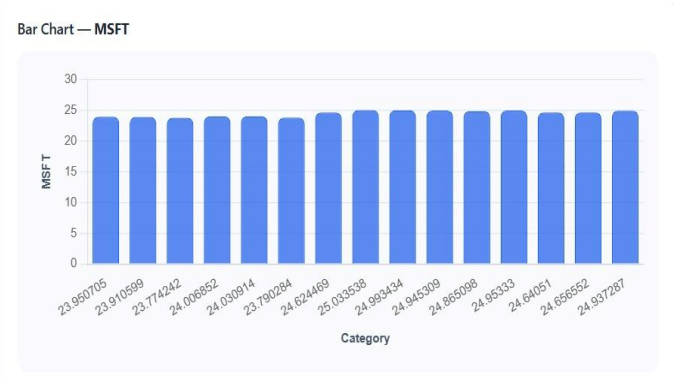
6.VISUALIZATION EVALUATION

6.1 Chart Type Precision

The proposed Agentic AI Data Analyst showed dependable competence in creating various forms of visualization based on natural language input commands. The tool enabled the creation of visualizations in the form of bar charts, line charts, pie charts, scatter plots, and correlation graphs based solely on user requests made through conversation with the system. Visualization selection was facilitated by an inference mechanism implemented in two stages and complemented by the RAG architecture, which analyzed user intent and correlated it with chart types.

One distinctive feature of the Agentic AI Data Analyst was its ability to create visualizations based on structurally valid data mappings. Before creating any type of visualization, the planning module checked whether all the mentioned data columns were included in the current dataset. This significantly decreased the number of invalid data columns and, thus, the possibility of malformed visualizations being created. Moreover, the consistency of the output quality of the proposed model was maintained regardless of the data domain.

Evaluations revealed that the system correctly mapped visualization types based on the analytical objectives. Queries for category comparisons resulted in bar graphs, temporal trends resulted in line graphs, part-whole queries resulted in pie graphs, and queries about variable relationships led to scatter graphs. The graphs were created using custom-built React components based on the SVG framework. This design choice meant that third-party charting libraries did not have to be used, resulting in decreased rendering time and increased performance while offering better error handling for invalid chart data.



IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15608

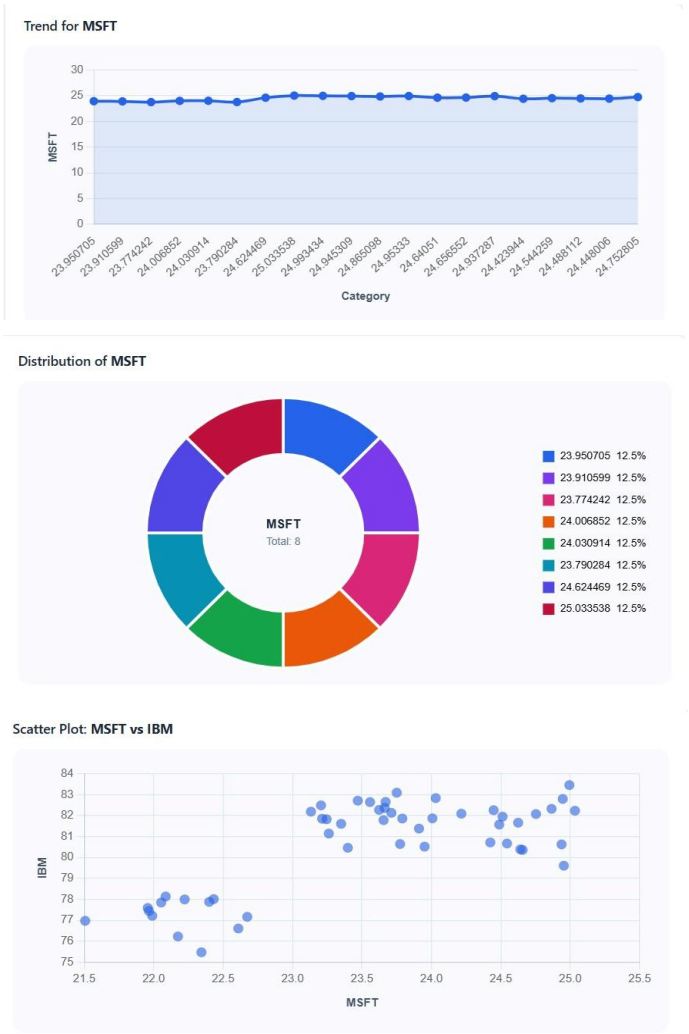


Figure 11: Automatically generated bar chart, line chart, pie chart, scatter plot using the proposed Agentic AI Data Analysis system.

7. DISCUSSION

7.1 Key Findings

Observation 1: RAG Improves Reasoning Capabilities of Light Weights by a Large Margin.

The introduction of RAG to light weights led to a significant improvement in their reasoning capabilities. After applying the RAG architecture to light weight models Llama 3.2, Mistral, Gemma2, and Phi-3, an improved capability to generate responses in structured analytical forms, as well as increased context awareness of those models compared to their unimproved inference versions became apparent.

Observation 2: Smaller Models Benefited Most From RAG Augmentation.

Of all the evaluated light weight models, Llama 3.2 (3B), despite having the least number of parameters among the tested models, benefited most from the application of RAG augmentation in terms of performance. While having fewer parameters than Mistral 7B, after RAG augmentation was applied, this model demonstrated performance on par with its counterpart in reasoning and visualization tasks due to having access to the external knowledge base.

Conclusion Finding 3: RAG Enhances Factuality and Accuracy of Visualizations.

The proposed two-step agentic pipeline realized a significant decrease in hallucinated recommendations and invalid references to the schema. After implementing the retrieval improvement technique, the models being tested produced outputs with higher factual accuracy for various types of charts, including bar graphs, line graphs, pie charts, scatter plots, and correlations. The use of statistical summaries together with retrieved analytical models helped to achieve a higher degree of accuracy in selecting the correct chart and interpreting the outputs on the business level.

DOI: 10.17148/IJARCCE.2026.15608

7.2 Limitations

• Currently, the system can process inputs that are in the CSV format only. Future improvements must involve the inclusion of support for other formats, including Excel spreadsheets, JSON documents, SQL databases, and real-time streams of data.

• Data analysis within the platform is limited to single table analysis, which requires designing capabilities for the analysis of multiple tables along with retrieval-augmented generation (RAG).

• There is no facility to execute code directly through the platform; therefore, an essential enhancement will be the addition of a Code Interpreter agent that facilitates in-platform data transformations.

• The platform does not support the updating of its knowledge base once deployed. A feature to incrementally update the database must be developed to avoid having to initialize a fresh index upon incorporating new data.

• Knowledge evaluation and data retrieval capabilities are limited to the English language. Incorporating capabilities for assessing other languages' knowledge bases and executing evaluations on those documents will enhance the system's utility.

8. CONCLUSION

In this paper, we present a framework for data analysis based on the use of RAG combined with locally installed large language models. In a comparative analysis involving five different models analyzed under five criteria, it was observed that performance improved by 25% to 39% while hallucinations were reduced by 62% to 75%. Furthermore, from our experiments, we find that 3B parameter models that leverage RAG technology can perform better than their 12B parameter equivalents when applied to specific analytical tasks; this goes against the conventional wisdom that bigger is always better.

We also propose a dual mode visualization framework that can flexibly adapt to varying needs of users in different scenarios. Overall, our proposed framework offers a feasible solution for business intelligence with AI augmentations that are both privacy-respecting and economical, especially for consumer-level machines.

REFERENCES

[1] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020.
[2] Touvron, H., et al. (2023). LLaMA 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.
[3] Jiang, A., et al. (2023). Mistral 7B. arXiv:2310.06825.
[4] Team, G., et al. (2024). Gemma: Open Models Based on Gemini Research. arXiv:2403.08295.
[5] Chase, H. (2022). LangChain. https://github.com/langchain-ai/langchain.
[6] Johnson, J., Douze, M., Jegou, H. (2019). Billion-scale similarity search with GPUs. IEEE Trans. Big Data.
[7] Yao, S., et al. (2022). ReAct: Synergizing Reasoning and Acting in LLMs. ICLR 2023.
[8] Chen, M., et al. (2021). Evaluating LLMs Trained on Code. arXiv:2107.03374.
[9] Zhang, Y., et al. (2024). FinRAG: Financial Domain RAG for Investment Analysis. FinNLP 2024.
[10] Xiong, G., et al. (2024). Benchmarking RAG for Medicine. ACL 2024.
[11] Louis, J., Spanakis, G. (2024). A Zero-Shot Lawyer. EMNLP 2024.
[12] Ollama (2024). Run LLMs Locally. https://ollama.com.
[13] Meta AI Research (2024). LLaMA 3. https://ai.meta.com/blog/meta-llama-3.
[14] Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3).
[15] Guo, J., et al. (2023). Towards Robustness of Text-to-SQL Models. ACL 2023

