UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Center For International Economics

Working Paper Series

Working Paper No. 2026-11

Time series forecasting in enterprises using an AI agent with times series MCP server

Li Chen

June 2026

CIE Center For International Economics

Time series forecasting in enterprises using an AI agent with times series MCP server

Li Chen*

Abstract

To address enterprise adoption challenges beyond model accuracy, including usability for non-expert users, trust and explainability, and cost-efficiency, this paper proposes a hybrid architecture in which a business AI agent acts as an orchestrator and a time series Model Context Protocol (MCP) server provides reusable forecasting capabilities along a seven-stage forecasting lifecycle. In the proposed architecture, the agent interprets the user request, reasons over the available context, invokes appropriate MCP tools, and translates structured tool outputs into business-oriented explanations. The implemented time series MCP server is evaluated on three real-world datasets under two experimental settings: a zero-shot forecasting setup, in which the agent compares the available forecasting tools, and a diagnostic-aware setup, in which the agent first analyzes data quality, seasonality, stationarity, structural breaks, and influencing factors before selecting a forecasting strategy.

The results show that forecasts produced through the MCP server can reach plausible quality levels across all datasets. The diagnostic-aware workflow improved forecast accuracy and explanation quality. The experiments further show that no single model family dominates across all settings: automatic ARIMA achieved the highest aggregate accuracy score but required the highest runtime, while Chronos-2 and Toto 2.0 provided competitive accuracy-runtime trade-offs. Exponential smoothing remained a fast and interpretable baseline. The findings suggest that an MCP-based architecture can make heterogeneous forecasting methods accessible, auditable, and cost-aware for enterprise AI agents, while still requiring clear task specification, governance, and human oversight.

Keywords : time series forecasting, Model Context Protocol, AI agents, AutoML, time series foundation models, enterprise analytics, explainable forecasting, design-based research.

*Corresponding author. E-mail: lichen@campus.uni-paderborn.de. University of Paderborn, Faculty of Business Administration and Economics

1 Introduction

Time series forecasting is one of the most widely used quantitative capabilities in enterprise data analytics. Sales forecasts guide production and inventory decisions. Financial forecasts support liquidity and profitability management and operational forecasts help infrastructure teams plan capacity. While the academic world focuses more on model fitting and forecast accuracy, time series forecasting in enterprise context involves an end-to-end lifecycle, in which business intent, data availability, domain constraints, model assumptions, uncertainty, interpretability, and operational governance interact (Rahman et al., 2025a; Zhao et al., 2025; Meisenbacher et al., 2022). The enterprise adoption challenge is therefore not only about improving model accuracy, but also making advanced forecasting methods usable, trustworthy, and economically deployable for recurring business decision processes, as pointed out by Goodwin et al. (2023) in his survey. This challenge is particularly visible in enterprises where domain experts understand the business question but do not necessarily possess statistical or programming expertise.

For many years, classical methods like ARIMA, exponential smoothing and their automated variants have been strong interpretable baselines. Many low-code business applications include this methods. Hyndman and Khandakar (2008) describe automatic algorithms for exponential smoothing and ARIMA as robust enough to support large-scale univariate forecasting, especially when organizations need forecasts for many product lines or business units. Chen and Feng (2025) introduce approaches to adopt advanced semi-parametric methods like $semi-ARMA$ and $deseats$ into enterprise use cases cost-efficiently and business user friendly. Nevertheless, the practical workflow still requires decisions about data aggregation, calendar regularization, missing values, outliers, seasonality, structural breaks, validation windows, and the communication of forecast uncertainty. These decisions are not automatically solved by a model-fitting function. In enterprise environments, they are often distributed across data engineers, analysts, business users, and IT operations (Crisan and Fiore-Gartland, 2021).

AutoML addresses part of this gap by automating feature engineering, model selection, and hyperparameter optimization (Hutter et al., 2019). In business analytics, Schmitt (2023) argues that AutoML can bridge the shortage of machine-learning experts by enabling fast prototyping and shortening development cycles, even if manually tuned models may still outperform AutoML in controlled cases. In the specific field of forecasting, AutoGluon-TimeSeries combines statistical, machine-learning, and deep-learning forecasters and produces probabilistic forecasts with a compact Python interface (Shchur et al., 2023). These systems are important because they reduce technical barriers and make model comparison more systematic. Yet the enterprise challenge remains broader than model selection: AutoML usually starts after the analytical task has already been translated into a machine-readable modeling problem, and it often ends before results are fully explained, governed, deployed, and monitored.

In recent years, a second paradigm of time series foundation models emerges. These models are

pretrained on large collections of time series and are designed to generalize to new forecasting tasks without task-specific training. Benchmarks such as GIFT-Eval emphasize the need for broad evaluation across domains, frequencies, horizons, and probabilistic outputs (Aksu et al., 2024b). Recent models such as Chronos-2 extend the foundation-model paradigm from univariate inference to universal forecasting with multivariate and covariate-aware in-context learning (Ansari et al., 2025), while Toto 2.0 argues that time series foundation models can benefit from scaling and sets strong results on several benchmarks (Khwaja et al., 2026). These developments are promising for enterprises because they can reduce the need to train a separate model for each use case. However, the practical value of LLM-inspired or foundation model approaches remains contested. (Tan et al., 2024) show that several LLM-based forecasting methods do not convincingly outperform simpler classic models and can incur however much higher computational cost. This implies that enterprises should treat foundation models as one option in a model portfolio, not as a universal replacement for interpretable baselines or AutoML.

With recent development of AI agents and its application to time series forecasting (Yao et al., 2023; Zhao et al., 2025; Schick et al., 2023), this paper proposes a hybrid approach. Instead of choosing between AutoML, classical statistics, and time series foundation models, it exposes them as tools of a time series MCP server that can be attached to enterprise AI agents. MCP is an open protocol for integrating large language model applications with external tools and data sources through standardized client-server interactions (Model Context Protocol, 2025; Luo et al., 2025). In the proposed architecture, the agent layer interprets user intent, asks clarifying questions, plans the forecasting workflow, and summarizes results. The MCP server layer includes specialized time series tools for data loading, preprocessing, diagnosing, training, forecasting and reporting. This separation keeps the reasoning and communication strengths of LLM agents while grounding existing algorithms in explicit tools which can be called by AI agents based on runtime reasoning.

This architectural argument is consistent with enterprise architecture principles by TOGAF (The Open Group, 2022). From this perspective, a forecasting MCP server can be treated as a reusable architecture building block rather than as a forecasting feature embedded separately in every application. A central server can be governed, monitored, extended, and connected to multiple agent clients such as SAP Joule, Microsoft Copilot, or other MCP-compatible clients. This modularity supports business-driven reuse, avoids duplicated implementation, and enables consistent controls for data access, model dependencies, cost limits, and auditability.

The research questions of this paper can be summarized as follows:

- **Research Question 1 (RQ1):** How can an enterprise-ready MCP server for time series forecasting be designed to address enterprise adoption challenges in forecasting, particularly usability for non-expert users, trust and explainability, and cost-efficient integration?

- **Research Question 2 (RQ2):** Do forecasting results produced through AI agents with such an MCP server provide better lifecycle coverage while maintaining good forecasting quality, transparency, and cost-efficiency?

The first question is addressed through architecture design, tool mapping, and lifecycle coverage. The second question is addressed through an experimental design and an application on public real-world datasets.

The paper is structured as follows. Section 2 reviews related work on AutoML, time series foundation models, LLM-based data science agents, MCP integration, and lifecycle models for data science tasks. Section 3 defines a seven-stage time series forecasting lifecycle and derives a detailed architecture proposal based on the identified architectural goals. Section 4 describes the implementation of the time series MCP server, including its tools for each lifecycle stage. It also illustrates an example agentic workflow using a system prompt, reusable skills, and MCP server tools. In Section 5, the MCP server is applied to three real-world datasets with different characteristics, and the results are analysed in relation to the main research questions. Section 6 concludes the paper.

2 Related work

2.1 AutoML and automated time series forecasting

Research on automating machines learning tasks including forecasting began in 1990s with focus on hyperparameter optimization and model selections. Later on in 2000s, the focus shift on building entire auto-ml pipelines and integration in enterprise (Hutter et al., 2019). Hyndman and Khandakar (2008) show that automatic forecasting is required in many business settings because organizations often need forecasts for thousands of product or operational series and cannot rely on a statistically trained expert for each series. Their forecast package automates model identification, parameter estimation, and prediction interval generation for exponential smoothing and ARIMA models. This work established a practical principle that remains relevant: automation is valuable when it is robust, scalable, and transparent enough to be used repeatedly without manual tuning for every time series. Modern AutoML generalizes this idea from model-specific automation to workflow-level automation. Schmitt (2023) frames AutoML as a response to the shortage of analytics expertise in business environments and emphasizes its value for fast prototyping and shorter development cycles. The paper also cautions that AutoML is not necessarily superior to expert manual tuning, which is important for enterprise adoption: AutoML should be understood as a productivity and accessibility mechanism rather than as a guarantee of optimal model performance.Crisan and Fiore-Gartland (2021) add a human-centered perspective by showing that enterprise users have different desired levels of automation depending on their expertise. Their interviews reveal a tension between speed and human oversight, and they highlight the role of visual analytics and human-in-the-loop design in making automation trustworthy. For forecasting

specifically, AutoGluon-TimeSeries demonstrates how AutoML can support probabilistic forecasting by combining statistical models, machine-learning approaches, and ensembles (Shchur et al., 2023) . Its design goals of ease of use, robustness, and short training time are aligned with enterprise needs, and its support for point and quantile forecasts is particularly relevant because business decisions often require uncertainty rather than single-number predictions. The limitation, however, is that AutoML still presupposes that the task is already expressed as a forecasting dataset with suitable targets, covariates, frequency, validation windows, and metrics. Business understanding, data-quality diagnosis, explanation, deployment, and monitoring remain partly outside the AutoML boundary. Therefore, AutoML is a necessary but insufficient component of an end-to-end enterprise forecasting architecture.

The present paper builds on this AutoML literature by treating AutoML as one tool family within a broader agentic workflow. The MCP server does not replace AutoML. It wraps AutoML together with diagnostics, quality handling, classical baselines, decomposition, and foundation models. This design reflects the human-in-the-loop insight of Crisan and Fiore-Gartland (2021). Business users should not be forced to choose between manual statistical work and opaque automation. Instead, the system should expose automation while preserving intermediate evidence, warnings, model comparisons, and opportunities for human review. This is also relevant to enterprise environments, where forecasts often enter planning cycles, financial reporting processes, or supply-chain decisions. In such contexts, unexplained automation may reduce manual work but increase organizational risk.

2.2 Time series foundation models and LLM-based forecasting

Time series foundation models extend the idea of pretraining to temporal data. Instead of fitting a model from scratch for each task, a single model is pretrained on large and heterogeneous time series corpora and applied to new tasks in a zero-shot or few-shot manner. Aksu et al. (2024b) argue that the field requires benchmarks that cover multiple domains, frequencies, prediction lengths, and probabilistic evaluation settings. GIFT-Eval addresses this gap by collecting 23 datasets across seven domains and 10 frequencies, with both univariate and multivariate settings. This benchmark orientation is important for enterprise research because enterprise time series are heterogeneous: monthly sales, weekly demand, daily inventory, hourly energy consumption, and minute-level telemetry cannot be evaluated through a single narrow dataset. Chronos-2 represents a recent step toward universal forecasting. Ansari et al. (2024, 2025) describe a pretrained model that handles univariate, multivariate, and covariate-informed tasks through in-context learning and group attention. This is significant because many enterprise use cases depend on related time series and exogenous drivers. Retail demand may depend on promotions and holidays; infrastructure capacity may depend on request volume and service configuration; energy consumption may depend on weather and tariffs. A foundation model that can use related series and known future covariates could therefore simplify deployment compared

with maintaining many task-specific models.

Toto 2.0 provides another recent example, focusing on the scaling behavior of time series foundation models. Khwaja et al. (2026) report a family of open-weight models ranging from small to large parameter counts and argue that forecast quality improves reliably with scale on benchmarks such as BOOM, GIFT-Eval, and TIME. For enterprises, the implication is nuanced. Scaling may improve accuracy, but larger models also raise inference cost, hardware requirements, latency, governance, and deployment complexity. Consequently, model scale should be treated as a design budget, not as an unquestioned objective.

However, the usefulness of LLMs for forecasting remains actively debated. Tan et al. (2024) show through ablation studies that several LLM-based forecasting methods can perform similarly or worse than simpler alternatives while requiring substantially more computation. Their conclusion does not rule out language models for time series reasoning, explanation, or multimodal tasks; rather, it challenges the assumption that pretrained language models directly transfer sequence-modeling advantages to numerical forecasting. This distinction is central to the present paper. The LLM is not proposed as the primary numerical forecaster. Instead, it is used as an orchestrator that interprets business intent, selects tools, and explains results, while numerical forecasting is delegated to specialized methods. Recent surveys on time series reasoning and agentic systems reinforce this separation. Chang et al. (2025) argue that time series reasoning should keep temporal evidence visible and aligned with intermediate conclusions, and they emphasize tool use, decomposition, verification, cost, and reproducibility as design dimensions. This supports an architecture in which forecasting workflows are explicit and inspectable. A foundation model can be useful when it improves probabilistic forecasts, but it should be invoked under controlled conditions and compared against baselines rather than hidden inside an opaque agent response.

2.3 LLM-Based data science Agents

LLM-based data science agents are increasingly studied as systems that can plan, reason, invoke tools, and coordinate multi-step workflows (Yao et al., 2023; Zhao et al., 2025; Schick et al., 2023). ReAct further demonstrates that interleaving reasoning traces and actions improves interpretability and task success because the model can update its plan based on observations from external environments (Yao et al., 2023). These ideas are directly relevant to enterprise forecasting: the agent should not merely produce a forecast from memory, but should call tools for loading data, checking quality, fitting models, evaluating metrics, and summarizing evidence.

Data science agent surveys provide a lifecycle view of this trend. LLM-Based time series agents are conceptualized as an end-to-end workflow, comprehensively formalized by architectures like TimeSeriesScientist, which deploys specialized sub-agents Curator, Planner, Forecaster, Reporter to minimize human intervention (Zhao et al., 2025) . Toolformer shows that language models can learn to call

external tools such as calculators, search systems, translation systems, and calendars, deciding when to call a tool and how to incorporate the result (Schick et al., 2023). The LAMBDA (Large Model Based Data Agent) framework establishes a code-free multi-agent system simulating human collaboration workflows, specifically "programmer" and "inspector" roles, to intuitively translate natural language instructions into executable data analysis pipelines, which positioned LLM-based data agents as a way to lower entry barriers for users without deep expertise in statistics, programming, or machine learning (Sun et al., 2025) . Chen et al. (2025) organize data science agents by roles, execution, knowledge integration, reflection, and loop-based workflows. Rahman et al. (2025b) map data science agents to six lifecycle stages: business understanding and data acquisition, exploratory analysis and visualization, feature engineering, model building and selection, interpretation and explanation, and deployment and monitoring. They also highlight a gap: many systems emphasize exploratory analysis and modeling while giving less attention to business understanding, deployment, monitoring, trust, safety, and governance. Evaluating the utility of these forecasts relies on frameworks like XForecast, which utilizes "simulatability" metrics to mathematically measure how effectively Natural Language Explanations (NLEs) generated by LLMs assist humans in forming mental models of the underlying forecasting mechanics (Aksu et al., 2024a) . This lifecycle gap is particularly problematic for enterprise forecasting. A technically accurate forecast may still fail if the business question is misunderstood, if the input data contains undocumented gaps, if outliers are silently removed, if the selected model cannot be explained, if uncertainty is not communicated, or if the model is not monitored after deployment. The proposed MCP architecture responds to this gap by making lifecycle stages explicit. Tools are not only provided for model building, but also for data loading, data quality, diagnostics, interpretation, and deployment-oriented metadata. The LLM agent can then translate the user's natural language request into a traceable workflow.

2.4 Model Context Protocol as integration architecture

MCP provides a standardized way for LLM applications to connect to external data sources and tools. The official specification describes MCP as an open protocol that enables integration between LLM applications and external tools and context providers (Model Context Protocol, 2025). The conceptual value is architectural separation: the host or client application manages user interaction and orchestration, while MCP servers expose domain-specific tools under structured contracts. Anthropic's introduction of MCP similarly emphasizes secure, two-way connections between AI-powered tools and data sources (Anthropic, 2024). For enterprises, this is attractive because specialized capabilities can be implemented once and reused across multiple agent interfaces. In the proposed forecasting architecture, MCP is not merely a technical transport layer. It is a governance boundary. The MCP server can enforce input schemas, dependency boundaries, file-size limits, model-context limits, structured error handling, and result metadata. The agent client can remain flexible and user-facing, but numerical operations are executed by tools that return auditable outputs. This is different from prompting an

LLM to write and run ad hoc code for every question. It also differs from embedding forecasting logic directly into a single business application. The MCP server functions as a reusable analytical capability layer. The literature therefore motivates three design requirements. First, forecasting automation must cover more than model selection; it must support the full lifecycle. Second, LLMs should be used where they are strongest: interaction, planning, explanation, and tool orchestration, not ungrounded numerical forecasting. Third, enterprise adoption requires modular and governed integration. The time series MCP server proposed in this paper is designed to meet these requirements by combining AutoML, foundation models, classical methods, diagnostics, and workflow metadata in one agent-accessible architecture.

2.5 Lifecycle of time series forecasting

Existing lifecycle models for data science provide a useful starting point, but they are not sufficiently specific for time series forecasting tasks in enterprise. For example, Rahman et al. (2025b) describe a general data science agent lifecycle with six stages: business understanding and data acquisition (S1), exploratory analysis and visualization (S2), feature engineering (S3), model building and selection (S4), interpretation and explanation (S5), and deployment and monitoring (S6). This lifecycle is appropriate for general data science tasks, but time series forecasting introduces additional requirements that are closely tied to the temporal structure of the data, the decision horizon, the forecast frequency, and the operational use of forecast results. Hyndman and Athanasopoulos (2018) argue that forecasting goals, available data, and suitable methods need to be determined before the actual forecasting steps are executed. They also emphasize the importance of time plots, seasonal plots, lag plots, autocorrelation analysis, decomposition, time series features, residual diagnostics, prediction intervals, point and distributional accuracy, and time-series cross-validation. This indicates that time series forecasting is not only a modeling task, but a structured analytical process that combines business framing, temporal data preparation, diagnostics, model selection, evaluation, and communication of uncertainty. Automated forecasting research provides a complementary, more technical view of the forecasting pipeline. Meisenbacher et al. (2022) summarize the typical automated time series forecasting design process into five pipeline sections: data preprocessing, feature engineering, hyperparameter optimization, forecasting method selection, and forecast ensembling. This perspective is valuable because it highlights the importance of systematic pipeline automation. However, it remains largely model- and pipeline-centric and therefore does not fully cover the business-facing and operational stages that are critical in enterprise environments. Recent agentic forecasting work extends this view further. Zhao et al. (2025) describe forecasting as a workflow of diagnostics-guided curation, planning, model fitting and validation, ensembling, and transparent reporting. They explicitly argue that the dominant cost of forecasting often lies not in model fitting itself, but in preprocessing, validation, ensembling, and reporting. This observation is particularly relevant for enterprise forecasting, where users often need trustworthy explanations, reusable workflows, and operational integration rather than isolated model

outputs.

3 Architecture goals and proposal

3.1 Lifecycle proposal for time series forecasting

Following these prior works, this paper conceptualizes enterprise time series forecasting as a seven-stage lifecycle. The proposed lifecycle is not intended to replace existing data science lifecycle models, but to adapt them to the specific characteristics of time series forecasting. It makes explicit those stages that are often hidden in generic data science workflows but are essential for reliable forecasting in enterprise contexts.

1. **Business understanding and forecast target setting.** The lifecycle starts with the clarification of the business decision that the forecast should support. This includes the definition of the target variable, forecast horizon, forecast frequency, aggregation level, relevant dimensions, business constraints, and acceptable level of uncertainty. In enterprise settings, this stage is essential because the same historical data can support different forecasting tasks depending on the planning horizon and decision context. For example, a monthly sales forecast for strategic planning requires a different design than a daily demand forecast for operational replenishment.

2. **Data integration and preprocessing.** In the second stage, relevant internal and external data sources are identified, accessed, integrated, and transformed into a consistent time series format. This includes timestamp parsing, frequency alignment, aggregation or disaggregation, duplicate handling, missing value treatment, outlier detection, and the integration of known future covariates such as holidays, prices, promotions, weather data, or planned business events. In contrast to general data preparation, forecasting data preparation must preserve the temporal ordering of observations and avoid information leakage from the future.

3. **Time series exploration and diagnostic profiling.** The third stage focuses on understanding the temporal structure of the data. Typical diagnostics include visual inspection, trend and seasonality analysis, autocorrelation analysis, stationarity checks, decomposition, intermittency detection, structural break detection, and residual or noise pattern analysis. This stage provides the empirical basis for later modeling decisions. For an agentic forecasting system, this stage can be interpreted as a perception and diagnostic layer: the agent should inspect the time series before selecting tools or models.

4. **Forecasting strategy and feature representation.** The fourth stage translates business requirements and diagnostic findings into a forecasting strategy. This includes decisions about univariate versus multivariate forecasting, local versus global models, direct versus recursive multi-step forecasting, deterministic versus probabilistic forecasting, and the use of statistical

models, machine learning models, AutoML, or time series foundation models. It also includes the construction of forecasting-specific features such as lags, rolling statistics, calendar features, event indicators, and covariate representations.

5. **Model training, selection, and forecasting.** The fifth stage contains model fitting and model comparison. Candidate methods are trained and evaluated through temporally valid validation procedures, such as rolling-origin evaluation or holdout windows that respect the chronological order of the data. Hyperparameter optimization and automated model selection can be used to improve model performance. Ensembling can further increase robustness by combining complementary model families. In enterprise forecasting, this stage should also include simple statistical baselines, because they provide an important benchmark for evaluating whether more complex or expensive methods are justified.

6. **Forecast evaluation and explanation.** The sixth stage evaluates whether the forecast is accurate, reliable, and useful for the business decision. This includes point forecast metrics, probabilistic forecast metrics, residual diagnostics, bias checks, prediction interval evaluation, and comparison against naive or seasonal naive baselines. In addition to technical evaluation, the results must be explained in a way that is understandable to business users. This includes explaining the selected method, detected data quality issues, uncertainty ranges, main drivers, limitations, and recommended interpretation of the forecast.

7. **Deployment and monitoring.** The final stage operationalizes the forecasting workflow. This includes making forecasts available through enterprise applications, dashboards, planning systems, or AI agents. It also includes monitoring data freshness, forecast accuracy over time, error patterns by horizon and segment, structural changes, drift, and retraining needs. In enterprise environments, deployment and monitoring are not merely technical afterthoughts. They are necessary to ensure that forecasting remains reliable under changing business conditions.

This lifecycle provides the conceptual foundation for the architecture developed in this paper. It shows that an enterprise forecasting system must support more than model execution. It must help users define the forecasting task, access and validate temporal data, perform diagnostics, select suitable methods, evaluate uncertainty, explain results, and monitor forecasts after deployment. These requirements motivate an architecture in which the LLM is not responsible for executing statistical computation directly. Instead, the agent orchestrates specialized forecasting tools exposed through a MCP server, while the MCP server provides structured, reusable, and auditable analytical capabilities across the forecasting lifecycle.

3.2 Architecture goals

Enterprise forecasting tasks usually involve multiple stakeholders and operational constraints. Business users need a conversational interface that understands their domain language and translates their questions into forecasting tasks. Data and analytics teams need reliable methods, reproducible outputs, and transparent quality checks. IT and enterprise architecture teams need reusable services that can be governed, monitored, and extended centrally. A single monolithic forecasting application would struggle to satisfy these requirements across multiple business domains. The proposed architecture therefore implements forecasting capabilities as MCP tools that can be consumed by different AI agents and enterprise applications. This design is consistent with enterprise architecture principles that emphasize reusable building blocks, interoperability, governance, and separation of concerns (The Open Group, 2022).

The first design goal is **lifecycle completeness and analytical quality**. As discussed in the previous subsection, time series forecasting requires a lifecycle that covers business framing, temporal data preparation, diagnostics, forecasting strategy design, model selection, evaluation, explanation, and monitoring. Most forecasting systems focus primarily on model fitting. In contrast, an enterprise-ready architecture must support the preparatory and post-modeling stages because many forecasting failures originate before model training or after forecast generation. Missing values, outliers, structural breaks, inappropriate aggregation, wrong forecast horizons, and weak monitoring can dominate the accuracy difference between algorithms. Therefore, the architecture should expose not only forecasting models, but also tools for data validation, quality analysis, diagnostics, decomposition, evaluation, and reporting.

The second design goal is **business user acceptance**. Business users should be able to access forecasting capabilities through natural language and familiar enterprise work environments rather than through specialized statistical software. However, natural language interaction alone is not sufficient. The agent must convert user intent into a structured forecasting workflow and make the reasoning behind tool choices transparent. For example, when a user asks for a forecast of regional demand, the agent should identify the relevant data source, load the data at the correct frequency, check missingness and outliers, test for seasonality, select suitable candidate models, compare holdout metrics, and explain uncertainty in business terms. The MCP tools provide the structured analytical operations, while the agent provides the dialogue, planning, and explanation layer.

The third design goal is **cost efficiency and reusibility**. A central MCP server can be implemented once and connected to multiple agents, teams, or applications. This supports reuse and reduces duplicated development effort. Cost-efficiency is achieved in three ways. First, lightweight statistical tools can be used when they are sufficient, avoiding unnecessary foundation-model inference. Second, expensive optional dependencies such as AutoML frameworks or large time series foundation models can be loaded only when invoked. Third, the same server can serve multiple clients, meaning that

governance, logging, security review, and method extension are centralized rather than repeated in each application. This is particularly important in enterprise settings, where forecasting capabilities are often needed across multiple departments, planning processes, and analytical applications.

The fourth design goal is **method pluralism and controlled extensibility**. The architecture should not assume that one forecasting paradigm is universally superior. Classical statistical methods, automated machine learning, and time series foundation models each have different strengths, limitations, cost profiles, and governance implications. Classical methods can provide strong and interpretable baselines. AutoML can automate model search and improve robustness across heterogeneous series. Foundation models can provide rapid zero-shot or few-shot forecasts, especially when training data is limited or fast prototyping is required. The architecture should therefore make these methods available as comparable tools under a shared data and output contract. New methods should be added as additional MCP tools without requiring a redesign of the user interface or orchestration layer.

The fifth design goal is **auditability and trust**. Since forecasting results are often used for further planning, budgeting, inventory, staffing, and operational decisions, the architecture must enable auditability and trust. Each tool call should return structured metadata, including input assumptions, data-quality warnings, model configuration, evaluation metrics, uncertainty information, and possible limitations. This allows the agent to generate transparent explanations and enables audit of workflow. Trust is therefore not created only by model accuracy, but by a combination of reproducible computation, explicit diagnostics, comparable metrics, and understandable reporting.

3.3 Architecture proposal

The proposed architecture consists of four layers, as shown in Figure 1. The first layer is the **user interface layer**. It contains enterprise-facing agent entry points such as SAP Joule, Microsoft Copilot, or custom agent applications. This layer is responsible for interaction with business users and allows forecasting capabilities to be embedded into familiar work environments.

The second layer is the **orchestration layer**. It contains the LLM-based agent orchestration mechanism, which can be enhanced with system prompts, skills, workflow rules, and domain-specific instructions. This layer interprets the user request, clarifies missing information, plans the forecasting workflow, invokes suitable MCP tools, and synthesizes the final answer. It is also responsible for translating technical outputs into business-readable explanations.

The third layer is the **MCP server layer**. This layer contains registered time series tools with structured input and output schemas. Tool discovery and communication between the orchestration layer and the MCP server follow the standard MCP definition (Model Context Protocol, 2025). The tools can cover multiple stages of the proposed forecasting lifecycle, including data preparation, data

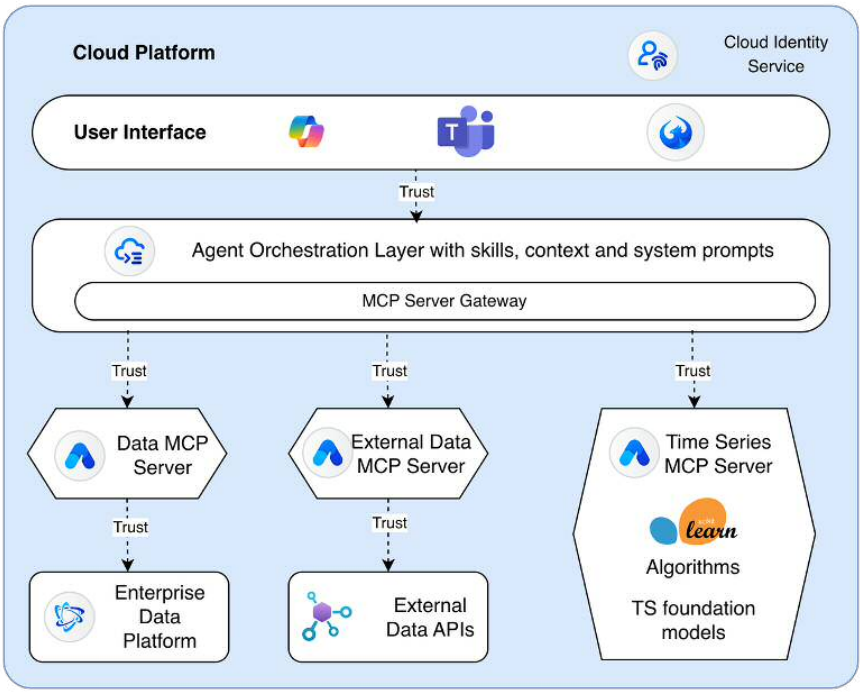

Figure 1: Architecture

quality checks, diagnostics, decomposition, model execution, forecast evaluation, and reporting. The MCP server therefore acts as a reusable analytical capability layer between the agent and the underlying forecasting methods.

The fourth layer is the **data layer**. It connects enterprise-internal data sources, business semantic models, and external data sources. In an enterprise context, this layer is particularly important because forecasting usually depends not only on historical target values, but also on master data, hierarchies, planning dimensions, calendars, prices, promotions, and other business context. The architecture therefore separates data access and semantic integration from model execution, allowing organizations to govern data access independently from forecasting tool logic.

This layered architecture provides several benefits. First, the MCP server approach supports **plug-and-play integration** into existing business agents. Adding the time series MCP server to an existing business agent makes forecasting capabilities available without rebuilding the agent from scratch. Removing or replacing the server does not require a complete redesign of the user interface or orchestration layer. This supports phased adoption: an organization may start with diagnostics and classical baselines, then later enable AutoML or foundation-model tools after governance and cost controls are established.

Second, the architecture supports **shared infrastructure, operation, and governance**. Instead of implementing forecasting logic separately in each application, the MCP server centralizes reusable forecasting tools. This enables consistent logging, versioning, access control, quality rules, and method

extension. From an enterprise architecture perspective, the MCP server can be interpreted as a reusable architecture building block. Its business value is democratized access to forecasting; its application value is reusable integration with agents; its data value is a common time series contract and quality metadata; and its technology value is a governed tool layer with clear provider boundaries. This positions the artifact not as a one-off prototype, but as an enterprise capability that can be maintained, monitored, and extended over time (The Open Group, 2022).

Third, the architecture enables **forecasting-specific system prompts and skills**. Experienced business users or analytics experts can encode recurring forecasting patterns in natural language. A forecasting-specific system prompt can instruct the agent to clarify the business horizon, aggregation level, target metric, dimensions, and acceptable uncertainty before model selection. It can also instruct the agent to inspect missing values, outliers, stationarity, seasonality, and structural breaks before interpreting model results. Skills can encode reusable workflow patterns, such as demand-planning workflows, financial forecasting workflows, or operational capacity forecasting workflows. This reduces blind hyperparameter search because the agent can use business context and diagnostics to narrow the set of relevant methods before invoking expensive tools.

Fourth, the architecture provides **stronger lifecycle coverage than a pure AutoML architecture**. AutoML can automate parts of model selection and hyperparameter optimization, but it typically does not fully address business-question clarification, enterprise data access, diagnostics explanation, governance, and post-deployment monitoring. The MCP-based approach allows AutoML to be used as one tool family within a broader forecasting lifecycle rather than as the complete architecture.

Fifth, the architecture provides **more control than a pure foundation-model architecture**. Time series foundation models can be valuable for rapid zero-shot probabilistic forecasting, but they should be compared with interpretable baselines and AutoML methods under the same data contract. The MCP server makes this comparison systematic by returning metrics, warnings, quantiles, model metadata, and diagnostic information in compatible formats. This enables the agent to choose methods based on task requirements, data characteristics, cost constraints, and governance rules rather than relying on a single model family.

Overall, the architecture operationalizes the proposed time series forecasting lifecycle by assigning different responsibilities to different layers. The user interface layer supports accessibility, the orchestration layer supports planning and explanation, the MCP server layer supports reusable analytical execution, and the data layer supports enterprise integration. This separation of concerns is central to the proposed design: the LLM agent should not replace statistical forecasting methods, AutoML, or foundation models. Instead, it should orchestrate them in a transparent, auditable, and business-oriented forecasting workflow.

4 Implementation of the time series MCP server

This section describes the implementation of the proposed time series MCP server and relates its current tool catalogue to the seven-stage forecasting lifecycle introduced in Section 3. The implementation exposes reusable time series capabilities as MCP tools that can be discovered and invoked by an AI agent. The current implementation registers 14 tools across six functional groups: server discovery, data preparation, data quality, diagnostics, decomposition, and forecasting. The server therefore provides an analytical capability layer that enables the agent to construct an auditable forecasting workflow from an initial business question to a structured model output. An abstract of the README.md is attached in Appendix A.

4.1 MCP tools for each stage

Table 1 summarizes the mapping between the proposed time series forecasting lifecycle and the implemented MCP tools. The mapping distinguishes between stages that are directly supported by executable MCP tools and stages that are only partially supported because they require business context, enterprise system integration, or human approval.

4.1.1 Tools for S1: Business understanding and forecast target setting

The first lifecycle stage is primarily a preparatory step for the business agent. At this stage, the agent derives the user's forecasting intention by combining information from the current user prompt, available business context, historical agent memory, the system prompt, domain knowledge, and the currently exposed MCP tools. The MCP server does not independently infer the business objective. Instead, it provides the technical interface through which a clarified business objective can be transformed into an executable forecasting specification.

The MCP server supports this stage indirectly through tool discovery and structured parameterization. The *list_tools* tool allows the AI agent to inspect the currently available analytical capabilities before planning a workflow. Forecasting tools expose parameters such as *forecast_steps*, *target_metrics*, interval levels, quantile levels, model identifiers, context length, and holdout settings. The loader exposes parameters such as *frequency*, *aggregation*, *value_columns*, and *dimension_columns*. These arguments materialize a business decision as an executable forecasting task.

4.1.2 Tools for S2: Data integration and preprocessing

The second lifecycle stage is directly supported by three implemented tools: *time_series_loader*, *time_series_missing_data_handler*, and *time_series_outlier_handler*. These tools transform raw tabular and multi-dimensional data into a regular and quality-annotated time series representation that can

Lifecycle stage	**Implemented MCP tools**
S1: Business understanding and forecast target setting	• *list_tools*; • tool arguments such as *value_columns*, *dimension_columns*, *frequency*, *aggregation*, *target_metrics*, *forecast_steps*, interval levels, and quantile levels;
S2: Data integration and preprocessing	• *time_series_loader*; • *time_series_missing_data_handler*; • *time_series_outlier_handler*;
S3: Time series exploration and diagnostic profiling	• *time_series_unit_root_test*; • *time_series_seasonality_test*; • *time_series_structural_break_test*; • *time_series_deseats_decompose*;
S4: Forecasting strategy and feature representation	• Forecasting strategy and metadata are not implemented as separate MCP tools; instead, the agent reasons iteratively over the outputs of S3 and S5 tools and the available business context;
S5: Model training, selection, hyperparameter optimization, and ensembling	• *time_series_arima*; • *time_series_exponential_smoothing*; • *time_series_deseats_forecast*; • *time_series_chronos2_forecast*; • *time_series_toto2_forecast*; • *time_series_automl_forecast*;
S6: Forecast evaluation and explanation	• metrics returned by *time_series_arima*, *time_series_exponential_smoothing*, *time_series_chronos2_forecast*, and *time_series_toto2_forecast*; • leaderboard from *time_series_automl_forecast*;
S7: Deployment and monitoring	• Deployment and monitoring are not implemented as direct MCP tools; however, structured MCP outputs can be used by downstream applications for deployment and subsequent monitoring;

Table 1: MCP tools for each lifecycle stage

be consumed by later lifecycle stages.

The *time_series_loader* reads data from supported structured files or connected data sources. It parses a time column, applies an optional date range, buckets observations into a regular frequency, aggregates duplicate observations within the same time bucket, and constructs one or more series from metric and dimension combinations. Supported normalized frequencies include daily, monthly, quarterly, and yearly frequencies. With this design, business time series stored in multi-dimensional analytical models, such as sales data organized by product, country, dealer, and date, can be converted into a common multi-series *SeriesCollection*.

The *time_series_missing_data_handler* identifies incomplete observations and applies a selected or automatic treatment strategy. Implemented strategies include no intervention, dropping incomplete series, forward or backward filling, interpolation, seasonal median replacement, and rolling median replacement. The purpose of this tool is not only to repair incomplete data, but also to preserve information about the intervention so that later diagnostic and forecasting results can be interpreted in light of the underlying data quality.

The *time_series_outlier_handler* detects anomalous observations and records their locations, timestamps, and possible consecutive-break evidence. The implemented detection methods include median absolute deviation, interquartile-range detection, and rolling Hampel-style detection. The tool can flag outliers, interpolate them, or replace them with seasonal medians. In enterprise sales data, this distinction is important because apparent outliers may correspond to actual business events, such as promotions, product launches, supply recovery effects, or market shocks. Consequently, the tool enables the agent to retain outlier evidence instead of automatically removing potentially meaningful business signals.

4.1.3 Tools for S3: Time series exploration and diagnostic profiling

The third lifecycle stage is supported by three diagnostic tools and one decomposition tool. These tools provide evidence about the temporal structure of the data before a forecasting strategy and concrete forecasting methods are selected.

The *time_series_unit_root_test* provides stationarity diagnostics. It supports ADF- and KPSS-based checks and returns recommendations such as stationary, difference once, possible trend or regime change, or inconclusive. The tool is designed to guide the agent in deciding whether differencing, trend modelling, or a regime-change-aware strategy should be considered in later stages.

The *time_series_seasonality_test* evaluates seasonal evidence. It can infer standard seasonal periods from the data frequency, such as 12 for monthly data and 4 for quarterly data, or accept an explicit seasonal period. The tool combines several evidence types, including seasonal-lag autocorrelation, a seasonal-strength statistic, and an ANOVA-style statistic by default. The result is relevant both for

model selection and for the choice of seasonal parameters.

The *time_series_structural_break_test* detects possible regime changes. It supports known-break testing, unknown-break search, multi-break fallback detection, and variance-shift search. The tool maps candidate break positions back to timestamps, which allows an agent or analyst to relate statistical evidence to business events such as price changes, promotions, supply disruptions, or market shocks such as the COVID-19 pandemic.

The *time_series_deseats_decompose* tool provides decomposition using the DeSeaTS algorithm (Schulz, 2026). It returns trend, seasonal component, fitted signal, residuals, deseasonalized values, detrended values, selected bandwidth, and method options. This tool is particularly useful when the agent needs to explain individual components of a time series, for example the development of trend or seasonality in regional car sales.

4.1.4 Tools for S4: Forecasting strategy and feature representation

The current implementation supports forecasting strategy selection through the combination of diagnostics and multiple forecasting tools. The agent can use outputs from S2 and S3 to decide between transparent statistical baselines, decomposition-based forecasting, zero-shot time series foundation models, and AutoML. For example, strong seasonality and stable historical patterns may motivate Holt-Winters exponential smoothing or DeSeaTS. Short histories or rapid prototyping requirements may motivate Chronos-2. Heterogeneous product-region series may motivate AutoGluon as an AutoML-based model-selection boundary. Toto 2.0 currently provides the most explicit support for multivariate representation through *group_by_dimensions*, which allows the adapter to place aligned target series into a joint inference group.

At the same time, forecasting strategy and feature representation are not implemented as a separate MCP tool. The server does not yet provide dedicated functionality for covariates such as event indicators, holiday effects, promotion variables, prices, weather information, or planned business actions. This remains an important extension direction because many enterprise forecasting tasks require known-future or exogenous drivers in addition to historical target values.

4.1.5 Tools for S5: Model training, selection and forecasting

The current MCP server provides six forecasting tools: *time_series_arima*, *time_series_exponential_smoothing*, *time_series_deseats_forecast*, *time_series_chronos2_forecast*, *time_series_toto2_forecast*, and *time_series_automl_forecast*.

The *time_series_arima* tool uses *statsmodels* to fit ARIMA and SARIMA specifications. It supports explicitly provided orders as well as a bounded information-criterion search. The tool can return fitted

parameters, information criteria, prediction intervals, warnings, and optional holdout metrics. When holdout evaluation is requested, the selected specification is evaluated on the reserved tail of the series and is then refitted on the complete observed history for the final future forecast. This makes ARIMA/SARIMA a well-interpretable baseline for both non-seasonal and seasonal enterprise time series. This tool supports forecasting considering influencing factors as covariates.

The *time_series_exponential_smoothing* tool uses optimized Holt-Winters exponential smoothing through *statsmodels*. It supports level, trend, seasonality, damping, additive and multiplicative components, and optional holdout evaluation. Forecast intervals are estimated through simulation from the fitted model. This tool is a suitable baseline when level, trend, and seasonal components are central to the forecasting task.

The *time_series_deseats_forecast* tool invokes the DeSeaTS algorithm and returns Semi-ARMA forecasts, confidence intervals, selected bandwidth, residual model order, decomposition context, and warnings. Forecasting with DeSeaTS should mainly be used for short horizons because trend and seasonal components are extrapolated. This explicit methodological indication helps the agent select the method appropriately and avoid overstating long-horizon forecast reliability.

The *time_series_chronos2_forecast* tool calls the Chronos-2 *predict_df* interface for zero-shot forecasting. It returns point forecasts, requested quantiles, model identifier, device, context length, forecast horizon, zero-shot metadata, warnings, and optional holdout metrics. The implementation enforces context-length and prediction-length limits, including the published maximum context length of 8192 and maximum prediction length of 1024. The current adapter exposes Chronos-2's covariate-informed capabilities (Ansari et al., 2025).

The *time_series_toto2_forecast* tool integrates Datadog Toto 2.0 through its tensor-oriented interface. It can forecast selected series independently or group aligned series into a multivariate inference request. The adapter constructs target tensors and boolean target masks, allowing missing observations in the model context. It returns fixed quantile levels, a median point forecast, model identifier, device, context length, group identifier, number of variates, decoding settings, and warnings. The default model identifier is *Datadog/Toto-2.0-22m* (Khwaja et al., 2026). The current tool exposes zero-shot time series forecasting capability.

The *time_series_automl_forecast* tool integrates AutoGluon TimeSeries. It maps complete selected series into an AutoGluon *TimeSeriesDataFrame*, initializes a *TimeSeriesPredictor*, and supports the configuration of forecast horizon, quantile levels, evaluation metric, presets, candidate model names, validation-window count, ensembling behavior, and training time limit. The result contains the selected model, leaderboard, configuration, point forecasts, and quantile forecasts (Shchur et al., 2023). This tool is enhanced to support forecasting with covariates.

4.1.6 Tools for S6: Forecast evaluation and explanation

The current implementation does not yet include a unified cross-model backtesting or comparison tool. Therefore, evaluation is only partially supported through the structured outputs of the forecasting tools in S5. These outputs provide the evidence base for evaluation and explanation in structured formats such as JSON, but they do not automatically generate a complete evaluation report.

The reasoning process that compares candidate models, evaluates forecasting quality, iterates over model choices and parameters, explains uncertainty, and translates technical outputs into business language is performed by the agent. This separation is consistent with the overall architecture: the MCP server provides auditable computational evidence, while the agent performs orchestration and explanation.

4.1.7 Tools for S7: Deployment and monitoring

The current MCP server provides only limited support for deployment and monitoring. It can provide structured outputs for downstream deployment upon request, but dedicated tools for model deployment and monitoring depend strongly on the hosting infrastructure and the internal governance concept of the enterprise. In the proposed architecture, deployment and monitoring should be handled by the enterprise host environment or by surrounding workflow orchestration. These aspects are therefore outside the main focus of this paper.

4.1.8 Tool discovery and grounding

The current MCP server provides the *list_tools* capability to help business agents discover available capabilities before planning a workflow. The server also exposes guidance resources, including a tool guide and data contract documentation. These resources help the agent understand available tools and expected input-output conventions. This improves robustness because the agent does not rely only on pretrained knowledge or vague prompt instructions, but can ground its planning in structured tool documentation.

4.2 An example agent workflow with the MCP server

With the MCP server connected, a business AI agent can invoke the exposed time series tools based on contextual reasoning. This context typically includes the system prompt, the user prompt, reusable skills, MCP tool definitions, and server-provided resources. Examples of a user prompt, a system prompt, and a forecasting-specific skill are provided in Appendix B. Based on this setup, the resulting agentic workflow can be described as follows.

First, the agent loads the system prompt, interprets the user prompt, and identifies the relevant

skill, namely the *Car Sales Forecasting Skill*. It then invokes *list_tools* to discover the available tools and derives initial parameters such as frequency, aggregation, target metric, and regional dimension from the business context.

Second, the agent invokes *time_series_loader* to load the data and identify the time column, quantity metric, monthly frequency, sum aggregation, and region dimension. The output is a *SeriesCollection* containing one sales series per region. This creates the regular monthly structure required by the subsequent quality, diagnostic, and forecasting tools.

Third, the agent invokes *time_series_missing_data_handler*. If short gaps exist, interpolation or seasonal median replacement may be appropriate depending on the history length and seasonal period. The resulting imputation flags are preserved. The agent then invokes *time_series_outlier_handler*. For sales data, it may be preferable to flag outliers rather than automatically replace them, because sales spikes may correspond to real campaigns, product launches, or supply catch-up effects.

Fourth, the agent invokes the diagnostic tools. The seasonality test checks whether monthly seasonal behavior is present. The structural-break test checks whether regime changes are present, for example after a product launch or market disruption. The unit-root test provides stationarity evidence and helps determine whether differencing or trend-sensitive models may be appropriate. If seasonality and changing seasonal structure are relevant, the agent may additionally invoke *time_series_deseats_decompose*.

Fifth, the agent selects forecasting tools. For transparent baselines, it can invoke *time_series_arima* and *time_series_exponential_smoothing*. If the deployment supports AutoGluon and the user accepts the runtime cost, *time_series_automl_forecast* can provide a model-selection benchmark and leaderboard. If rapid zero-shot probabilistic forecasting is required, the agent can invoke *time_series_chronos2_forecast*. Toto 2.0 is less central for this sales-by-region example unless multiple aligned metrics or grouped multivariate targets are available.

Sixth, the agent compares the results. It should consider holdout metrics where available, interval or quantile width, warnings, model metadata, and data-quality flags. The agent should avoid claiming that a model is best if the results were not evaluated under the same validation design. It should also explain when advanced methods do not provide clear added value over transparent baselines.

Finally, the agent generates a forecast explanation. The explanation should include the forecast horizon, aggregation level, data-quality findings, diagnostic evidence, selected methods, evaluation evidence, uncertainty interpretation, and limitations. If the forecast is intended for operational use, the agent should recommend that the enterprise host persist the forecasts.

4.3 Other implementation notes

4.3.1 Canonical data contract

The core data abstraction is a regular multi-series *SeriesCollection*. It contains dataset metadata, a normalized time index, metric columns, dimension columns, aggregation rules, individual series identities, quality indicators, and warnings. This abstraction is necessary because enterprise time series are often stored across multiple dimensions such as product, customer segment, country, and date. A common collection contract allows all later tools to operate on the same explicit set of series.

Furthermore, the contract allows quality provenance to travel through the workflow. Flags for missing data and outliers are preserved so that the final forecast explanation can distinguish between original observations and repaired values. This improves transparency and trust in an enterprise setting. The current contract does not yet represent covariates, hierarchy definitions, or multiple calendar systems, which remain important extensions for future work.

4.3.2 Structured error handling

The server provides a structured error contract to improve agent reliability. In an LLM-agent workflow, validation errors, data-quality errors, and estimation failures can often be corrected by the agent or user through adjusted methods, parameters, or prompts. Structured errors allow the agent to distinguish between these cases and respond with an appropriate recovery strategy rather than hallucinating a successful result.

The current server also includes configurable resource limits for file size, row count, number of series, number of timestamps, model context length, prediction length, subprocess timeouts, and AutoML runtime. The agent should communicate when a request is constrained because of these limits.

4.3.3 Security and governance

The MCP server should be deployed with an appropriate authentication and authorization concept. Authentication ensures that the server can only be accessed by authenticated users or approved agent applications. Authorization ensures that each user can access only those tools, data sources, and output locations for which they have permission.

5 Application on real-world data

This section applies the implemented time series MCP server to three real-world datasets. The application is aligned with the two research questions introduced in Section 1. The experiments evaluate the proposed architecture in realistic forecasting situations with different data frequencies, business

Dataset	Description	Reason for selection
Rossmann Store Daily Sales	Daily sales data from 2013-01-01 to 2015-07-31 for 1115 stores. The dataset contains high-volume sales data with influencing variables such as open indicator, promotion indicator, state holiday, and school holiday.	Represents a retail sales forecasting problem with daily frequency, weekly patterns, promotion effects, holidays and store closures observations.
Monthly passenger car registrations by country	Monthly passenger car registration numbers from 1994-01 to 2024-12 for 5 different European countries Belgium, France, Germany, Greece, and the United Kingdom.	Represents an aggregated automotive demand forecasting problem with monthly frequency, seasonality, outliers, and structural breaks with long histories
Yearly global electric car stock	Yearly forecasting of electric car stock from 2005 to 2025 by region, including China, Europe, the United States, Rest of the World, and total stock.	Represents a long-term automotive transformation indicator with short annual series, strong trend growth, limited observations, and high strategic relevance.

Table 2: Selected datasets for the real-world application

domains, data-quality characteristics, and modelling constraints. The main focus is therefore not only forecast accuracy, but also lifecycle coverage, tool orchestration, diagnostic reasoning, explainability, operational effort, and method selection under enterprise constraints.

5.1 Agent client selection

The Codex extension of Visual Studio Code is used as the agent client. The time series MCP server is configured as an external MCP server that can be discovered and invoked by the agent. In this experimental setup, the agent does not implement forecasting algorithms directly. Instead, it interprets the user prompt, plans the workflow, selects and invokes MCP tools, compares structured tool outputs, and generates a forecast explanation. This setup reflects the central architectural assumption of this paper. The LLM-based agent acts as an orchestrator and communication layer, while the MCP server provides reusable and auditable time series capabilities.

5.2 Data selection

For the application, 3 datasets covering retail store sales, automotive market demand, and global electric vehicle stock development are selected as shown in Table 2. The selected datasets are intended to reflect enterprise relevant diversity across domains, frequencies, granularities, seasonal patterns, structural breaks and data quality issues.

The dataset for daily Rossmann store sales is a public Kaggle competition dataset (Kaggle, 2015).

The yearly global electric car stock dataset is based on a public Kaggle dataset on global electric vehicles (Kaggle, 2025). The monthly passenger car registration dataset for selected European countries is published by the OECD Data Explorer under the indicator "First registrations of brand new vehicles" (OECD, 2026).

5.3 Experiment design and metrics

Two experiments are designed to evaluate forecasting capability and quality of the MCP server. Experiment 1 investigates forecasting capability and quality of all forecasting tools under a concise prompt, which might be given by business users with limited knowledge about algorithms. Experiment 2 examines if method selection, explanation quality, and forecast accuracy can be improved by enriched prompts with explicit diagnostic steps.

5.3.1 Experiment 1: Zero-shot forecasting

This experiment investigates the forecasting capability of all available forecasting tools using a simple prompt, which might be given by business users with limited knowledge about algorithms. The same prompt structure is used for each dataset so that comparisons can be made among model families. The candidate tools include automatic ARIMA, exponential smoothing, DeSeaTS, AutoML, Chronos-2, and Toto 2.0 where applicable. Note that the term *zero-shot* in this experiment, means that the agent is not given a manually specified modelling strategy. Instead, it is asked to use the available MCP tools directly and infer appropriate parameters.

> **Experiment prompt 1:**
> "Load dataset for **rossman store daily sales considering inflencing factors** from folder /docs. Use the time series MCP server tools to **forecast last 4 weeks with known influencing factor values for each store** and compare automatic ARIMA, exponential smoothing, DeSeaTS, AutoML, Chronos-2, and Toto 2.0 where applicable. Report MAPE, MASE working time, warnings, and model metadata."

> **Experiment prompt 2:**
> "Load dataset for **monthly passenger car registrations by country** from folder docs. Use the time series MCP server tools to **forecast next 12 periods** and compare automatic ARIMA, exponential smoothing, DeSeaTS, AutoML, Chronos-2, and Toto 2.0 where applicable. Report MAPE, MASE working time, warnings, and model metadata."

> **Experiment prompt 3:**
> "Load dataset for **yearly global electric car stock** from folder docs. Use the time series MCP server tools to **forecast next 3 years** and compare automatic ARIMA, exponential

smoothing, DeSeaTS, AutoML, Chronos-2, and Toto 2.0 where applicable. Report MAPE, MASE working time, warnings, and model metadata."

5.3.2 Experiment 2: Diagnostic-aware forecasting

In this experiment, the agent is explicitly instructed to diagnose each dataset before selecting a forecasting strategy. The agent is asked to check missing data, outliers, structural breaks, seasonality, and possible influencing factors. Based on this diagnostic evidence, the agent selects an appropriate forecasting strategy using the available MCP tools.

The comparison between Experiment 1 and Experiment 2 evaluates whether deeper diagnostic reasoning improves forecasting quality and explanation quality. It also tests whether prompt design influences the agent's forecasting capability and quality.

Experiment prompt 4:
"Load dataset for **daily Rossmann store sales** from folder docs. Diagnostic the loaded dataset including missing data, outliers, structural break, seasonality and possibly influencing factors and select a best forecasting strategy based on available forecasting tools to **forecast last 4 weeks with known influencing factor values for each store**. Report selected model, parameters, explaination for selection, forecasting results and MAPE, MASE, working time, warnings."

Experiment prompt 5:
"Load dataset for **monthly passenger car registrations by country** from folder docs. Diagnostic the loaded dataset including missing data, outliers, structural break, seasonality and possibly influencing factors and select a best forecasting strategy based on available forecasting tools to **forecast next 12 periods**. Report selected model, parameters, explaination for selection, forecasting results and MAPE, MASE, working time, warnings."

Experiment prompt 6:
"Load dataset for **yearly global electric car stock** from folder docs. Diagnostic the loaded dataset including missing data, outliers, structural break, seasonality and possibly influencing factors and select a best forecasting strategy based on available forecasting tools to forecast next 3 years. Report selected model, parameters, explaination for selection, forecasting results and MAPE, MASE, working time, warnings."

5.3.3 Accuracy and performance metrics

Point forecast accuracy is measured using MAPE and MASE. MASE is particularly useful for comparing across series because it scales forecast errors relative to a naive benchmark. MAPE is included

Dataset	Forecasting tool	MAPE %	MASE
Daily Rossmann store sales	Chronos-2	10,55	0,3121
Daily Rossmann store sales	Toto 2.0	11,90	0,3807
Daily Rossmann store sales	AutoML	12,15	0,3368
Daily Rossmann store sales	automatic ARIMA	13,28	0,3649
Daily Rossmann store sales	exponential smoothing	17,31	0,4792
Daily Rossmann store sales	DeSeaTS	119,70	3,6603
Monthly Passenger Car Registration	exponential smoothing	7,50	0,5750
Monthly Passenger Car Registration	AutoML	8,02	0,5870
Monthly Passenger Car Registration	Chronos-2	8,23	0,6280
Monthly Passenger Car Registration	automatic ARIMA	8,67	0,6520
Monthly Passenger Car Registration	Toto 2.0	9,66	0,7200
Monthly Passenger Car Registration	DeSeaTS	21,79	1,5900
Yearly global electric car stocks	automatic ARIMA	2,39	0,6500
Yearly global electric car stocks	exponential smoothing	4,33	1,1800
Yearly global electric car stocks	DeSeaTS	4,35	1,1900
Yearly global electric car stocks	Toto 2.0	4,57	1,2000
Yearly global electric car stocks	Chronos-2	15,07	3,6900
Yearly global electric car stocks	AutoML	25,80	5,9400

Table 3: Results of Experiment 1: Zero-Shot Forecasting

because it is widely understood in business contexts, especially in SAP enterprise context.

5.4 Experiment results and analysis

This section reports and analyses the results of the experiments following the research questions of this paper.

5.4.1 Results of Experiment 1: Zero-shot forecasting

Table 3 presents the results of Experiment 1 for all combinations of datasets and forecasting tools. The evaluation reports MAPE and MASE as point forecast accuracy metrics. The best-performing method for each dataset is marked in green. Rows marked in yellow indicate methods with practically useful forecasting quality for the respective enterprise use case, especially where MASE is clearly below 1 and MAPE remains within a reasonable range.

For the Rossmann dataset, the agent loaded the 1115 store-level daily sales series for the period from 2015-01-01 to 2015-07-03 for training . The final four weeks from 2015-07-04 to 2015-07-31, were used as the holdout period. This resulted in 1115 store-level time series and 31220 forecast points. Known future influencing factors, including store-open status, promotion status, school holidays, state holidays, and weekday indicators, were used as covariates for those methods that supported them, namely Chronos-2, AutoML, and automatic ARIMA. As shown in Table 3, Chronos-2 achieved the best MASE on the Rossmann dataset, followed by AutoML and automatic ARIMA. Toto 2.0 also produced practically useful results, although it did not use exogenous factors. DeSeaTS performed

Dataset	Forecasting tool	MAPE %	MASE
Daily Rossmann store sales	global method auto ARIMA	10,41	0,2930
Monthly Passenger Car Registration	country-specific method	6,328	0,4850
Yearly global electric car stocks	global method auto ARIMA	2,39	0,6470

Table 4: Results of Experiment 2: Diagnostic-aware Forecasting

extremely poorly on this daily panel. A plausible explanation is that the method is not well matched to the long-period forecasting problem and does not incorporate influencing factors in the current tool interface.

For the monthly passenger car registration dataset, the agent loaded 372 monthly observations from 1994-01 to 2024-12 for five complete country series. A 12-month holdout was used by training through 2023-12 and forecasting 2024-01 to 2024-12. Table 3 shows that exponential smoothing achieved the best average result, followed closely by AutoML, Chronos-2, automatic ARIMA, and Toto 2.0. This result is notable because the best method is a transparent statistical baseline rather than a more complex foundation model or AutoML pipeline. DeSeaTS produced weaker holdout performance and returned warnings that Semi-ARMA forecasts are mainly intended for short horizons.

For the yearly global electric car stock dataset, the agent initially treated the aggregated column *Total (millions)* as the forecasting target. The evaluation used 2005–2022 as the training period and 2023-2025 as the holdout period. Table 3 shows that automatic ARIMA achieved the best holdout result, with a MASE clearly below 1. All other methods reported MASE values above 1 and therefore did not outperform the annual naive benchmark. From an enterprise forecasting perspective, these results would not be sufficient for operational use without further analysis.

In addition, the intended forecast target was electric car stock per region, rather than the aggregated total. The agent initially selected the aggregated column and therefore interpreted the target incorrectly. After revising the prompt with a more explicit instruction, such as **"forecast next 3 years"**, the forecast target was correctly identified. This observation highlights the importance of precise prompting when AI agents are used for analytical workflows.

5.4.2 Results of Experiment 2: Diagnostic-aware forecasting

Table 4 shows result of Experiment 2, where the agent was explicitly instructed to diagnose the dataset, select a forecasting strategy and then execute the forecasting task. In contrast to Experiment 1, the agent did not run all forecasting tools. Instead, it selected a subset of tools and model configurations based on diagnostic evidence. All selected strategies returned MASE values below 1 and reasonable MAPE values.

For the Rossmann dataset, the diagnostic-aware workflow identified weekly seasonality, store-level heterogeneity, the deterministic role of the *Open* variable, and the importance of known future op-

erational drivers. The agent selected a per-store ARIMAX strategy with order $(1, 0, 1)$ and known future covariates. The covariates included *Open*, *Promo*, *SchoolHoliday*, a state-holiday indicator, and weekday indicators. The diagnostic-aware Rossmann workflow achieved a MAPE of 10.41% and a MASE of 0.293 across 31,220 point forecasts. This improved over the best MASE from Experiment 1. The agent also reported important warnings: 403 stores had convergence warnings, 91 stores had non-stationary starting AR parameter warnings, and negative forecasts were clipped to zero. Known closed store-days were forced to zero. These details are relevant for trust and explainability for business users in an enterprise context.

For the passenger car registration dataset, 84 outliers was found, including COVID-related shocks in March and April 2020 and United Kingdom registration spikes in March and September. The agent also identified significant structural breaks for Belgium and Greece, strong monthly seasonality across all series. Instead of selecting one global method for all countries, the agent selected country-specific winners based on the 2024 holdout. The mean selected holdout MAPE was 6.328%, and the mean selected MASE was 0.485. This improved over the best single model in Experiment 1, where exponential smoothing achieved average MAPE of 7.50% and MASE of 0.575. The result supports the value of diagnostic-aware and series-specific method selection.

For the yearly global electric car stock dataset, the recent rapid-growth years from 2020 to 2025 were flagged as outliers, but the agent correctly interpreted them as growth-regime signals rather than bad records. A significant structural break was detected at 2021-12-31. Seasonality was not applicable for annual data. Stationarity diagnostics indicated non-stationarity and recommended differencing once. The agent selected direct nonseasonal ARIMA$(1, 2, 2)$ based on a 3-year holdout. The selected ARIMA$(1, 2, 2)$ model achieved holdout MAPE of 2.39% and MASE of 0.6470, which matches the best result from Experiment 1 and confirms that, for this dataset, diagnostic-aware reasoning did not change the winning model but improved the explanation. The agent reported limitations about narrow forecast intervals rapid growth periods in the dataset. This provides crucial hint about uncertainty for business users.

5.4.3 Overall forecasting capability, quality and agent behavior

Across both experiments, the AI agent successfully invoked the configured MCP server and executed the requested forecasting workflows. In most cases, the agent correctly inferred the forecasting target, time frequency, evaluation period, and applicable tools. The main exception occurred in the yearly global electric car stock dataset, where the agent initially selected the aggregated total column instead of the intended regional target. This issue was resolved through a corrected follow-up prompt with a more explicit target instruction. The observation is important for the overall argument of the paper: natural language interaction improves usability, but business-critical analytical tasks still require prompt clarity, validation, and human review.

The results demonstrate that the proposed MCP server can produce forecasts of plausible quality across different real-world datasets. In Experiment 1, the best-performing tools produced acceptable MASE values below 1 for all three datasets. In Experiment 2, diagnostic-aware forecasting slightly improved the results for the daily Rossmann store sales dataset and the monthly passenger car registration dataset. For the yearly global electric car stock dataset, the quantitative performance was preserved while the diagnostic explanation improved substantially.

This confirms the technical forecasting capability of the proposed MCP server architecture. More importantly, it shows that the value of the architecture is not limited to raw model performance. The agent was able to incorporate diagnostics, data-quality information, warnings, model metadata, and business constraints into the forecasting workflow. This is directly related to the enterprise adoption challenges addressed in this paper: usability, trust, explainability, and cost-aware integration.

5.4.4 Comparison between Experiment 1 and Experiment 2

Experiment 1 and Experiment 2 differ primarily in the role assigned to the agent. In Experiment 1, the agent was explicitly asked to call and compare all forecasting tools. As a result, it reported results for all available forecasting methods, including methods that were not well suited to some datasets. In Experiment 2, the agent was asked to diagnose the dataset and select a suitable strategy. Consequently, it invoked fewer tools and produced a more focused workflow.

The comparison indicates that the agent follows the intent encoded in the prompt. When the prompt asks for exhaustive model comparison, the agent behaves like a benchmarking assistant. When the prompt asks for diagnostic-aware strategy selection, the agent behaves more like an analytical forecaster. This implies that the same MCP server can support both exploratory benchmarking and efficient decision-oriented forecasting, depending on the system prompt, user prompt, and skill design.

Diagnostic-aware forecasting improved the reported accuracy for the daily Rossmann store sales dataset and the monthly passenger car registration dataset. For Rossmann, MASE improved from 0.3121 to 0.2930. For monthly passenger car registrations, MASE improved from 0.5750 to 0.4850. For the yearly electric car stock dataset, MASE remained nearly unchanged, moving from 0.6500 to 0.6470. Although the numerical improvement for the third dataset is small, the diagnostic-aware workflow added important interpretation about non-stationarity, structural break behaviour, and uncertainty limitations. Therefore, Experiment 2 improves not only accuracy in some cases, but also the transparency and relevance of the forecast explanation.

5.4.5 Comparison among forecasting models

In an enterprise context, both forecasting quality and cost-efficiency matter when selecting forecasting methods. To compare forecasting tools across datasets, a simple accuracy score is calculated from

Forecasting tool	Accuracy score	Average runtime (sec)
Chronos-2	2,5	30,58
Toto 2.0	2	3,98
AutoML	2	66,24
automatic ARIMA	3,5	304,80
exponential smoothing	1,5	13,94
DeSeaTS	0	43,35

Table 5: Accuracy score and runtime by forecasting tool

Experiment 1. The best-performing method for each dataset, marked in green in Table 3, receives 1.5 points. Methods marked in yellow receive 1 point because they provide practically useful forecast quality. Methods with poor forecast quality receive 0 points. The average runtime across the three datasets is used as a proxy for cost-efficiency. The cost of AI token consumption is not included because it was not technically possible to collect token consumption information for each MCP tool call.

Based on Experiment 1, the resulting score and average runtime for each forecasting tool are shown in Table 5. Automatic ARIMA achieved the highest accuracy score of 3.5. However, its average runtime of about five minutes was substantially higher than the runtimes of the foundation model tools Chronos-2 and Toto 2.0. Chronos-2 and Toto 2.0 also achieved good overall accuracy scores and were computationally more attractive. Exponential smoothing was very efficient and performed best on the monthly passenger car registration dataset, but its overall score was lower because it was less competitive on the Rossmann and electric car stock datasets. DeSeaTS did not receive an accuracy score in these experiments because it did not provide competitive results on the selected holdouts.

The model comparison provides three main findings. First, no single forecasting tool dominates across all datasets when both accuracy and runtime are considered. Second, classical statistical methods remain highly relevant in enterprise forecasting because they are interpretable and can achieve strong accuracy, especially for long monthly or annual series. Third, foundation models can provide a favourable trade-off between accuracy and runtime in some contexts. Chronos-2 was the best method for the Rossmann dataset and achieved a strong overall score with moderate runtime, while Toto 2.0 was operationally very fast and produced useful results in several cases. This supports the architectural decision to expose multiple forecasting methods through a common MCP server rather than embedding one fixed forecasting model into the agent.

6 Concluding Remarks

To further improve the usability, transparency, and cost-efficiency of time series forecasting in enterprise contexts, this paper proposed a hybrid architecture in which an LLM-based business agent acts as an orchestrator and a time series MCP server provides reusable forecasting-related capabilities. More specifically, the agent interprets the user request, performs context-aware reasoning based on system

prompts and skills, invokes MCP tools, and translates structured tool outputs into business-oriented explanations. The MCP server encapsulates concrete forecasting capabilities along the forecasting lifecycle, ranging from data loading, data diagnostics, and preprocessing to model selection, forecasting, and result reporting.

With respect to RQ1, this paper proposed an architecture concept that aligns with established enterprise architecture goals derived from TOGAF, provides broad coverage of the time series forecasting lifecycle, and integrates forecasting tools based on classical models, foundation models, and AutoML. Based on this concept, a time series MCP server was implemented. The technical functionality of the implemented tools was confirmed through the application scenarios and experiments presented in this paper.

Regarding RQ2, the experiments on three real-world datasets show that forecasts produced through the MCP server can reach plausible quality levels while improving lifecycle coverage and explanation. In the zero-shot forecasting experiment, the best-performing methods achieved MASE values below 1 across all three datasets. These results show that no single model family dominates across all forecasting settings. Instead, model suitability depends on frequency, data volume, seasonality, covariate availability, structural breaks, and forecast horizon. The diagnostic-aware experiment further demonstrated the value of lifecycle-oriented agent orchestration. By explicitly prompting the agent to diagnose missing data, outliers, seasonality, stationarity, structural breaks, and influencing factors before selecting a forecasting strategy, forecast quality improved for the Rossmann store sales dataset and the monthly passenger car registration dataset. For the yearly global electric car stock dataset, the quantitative result was preserved, while the explanation improved. These findings suggest that the added value of the MCP-based approach lies not only in accuracy, but also in the ability to improve other stages of forecasting lifecycle related to diagnostic, report to business users and auditability. The comparison among forecasting tools also highlights the importance of cost-aware method selection. Automatic ARIMA achieved the highest aggregate accuracy score in the experiments, but it also required the highest average runtime. Chronos-2 and Toto 2.0 provided competitive accuracy-runtime trade-offs, while exponential smoothing remained a fast and interpretable baseline.

This paper has several limitations. First, only the ARIMA, Chronos-2, and AutoML forecasting tools currently support covariate-aware forecasting. This is particularly relevant for enterprise datasets such as retail sales, where known future external factors can strongly influence forecast quality. Second, the current implementation does not yet include a unified model comparison or backtesting tool that evaluates all forecasting methods under identical rolling-origin settings. Third, deployment and monitoring are not implemented as MCP tools and must be handled by the enterprise host environment. Fourth, the experiments show that AI-agent workflows remain sensitive to prompt formulation. In the global electric car stock experiment, the agent initially selected the aggregated total column instead of the intended regional target. This illustrates that natural language interaction improves usability, but

does not remove the need for validation, clear task specification, and human oversight in enterprise decision processes. Future work could therefore extend the MCP server to address these limitations.

Acknowledgments

This work was supervised by Professor Dr. Yuanhua Feng at University Paderborn. I am very grateful for his valuable guidance and support throughout this research. Furthermore, I thank my family and friends in Paderborn for their mental and physical support during my research. During the preparation of this work, the author used ChatGPT and Gemini to check for grammatical errors and refine some texts and code. During implementation of the proposed MCP Server, Codex was used to support coding. The author reviewed and edited the texts and code thereafter and takes full responsibility for the content.

References

Taha Aksu, Chenghao Liu, Amrita Saha, Sarah Tan, Caiming Xiong, and Doyen Sahoo. Xforecast: Evaluating natural language explanations for time series forecasting, 2024a. URL https://arxiv.org/abs/2410.14180.

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024b. URL https://arxiv.org/abs/2410.10393.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024. URL https://arxiv.org/abs/2403.07815.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025. URL https://arxiv.org/abs/2510.15821.

Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, November 2024. Accessed: 2026-05-25.

Ching Chang, Yidan Shi, Defu Cao, Wei Yang, Jeehyun Hwang, Haixin Wang, Jiacheng Pang, Wei Wang, Yan Liu, Wen-Chih Peng, and Tien-Fu Chen. A survey of reasoning and agentic systems in time series with large language models, 2025. URL https://arxiv.org/abs/2509.11575.

Ke Chen, Peiran Wang, Yaoning Yu, Xianyang Zhan, and Haohan Wang. Large language model-based data science agent: A survey. *arXiv preprint arXiv:2508.02744*, 2025. URL https://arxiv.org/abs/2508.02744.

Li Chen and Yuanhua Feng. Forecasting of trend stationary time series in sap using a data-driven semiparametric arma model. Working paper, Paderborn University, 2025.

Anamaria Crisan and Brittany Fiore-Gartland. Fits and starts: Enterprise use of automl and the role of humans in the loop. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery. doi: 10.1145/3411764.3445775. URL https://doi.org/10.1145/3411764.3445775.

P. Goodwin, J. Hoover, S. Makridakis, F. Petropoulos, and L. Tashman. Business forecasting methods: Impressive advances, lagging implementation. *PLoS One*, 18:e0295693, 12 2023.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer Cham, 1 edition, 2019. ISBN 978-3-030-05318-5. doi: 10.1007/978-3-030-05318-5.

R.J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts.com, Melbourne, Australia, 2nd edition, 2018.

Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 27(3):1–22, 2008. doi: 10.18637/jss.v027.i03.

Kaggle. Rossmann store sales. https://www.kaggle.com/c/rossmann-store-sales, 2015. Public Kaggle competition dataset. Accessed: 2026-06-05.

Kaggle. Global electric vehicle dataset 2025. https://www.kaggle.com/datasets/padmapiyush/global-electric-vehicle-dataset-2023, 2025. Public Kaggle dataset used for yearly global electric car stock. Accessed: 2026-06-05.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukova, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era. Technical report, Datadog AI Research, May 2026. URL https://github.com/DataDog/toto.

Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. Mcp-universe: Benchmarking large language models with real-world model context protocol servers, 2025. URL https://arxiv.org/abs/2508.14704.

Stefan Meisenbacher, Marian Turowski, Kaleb Phipps, Martin Rätz, Dirk Müller, Veit Hagenmeyer, and Ralf Mikut. Review of automated time series forecasting pipelines. *CoRR*, abs/2202.01712, 2022. URL https://arxiv.org/abs/2202.01712.

Model Context Protocol. Model context protocol specification, version 2025-03-26. https://modelcontextprotocol.io/specification/2025-03-26, March 2025. Accessed: 2026-05-25.

OECD. First registrations of brand new vehicles. https://data-explorer.oecd.org/vis?df%5Bag%5D=OECD.ITF&df%5Bds%5D=dsDisseminateFinalDMZ&df%5Bid%5D=DSD_ST%40DF_STREG&lc=en, 2026. OECD Data Explorer dataset. Accessed: 2026-06-05.

Mizanur Rahman, Amran Bhuiyan, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Ridwan Mahbub, Ahmed Masry, Shafiq Joty, and Enamul Hoque. LLM-based data science agents: A survey of capabilities, challenges, and future directions. *arXiv preprint arXiv:2510.04023*, 2025a. URL https://arxiv.org/abs/2510.04023.

Mizanur Rahman, Amran Bhuiyan, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Ridwan Mahbub, Ahmed Masry, Shafiq Joty, and Enamul Hoque. Llm-based data science agents: A survey of capabilities, challenges, and future directions, 2025b. URL https://arxiv.org/abs/2510.04023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Marc Schmitt. Automated machine learning: AI-driven decision making in business analytics. *Intelligent Systems with Applications*, 18:200188, 2023. doi: 10.1016/j.iswa.2023.200188.

Dominik Schulz. The r package deseats for data-driven trend and seasonality estimation in time series. Working Papers CIE 169, Paderborn University, CIE Center for International Economics, March 2026. URL https://ideas.repec.org/p/pdn/ciepap/169.html. RePEc:pdn:ciepap:169.

Oleksandr Shchur, Ali Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Bernie Wang. Autogluon–timeseries: Automl for probabilistic time series forecasting. In Aleksandra Faust, Roman Garnett, Colin White, Frank Hutter, and Jacob R. Gardner, editors, *Proceedings of the Second International Conference on Automated Machine Learning*, volume 224 of *Proceedings of Machine Learning Research*, pages 9/1–21. PMLR, 2023. URL https://proceedings.mlr.press/v224/shchur23a.html.

Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. Lambda: A large model based data agent. *Journal of the American Statistical Association*, 2025. doi: 10.1080/01621459.2025.2510000. URL https://doi.org/10.1080/01621459.2025.2510000. Online first.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://openreview.net/forum?id=DV15UbHCY1.

The Open Group. *The TOGAF® Standard, 10th Edition - Introduction and Core Concepts*. van Haren Publishing, 2022. URL https://pubs.opengroup.org/togaf-standard/index.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*, ICLR 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. Timeseries-scientist: A general-purpose ai agent for time series analysis, 2025. URL https://arxiv.org/abs/2510.01538.

A Appendix A: README excerpt of the implemented MCP server

README excerpt of `time-series-analysis-mcp`

```
time-series-analysis-mcp is a Model Context Protocol (MCP) server for
reproducible time-series analysis and forecasting. It exposes a structured
tool catalogue for loading tabular time-series data, repairing data-quality
issues, running diagnostics, decomposing seasonal components, and producing
forecasts with statistical, decomposition-based, foundation-model, and AutoML
methods.

The server was developed as a scientific-paper implementation artifact. This
README is written so it can be used as an appendix describing the implemented
MCP server, its public interface, and representative forecasting tasks.

Project Metadata

- Public repository: https://github.com/LiChenStuttgart/time-series-analysis-mcp
- Author: LiChenStuttgart, 3380836@gmail.com
- Package name: time-series-analysis-mcp
- Current package version: 0.1.0
- License: MIT
- Python requirement: >=3.10
- MCP transport: stdio through mcp.server.fastmcp.FastMCP

Overall Summary

The MCP server acts as a bridge between an LLM-capable MCP client and a
time-series analysis library. It registers 14 public tools and 2 MCP resources.
The main workflow is:

1. Discover tools with list_tools or the guide://time-series-analysis resource.
2. Load tabular data into the canonical SeriesCollection contract with
   time_series_loader.
3. Handle missing values and outliers while preserving audit metadata.
4. Run stationarity, seasonality, or structural-break diagnostics.
5. Decompose series with DeSeaTS when seasonal components are part of the
   analysis.
6. Forecast with ARIMA/SARIMA, Holt-Winters exponential smoothing, DeSeaTS
   Semi-ARMA, Chronos-2, Toto 2.0, or AutoGluon TimeSeries.

Downstream tools accept one of three input styles:

- series_collection: the dataset object returned by time_series_loader or by a
  previous quality-handling tool.
- data: a direct list of numeric values, which is converted into a synthetic
  single-series collection.
- file_path plus loader arguments such as time_column, value_columns,
  dimension_columns, and frequency.

Tool outputs are JSON strings by default. Most tools also accept output_format
as json, markdown, or text, and save_path to persist a rendered result.

MCP Resources

- guide://time-series-analysis: live registry-generated tool catalogue, grouped
  by server, data preparation, data quality, diagnostics, decomposition, and
  forecasting.
- guide://data-contracts: canonical input and output schema documentation for
  SeriesCollection, quality flags, and forecast result objects.

Tool Catalogue

Group: server
Tool: list_tools
Description: Lists registered MCP tools, groups, descriptions, and usage guidance.

Group: data preparation
Tool: time_series_loader
Description: Loads CSV, TXT, JSON, XLSX, or XLS files and builds regular
multi-series SeriesCollection objects.

Group: data quality
Tool: time_series_missing_data_handler
Description: Detects and handles nulls, NaNs, and calendar gaps with strategies
such as auto, interpolation, seasonal median, rolling median, forward fill,
backward fill, or drop series.
```

```
Group: data quality
Tool: time_series_outlier_handler
Description: Detects point outliers with MAD, IQR, rolling Hampel, or auto
selection, then flags, winsorizes, interpolates, nulls, or seasonally replaces
them.

Group: diagnostics
Tool: time_series_unit_root_test
Description: Runs stationarity and unit-root diagnostics and can return
differencing recommendations.

Group: diagnostics
Tool: time_series_seasonality_test
Description: Scores and tests seasonal structure for daily, monthly, quarterly,
yearly, or explicit seasonal periods.

Group: diagnostics
Tool: time_series_structural_break_test
Description: Detects structural breaks and change points, including possible
level or regime shifts.

Group: decomposition
Tool: time_series_deseats_decompose
Description: Calls the R DeSeaTS package to produce trend, seasonality, fitted
values, residuals, deseasonalized values, and detrended values.

Group: forecasting
Tool: time_series_arima
Description: Fits optimized ARIMA, SARIMA, or ARIMAX-style models with optional
auto order selection, holdout metrics, prediction intervals, and known-future
covariates.

Group: forecasting
Tool: time_series_exponential_smoothing
Description: Fits optimized Holt-Winters exponential smoothing models with level,
trend, optional seasonality, simulation-based intervals, and holdout metrics.

Group: forecasting
Tool: time_series_deseats_forecast
Description: Calls DeSeaTS Seasonal Semi-ARMA forecasting for short-horizon
seasonal forecasts and interval bands.

Group: forecasting
Tool: time_series_chronos2_forecast
Description: Generates zero-shot probabilistic forecasts with optional Amazon
Chronos-2 weights, quantiles, covariates, and holdout metrics.

Group: forecasting
Tool: time_series_toto2_forecast
Description: Generates zero-shot probabilistic forecasts with optional Datadog
Toto 2.0 weights and can model aligned series together using group_by_dimensions.

Group: forecasting
Tool: time_series_automl_forecast
Description: Uses AutoGluon TimeSeries for candidate model fitting, validation
scoring, model selection, optional ensembling, leaderboard output, and
probabilistic forecasts.
```

B Appendix B: Example of user prompt, system prompt and skills

Example of user system prompts and skills

```
User prompt
Assume that a business user sends the following prompt to the business agent:

"Forecast monthly passenger car sales for the next six months by region.
Use the available historical sales data, check data quality and seasonal patterns,
compare suitable forecasting methods, and explain the resulting forecasts,
prediction intervals, and main limitations."


System prompt
A forecasting-specific system prompt should guide and constrain the agent's
behavior. A suitable system prompt for this use case can be expressed as follows:

"You are an enterprise time series forecasting agent. Before invoking forecasting
tools, clarify the business target, time frequency, aggregation level, dimensions,
forecast horizon, and required uncertainty output. Use the time series MCP server
for data loading, quality checks, diagnostics, decomposition, and forecasting.
Do not select an expensive forecasting method before checking data quality,
seasonality, stationarity, and structural breaks. Prefer transparent baselines
as comparison points. When presenting results, explain data quality interventions,
diagnostic evidence, selected methods, uncertainty intervals or quantiles, and
limitations."


Car Sales Forecasting Skill

"Purpose:
Generate an auditable sales forecast for passenger cars by region.

Trigger:
Use this skill when the user asks for future passenger-car sales or a sales
forecast by region or market.

Workflow:
1. Call list_tools to check available MCP tools for time series analysis and
   data access.

2. Load sales data with time_series_loader from the source system SAP Datasphere
   using monthly frequency and sum aggregation.

3. Handle missing values with time_series_missing_data_handler and outliers with
   time_series_outlier_handler.

4. Run diagnostics:
   - time_series_seasonality_test
   - time_series_structural_break_test
   - time_series_unit_root_test

5. Generate baseline forecasts with iteratively selected parameters:
   - time_series_exponential_smoothing
   - time_series_arima

6. If available and justified, compare with:
   - time_series_automl_forecast
   - time_series_chronos2_forecast

7. Compare metrics, intervals or quantiles, warnings, and data quality flags.

8. Explain the forecast by region, including uncertainty, detected patterns,
   selected method, and limitations.

Rules:
- Always include at least one transparent baseline.
- Do not remove sales outliers automatically; flag them first."
```