RESEARCH-ARTICLE

AutoDW-TS: Automated Data Wrangling for Time-Series Data

LEI LIU, Fujitsu America, Inc., Sunnyvale, CA, United States

SO HASEGAWA, Fujitsu America, Inc., Sunnyvale, CA, United States

SHAILAJA KEYUR SAMPAT, Fujitsu America, Inc., Sunnyvale, CA, United States

MEHDI BAHRAMI, Fujitsu America, Inc., Sunnyvale, CA, United States

WEI-PENG CHEN, Fujitsu America, Inc., Sunnyvale, CA, United States

KODAI TOYOTA, Fujitsu Limited, Minato, Tokyo, Japan

View all

Open Access Support provided by:

Fujitsu America, Inc.

Fujitsu Limited

Published: 10 November 2025

Citation in BibTeX format

CIKM '25: The 34th ACM International Conference on Information and Knowledge Management
November 10 - 14, 2025
Seoul, Republic of Korea

Conference Sponsors:
SIGWEB
SIGIR

AutoDW-TS: Automated Data Wrangling for Time-Series Data

Lei Liu
So Hasegawa
Shailaja Keyur Sampat
Mehdi Bahrami
Wei-Peng Chen
lliu@fujitsu.com
shasegawa@fujitsu.com
ssampat@fujitsu.com
mbahrami@fujitsu.com
wchen@fujitsu.com
Fujitsu Research of America Inc.
Santa Clara, California, USA

Kodai Toyota
Takashi Kato
Takumi Akazaki
Akira Ura
Tatsuya Asai
toyota.kodai@fujitsu.com
t_kato@fujitsu.com
akazaki.takumi@fujitsu.com
ura.akira@fujitsu.com
asai.tatsuya@fujitsu.com
Fujitsu Research
Kawasaki, Kanagawa, Japan

Abstract

Data wrangling — the process of preparing raw data for analysis through cleansing, transformation, and enrichment — is a critical step in the data science pipeline. Its importance is amplified for time-series data, which underpins many applications, with forecasting being one of the most prominent tasks. Yet, current practices remain largely manual, time-consuming, and error-prone, limiting productivity and scalability. In this paper, we introduce AutoDW-TS, an automated approach to time-series data wrangling powered by Large Language Models (LLMs). Our method offers an end-to-end pipeline, automating key stages such as table merging, prediction engineering, cleansing, imputation, and enrichment. To support diverse use cases, we developed multiple systems, including an interactive AutoDW-TS WebApp, Web APIs, and an AI agent. We share insights from developing and deploying these systems, along with results from an extensive evaluation across 38 time-series benchmarks. Our findings show that AutoDW-TS significantly improves forecasting performance, demonstrating its effectiveness and potential to transform time-series data preparation at scale.

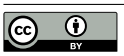
CCS Concepts

• **Information systems** → **Data management systems**; • **Computing methodologies** → **Artificial intelligence**.

Keywords

Data Wrangling, Large Language Models, Time-Series Forecasting

ACM Reference Format:
Lei Liu, So Hasegawa, Shailaja Keyur Sampat, Mehdi Bahrami, Wei-Peng Chen, Kodai Toyota, Takashi Kato, Takumi Akazaki, Akira Ura, and Tatsuya Asai. 2025. AutoDW-TS: Automated Data Wrangling for Time-Series Data. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746252.3761524



CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761524

1 Introduction

In the data science workflow, data wrangling is a crucial step that transforms raw datasets into high-quality, analysis-ready data [22, 66]. It includes merging tables, correcting errors, imputing missing values, and enriching datasets through transformations or external sources [16, 30]. Despite its importance, wrangling remains one of the most time-consuming tasks, with data professionals spending up to 80% of their time on preparation [62, 78, 81]. This inefficiency limits time for modeling and insight generation. As datasets grow in size and complexity, the need for scalable, consistent, and low-error solutions has intensified. Automating data wrangling reduces manual effort, minimizes errors, and accelerates the data pipeline.

Data wrangling underpins many applications, including analytics, business intelligence, and machine learning (ML) [12, 65]. Among these, time-series forecasting is especially impactful in domains such as finance, healthcare, and energy [13]. The accuracy of time-series forecasts relies heavily on the quality and consistency of the input data [41]. However, such data often exhibit issues such as missing timestamps, inconsistent formats, and limited feature richness, which can significantly impair model performance [11, 32].

To this end, we introduce **AutoDW-TS**, an automated framework for time-series data wrangling. Leveraging LLMs, AutoDW-TS cleans, structures, and enriches datasets with minimal human input. It extends our earlier work on tabular data [49] with significant improvements in prediction engineering, imputation, and cleansing for time-series data, along with new modules for table merging and external enrichment. Our key contributions are:

- **AutoDW-TS Framework:** We propose a comprehensive, end-to-end solution for automating time-series data wrangling. AutoDW-TS introduces novel techniques that eliminate the need for extensive manual configuration. Unlike traditional tools that are often labor-intensive and error-prone, AutoDW-TS streamlines the pipeline, reducing effort while improving forecast accuracy.
- **System and Deployment:** We developed multiple systems, including an interactive AutoDW-TS WebApp, Web APIs, and an AI agent that collaborates with other agents in broader automation ecosystems. These have been deployed in real-world environments for both public and enterprise users, demonstrating AutoDW-TS's practicality, scalability, and versatility.

- **Evaluation:** We evaluated AutoDW-TS on 38 time-series benchmarks. Results show AutoDW-TS substantially enhances forecasting performance, highlighting its effectiveness and potential for large-scale time-series data preparation.

2 Related Works

2.1 Data Wrangling

Historically, data wrangling has relied on manual processes using Python [53, 54] and R [8, 33], with libraries like Pandas [61] and dplyr [58] providing functions for cleaning, transformation, and filtering [16, 66]. While flexible, manual approaches are time-consuming and error-prone, especially for large or messy datasets [30]. To address this, AI-enhanced platforms such as Alteryx [5], DataRobot [19], and dotData [21] offer semi-automated, user-friendly wrangling. Despite being low- or no-code, they often require manual pipeline configuration and workflow tailoring, limiting full automation. Recently, LLMs have been explored to generate Python code for wrangling tasks [2, 14, 38]. However, they are limited by context window size and lack tabular-specific training, often producing incomplete or incorrect code. For instance, [38, 39] report that state-of-the-art LLMs achieve low correctness in some wrangling tasks, highlighting the gap between LLM capabilities and the demands of production-level automation. In our previous work [49], we introduced AutoDW, an automated wrangling framework for general-purpose tabular data. Rather than relying on LLMs to generate code, AutoDW uses them to recommend configurations and strategies, while the actual execution is performed by our own processing modules. However, AutoDW does not yet address the unique challenges of time-series data, which require tailored approaches due to temporal dynamics and structural complexity.

2.2 Time-Series Forecasting and AutoML

Time-series forecasting is a central topic in ML and statistics, with extensive research on classical and modern methods. Traditional methods such as ARIMA and exponential smoothing models have long served as the foundation for univariate forecasting tasks [9, 40]. More recent developments have focused on deep learning architectures, including recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer-based models, which have shown promise in capturing complex temporal dependencies in multivariate settings [44, 47, 84]. Hybrid models that integrate statistical and neural network approaches have also gained popularity, aiming to leverage the strengths of both paradigms [73].

In parallel, Automated Machine Learning (AutoML) [4, 35, 42] frameworks for time-series forecasting have emerged to automate the pipeline of model selection, hyperparameter tuning, and sometimes even data preprocessing. Tools such as Auto-sklearn [26], Prophet [77], H2O AutoML [45], AutoGluon-TimeSeries [72], SapientML [68], auto-sktime [86], and Amazon SageMaker Autopilot [6] have extended their capabilities to include time-series forecasting components. These systems aim to lower the barrier to entry for users by offering end-to-end solutions that require minimal manual intervention, while still delivering competitive performance. However, most of these tools focus on model automation and often assume that data is already well-prepared, leaving out the critical step of automated time-series data wrangling [15, 35, 46].

3 AutoDW-TS: Technical Details

3.1 Solution Overview

Figure 1 illustrates the AutoDW-TS framework. The process begins with the input of one or more tabular datasets (e.g., CSV, XLS, TSV, etc.). If multiple datasets are uploaded, they will be combined into a single dataset, as explained in Section 3.2. A prediction engineering module is then automatically initialized to predict the most likely task specification for the time-series forecasting. This includes identifying the ID, target, time, series columns and determining the forecast frequency by leveraging the power of LLMs, as will be detailed in Section 3.3. Next, an FTI algorithm is applied to predict the feature type for each column in the time-series dataset, as provided in Section 3.4. Based on the FTI results, data wrangling options are recommended, typically involving data cleansing, data imputation, and data enrichment, as detailed in Sections 3.5, 3.6, and 3.7 respectively. The wrangled dataset can then be used for downstream applications such as time-series forecasting.

3.2 Table Merge

Time-series data often spans multiple tabular files, either due to varied data sources or intentional segmentation for management and analysis. To enable unified data preparation and ML, we develop a Table Merge module that automatically consolidates these files into one comprehensive table. While this is a well-known problem, existing tools often require a human-in-the-loop setting [18, 21] or assume specific conditions like matching column names or predefined key relationships (e.g., SQL Server [69], MS Power BI [7]). In contrast, our solution is fully automated and does not rely on such assumptions. The merge process (Figure 1(A)) involves 6 steps:

(1) **Master Table Determination:** After the datasets are uploaded, an LLM analyzes column headers to identify the target column, typically representing a key business indicator. The table containing this target column is designated as the master table.
(2) **Joinable Column Discovery:** We use a value-based schema matching method with Shannon Entropy [71] to detect joinable columns. For each column pair $c_i \in T_1$, $c_j \in T_2$, where T_1 and T_2 are a pair of datasets, the entropy is calculated as:

$$H(c_i, c_j) = -\sum_{v \in V} p(v) \cdot \log_2 p(v)$$

where V is the set of non-empty values in c_i that also appear in c_j, and $p(v)$ is the probability of value v occurring in the intersection. Column pairs with entropy values greater than a predefined threshold (e.g., 1) are considered joinable.
(3) **Merge Type Classification**: If most columns are joinable one-to-one, the tables are 'Unionable' and merged row-wise (e.g., daily sales data). If only a few columns match, they are 'Column-Joinable' (e.g., product and order table join via 'product_id').
(4) **Unionable Merge:** Unionable tables are merged first. Merged tables are then removed, retaining the result and any remaining column-joinable tables.
(5) **Join Order for Column-Joinable Tables**: We assess cardinality (e.g. one-to-one, one-to-many) and build a weighted, directed graph based on entropy to determine the optimal join order, ensuring all tables connect to the master table via shortest paths.

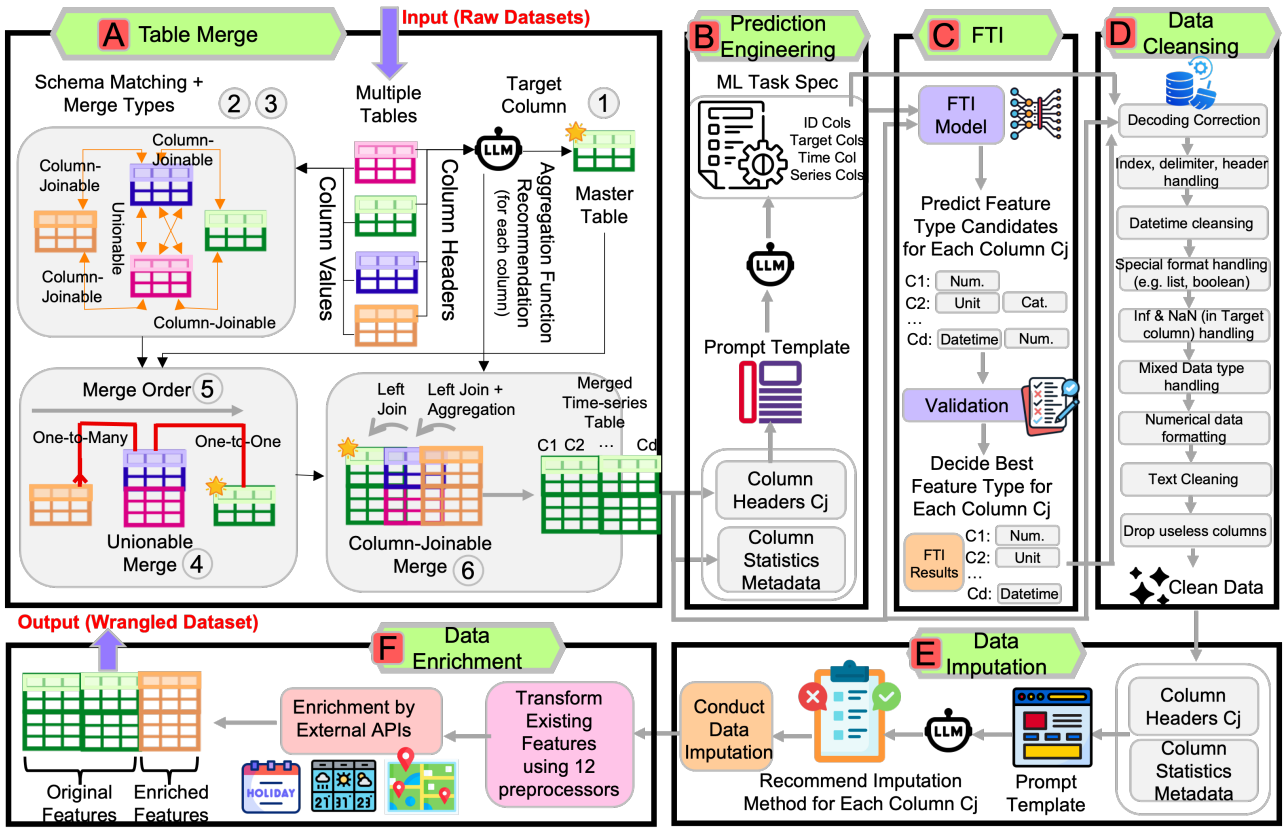


Figure 1: Overview of AutoDW-TS: An end-to-end framework for automated data wrangling in time-series datasets.

(6) **Column-Joinable Merge and Aggregations:** Following the directed join order from the graph, all column-joinable tables are merged iteratively using left joins to preserve the master table's data distribution, which is essential for machine learning. For one-to-many or many-to-many relationships, an LLM recommends the top three aggregation functions — selected from minimum, maximum, count, sum, average (mean), or mode — to incorporate key statistical information during the join [60].

3.3 Prediction Engineering

As shown in Figure 1(B), Prediction Engineering automatically recommends ML task configurations for time-series forecasting, typically including the following columns:

- **ID columns**: distinguish between different time series.
- **Target columns**: numeric values to be predicted.
- **Time column**: timestamps or date-time values.
- **Series columns**: describe the values in the time series, including the Target columns. These are optional and tool-dependent.

The task configuration also includes the *forecast frequency* (i.e., prediction length), often inferred from the time column, which specifies the interval to forecast. A valid dataset must include at least ID, Time, and Target columns. For univariate data lacking an ID column, AutoDW-TS adds one automatically.

Specifying configurations manually in AutoML tools can be time-consuming and error-prone, especially for datasets with many columns. To automate this, we design a prompt template and use an LLM to recommend configurations. The prompt lists all column names and definitions of ID, Target, Time, and Series columns. An optional example can be included for in-context learning. For each column, required statistical metadata is provided—including column type, data type, sample values, and summary statistics (e.g., mean, standard deviation, and value range for numeric columns, and the number of unique categories and top frequent values for categorical columns). This metadata is crucial, as column names are often arbitrary and semantically uninformative.

3.4 Feature Type Inference (FTI)

The FTI module (Figure 1(C)) predicts feature types for each column in time-series data and determines appropriate enrichment methods for later stages. Built on our previous work [49], FTI in AutoDW-TS includes significant extensions for time-series support. Unlike existing tools [23, 70] that support only a few feature types, our FTI expands the taxonomy to 12 classes: *Numerical*, *Categorical*, *Datetime*, *Sentence*, *URL*, *Embedded-Number*, *List*, *Ignorable ID*, *Unit*, *Range*, *Inequality Sign*, and *Formatted ID*. This finer-grained classification enables enrichment tailored to each column's semantics.

We train a two-stage LightGBM [43] classifier to predict candidate feature types, followed by a validation procedure to select the best type [49]. To handle time-series data, the validation incorporates time-series-specific feature engineering. For columns identified as *Series Columns*, lagged features and rolling window

aggregations [20] are generated and appended to the dataset. Each candidate feature type is paired with its enrichment option, and the resulting pipeline is evaluated using a fixed ML model. The type achieving the highest performance is selected as optimal.

3.5 Data Cleansing

Effective data cleansing, by eliminating errors, inconsistencies, and irrelevant entries, plays a vital role in the data preprocessing workflow, as it directly influences the accuracy and robustness of subsequent ML tasks [16, 31]. The data cleansing component in AutoDW-TS builds upon [49], incorporating several enhancements tailored for time-series datasets. Specifically, this module includes procedures such as decoding the dataset's structure, identifying correct delimiters, and determining the appropriate tabular headers. It also supports specialized processing for datetime, list, and boolean fields. To meet the stringent input requirements of ML algorithms, the module further processes infinite values, removes rows with missing values in the target column, and drops uninformative features (e.g., columns with constant values). It also resolves issues related to inconsistent data types within a single column by enforcing type uniformity. For textual data, operations like trimming spaces and standardizing formatting are performed. Additionally, given that time columns in time-series data may contain multiple datetime formats, the module normalizes these into a consistent format to ensure proper temporal analysis, as shown in Figure 1(D).

3.6 Data Imputation

Imputation is a key step in data preprocessing that aims to fill in missing values to enable reliable downstream analysis. In time-series data, this task is particularly challenging due to temporal dependencies, seasonality, and potential multivariate interactions across time steps [63]. Traditional imputation methods for time-series data include statistical techniques such as forward fill, backward fill, interpolation, and more advanced model-based approaches like K-Nearest Neighbors (KNN) [51], iterative imputation [37], and deep learning-based strategies [24]. Recently, LLMs have emerged as powerful tools for data imputation by performing imputation based on contextual cues embedded in the surrounding data [57]. These methods prompt LLMs with partial tabular data to generate plausible imputed values in natural language or structured formats.

In AutoDW-TS, we propose a low-cost yet effective LLM-based imputation approach that recommends the most suitable imputation method for each column, rather than relying on an LLM to directly fill in missing values. Our method prompts the LLM to reason about the optimal imputation strategy using relevant analytical context. For numerical columns, the prompt includes information such as the column name, data type, feature type, sample values, missing value percentage, mean, minimum, maximum, standard deviation, missing patterns, seasonality, and trend. Missing patterns are described in the form "there are M blocks with N consecutive `NaNs`." Seasonality is identified by detecting periodic patterns via autocorrelation [9], while trends are detected using linear regression [40]. Based on this information, the LLM selects from a set of imputation methods, including forward fill, backward fill, interpolation, moving average, seasonality-based imputation, and KNN imputation. For categorical columns, we additionally include statistics such as the number of categories, number of unique values, and most frequent values, and ask the LLM to choose from forward fill, backward fill, seasonality-based imputation, or mode imputation.

3.7 Data Enrichment

The purpose of data enrichment (Figure 1(F)) is to generate new features from existing columns, ultimately improving the performance of downstream tasks such as ML model accuracy. In AutoDW-TS, two enrichment options are supported: transforming existing features (Section 3.7.1) and leveraging external data (Section 3.7.2).

3.7.1 Data Enrichment by Transforming Existing Features. For each column, FTI predicts the most appropriate feature type. Based on the predicted feature type, a specific preprocessor is applied to transform the original feature and enrich the dataset, as detailed in [49]. For example, if a column is identified as a *URL* feature type, the URL preprocessor extracts and generates multiple features such as the domain, URL path, path depth, and filename. This process is applied consistently across all 12 feature types supported by FTI, with each corresponding preprocessor responsible for transforming and enriching the dataset accordingly.

3.7.2 Data Enrichment by External Data. To enhance contextual understanding and predictive accuracy, we propose an automatic data enrichment framework that integrates external information from publicly available Web APIs for time-series data. For instance, incorporating weather and holiday data can improve forecasts of daily store sales, road traffic, or household electricity usage [76]. AutoDW-TS uses four Web APIs (Table 1) for this purpose. Since each API requires specific endpoint formatting, the process begins by identifying columns suitable for constructing API requests. Then, API calls are executed row by row using dynamic endpoints, and the responses are parsed to append new features to the dataset. As not all enriched features are useful, a time-series-aware feature selection step is applied to retain only the most relevant ones.

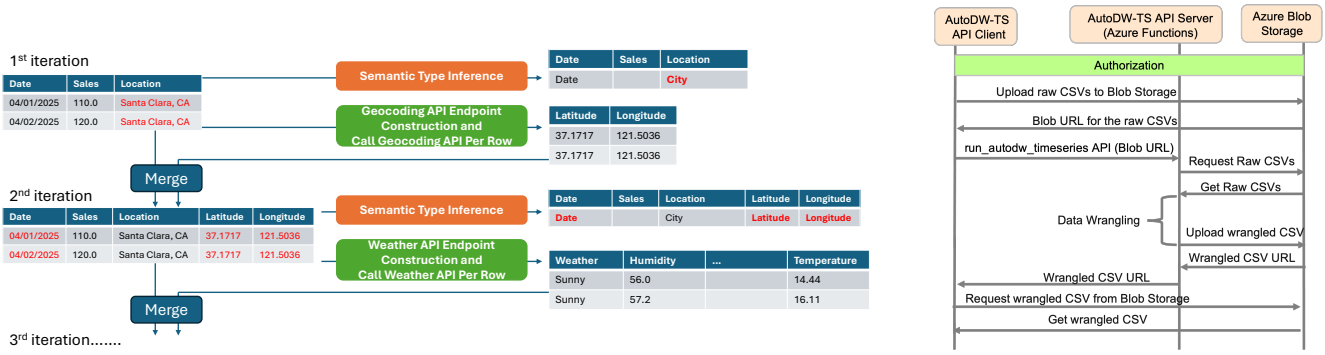
Step 1 - LLM-based Enrichment via API: This enrichment step operates in multiple iterations (Fig. 2(A)) until no new API calls are generated. It starts with zero-shot LLM-based classification to infer each column's semantic type using its name and sample values. These inferred types are matched against predefined input requirements for each API (Table 1). If a column's type satisfies an API's input condition, it is deemed eligible for endpoint construction.

Executable APIs are then selected based on semantic matches. For the *Holiday*, *Weather*, and *Reverse Geocoding* APIs, each row's value is inserted into an endpoint template with placeholders. For the *Geocoding API*, multiple attributes (e.g., *Country*, *State*, *City*) are combined to form a full address. Responses are parsed to extract the output attributes listed in Table 1 and added to the dataset. While most APIs allow straightforward extraction, some require special handling due to unique response formats, as detailed below.

(1) **Forecast vs. Actual Data:** The *Weather API* provides both forecast and actual weather data. AutoDW-TS prioritizes forecast data for consistency during training and testing, aligning with real-world scenarios where only future information is available at prediction time. If forecast data is unavailable — typically when the prediction date is too far ahead — actual weather data

Table 1: Catalog of Web APIs utilized by AutoDW-TS for external knowledge-based enrichment.

API Name	Host	List of Inputs	List of Outputs
Holiday API	Nager.Date [56]	Country, Date	Holiday Flag
Geocoding API	OpenStreetMap [59]	Address	Latitude, Longitude
Reverse Geocoding API	OpenStreetMap [59]	Latitude, Longitude	Address
Weather API	Open-Meteo [85]	Latitude, Longitude, Date	Weather, Temperature, Feels_Like, Temperature, Humidity, Dew Point, Pressure, Precipitation, Clouds, Wind_Speed, Wind_Degree, Wind_Gust



(A) (B)

Figure 2: (A) Data enrichment by external APIs in AutoDW-TS. (B) AutoDW-TS WebAPI deployment for large file support.

from the same date one year earlier is used as a substitute. To simulate the uncertainty inherent in forecasts, random noise is added to this historical data. Similarly, if historical forecast data is inaccessible, noise is added to the corresponding actual historical records to preserve consistent modeling conditions.

(2) **Temporal Aggregation:** APIs like *Weather* and *Holiday* provide daily data, which must be aggregated when the timeseries granularity is coarser (e.g., weekly/monthly). For holidays, the total number of holidays per interval is computed. For weather data, numerical values (e.g., temperature, humidity) are averaged, while categorical data (e.g., sunny, rainy) are represented by the mode to capture the dominant pattern in the interval.

Step 2 - Feature Selection: API enrichment can introduce many features, especially from the *Weather API*, which may return dozens of attributes per call. To avoid overfitting and reduce redundancy, feature selection is performed using a Random Forest [10] model, removing enriched features with importance scores below a specified threshold and retaining only high-impact variables.

This two-step process produces an enriched time-series dataset with semantically relevant external context, making it more suitable for downstream prediction tasks. Many real-world datasets, such as store sales data, lack explicit geographic information. AutoDW-TS enables enrichment in such cases by allowing users to provide geographical information (e.g., state, city, address) as free-text input. A new column is generated from this input to construct API endpoints for *Geocoding*, *Holiday*, and *Weather* APIs, thereby enhancing the dataset with spatially aware external signals.

4 AutoDW-TS Industry Deployment

To support diverse usage scenarios, we developed multiple systems, including an interactive AutoDW-TS WebApp and AutoDW-TS Web APIs, which have been deployed in real-world environments. In anticipation of emerging trends, we have also developed an AutoDW-TS AI agent designed to interoperate with other AI agents for broader automation use cases. The LLM used in all the deployments is GPT-4, provided by Azure OpenAI with API version `2024-02-15-preview`.

The preliminary version of the AutoDW-TS tool has been publicly accessible through the Fujitsu Research Portal[1] since June 2024. An advanced version, with many additional functionalities, is available exclusively to business users on the Fujitsu Kozuchi Platform[2], with plans for a public release at a later date. A demonstration video showcasing the core operations of AutoDW-TS can be viewed at[3].

4.1 AutoDW-TS WebApp

The AutoDW-TS WebApp is developed using Streamlit [25], providing a user-friendly GUI for interacting with and customizing AutoDW-TS operations. The full AutoDW-TS pipeline, illustrated in Figure 1, is implemented within the WebApp. Users can upload one or more datasets, after which the WebApp automatically executes the complete data wrangling process. While recommended configurations and wrangling options are displayed, users can modify these suggestions to customize the pipeline according to their needs. The application is containerized and deployed via Azure Virtual Machines (VM) [50] and Azure Kubernetes Service [3]. Upon request, a dedicated VM instance is provisioned automatically for each user, ensuring scalability and consistent performance.

4.2 AutoDW-TS Web APIs

Complementing the WebApp, which emphasizes user interaction, we have also implemented and deployed AutoDW-TS Web APIs for easier system integration and parallel processing. These APIs are built using the JSON-RPC protocol [74] and deployed via Azure

[1] https://en-documents.research.global.fujitsu.com/auto-data-wrangling/
[2] https://www.fujitsu.com/global/services/kozuchi/
[3] https://youtu.be/pu2PewtCxs4

Functions [28] and Azure API Management [52]. The core API, `run_autodw_timeseries`, performs end-to-end data wrangling on a list of input datasets, following the same pipeline shown in Figure 1. It also supports optional parameters, including task specifications, feature types, wrangling options, and test datasets, allowing users to tailor the process to specific needs. Additionally, we have deployed APIs for each individual module in the pipeline — such as prediction engineering, FTI, and each data enrichment preprocessor — totaling 25 Web APIs. This modular design gives users fine-grained control over each step of the data wrangling process.

4.3 AutoDW-TS AI Agent

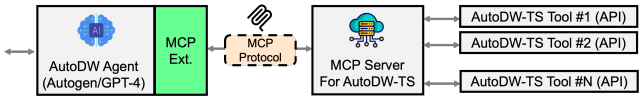


Figure 3: AutoDW-TS AI agent.

AI agents are becoming an increasingly important paradigm for automating complex workflows, enabling systems to make decisions, coordinate tasks, and interact with other agents or users flexibly and intelligently [1, 55]. The AutoDW-TS AI Agent plays a critical role in automating and scaling time-series data wrangling through intelligent decision-making and tool orchestration. As shown in Figure 3, it integrates multiple AutoDW-TS tools, each corresponding to a specific AutoDW-TS function, enabling fine-grained control over various wrangling operations. The agent is implemented using the AutoGen framework [27] and communicates with the AutoDW-TS Model Context Protocol (MCP) server via the MCP protocol [64], ensuring structured and consistent interaction with backend services. At its core, the agent features a tool selector module that maintains knowledge of each AutoDW-TS MCP tool's capabilities, input requirements, and output formats. This allows the agent to automatically select the most appropriate tool based on the current context of the conversation. While the AutoDW-TS AI Agent has not yet been deployed, we plan to make it available soon. Once deployed, it will support agent-to-agent (A2A) communication [79], enabling collaboration with other AI agents to accomplish more complex, multi-step data tasks.

4.4 Experience Learned from Deployment

- **Use Cases:** Data wrangling is essential across a wide range of industries, extending beyond ML applications. For instance, users of data-lake platforms often require comprehensive data analytics following the wrangling process. To address these broader needs, AutoDW-TS has been extended to support additional use cases such as data insight discovery, visualization, and Exploratory Data Analysis (EDA) [80] report generation. To maximize its impact, future enhancements should continue to expand AutoDW-TS's capabilities to support an even wider array of scenarios.
- **Large File Support:** Deployment trials revealed that Web API platforms like Azure Functions have payload size limitations [48], restricting their use with large datasets. To address this, we integrated Azure Blob Storage [75] into the deployment architecture. For large files, the AutoDW-TS Web API client uploads datasets directly to Blob Storage and passes the file URL to the API. Azure Functions then retrieve the file directly from Blob Storage, thereby avoiding payload size constraints. The detailed architecture for large file support is illustrated in Fig. 2(B).
- **Data Privacy:** Data privacy [34, 83] has emerged as a key concern, particularly for users reluctant to upload data to the AutoDW-TS WebApp/Web APIs or expose it to external LLMs such as GPT-4. To address this, AutoDW-TS supports on-premise deployment with enterprise-grade LLMs (e.g., Cohere [17]) or capable open-source LLMs, though this may involve some performance trade-offs. In addition, exploring techniques such as anonymizing sensitive data and using partial datasets for wrangling [29, 82] can further mitigate privacy risks.

5 Evaluation and Discussion

To evaluate AutoDW-TS's performance for data wrangling and its benefits for time-series forecasting, we conducted a comprehensive benchmark evaluation using 38 datasets that are commonly used for time-series forecasting, as summarized in Table 2. These datasets vary widely in the number of rows and columns, covering both univariate and multivariate time-series forecasting problems. For comparison, we first used the original training datasets with AutoML to obtain the ML model/pipeline and evaluated the Root Mean Square Error (RMSE) scores [36] on the testing dataset using the generated ML model/pipeline. After that, we used AutoDW-TS to clean and enrich the original datasets before applying them to AutoML to get the revised RMSE scores. We selected two time-series forecasting tools for performance evaluation: Prophet [77], a widely used forecasting model, and SapientML [68] with time-series extensions, a recent AutoML system that has been shown to outperform many existing alternatives [68]. For AutoDW-TS, we provide the ground-truth task configuration as input, bypassing the prediction engineering module to ensure that all comparisons are based on identical configurations. The rest of the data wrangling is fully automated using AutoDW-TS's recommended options.

Experimental results are shown in Table 2. Several failed cases with the original datasets were resolved after using AutoDW-TS, underscoring the importance of data cleansing in correcting diverse dataset errors in time-series tasks. More importantly, results from both SapientML and Prophet demonstrate that AutoDW-TS enhances data quality through data enrichment, leading to improved forecasting performance. After applying AutoDW-TS, the number of benchmarks where the tool outperformed the baseline increased significantly (26 vs. 8 for SapientML, 15 vs. 4 for Prophet).

We also analyzed the percentage change in RMSE after applying AutoDW-TS. For the 8 datasets where RMSE worsened when using SapientML, the average degradation was 6.41%. In contrast, among the 19 datasets that showed improved RMSE (excluding the 7 previously failed cases), the average improvement was 13.88%. A similar trend was observed for Prophet: in the 4 degraded cases, the average RMSE increased by 5.36%, whereas the improved cases exhibited an average RMSE reduction of 21.41%.

These results indicate that, while AutoDW-TS may not improve forecasting performance for every dataset, the magnitude of degradation is typically much smaller than the improvements it delivers. Notably, these averages exclude datasets that failed under AutoML tools but succeeded after applying AutoDW-TS. Including them

Table 2: Benchmark results of AutoDW-TS. (Freq.: Forecast frequency. Wrangled: Dataset after applying AutoDW-TS with additional features. w/o ADW: Without using AutoDW-TS. w/ ADW: With AutoDW-TS.)

Benchmark	Type	Freq.	Train		Test	Wrangled	SapientML		Prophet	
			Row	Col	Row	Col	w/o ADW	w/ ADW	w/o ADW	w/ ADW
DLF DATA SET 25-09-2019-TO-23-09-2021	Multivariate	Daily	400	15	101	1549	Failed	$\mathbf{2.511 \times 10^9}$	7.670×10^9	7.670×10^9
Walmart Sales Dataset of 45 stores	Multivariate	7 days	5130	8	1305	9	1.388×10^5	$\mathbf{1.352 \times 10^5}$	1.365×10^5	1.365×10^5
Max Planck Weather Dataset	Multivariate	10 mins	336440	15	84111	17	Failed	**9.554**	9.501	**6.382**
Gold rates (1985 - Present)	Multivariate	Daily	8091	7	2023	9	3.275×10^4	3.275×10^4	1.365×10^5	1.365×10^5
Pharma sales data	Multivariate	Daily	1684	13	422	15	6.648	**6.273**	**8.252**	8.842
Worldwide Commodity Prices	Multivariate	Monthly	347	64	87	66	$\mathbf{8.660 \times 10^1}$	9.273×10^1	1.194×10^2	1.194×10^2
World Deaths and Causes	Multivariate	Yearly	5800	35	1473	36	Failed	$\mathbf{6.091 \times 10^2}$	Failed	$\mathbf{4.712 \times 10^2}$
Daily Electricity Price and Demand Data	Multivariate	Daily	1684	14	422	16	1.083×10^4	$\mathbf{1.045 \times 10^4}$	Failed	$\mathbf{9.219 \times 10^3}$
NIFTY-50 Stock Market Data	Multivariate	Daily	188123	15	47069	15	$\mathbf{3.128 \times 10^3}$	3.392×10^3	3.829×10^3	3.829×10^3
S&P 500 Stock Data	Multivariate	Daily	495125	7	123915	8	3.383×10^1	$\mathbf{3.347 \times 10^1}$	2.554×10^1	2.554×10^1
NEW BITCOIN DataSet for the 2023	Multivariate	Minutely	35712	12	8928	13	5.359×10^2	$\mathbf{1.313 \times 10^2}$	2.537×10^3	$\mathbf{1.454 \times 10^3}$
Bike Sharing Washington DC	Multivariate	Daily	2337	29	585	17	Failed	$\mathbf{2.926 \times 10^3}$	Failed	$\mathbf{2.754 \times 10^3}$
Mobile Application User Statistics	Multivariate	Hourly	135	5	34	775	2.982×10^1	$\mathbf{2.148 \times 10^1}$	3.633×10^1	$\mathbf{2.856 \times 10^1}$
Gold Rate History in TamilNadu (2006-2020)	Multivariate	Daily	3976	6	995	7	Failed	$\mathbf{1.640 \times 10^2}$	$\mathbf{6.475 \times 10^2}$	6.493×10^2
Shampoo Sales Dataset	Univariate	Monthly	28	2	8	4	9.100×10^1	$\mathbf{7.252 \times 10^1}$	2.298×10^2	2.298×10^2
Egg Sales Dataset	Univariate	Daily	10234	2	358	5	$\mathbf{8.056 \times 10^1}$	8.112×10^1	$\mathbf{2.674 \times 10^1}$	2.687×10^1
Electricity Production	Univariate	Monthly	317	2	80	4	5.247	5.247	4.443	4.443
Air Baggage Complaints	Multivariate	Monthly	201	8	51	15	7.513×10^3	$\mathbf{4.632 \times 10^3}$	4.609×10^3	4.609×10^3
Wind Speed Prediction	Multivariate	Daily	5259	9	1315	12	**4.472**	4.494	4.722	4.722
Monthly Sunspot Dataset	Univariate	Monthly	2612	3	653	5	$\mathbf{7.772 \times 10^1}$	7.789×10^1	9.040×10^1	$\mathbf{8.995 \times 10^1}$
Delhi Daily Climate	Multivariate	Daily	1462	5	114	7	3.753	**3.508**	2.667	2.667
Monthly Global Temperature	Univariate	Monthly	2630	3	658	5	2.743×10^{-1}	$\mathbf{2.407 \times 10^{-1}}$	2.563×10^{-1}	$\mathbf{2.528 \times 10^{-1}}$
Annual Global Temperature	Univariate	Yearly	218	3	56	4	2.217×10^{-1}	$\mathbf{2.041 \times 10^{-1}}$	3.549×10^{-1}	$\mathbf{3.275 \times 10^{-1}}$
ETT (Electricity Transformer Temperature)	Multivariate	Hourly	13936	8	3484	10	9.549	**8.761**	1.407×10^1	**9.850**
Household Power Consumption	Multivariate	Minutely	1660207	9	415052	11	3.052	**2.492**	**3.693**	4.290
Rossmann-store-sales	Multivariate	Daily	975164	9	42045	10	$\mathbf{1.066 \times 10^3}$	1.203×10^3	1.054×10^3	1.054×10^3
Real-world sales	Multivariate	Daily	7070	4	2352	4	3.894×10^3	3.894×10^3	1.485×10^4	1.485×10^4
Tetuan City Power Consumption	Univariate	Daily	291	7	73	9	2.118×10^3	$\mathbf{2.099 \times 10^3}$	2.008×10^3	2.008×10^3
MaunaLoa	Univariate	Monthly	153	3	39	4	Failed	**1.684**	Failed	$\mathbf{9.236 \times 10^{-1}}$
Houston Power Usage	Univariate	Daily	1198	6	300	14	1.170×10^1	$\mathbf{1.144 \times 10^1}$	5.604	5.604
Pollution	Univariate	Hourly	35059	13	8765	11	Failed	$\mathbf{7.374 \times 10^1}$	Failed	Failed
Australia Electricity Demand	Univariate	Daily	1347	4	337	8	8.787×10^3	$\mathbf{8.768 \times 10^3}$	9.938×10^3	$\mathbf{9.794 \times 10^3}$
French Bakery Sales	Univariate	Daily	480	4	120	13	1.671×10^2	$\mathbf{1.532 \times 10^2}$	4.378×10^2	$\mathbf{3.638 \times 10^2}$
Walmart Sales of Single Store	Univariate	Weekly	91	4	23	772	7.008×10^4	$\mathbf{6.562 \times 10^4}$	2.436×10^5	$\mathbf{2.067 \times 10^5}$
Air pollution measurements	Multivariate	Hourly	4864	12	2247	14	5.944×10^1	$\mathbf{4.913 \times 10^1}$	Failed	Failed
rohlik-orders-forecasting-challenge	Multivariate	Daily	6943	18	397	18	$\mathbf{1.169 \times 10^3}$	1.195×10^3	2.504×10^3	2.504×10^3
Temperature forecast	Multivariate	15 mins	58960	9	5360	10	**8.958**	1.148×10^1	8.008	**2.796**
cart-time-series	Multivariate	Daily	12155	3	3039	5	2.020×10^1	2.020×10^1	2.562×10^1	2.562×10^1
#Winners							**8**	**26**	**4**	**15**

Table 3: Prediction engineering results.

Prediction	Prompt w/o Statistics	Prompt w Statistics
Target Columns	76.3%	84.2%
Series Columns	71.1%	78.9%
ID Columns	78.9%	86.8%
Date Column	100%	100%

would further emphasize the effectiveness of AutoDW-TS in enhancing time-series forecasting.

As previously noted, the benchmark study uses ground-truth task configurations, bypassing prediction engineering. To assess the accuracy of prediction engineering, we applied the method described in Section 3.3 to the 38 datasets, with results summarized in Table 3. Results show that LLMs such as GPT-4 can accurately infer task configurations for time-series forecasting with sufficient context. Accuracy improves further when column statistical metadata is included in the prompt, as column names alone—often arbitrary and lacking semantic meaning—are inadequate. Prediction engineering enables end-to-end automation in AutoDW-TS, significantly reducing the need for manual task configuration.

Finally, we conducted ablation studies using SapientML. For benchmarks with multiple tables, training on the merged table improved average RMSE by 5.42% compared to using only the master table. Replacing the imputation module (Section 3.6) with a baseline method—linear interpolation combined with forward/backward fill for missing values at the beginning and end—led to a 2.13% performance drop. Disabling enrichment via external APIs reduced performance by 3.46%. These findings indicate that each module in AutoDW-TS contributes to improved forecasting accuracy.

6 Conclusions

In this paper, we introduced AutoDW-TS, which represents a significant step toward automating the complex and labor-intensive process of time-series data wrangling. By leveraging the power of LLMs, our system delivered an end-to-end pipeline that reduces the need for manual intervention, thereby improving both efficiency and accuracy. Through the development and deployment of practical tools — including a WebApp, Web APIs, and an AI agent — we demonstrate the feasibility and impact of integrating automation into real-world time-series workflows at scale. Our experimental results across 38 benchmarks validate the effectiveness of AutoDW-TS in enhancing forecasting performance. As data-driven applications continue to grow, we believe that automated solutions like AutoDW-TS will be essential in scaling time-series analytics and enabling broader, more intelligent data automation ecosystems.

Disclosure of Generative AI Usage

This section provides a full disclosure of how Generative AI (GenAI) tools and large language models (LLMs) were used at different stages of the research, including system implementation, data processing, and manuscript preparation.

In AutoDW-TS, GenAI and LLMs are leveraged to support end-to-end automation across the data wrangling pipeline. The technical implementation of this approach is described in detail in the main body of the paper. Throughout the process, multiple prompts are dynamically generated and submitted to GenAI/LLMs, enabling assistance at various stages of data wrangling.

For instance, during the table merge phase, prompts may be constructed from a predefined template to obtain recommendations for aggregation functions, as described in Section 3.2. Within this template, variables enclosed in {...} serve as placeholders. AutoDW-TS dynamically populates these placeholders with content from the input datasets. For example, {max_recommendations} specifies the maximum number of aggregation function recommendations an LLM should return (default = 3), while {column_names_desc} lists the input dataframe columns along with optional metadata for each column.

LLM Prompt Template for Table Merge

```
In a database, two tables may have a one-to-many join
     relationship. For example, one author may have multiple
      books, a person may have multiple children, or a
     doctor may have multiple patients. In such cases, when
     two tables are joined, aggregation functions are
     applied to the secondary table where multiple matches
     exist. Commonly used aggregation functions include: (1)
      Count, (2) Sum, (3) Minimum, (4) Maximum, (5) Average,
      and (6) Mode.

The choice of aggregation function depends on the type of
     the column (e.g., numerical, categorical, date/time,
     text) and the information it contains. For example, for
      a numeric column like 'Phone Number' or a text column
     such as 'Address', Count is the most appropriate
     aggregation function as it indicates how many matches
     exist in the secondary table. It does not make sense to
      calculate the minimum, maximum, or average of a 'Phone
      Number' or 'Address'; therefore, those are not useful
     aggregation functions. For numerical columns such as '
     Salary', Average, Minimum, or Maximum are better
     choices since they provide information about the
     average, minimum, and maximum salary of the matching
     records. For numerical columns such as 'Quantity', the
     Sum function is more appropriate as it shows the total
     quantity for the matching records. If a column is
     categorical, such as 'Country' or 'Gender', Mode is the
      most suitable aggregation function as it identifies
     the most frequent value among the matching records.

As a database engineer, your task is to recommend the best
     aggregation function for a dataset with the following
     column information: {column_names_desc}.

For each column, provide between 0 and {max_recommendations}
      recommendations for aggregation functions.

Provide the answer in JSON format, with the column name as
     the key and a list of aggregation functions as the
     value. Order the aggregation functions by their
     suitability (from best to worst). Only provide the JSON
      dictionary without indentation or comment symbols
     (```). Do not include a long explanation.
```

Similarly, during the prediction engineering phase, prompts may be constructed from a predefined template to request recommendations from the model for time-series forecasting task configurations. In this template, {input_column_names} specifies the list of column names, while {column_stat_desc} provides the statistical metadata for each column, as described in Section 3.3.

LLM Prompt Template for Prediction Engineering

```
You are provided with a tabular dataset for time-series data
      containing the following columns: {input_column_names}

Please classify the columns from the above list into four
     different configurations for time-series forecast
     machine learning. These four configurations are:

(1) Target columns (list): columns that can be used as the
     target if you want to create a time-series forecast
     machine learning model over this dataset. In most cases
     , there is only one target column.

(2) ID columns (list): columns that distinguish different
     series.

(3) Time column (str): the name of the time column.

(4) Series columns (list): columns describing values in the
     time series. Series columns can include target columns,
      but they should not include the time column.

Provide the answer in JSON format with 4 keys:
     target_columns, id_columns, time_column, and
     series_columns.

Only provide the JSON response without indentation. Do not
     include new lines or extra whitespaces in the response.
      Do not include long explanations. If a column does not
      belong to any of these configurations, you do not need
      to include it.

To assist your prediction, column statistic metadata for
     each column is provided as follows: {column_stat_desc}
```

During the data imputation stage, a prompt can be generated using the template below and submitted to a GenAI/LLM to obtain recommendations for suitable imputation methods. The placeholder {dataset_column_stat_desc} represents the statistical metadata of the dataset and its columns, and is dynamically populated by AutoDW-TS based on the input data. Note that the brief description of each imputation method is optional in the current template, since most of these methods are well-established and already familiar to LLMs. However, if a customized or novel imputation method is included, the corresponding description should be added to the template. Additional details on the generation of this content are provided in Section 3.6.

In addition to these prompts, a system prompt can be included to provide context, such as "You are the best machine learning engineer" (or data scientist, database engineer, depending on the use case or template), to better guide the LLM's behavior. As noted in Section 4, AutoDW-TS uses GPT-4, accessed via Azure OpenAI with API version `2024-02-15-preview`.

LLM Prompt Template for Data Imputation

```
For the following columns in the time-series forecast
    dataset, missing values and column/dataset statistics
    are presented as follows:

{dataset_column_stat_desc}

Please recommend the most appropriate imputation method (
    choose from: Forward Fill, Backward Fill, Interpolation
    , Moving Average, Seasonality Imputation, or KNN
    Imputation) for each column. Descriptions of each
    imputation method are provided below:

Forward Fill: Fills missing values using the last available
    observation. Best for data where the last known value
    reasonably persists over time, e.g., stock prices
    between trades or sensor readings.
Backward Fill: Fills missing values using the next available
     observation. Suitable when future values can
    approximate missing points, e.g., scheduled events or
    planned measurements.
Interpolation: Estimates missing values between neighboring
    points (linear, spline, or polynomial). Effective when
    data changes gradually or smoothly over time, e.g.,
    temperature, sales trends.
Moving Average: Replaces missing values with the average of
    surrounding observations within a window. Useful for
    noisy data where smoothing is acceptable, e.g., sensor
    data or aggregated metrics.
Seasonality Imputation: Fills missing values based on
    recurring seasonal patterns, e.g., using the same
    period from previous cycles. Ideal for strongly
    seasonal data, e.g., electricity demand, retail sales,
    or website traffic.
KNN Imputation: Uses k-nearest neighbors in feature space to
     impute missing values based on similar observations.
    Appropriate when correlations exist between multiple
    columns, e.g., multivariate time-series datasets with
    related features.

Provide the recommended imputation method for each column.

Provide the answer in JSON format, with each column name as
    a key and the recommended imputation method as its
    value. Only provide the JSON response without
    indentation, and do not include any explanations.
```

Moreover, for data enrichment through feature transformation, as detailed in Section 3.7.1, several preprocessors leverage LLMs. For example, if a column is identified as a sentence feature, an LLM can be used by the sentence preprocessor to generate text embeddings [67] for that column. These applications of LLMs build on our previous work, AutoDW [49], but they are not the main focus of this paper.

Finally, GenAI/LLM was also utilized during the writing process to revise grammar and refine word choices, thereby enhancing the overall clarity and presentation of the paper.

References

[1] Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B Divya. 2025. Agentic AI: Autonomous intelligence for complex goals–a comprehensive survey. *IEEE Access* (2025).

[2] Ashlesha Akella, Abhijit Manatkar, Krishnasuri Narayanam, and Sameep Mehta. 2025. CodeGenWrangler: Data Wrangling task automation using Code-Generating Models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*. 949–960.

[3] Azure Kubernetes Service (AKS). accessed May 22, 2025. https://azure.microsoft.com/en-us/products/kubernetes-service

[4] Ahmad Alsharef, Karan Aggarwal, Sonia, Manoj Kumar, and Ashutosh Mishra. 2022. Review of ML and AutoML solutions to forecast time-series data. *Archives of computational methods in engineering* 29, 7 (2022), 5297–5311.

[5] Alteryx. accessed May 10, 2025. https://www.alteryx.com/

[6] Amazon Web Services. accessed May 21, 2025. Amazon SageMaker Autopilot now supports time series data. https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-autopilot-now-supports-time-series-data/

[7] Power BI. accessed May 22, 2025. https://www.microsoft.com/en-us/power-platform/products/power-bi

[8] Bradley C Boehmke. 2016. *Data wrangling with R*. Springer.

[9] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[10] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[11] Lukas Budach, Moritz Feuerpfeil, Nina Ihde, Andrea Nathansen, Nele Noack, Hendrik Patzlaff, Felix Naumann, and Hazar Harmouch. 2022. The effects of data quality on machine learning performance. *arXiv preprint arXiv:2207.14529* (2022).

[12] Li Cai and Yangyong Zhu. 2015. The challenges of data quality and data quality assessment in the big data era. *Data science journal* 14 (2015), 2–2.

[13] Chris Chatfield. 2000. *Time-series forecasting*. Chapman and Hall/CRC.

[14] Wei-Hao Chen, Weixi Tong, Amanda Case, and Tianyi Zhang. 2025. Dango: A Mixed-Initiative Data Wrangling System using Large Language Model. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–28.

[15] Yi-Wei Chen, Qingquan Song, and Xia Hu. 2021. Techniques for automated machine learning. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 35–50.

[16] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 international conference on management of data*. 2201–2206.

[17] Cohere. accessed May 10, 2025. https://cohere.com/

[18] Dataiku. 2025. Dataiku technical documentation. https://doc.dataiku.com/dss/latest/other_recipes/join.html#join-joining-datasets

[19] DataRobot. accessed May 10, 2025. https://www.datarobot.com/

[20] Guozhu Dong and Huan Liu. 2018. *Feature engineering for machine learning and data analytics*. CRC press.

[21] dotData. accessed May 10, 2025. https://dotdata.com/

[22] Florian Endel and Harald Piringer. 2015. Data Wrangling: Making data useful again. *IFAC-PapersOnLine* 48, 1 (2015), 111–112.

[23] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505* (2020).

[24] Chenguang Fang and Chen Wang. 2020. Time series data imputation: A survey on deep learning approaches. *arXiv preprint arXiv:2011.11347* (2020).

[25] Streamlit A faster way to build and share data apps. accessed May 10, 2025. https://streamlit.io/

[26] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. *Advances in neural information processing systems* 28 (2015).

[27] AutoGen: A framework for building AI agents and applications. accessed May 22, 2025. https://microsoft.github.io/autogen/stable//index.html

[28] Azure Functions. accessed May 10, 2025. https://azure.microsoft.com/en-us/products/functions

[29] Benjamin CM Fung, Ke Wang, and S Yu Philip. 2007. Anonymizing classification data for privacy preservation. *IEEE transactions on knowledge and data engineering* 19, 5 (2007), 711–725.

[30] Tim Furche, Georg Gottlob, Leonid Libkin, Giorgio Orsi, and Norman W Paton. 2016. Data wrangling for big data: Challenges and opportunities. In *19th International Conference on Extending Database Technology*. 473–478.

[31] Venkat Gudivada, Amy Apon, and Junhua Ding. 2017. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software* 10, 1 (2017), 1–20.

[32] Nitin Gupta, Shashank Mujumdar, Hima Patel, Satoshi Masuda, Naveen Panwar, Sambaran Bandyopadhyay, Sameep Mehta, Shanmukha Guttula, Shazia Afzal, Ruhi Sharma Mittal, et al. 2021. Data quality for machine learning tasks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 4040–4041.

[33] Wickham Hadley and Grolemund Garrett. 2016. R for data science: import, tidy, transform, visualize, and model data. *O'Reilly Media, Inc* (2016).

[34] Feng He, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou, and Philip S Yu. 2024. The emerged security and privacy of llm agent: A survey with case studies. *arXiv preprint arXiv:2407.19354* (2024).

[35] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622.

[36] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. 2023. Forecast evaluation for data scientists: common pitfalls and best practices. *Data Mining and Knowledge Discovery* 37, 2 (2023), 788–832.

[37] Hanwen Hu, Shiyou Qian, Dingyu Yang, Jian Cao, and Guangtao Xue. 2024. Iterative Time Series Imputation by Maintaining Dependency Consistency. *ACM Transactions on Knowledge Discovery from Data* 19, 1 (2024), 1–24.

[38] Junjie Huang, Daya Guo, Chenglong Wang, Jiazhen Gu, Shuai Lu, Jeevana Priya Inala, Cong Yan, Jianfeng Gao, Nan Duan, and Michael R Lyu. 2024. Contextualized Data-Wrangling Code Generation in Computational Notebooks. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 1282–1294.

[39] Yiming Huang, Jianwen Luo, Yan Yu, Yitong Zhang, Fangyu Lei, Yifan Wei, Shizhu He, Lifu Huang, Xiao Liu, Jun Zhao, et al. 2024. DA-Code: Agent Data Science Code Generation Benchmark for Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 13487–13521.

[40] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice*. OTexts.

[41] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. 2020. Overview and importance of data quality for machine learning tasks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3561–3562.

[42] Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. 2021. Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–36.

[43] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[44] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. 2017. Time-series extreme event forecasting with neural networks at uber. In *International conference on machine learning*, Vol. 34. sn, 1–5.

[45] Erin LeDell and Sebastien Poirier. 2020. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, Vol. 2020. ICML San Diego, CA, USA.

[46] Liyao Li, Haobo Wang, Liangyu Zha, Qingyi Huang, Sai Wu, Gang Chen, and Junbo Zhao. 2023. Learning a Data-Driven Policy Network for Pre-Training Automated Feature Engineering. In *2023 International Conference on Learning Representations (ICLR)*.

[47] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.

[48] Azure Functions Service Limit. accessed May 22, 2025. https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale#service-limits

[49] Lei Liu, So Hasegawa, Shailaja Keyur Sampat, Maria Xenochristou, Wei-Peng Chen, Takashi Kato, Taisei Kakibuchi, and Tatsuya Asai. 2024. AutoDW: Automatic Data Wrangling Leveraging Large Language Models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2041–2052.

[50] Azure Virtual Machines. accessed May 22, 2025. https://azure.microsoft.com/en-us/products/virtual-machines

[51] Ramasamy Malarvizhi and Antony Selvadoss Thanamani. 2012. K-nearest neighbor in missing data imputation. *Int. J. Eng. Res. Dev* 5, 1 (2012), 5–7.

[52] Azure API Management. accessed May 22, 2025. https://azure.microsoft.com/en-us/products/api-management

[53] Wes McKinney. 2013. *Python for data analysis*. " O'Reilly Media, Inc.".

[54] Wes McKinney et al. 2010. Data structures for statistical computing in Python.. In *SciPy*, Vol. 445. 51–56.

[55] San Murugesan. 2025. The rise of agentic AI: implications, concerns, and the path forward. *IEEE Intelligent Systems* 40, 2 (2025), 8–14.

[56] Nager.Date. accessed May 22, 2025. https://date.nager.at/

[57] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *arXiv preprint arXiv:2205.09911* (2022).

[58] A Grammar of Data Manipulation dplyr. accessed May 10, 2025. https://dplyr.tidyverse.org/

[59] OpenStreetMap. accessed May 22, 2025. https://www.openstreetmap.org

[60] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[61] pandas Python Data Analysis Library. accessed May 10, 2025. https://pandas.pydata.org/

[62] Norman Paton. 2019. Automating data preparation: Can we? should we? must we?. In *21st International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data*.

[63] Irfan Pratama, Adhistya Erna Permanasari, Igi Ardiyanto, and Rini Indrayani. 2016. A review of missing values handling methods on time-series data. In *2016 international conference on information technology systems and innovation (ICITSI)*. IEEE, 1–6.

[64] MCP: Model Context Protocol. accessed May 22, 2025. https://modelcontextprotocol.io/

[65] Foster Provost and Tom Fawcett. 2013. Data science and its relationship to big data and data-driven decision making. *Big data* 1, 1 (2013), 51–59.

[66] Tye Rattenbury, Joseph M Hellerstein, Jeffrey Heer, Sean Kandel, and Connor Carreras. 2017. *Principles of data wrangling: Practical techniques for data preparation*. " O'Reilly Media, Inc.".

[67] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[68] Ripon K Saha, Akira Ura, Sonal Mahajan, Chenguang Zhu, Linyi Li, Yang Hu, Hiroaki Yoshida, Sarfraz Khurshid, and Mukul R Prasad. 2022. SapientML: synthesizing machine learning pipelines by learning from human-writen solutions. In *Proceedings of the 44th International Conference on Software Engineering*. 1932–1944.

[69] Microsoft SQL Server. accessed May 22, 2025. https://www.microsoft.com/en-us/sql-server

[70] Vraj Shah, Jonathan Lacanlale, Premanand Kumar, Kevin Yang, and Arun Kumar. 2021. Towards benchmarking feature type inference for automl platforms. In *Proceedings of the 2021 international conference on management of data*. 1584–1596.

[71] Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423.

[72] Oleksandr Shchur, Ali Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Bernie Wang. 2023. AutoGluon–TimeSeries: AutoML for probabilistic time series forecasting. In *International Conference on Automated Machine Learning*. PMLR, 9–1.

[73] Slawek Smyl. 2020. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International journal of forecasting* 36, 1 (2020), 75–85.

[74] JSON-RPC 2.0 Specification. accessed May 10, 2025. https://www.jsonrpc.org/specification

[75] Azure Blob Storage. accessed May 22, 2025. https://azure.microsoft.com/en-us/products/storage/blobs

[76] Elham Taghizadeh. 2017. Utilizing artificial neural networks to predict demand for weather-sensitive products at retail stores. arXiv:1711.08325 [cs.LG] https://arxiv.org/abs/1711.08325

[77] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.

[78] Overcoming the 80/20 Rule in Data Science. accessed May 10, 2025. https://www.pragmaticinstitute.com/resources/articles/data/overcoming-the-80-20-rule-in-data-science/

[79] Announcing the Agent2Agent Protocol (A2A). accessed May 22, 2025. https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/

[80] John Wilder Tukey et al. 1977. *Exploratory data analysis*. Vol. 2. Springer.

[81] Jake VanderPlas. 2016. *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.".

[82] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. 2024. On protecting the data privacy of large language models (llms): A survey. *arXiv preprint arXiv:2403.05156* (2024).

[83] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* 4, 2 (2024), 100211.

[84] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2114–2124.

[85] Patrick Zippenfenig. 2023. *Open-Meteo Weather API*. doi:10.5281/zenodo.7970649

[86] Marc-André Zöller, Marius Lindauer, and Marco F Huber. 2024. Auto-sktime: Automated Time Series Forecasting. In *International Conference on Learning and Intelligent Optimization*. Springer, 456–471.