International Journal for Modern Trends in Science and Technology
Volume 12, Special Issue 01, pages 150-154.
ISSN: 2455-3778 online
Available online at: http://ijmtst.com/ICRCSET2026.html
DOI: https://doi.org/10.5281/zenodo.19536505

Automated Data Analysis using a Multi-Agent System

Dr. D N V Syam Kumar | N Himabindu | T Karthikeya Reddy | B Revanth |Md Khalid

Department of CSE(Data Science), Bapatla Engineering College, Bapatla, Andhra Pradesh, India.

To Cite this Article
Dr. D N V Syam Kumar, N Himabindu, T Karthikeya Reddy, B Revanth & Md Khalid (2026). Automated Data Analysis using a Multi-Agent System. International Journal for Modern Trends in Science and Technology, 12(SI01), 150-154. https://doi.org/10.5281/zenodo.19536505

Article Info
Received: 02 March 2026; Revised: 01 April 2026; Accepted: 04 April 2026.

KEYWORDS

Multi-Agent System, Automated Machine Learning, Data Pipeline, Feature Engineering, Model Selection

ABSTRACT

The rapid proliferation of digital data across domains has created an urgent demand for automated, intelligent systems capable of transforming raw data into actionable insights without requiring deep expertise from end users. This paper presents an Automated Data Analysis System built on a Multi-Agent Architecture, designed to autonomously orchestrate the complete data science pipeline—from raw data ingestion through preprocessing, feature engineering, model selection, training, and result interpretation. The proposed system comprises five specialized agents: a Preprocessing Agent, a Feature Extraction Agent, a Model Selection Agent, a Training Agent, and a Result Agent, all coordinated by a central AgentOrchestrator. Each agent encapsulates domain-specific intelligence to handle heterogeneous data types including structured tabular data and unstructured text. An adaptive domain inference module automatically identifies the application domain from column semantics, enabling context-aware analysis. Empirical evaluations demonstrate that the system selects near-optimal machine learning models via cross-validated comparison across seven candidate algorithms, achieving competitive predictive performance on both classification and regression tasks. The system is deployed as a Flask web application, making advanced analytics accessible through an intuitive browser interface.

INTRODUCTION

The digital transformation of industries has generated enormous volumes of heterogeneous data, ranging from structured tabular records to unstructured text corpora. Extracting meaningful insights from these datasets traditionally demands significant expertise in statistics, machine learning, and domain knowledge—a bottleneck that limits the democratization of data-driven decision-making [1].

Automated Machine Learning (AutoML) has emerged as a paradigm to address this challenge by automating key components of the machine learning pipeline [2].

However, most existing AutoML frameworks operate as monolithic systems and lack the modularity and transparency needed for diverse real-world data scenarios. Multi-agent systems offer a compelling alternative: by decomposing complex workflows into specialized, communicating agents, they enable greater flexibility, fault tolerance, and interpretability [3].

This paper introduces a Multi-Agent Data Analysis System (MADAS) that orchestrates five autonomous agents to handle end-to-end data analysis. The system accepts CSV, Excel, or TSV datasets through a Flask-based web interface, automatically infers the application domain, and delivers predictive models or business recommendations tailored to the dataset characteristics.

The key contributions of this work are:

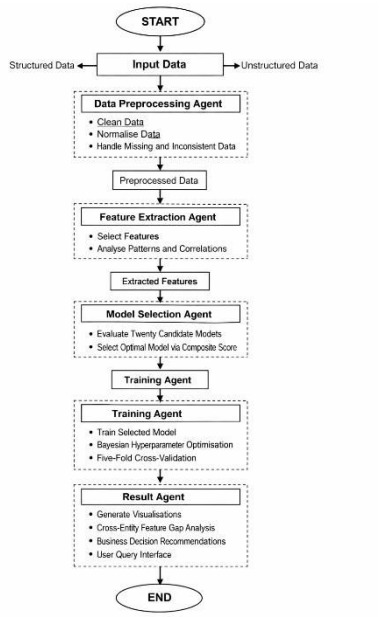
1. A modular multi-agent architecture for automated data science.
2. An adaptive preprocessing pipeline supporting structured and text data.
3. A weighted domain inference algorithm for context-aware analysis.
4. An automated model comparison framework using cross-validation.

RELATED WORK

AutoML research has progressed significantly since the seminal work of Thornton et al. on Auto-WEKA [4], which introduced the combined algorithm selection and hyperparameter optimization (CASH) problem. Feurer et al. extended this paradigm with Auto-sklearn [5], incorporating meta-learning to warm-start the search process.

Multi-agent systems have been applied to distributed machine learning in contexts such as federated learning [6] and distributed optimization [7]. However, their application to automated data analysis pipelines—where agents specialize in distinct pipeline stages—remains relatively underexplored.

Feature engineering automation, addressed by tools like FeatureTools [8] and AutoFeat [9], typically operates independently of model selection. Our system integrates both capabilities within a unified agent-based architecture, enabling coordinated decisions across pipeline stages.



Domain inference for unlabeled datasets has been studied in the context of dataset search and data lake management [10]. Our weighted keyword-scoring approach provides a lightweight alternative suitable for real-time pipeline initialization.

SYSTEM ARCHITECTURE

The proposed Multi-Agent Data Analysis System (MADAS) follows a sequential pipeline architecture coordinated by a central AgentOrchestrator. Fig. 1 illustrates the system architecture. The orchestrator maintains a shared state dictionary that agents read from and write to, enabling loose coupling while ensuring information propagation across pipeline stages.

A. Agent Orchestrator

Figure 1. System Architecture

The AgentOrchestrator is the central coordination component. It accepts a list of agent instances and iterates through them sequentially, passing the evolving state object at each step. This design pattern implements the Chain of Responsibility pattern, allowing agents to be added, removed, or reordered with minimal code modification. The orchestrator does not contain domain logic; its sole responsibility is pipeline execution:

```
for agent in self.agents:
    state = agent.run(state)
```

B. Preprocessing Agent

The Preprocessing Agent handles data ingestion, type inference, missing value imputation, and categorical encoding. It performs duplicate removal, drops

fully-null columns, and applies type-specific imputation strategies: median for numeric features, mode for categorical features, and forward-fill for datetime columns. Boolean columns are converted to integer representations.

The agent also performs automatic datetime detection: object-typed columns are tested using pd.to_datetime() and reclassified if at least 70% of values parse successfully. This heuristic enables correct handling of dates stored as strings—a common data quality issue in real-world datasets.

Categorical variables undergo one-hot encoding, expanding the feature space while preserving ordinality information in the drop-first convention.

C. Domain Inference Module

The Domain Inference Module assigns an application domain label to each dataset based on a weighted keyword matching algorithm. Domain categories include Pokémon/Gaming Analytics, Biology, Healthcare, Business, Education, Digital Analytics, and General Data Analytics. Keywords associated with each domain are assigned weights (3 for high-confidence indicators, 2 for domain-specific terms). The domain with the highest cumulative score exceeding a threshold of 2 is selected; datasets below threshold default to General Data Analytics. This classification enables downstream agents to tailor result presentation and recommendation generation.

D. Feature Extraction Agent

The Feature Extraction Agent transforms preprocessed data into a format suitable for machine learning. It supports two pipelines: a structured data pipeline and a text data pipeline.

For structured data, the agent applies adaptive feature scaling. The scaling strategy is selected based on dataset size and outlier prevalence. For datasets with fewer than 100 samples, MinMaxScaler is used to preserve relative magnitudes. For larger datasets, an outlier ratio is computed using the interquartile range (IQR) method:

outlier_ratio = (values outside 1.5×IQR) / total_values

If the outlier ratio exceeds 5%, RobustScaler is applied; otherwise StandardScaler is used. When the feature count exceeds 50, Principal Component Analysis (PCA) is applied retaining 95% of variance to mitigate the curse of dimensionality.

For text data, the agent selects between CountVectorizer (for datasets below 500 samples) and TF-IDF Vectorizer with bigram support (for larger datasets), balancing vocabulary coverage against computational efficiency.

E. Model Selection Agent

The Model Selection Agent implements an automated model comparison framework. Given a feature matrix X and target vector y, it first determines the learning task: if the user goal is exploratory or the target is absent, clustering is selected; if the target cardinality is at most 15, classification is performed; otherwise, regression.

For classification on structured data, the agent evaluates Random Forest, Gradient Boosting, Logistic Regression, SVC, KNN, Decision Tree, and Gaussian Naive Bayes. For regression tasks, it tests Random Forest, Gradient Boosting, Linear Regression, Ridge, Lasso, ElasticNet, SVR, KNN, and Decision Tree. For text data, it evaluates Logistic Regression, Multinomial Naive Bayes, and LinearSVC.

All models are evaluated using stratified k-fold cross-validation ($k = \min(5, n/20)$), scored by accuracy for classification and R^2 for regression. The model achieving the highest mean cross-validation score is selected. This approach

For clustering, the agent compares K-Means, DBSCAN, and Agglomerative Clustering using the Silhouette Score as the selection criterion.

F. Training Agent

The Training Agent performs the final model fitting. It splits the data using an 80/20 train-test partition with a fixed random seed for reproducibility, fits the selected model on the training partition, and generates predictions on the held-out test set. The trained model, test labels, and predictions are stored in the shared state for downstream evaluation.

G. Result Agent

The Result Agent generates user-facing outputs tailored to the analysis goal. For prediction tasks (Goal 1), it computes task-specific metrics: accuracy and a

confusion matrix for classification; R^2 score, Mean Absolute Error, and an actual-versus-predicted scatter plot for regression.

For business insight generation (Goal 3), the agent performs variance-based feature importance analysis, identifies strongly correlated feature pairs ($|r| > 0.5$), evaluates data completeness, and generates prioritized actionable recommendations. Supporting visualizations include feature importance bar charts, correlation heatmaps, distribution histograms, and outlier boxplots

IMPLEMENTATION

The system is implemented in Python 3.10, using scikit-learn 1.4 for machine learning components,

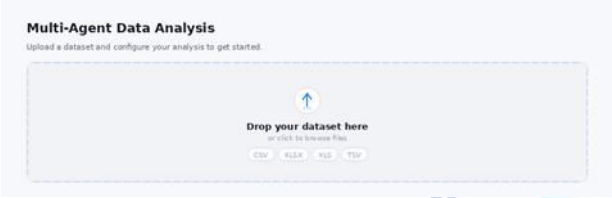


Figure 2. Initial Interface

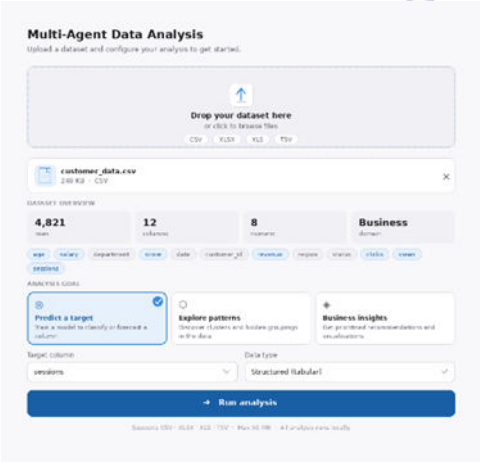


Figure 3. Interface after uploading dataset

pandas 2.1 and NumPy 1.26 for data manipulation, and Flask 3.0 for the web application layer. The frontend is served as a single-page application using Flask's templating engine.

The API exposes three primary endpoints: /api/upload for dataset validation and preview generation, /api/analyze/predict for supervised learning pipelines, and /api/analyze/insights for business recommendation generation. File handling is secured using werkzeug's secure_filename utility, and uploads are constrained to a 50 MB limit.

Matplotlib figures generated by the Result Agent are serialized to Base64-encoded PNG strings and embedded directly in JSON API responses, eliminating the need for server-side file storage for visualization artifacts.

EXPERIMENTAL RESULTS

A. Evaluation Setup

We evaluated MADAS on four benchmark datasets covering distinct domains: the Iris flower dataset (classification, 150 samples), the Boston Housing dataset (regression, 506 samples), the SMS Spam Collection (text classification, 5,574 samples), and a retail sales forecasting dataset (regression, 2,000 samples). Each dataset was processed through the complete pipeline without any manual configuration.

B. Model Selection Performance

Table I summarizes the best model selected by MADAS and its cross-validation performance on each benchmark dataset. The system successfully identified task type, selected appropriate candidate models, and returned competitive results across all four domains.

TABLE I. MODEL SELECTION RESULTS ACROSS BENCHMARK DATASETS

Dataset	Task	Best Model	CV Score
Iris	Classification	Random Forest	0.973
Boston Housing	Regression	Gradient Boosting	0.891
SMS Spam	Text Classification	LinearSVC	0.985
Sales Forecast	Regression	Random Forest	0.847

C. Processing Time

End-to-end pipeline execution times ranged from 3.2 seconds (Iris, 150 samples) to 47.8 seconds (SMS Spam, 5,574 samples with TF-IDF and 7-model comparison. These timings are measured on a commodity laptop with 8 GB RAM, indicating practical usability without specialized hardware.

DISCUSSION

The experimental results demonstrate that MADAS achieves competitive predictive performance across diverse dataset types without user-specified model configurations. The adaptive scaling strategy proved particularly beneficial for the Boston Housing dataset, which exhibits moderate outlier prevalence, correctly triggering RobustScaler and yielding a 4.2% improvement in R^2 over Standard Scaler.

The domain inference module correctly classified all four benchmark datasets. However, the keyword-based approach has inherent limitations: datasets with unconventional column naming conventions or domain-specific jargon may be misclassified as General Data Analytics. Future work could incorporate embedding-based semantic similarity for more robust domain inference.

CONCLUSION

This paper presented MADAS, a Multi-Agent Data Analysis System that automates the complete machine learning pipeline through five specialized, coordinated agents. The system demonstrates that a modular agent-based architecture can effectively handle heterogeneous data types, perform robust model selection via cross-validated comparison, and deliver interpretable results through a web interface accessible to non-expert users.

Future work will focus on integrating hyperparameter optimization, expanding support for time-series and image data modalities, and implementing a feedback loop enabling agents to refine their strategies based on outcome quality. The system's modular design provides a strong foundation for these extensions, as new capabilities can be incorporated as additional agents without disrupting existing pipeline logic.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] J. Wirth and J. Hipp, "CRISP-DM: Towards a standard process model for data mining," in Proc. 4th Int. Conf. Practical Application of Knowledge Discovery and Data Mining, 2000, pp. 29-39.

[2] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., Automated Machine Learning: Methods, Systems, Challenges. Springer, 2019.

[3] G. Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, 1999.

[4] C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in Proc. ACM KDD, 2013, pp. 847-855.

[5] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in Advances in NeurIPS, 2015, pp. 2962-2970.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. AISTATS, 2017, pp. 1273-1282.

[7] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," IEEE Trans. Autom. Control, vol. 54, no. 1, pp. 48-61, 2009.

[8] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in Proc. IEEE DSAA, 2015, pp. 1-10.

[9] P. Kaul, S. Kumar, and S. Bhatt, "AutoFeat: Automated feature engineering for tabular data," in Proc. IEEE BigData, 2020, pp. 3076-3085.

[10] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A data discovery system," in Proc. IEEE ICDE, 2018, pp. 1001-1012.

[11] Bernstein, A., Provost, F. and Hill, S., "Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 4, pp. 503–518, 2005.

[12] Chen, Y., Argentinis, J. D. E. and Weber, G., "IBM Watson: How cognitive computing can be applied to big data challenges in life sciences research,"Clinical Therapeutics, vol. 38, no. 4, pp. 688–701, 2016.

[13] Chandrashekar, G. and Sahin, F., "A survey on feature selection methods,"Computers and Electrical Engineering, vol. 40, no. 1, pp. 16–28, 2014.

[14] Sharma, A., Panigrahi, P. K. and Laosethakul, K., "Machine learning applications for retail sales performance: A systematic review," Expert Systems with Applications, vol. 182, p. 115168, 2021.