Agentic ML Builder

AAI-590 Capstone Project

Project Advisor Prof. Zahid Wani

Group 04

Mahesh Arcot Krishnamurthy | Ranjeet Das

10th November 2025

Agentic ML Builder – Dataset and Exploratory Data Analysis (EDA)

Overview

The **Agentic ML Builder MVP** uses **three key datasets**, each supporting a different ML capability within the system:

ML Capability	Dataset Name	Purpose	Source
Meta-Learning Model	OpenML Meta-Benchmark Dataset (e.g., B50/B100)	Learn relationships between dataset characteristics and best-performing ML algorithms	OpenML.org
Natural Language Understanding (NLU) Model	Synthetic Instruction-to-Intent Dataset	Map user's natural language project description → structured ML task type	Custom (generated via GPT-4 + labeled manually)
RAG Template Retrieval Base	ML Template Corpus (vectorized)	Store and retrieve code templates for ML tasks (classification, regression, NLP, vision)	Internal curated repository

---

Dataset 1: Meta-Learning Dataset (OpenML)

1. **Variables and Datatypes**

Variable	Description	Datatype
dataset_name	Unique name of dataset	String
num_instances	Number of rows/samples	Integer
num_features	Number of columns/features	Integer
num_numeric_features	Number of numeric columns	Integer
num_categorical_features	Number of categorical columns	Integer
missing_value_ratio	Percentage of missing data	Float
task_type	Classification, regression, clustering	Categorical

Variable	Description	Datatype
algorithm	Model tested (e.g., RandomForest, SVM)	Categorical
accuracy	Accuracy score achieved by algorithm	Float
f1_score	F1 metric (if classification)	Float
Rmse	Root Mean Squared Error (if regression)	Float

---

2. **Data Issues and Handling**

Issue	Source	Handling
Missing performance metrics (some datasets lack RMSE or F1)	Incomplete benchmarking on OpenML	Used SimpleImputer to fill missing metrics (mean substitution).
Imbalanced representation (too many classification vs regression)	Unequal benchmark coverage	Applied RandomUnderSampler to balance tasks.
Inconsistent encoding for algorithms (e.g., RandomForestClassifier vs rf)	Varying dataset sources	Normalized algorithm names via regex and mapping dictionary.
Outliers in dataset size (huge datasets skew training)	Large-scale benchmarks (e.g., CIFAR, ImageNet)	Clipped num_instances at 99th percentile for stability.

Best guess for issues:
These inconsistencies come from heterogeneous OpenML contributors who upload datasets with differing feature and metric availability.

---

3. **Relationship to Project Goal**

The **Meta-Learning Model** aims to predict the *best algorithm* given dataset metadata.

✅ Observed patterns:

- **High correlation** between num_features and algorithm type — tree-based models (Random Forest, XGBoost) perform better on high-dimensional data.

- **Task type** strongly predicts performance metric (F1 vs RMSE).
- **Missing value ratio** negatively correlates with accuracy — suggests preprocessing importance.

These relationships confirm that dataset metadata can be used to infer suitable algorithms, directly supporting the **Agentic ML Builder's AutoML intelligence**.

---

4. Transformations / Feature Engineering

New features were derived:

New Variable	Description	Formula
feature_density	Features per 1,000 instances	num_features / num_instances
data_complexity_score	Composite metric from missing values + categorical ratio	(missing_value_ratio * 0.4) + (num_categorical_features / num_features * 0.6)
performance_rank	Rank of algorithm performance per dataset	rank(-accuracy) or rank(rmse)

➞ These derived features improved the model's ability to generalize across different datasets.

---

5. Variable Relationships and Correlations

Pearson Correlation Heatmap Summary:

Variables	Correlation	Observation
num_features ↔ num_numeric_features	+0.94	Strong correlation — redundant, one dropped.
missing_value_ratio ↔ accuracy	-0.62	High missingness reduces performance.
feature_density ↔ task_type	0.73 (classification bias)	Classification tasks tend to have higher feature density.

➞ Implication: remove highly correlated features before model training (to avoid multicollinearity).

Dataset 2: NLU Dataset (Synthetic Instruction Dataset)

1. Variables and Datatypes

Variable	Description	Datatype
user_query	User's project description	String
problem_type	Classification / Regression / Clustering / NLP / Vision	Categorical
Domain	Tabular, Text, Image, Audio	Categorical
target_variable	Target column (if mentioned)	String / Null
algorithm_preference	Preferred algorithm (if mentioned)	String / Null
intent_confidence	Confidence score of prediction	Float

---

2. Data Issues and Handling

Issue	Source	Handling
Inconsistent phrasing (e.g., "predict" vs "forecast")	Natural user diversity	Normalized via text preprocessing and synonym expansion.
Missing labels for some synthetic queries	GPT-4 generation noise	Manually labeled missing intents.
Class imbalance (more classification tasks)	Natural bias in user prompts	Oversampled underrepresented classes via SMOTE-NC.

Best guess for issues:
Generated prompts reflect real-world overemphasis on classification-type problems.

---

3. Relationship to Project Goal

The dataset directly trains the **NLU model** to interpret user requests.

✅ Observed patterns:

- "Predict", "classify", "detect" → **Classification**
- "Estimate", "forecast", "predict numeric" → **Regression**
- "Group", "cluster" → **Unsupervised Learning**
- Presence of "text", "review", "document" → **NLP**
- Presence of "image", "detect object" → **Vision**

These linguistic patterns guide the model to extract structured configuration parameters automatically.

---

4. **Transformations / Feature Engineering**

- **Tokenization:** Used BERT tokenizer (WordPiece) for contextual embedding.
- **Feature extraction:** Embedded sentences using sentence-transformers before classification.
- **Synonym augmentation:** Added paraphrases for "detect", "forecast", etc., to improve robustness.

New variable: keyword_density = count of ML-related terms / total tokens.
Higher keyword density improved classification confidence.

---

5. **Variable Correlations**

- **keyword_density ↔ intent_confidence**: +0.71 (high correlation)
- **problem_type ↔ domain**: strong categorical dependency (e.g., NLP ↔ Text)
- **Multicollinearity check**: dropped target_variable for training (too sparse).

Implication: Model trained only on user_query, keyword_density, and domain features for stability.

Dataset 3: RAG Template Knowledge Base

1. **Variables and Datatypes**

Variable	Description	Datatype
template_id	Unique template identifier	String
task_type	Classification, Regression, NLP, Vision	Categorical
framework	PyTorch, TensorFlow, scikit-learn	Categorical
code_text	Template code content	Text
embedding_vector	Semantic embedding vector (1536-d)	Array[Float]
source_repo	GitHub / Azure ML Examples	String

---

2. **Data Issues and Handling**

Issue	Handling
Duplicate templates (minor variations)	Deduplicated using cosine similarity > 0.98
Missing framework tags	Inferred from import statements via regex
Code formatting inconsistencies	Auto-formatted using black formatter

---

3. **Relationship to Project Goal**

The RAG corpus provides reusable code blocks for generation.

- Templates tagged correctly with task_type correlate with **retrieval accuracy** (↑ 92% match).
- Embedding similarity thresholds tuned (optimal at 0.85 cosine similarity).

---

4. **Transformations / Feature Engineering**

- Generated embeddings using **Azure OpenAI text-embedding-3-large**.
- Added token_count feature = number of code tokens (to control retrieval bias).

---

5. **Variable Relationships**

Feature Pair	Correlation	Note
task_type ↔ framework	0.84	PyTorch templates dominate Vision tasks.
token_count ↔ embedding_norm	0.64	Longer templates have stronger embedding norms.

⚙️ **Summary Insights**

Question	Answer
1. What variables are present?	Variables differ per dataset — include metadata (numerical, categorical), language (text), and embeddings (vectors).
2. What issues were present and how were they handled?	Missing metrics, class imbalance, inconsistent encoding; handled with imputation, normalization, resampling.
3. How are variables related to project goal?	All are predictive of project task type or algorithm recommendation; clear correlations exist (e.g., task_type ↔ algorithm).
4. Were new variables created?	Yes — feature_density, data_complexity_score, keyword_density, and token_count improved predictive strength.
5. How are variables related to each other?	Moderate to strong correlations (multicollinearity handled via feature selection). These relationships helped in designing robust meta-learning and NLU models.

In conclusion:
Each dataset component directly enables a part of the **Agentic ML Builder's cognitive pipeline**:

- **NLU dataset → interprets user intent**
- **Meta-learning dataset → recommends algorithms**
- **RAG dataset → retrieves code templates**

Their EDA reveals strong internal consistency and useful variable relationships, ensuring the system can reason effectively from human input to working ML code scaffolds.

For more details refer to the PDF documents of the notebooks related to EDA on these three dataset both REAL and Synthetic dataset available in the repository

Google Drive:

https://drive.google.com/drive/folders/1CRTuBbuSVszhV0LePbwVppTkW5GWKcWY?usp=drive_link

Reports

1. AgenticMLBuilder_EDA.pdf
2. meta_learning_eda.ipynb and meta_learning_eda.pdf
3. nlu_rag_openml_eda.ipynb and nlu_rag_openml.pdf
4. nlu_rag_eda_synth.ipynb and nlu_rag_eda_synth.pdf

GitHub:

https://github.com/maheshbabu-usd/aai-590-group04-agentic-ml-builder

Reports

5. AgenticMLBuilder_EDA.pdf
6. meta_learning_eda.ipynb and meta_learning_eda.pdf
7. nlu_rag_openml_eda.ipynb and nlu_rag_openml.pdf
8. nlu_rag_eda_synth.ipynb and nlu_rag_eda_synth.pdf

References:

[1] Shi, Y., Wang, M., Cao, Y., Lai, H., Lan, J., Han, X., Wang, Y., Geng, J., Li, Z., Xia, Z., Chen, X., Li, C., Xu, J., Duan, W., & Zhu, Y. (2025). *Aime: Towards fully-autonomous multi-agent framework* (arXiv:2507.11988). arXiv. https://doi.org/10.48550/arXiv.2507.11988

[2] Higuchi, T., Henry, S., & Straight, E. (2025, October 1). *Introducing Microsoft Agent Framework: The open-source engine for agentic AI apps.* Azure AI Foundry Blog. https://devblogs.microsoft.com/foundry/introducing-microsoft-agent-framework-the-open-source-engine-for-agentic-ai-apps/

[3] Microsoft. (2025). agent-framework: A framework for building, orchestrating and deploying AI agents and multi-agent workflows [Computer software]. GitHub. https://github.com/microsoft/agent-framework

[4] Ashrafi, N., Bouktif, S., & Mediani, M. (2025). *Enhancing LLM code generation: A systematic evaluation of multi-agent collaboration and runtime debugging for improved accuracy, reliability, and latency* (arXiv preprint arXiv:2505.02133 v1). arXiv. https://doi.org/10.48550/arXiv.2505.02133

[5] Eken, B., Pallewatta, S., Tran, N. K., Tosun, A., & Babar, M. A. (2024). *A multivocal review of MLOps practices, challenges and open issues* (arXiv preprint arXiv:2406.09737v2). https://arxiv.org/abs/2406.09737

[6] OpenAI. (2024). *A practical guide to building agents* [PDF]. https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf