AUTOM^{3}L: AUTOMATED MULTIMODAL MACHINE LEARNING WITH LARGE LANGUAGE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Automated Machine Learning (AutoML) stands as a promising solution for automating machine learning (ML) training pipelines to reduce manual costs. However, most current AutoML frameworks are confined to unimodal scenarios and exhibit limitations when extended to challenging and complex multimodal settings. Recent advances show that large language models (LLMs) have exceptional abilities in reasoning, interaction, and code generation, which shows promise in automating the ML pipelines. Innovatively, we propose AutoM^{3}L, an Automated Multimodal Machine Learning framework, where LLMs act as controllers to automate training pipeline assembling. Specifically, AutoM^{3}L offers automation and interactivity by first comprehending data modalities and then automatically selecting appropriate models to construct training pipelines in alignment with user requirements. Furthermore, it streamlines user engagement and removes the need for intensive manual feature engineering and hyperparameter optimization. At each stage, users can customize the pipelines through directives, which are the capabilities lacking in previous rule-based AutoML approaches. We conduct quantitative evaluations on four multimodal datasets spanning classification, regression, and retrieval, which yields that AutoM^{3}L can achieve competitive or even better performance than traditional rule-based AutoML methods. We show the user-friendliness and usability of AutoM^{3}L in the user study. Code is available at: https://anonymous.4open.science/r/anonymization_code

1 INTRODUCTION

Multimodal data holds paramount significance in machine learning tasks, offering the capability to harness richer contextual insights. Yet, the inherent diversity of such modalities introduces complexities, particularly in selecting ideal model architectures and ensuring seamless synchronization of features across these modalities, resulting in a reliance on intensive manual involvement. Aspiring to diminish manual hand-holding in the ML pipeline, Automated Machine Learning (AutoML) has emerged (Hutter et al., 2019; Gijsbers et al., 2019; Vakhrushev et al., 2021; Weerts et al., 2020; Wang et al., 2021; Elshawi et al., 2019). However, a gaping void persists as the lion's share of AutoML solutions remains tailored predominantly for uni-modal data. AutoGluon[1] made the first attempt at multimodal AutoML but is beset with shortcomings. Firstly, it falls short of fully automated feature engineering, essential for adeptly managing multimodal data. Moreover, it imposes a pronounced learning curve to get familiar with its configurations and settings. This complexity contradicts the user-friendly automation ethos that AutoML initially epitomizes. Besides, its adaptability, constrained by preset settings like search space, model selection, and hyper-parameters, leaves much to be desired manually. Furthermore, expanding AutoGluon's capabilities by integrating new techniques or models often necessitates intricate manual code modifications, thus hampering its agility and potential for growth.

The scientific realm has been abuzz with the meteoric rise of large language models (LLMs), particularly due to their transformative potential in task automation (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023; Wei et al., 2022). Evolving beyond their foundational guise as text generators, LLMs have metamorphosed into autonomous powerhouses, adept at self-initiated planning and execution (Shen et al., 2023; Wang et al., 2023; Wu et al., 2023; Hong et al., 2023;

[1] https://github.com/autogluon/autogluon

Yao et al., 2022). Such an evolution presents a tantalizing prospect, namely the opportunity to significantly bolster the performance and adaptability of multimodal AutoML systems. Capitalizing on this potential, we introduce AutoM^{3}L, an innovative LLM framework for Automated Multimodal Machine Learning. Distinct from platforms like AutoGluon, which are tethered to fixed, predetermined pipelines, AutoM^{3}L stands out with its dynamic user interactivity. Specifically, it seamlessly weaves ML pipelines, tailoring them to user directives, achieving unparalleled scalability and adaptability from data pre-processing to model selection and optimization.

The major contributions are four-fold, summarized as follows. (1) We introduce a novel LLM framework, namely AutoM^{3}L which aims to automate the ML pipeline development for multimodal data. It enables users to derive accurate models for each modality from a large pool of models along with a self-generated executable script for cross-modality feature fusion using minimal natural language instructions. (2) We further spearhead the automation of feature engineering. Concretely, we leverage an LLM to filter out attributes that might hamper model performance and concurrently impute missing data. (3) Finally, we automate hyperparameter optimization with LLM via self-suggestions combined with the integration of external API calls. This can decisively negate the need for labor-intensive manual explorations. (4) We embark on comprehensive evaluations, comparing with conventional rule-based multimodal AutoML on a myriad of multimodal datasets. Moreover, user studies further underscored the distinct advantages of AutoM^{3}L in terms of its user-friendliness and a significantly diminished learning curve.

2 RELATED WORKS

AutoML. AutoML has emerged as a transformative paradigm to streamline the design, training, and optimization of ML models by minimizing the need for extensive human intervention. Current AutoML solutions predominantly fall into three categories: (i) training pipeline automation, (ii) automated feature engineering, (iii) hyperparameter optimization. Within the sphere of automated feature engineering, certain methodologies have carved a niche for themselves. For instance, DSM (Kanter & Veeramachaneni, 2015) and OneBM (Lam et al., 2017) have revolutionized feature discovery by seamlessly integrating with databases, curating an exhaustive set of features. In a complementary vein, AutoLearn (Kaul et al., 2017) adopts a regression-centric strategy, enhancing individual records by predicting and appending additional feature values. Concurrently, training pipeline and hyperparameter optimization automation have also seen significant advancements. For example, H2O AutoML (LeDell & Poirier, 2020) is particularly noteworthy for its proficiency in rapidly navigating an expansive pipeline search space, leveraging its dual-stacked ensemble models. However, a recurring challenge across these AutoML solutions is their predominant focus on uni-modal data, which limits their applicability to more complex multimodal data. Recognizing this gap, we introduce a novel LLM framework tailored specifically for multimodal AutoML scenarios.

Large Language Models. The domain of Natural Language Processing has undergone a paradigm shift with the introduction of LLMs (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023; Wei et al., 2022; Chung et al., 2022). With their staggering parameter counts reaching into the hundreds of billions, LLMs have showcased unparalleled versatility across diverse tasks. A testament to their evolving capabilities is Toolformer (Schick et al., 2023), which equips LLMs to interact with external utilities via API calls, thereby expanding their functional horizons. AutoGPT further exemplifies this evolution, segmenting broad objectives into tangible sub-goals, subsequently executed through prevalent tool APIs, such as search engines or code executors. Yet, as we embrace the potential of LLMs to manage AI tasks via API interactions, it's crucial to navigate the inherent intricacies. Model APIs, in particular, often require bespoke implementations, frequently involving pre-training phases which highlights the pivotal role of AutoML in refining and optimizing these intricate workflows. Our proposed AutoML framework aspires to bridge this gap, enabling fluid user-AI engagements through lucid dialogues and proficient code generation.

3 METHODS

We elaborate on the details of the five functional components in **Auto**mated **M**ulti-**M**odal **M**achine **L**earning (AutoM^{3}L): (1) modality inference, (2) automated feature engineering, (3) model selection, (4) pipeline assembly, and (5) hyperparameter optimization, as illustrated in Fig. 1.

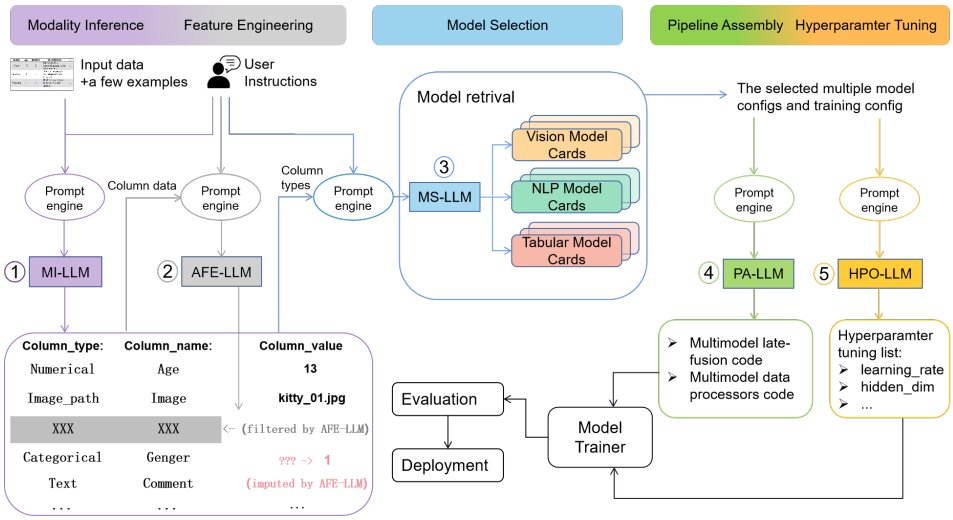

Figure 1: The overall framework of AutoM^{3}L. It consists of five stages: ① Infer the modality of each attribute in structured table data. ② Automate feature engineering for feature filtering and data imputation. ③ Select optimal models for each modality. ④ Generates executable scripts for model fusion and data processing to assemble the training pipeline. ⑤ Search optimal hyperparameters.

Organization of Multimodal Dataset. The multifaceted nature of multimodal data allows it to be represented in various formats, among which the JavaScript Object Notation (JSON) stands out as a prevalent choice. In this work, however, we prioritize structured tables for their distinct advantages. Not only do they offer a clear representation, capturing the interplay between different modalities, but they also adeptly aggregate information from varied formats into a unified structure. Contained within these tables is a diverse range of data modalities including images, text, and tabular data. For a comprehensive understanding of structured tabular, we direct readers to Appendix B.

Modality Inference Module. AutoM^{3}L begins with **M**odality **I**nference-LLM (MI-LLM) to identify the modality associated with each column in the structured table. Simplifying its operation and avoiding extra training costs, MI-LLM taps into in-context learning. Correspondingly, the guiding prompt to MI-LLM is tripartite, as showcased in Fig. 2(a): (1) An ensemble of curated examples is used for in-context learning. This ensemble assists MI-LLM in generating desired format responses and firmly establishing correlations between column names and their modalities. (2) A subset of the input structured table is included, containing randomly sampled data items paired with their respective column names. The semantic richness of this subset serves as a guiding light, steering the MI-LLM towards accurate modality identification. (3) User-specified directives do more than just instruct; they enrich the process with deeper context. Capitalizing on the LLM's exceptional interactivity, these directives refine the modality inference further. For example, a directive like "*this dataset delves into the diverse factors influencing animal adoption rates*" grants MI-LLM a contextual perspective, facilitating a more astute interpretation of column descriptors.

Automated Feature Engineering Module. Feature engineering shines as a crucial preprocessing phase, dedicated to tackling common data challenges, such as missing values. While many conventional AutoML solutions heavily depend on rule-based feature engineering, our AutoM^{3}L framework embraces the unmatched capabilities of LLMs to elevate this process. Specifically, we introduce the **A**utomatic **F**eature **E**ngineering-LLM (AFE-LLM), as depicted in Fig. 2(b). This module employs two distinct prompts, resulting in two core components: AFE-LLM$_{\text{filter}}$ and AFE-LLM$_{\text{imputed}}$. The former, AFE-LLM$_{\text{filter}}$, is adept at sifting through the data to eliminate irrelevant or superfluous attributes. On the other hand, AFE-LLM$_{\text{imputed}}$ is dedicated to data imputation, ensuring the completeness and reliability of vital data. Importantly, these components work in tandem, where after AFE-LLM$_{\text{filter}}$ refines the features, AFE-LLM$_{\text{imputed}}$ steps in to address any data gaps in the streamlined dataset. To enhance feature filtering, AFE-LLM$_{\text{filter}}$'s prompt integrates: (1) An ensemble of examples for in-context learning. More specifically, by strategically introducing attributes

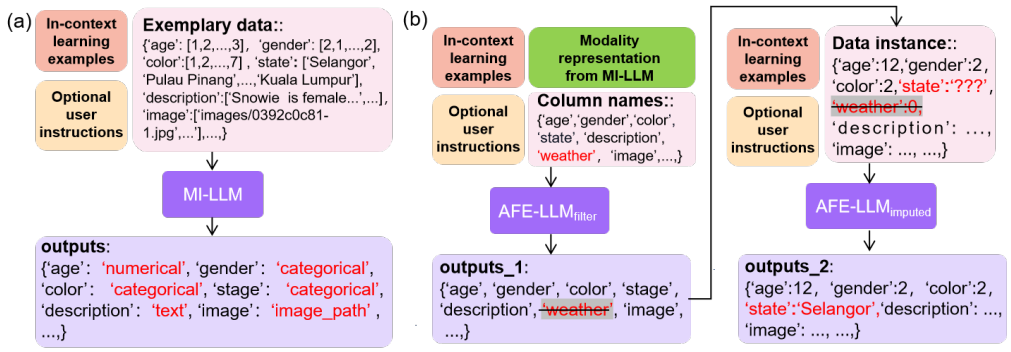

Figure 2: (a) Modality Inference with MI-LLM. It displays MI-LLM's capability to discern the modality of each column in a dataset. Attributes are highlighted with red annotations to signify the inferred modality. (b) Data Refinement with AFE-LLM. It emphasizes AFE-LLM's dual role in both feature filtering and data imputation. In the left part, attributes marked in red denote those that are filtered out, while on the right, red annotations identify attributes that have been subjected to imputation.

from various datasets and interlacing them with intentionally irrelevant ones, the AFE-LLM$_{\text{filter}}$ is oriented towards distinguishing and removing non-essential attributes. (2) Column names in the structured table, brimming with semantic information about each feature component, augment the LLM's ability to distinguish between pivotal and disposable attributes. (3) Modality inference results derived from MI-LLM, guiding the LLM to shed attributes of limited informational significance. For instance, when juxtaposing a continuous attribute like age with a binarized attribute indicating if someone is over 50, the latter, being somewhat redundant, can be identified for removal. (4) When available, user-defined directives or task descriptions can be embedded which aims to forge a connection between pertinent column names and the overarching task. Regarding data imputation, AFE-LLM$_{\text{imputed}}$ exploits its profound inferential prowess to seamlessly detect and fill data voids. The prompt for this facet encompasses: (1) Data points characterized by value omissions, enabling AFE-LLM$_{\text{imputed}}$ to fill these gaps by discerning patterns and inter-attribute relationships. (2) A selected subset of data instances that involve deliberately obfuscating attributes and juxtaposing them in Q&A pairs, laying down an inferential groundwork. (3) Where available, user-defined directives or task blueprints are incorporated, offering a richer context, and further refining the imputation process.

Model Selection Module. Upon successfully navigating through the modality inference and feature engineering stages, AutoM^{3}L moves to pinpoint the optimal model architecture for each of the data modalities. For model organization, the collection of the model candidates is termed a model zoo, where each model is stored as a model card. The model card captures a spectrum of details, from the model's name, type, the data modality it can be applied to, empirical performance metrics, its hardware requirements, and *etc*. To streamline the generation of these cards, we utilize LLM-enhanced tools such as ChatPaper (Yongle Luo, 2023) to obviate the need for tedious manual processes. Utilizing text encoders, we generate embeddings for these model cards, thereby allowing users to fluidly enhance the model zoo by appending new cards, as illustrated in Fig. 3(a). Afterward, to adeptly match each modality with the suitable model, we propose the **M**odel **S**election-LLM (MS-LLM). We interpret this task as a single-choice dilemma, where the context presents a palette of models for selection. However, given the constraints on context length, parading a complete array of model cards isn't feasible. Therefore, we first filter the model cards based on their applicable modality type, retaining only those that align with the specified data modality. Thereafter, a subset of the top 5 models is identified via text-based similarity metrics between the user's requirements and the model cards' descriptions. These top-tier model cards then become part of MS-LLM's prompt, which, when combined with user directives and data specifics, steers MS-LLM toward its ultimate decision, leading to the identification of the best-suited model for the discerned modality, as depicted in Fig. 3(b). In essence, the MS-LLM prompt fuses: (1) A selected subset of five model cards, offering a glimpse of potential model candidates. (2) An input context, blending data narratives and user directives. The data narrative demystifies elements like data type, label type, and evaluation stan-

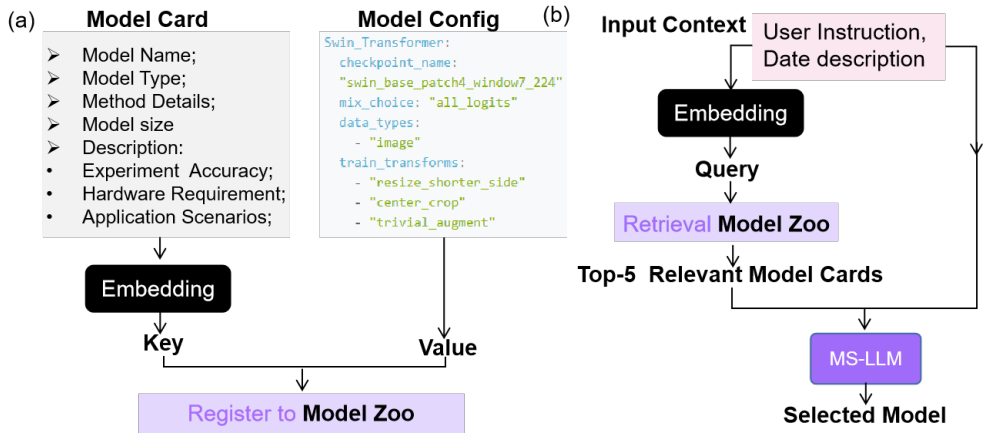

Figure 3: Illustration of the model zoo and MS-LLM. (a) Model addition process: This stage showcases how new models are incorporated into the model zoo, visualized as a vector database. The model card's embedding vector serves as the unique identifier or key, paired with its respective model configuration as the value. (b) Model retrieval process: This segment represents the model selection process. Given user directives, the system initiates a query, pinpointing the top 5 models that align with each input modality. From this refined subset, MS-LLM then determines and selects the most fitting model.

dards. Meanwhile, user directives can elucidate custom model requirements. For instance, a user stipulation expressed as "*deploy the model on the CPU device*" would guide MS-LLM to models primed for lightweight deployments.

Pipeline Assembly Module. Following the retrieval of uni-modal models, there's a crucial step of fusing these models We employ a late fusion strategy to integrate the multimodal data, where this process can be mathematically expressed as:

$$\begin{aligned} \texttt{feature}_i &= \texttt{feature\_adapter}_i(\texttt{model}_i(\texttt{x}_i)), \\ \texttt{logits}_{\texttt{fuse}} &= \texttt{fusion\_head}(\texttt{fusion\_model}(\texttt{concat}(\texttt{feature}_1, ..., \texttt{feature}_n))), \end{aligned} \tag{1}$$

where `concat` denotes concatenation, $\texttt{x}_i$ writes for the input data of modality i ($i = 1, \cdots, n$), $\texttt{feature\_adapter}_\texttt{n}$ functions to adapt the output of $\texttt{model}_\texttt{n}$ to a consistent dimension. Notably, both the `fusion_head` and `fusion_model` are the target models to be identified. However, determining the architectures for `fusion_head` and `fusion_model` is not practical to rely on rule-based methods, since these architectures depend on the number of input modalities. hence we formulate this process as a code generation task. Instead, we reframe this as a code generation challenge, wherein the **P**ipeline **A**ssembly-LLM (PA-LLM) is tasked with generating the necessary fusion model architecture, integrating features produced by each uni-modal model. Concretely, PA-LLM leverages the code generation capabilities of LLMs to produce executable code for both model fusion and data processors, as depicted in Fig. 4(a). This is achieved by supplying the module with model configuration files within the prompt. Similarly, data processors are synthesized based on the data preprocessing parameters detailed in the configuration file. PA-LLM allows us to automate the creation of programs that traditionally demanded manual scripting, simply by providing the requisite configuration files. A point of emphasis is our prioritization of integrating pre-trained models for text and visual data, primarily sourced from `HuggingFace` and `Timm`. This involves adapting the code to facilitate model loading. By establishing ties with the broader ML community, we've substantially amplified the versatility and applicability of our model zoo.

Automated Hyperparameter Optimization Module. Hyperparameters such as learning rate, batch size, hidden layer size within a neural network, loss weight and *etc* are commonly manually adjusted in conventional ML pipelines, which is thus labor intensive and time-consuming. While external tools like `ray.tune` have been invaluable, allowing practitioners to define hyperparameters and their search intervals for optimization, there remains a compelling case for greater automation. To bridge this gap, we propose the **H**yper**P**arameter **O**ptimization-LLM (HPO-LLM),

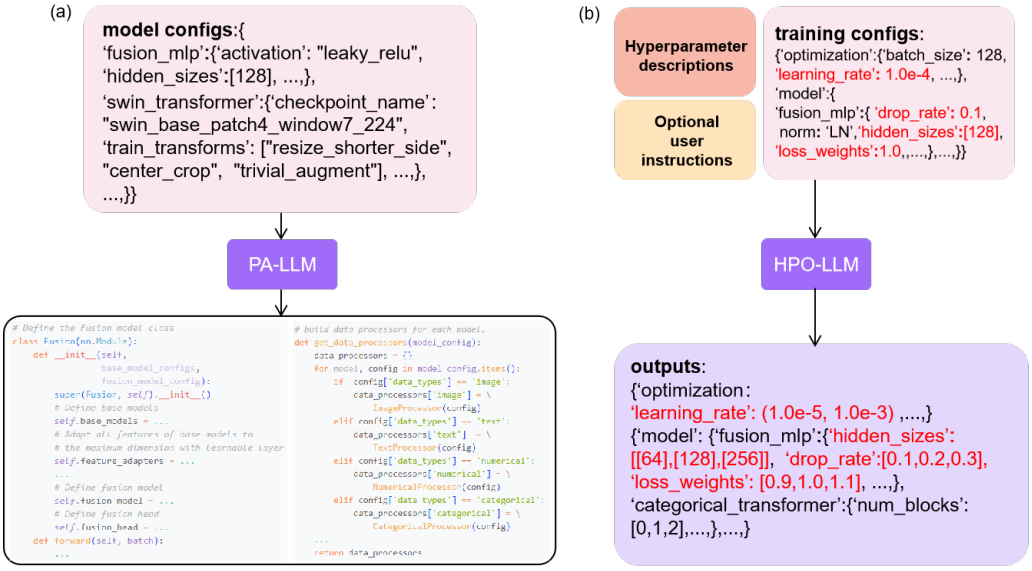

Figure 4: (a) The PA-LLM is responsible for generating executable code, ensuring seamless model training and data processing. (b) On the other hand, the HPO-LLM deduces optimal hyperparameters and defines appropriate search intervals for hyperparameter optimization.

building upon the foundational capabilities of `ray.tune`. The essence of HPO-LLM is its ability to ascertain optimal hyperparameters and their search intervals by meticulously analyzing a provided training configuration file, as visualized in Fig. 4(b). Harnessing the deep knowledge base of LLMs concerning ML training, we employ the HPO-LLM to generate comprehensive descriptions for each hyperparameter found within the configuration file. These descriptions, paired with the original configuration file, form the foundation of the prompt context for HPO-LLM. The module then embarks on identifying the hyperparameters primed for optimization, basing its proposals on preset values cataloged within the hyperparameter list. Delving into the specifics, the input prompt fed to HPO-LLM is multi-faceted: (1) It incorporates the training configuration file, brimming with the hyperparameter set, aiding HPO-LLM in cherry-picking hyperparameters ripe for optimization. (2) LLM-generated text descriptions for each hyperparameter, furnishing HPO-LLM with a nuanced understanding of each hyperparameter's implications. (3) Optional user directives, offering a personalized touch. Users can weave in additional instructions, guiding HPO-LLM's decision-making. This could encompass emphasizing certain hyperparameters based on unique requirements. By intertwining the capabilities of `ray.tune` with our HPO-LLM, we've pioneered an approach that takes hyperparameter optimization to new heights, marrying automation with enhanced acumen.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Datasets. To evaluate the efficacy of the `AutoM`^{3}`L` system, we conducted experiments on four multimodal datasets, all sourced from the Kaggle competition platform. These datasets encompass a range of tasks, namely classification, regression, and retrieval. For classification, we use two datasets, each characterized by distinct modalities: (1) PetFinder.my-Adoption Prediction (PAP): This dataset aims to predict pet adoptability, leveraging image, text, and tabular modalities. (2) Google Maps Restaurant Reviews (GMRR): It is curated to discern the nature of restaurant reviews on Google Maps, making use of image, text, and tabular modalities. Turning our attention to regression, we utilized the PetFinder.my-Pawpularity Contest dataset (PPC). This dataset's primary objective is to forecast the popularity of shelter pets, drawing insights from text and tabular modalities. For the retrieval-based tasks, we employed the Shopee-Price Match Guarantee dataset (SPMG), which aims to determine if two products are identical, hinging on data from image and text modalities. Our performance metrics include accuracy for classification tasks, the coefficient of

Table 1: Evaluation for modality inference. AutoM^{3}L can effectively determine the data modality, even on data that AutoGluon misclassifies.

Method	AutoGluon	AutoM^{3}L
PAP↑	0.4121	0.4080
PPC↓	1.0129	1.0129
GMRR↑	0.3727	0.4091
SPMG↑	0.9851	0.9851

Table 2: Evaluation for feature engineering. AutoM^{3}L filters out noisy features and performs data imputation, effectively mitigating the adverse effects of noisy data.

Method	AutoGluon	AutoM^{3}L
PAP↑	0.4022	0.4071
PPC↓	1.0131	1.0130
GMRR↑	0.3773	0.3893
SPMG↑	0.9837	0.9851

determination (R^2) for regression tasks, and the area under the ROC curve (AUC) for retrieval tasks. See Appendix B for more details.

Baseline. Given the scarcity of specialized multimodal AutoML frameworks, our experimental evaluations were exclusively performed using the AutoGluon framework. Setting up training pipelines in AutoGluon necessitated detailed manual configurations. This involved specifying which models to train and conducting a thorough pre-exploration to determine the parameters suitable for hyperparameter optimization and their respective search ranges. It's crucial to highlight that the automation and intelligence levels of AutoGluon remain challenging to quantify, and in this research, we innovatively measure them through the user study from the human perspective.

IRB Approval for User Study. The user study conducted in this research has received full approval from the Institutional Review Board (IRB). All methodologies, protocols, and procedures pertaining to human participants were carefully reviewed to ensure they align with ethical standards.

4.2 Quantitative Evaluation

We first carried out quantitative evaluations, drawing direct comparisons with AutoGluon with focus on the modality inference, automated feature engineering, and the automated hyperparameter optimization modules. For modality inference evaluation, apart from the modality inference component, all other aspects of the frameworks are kept consistent. For feature engineering and hyperparameter optimization, we aligned the modality inference from AutoGluon with the results of AutoM^{3}L to analyze their respective impacts on performance. Afterwards, we evaluate the pipeline assembly module in terms of intelligence and usability through user study in the next section, due to its inherent difficulty in quantitative assessment.

Evaluation for Modality Inference. Table 1 depicts the comparative performance analysis between AutoGluon's modality inference module and our LLM-based modality inference approach across various multimodal datasets. Within AutoGluon, modality inference operates based on a set of manually defined rules. For instance, an attribute might be classified as a categorical modality if the count of its unique elements is below a certain threshold. When we observe the results, it's evident that AutoM^{3}L offers accuracy on par with AutoGluon for most datasets. This similarity in performance can be primarily attributed to the congruence in their modality inference outcomes. However, a notable divergence is observed with the GMRR dataset, where AutoM^{3}L obtains 0.4091 accuracy, significantly outperforming AutoGluon's 0.3727. Upon closer inspection, we identified that AutoGluon misclassified the 'image_path' attribute as categorical, thereby neglecting to activate the visual model. Such an oversight underscores the robustness of our LLM-based modality inference approach, which adeptly deduces modality specifics from both column names and their associated data.

Evaluation for Feature Engineering. Table 2 illustrates the comparisons utilizing AutoGluon's data preprocessing module and our LLM-based automated feature engineering module on multimodal datasets. Given the completeness of these datasets, we randomly masked portions of the data and manually introduced noisy features from unrelated datasets to assess the effectiveness of automated feature engineering. Note that, AutoGluon lacks a dedicated feature engineering module for multimodal data, making this experiment a direct assessment of our automated feature engineer-

Table 3: Evaluation on the hyperparameter optimization. AutoM^{3}L's self-recommended search space rivals, and in some cases surpasses, manually tuned search spaces.

Method	PAP↑	PPC↓	GMRR↑	SPMG↑
AutoGluon w/o HPO	0.4121	1.0129	0.4091	0.9851
AutoGluon w/ HPO	0.4455	1.0128	0.4272	0.9894
AutoM^{3}L	0.4435	1.0118	0.4499	0.9903

ing. We observed that automated feature engineering, which implements feature filtering and data imputation, effectively mitigates the impact of noisy data. Across all test datasets, automated feature engineering showed improvements, with a notable 1.2% performance increase observed in the GMRR dataset.

Evaluation for Hyperparameter Optimization. We also conduct experiments to assess the capabilities of tje automated hyperparameter optimization module within AutoM^{3}L. Contrasting with AutoGluon, where users typically grapple with manually defining the hyperparameter search space, AutoM^{3}L streamlines this process. From Table 3, it's evident that the integration of hyperparameter optimization during the training phase contributes positively to model performance. Impressively, AutoM^{3}L matches AutoGluon's accuracy across all datasets. However, the standout advantage of AutoM^{3}L lies in its automation; while AutoGluon demands a manual, often tedious setup, AutoM^{3}L markedly reduces human intervention, offering a more seamless, automated experience.

4.3 User Study

Hypothesis Formulation and Testing. To assess AutoM^{3}L's effectiveness, we conducted a user study focused on whether the LLM controller can enhance the degree of automation within the multimodal AutoML framework. We formulated null hypotheses:

- **H1**: *AutoM^{3}L does **not** reduce time required for learning and using the framework.*
- **H2**: *AutoM^{3}L does **not** improve user action accuracy.*
- **H3**: *AutoM^{3}L does **not** enhance overall framework usability.*
- **H4**: *AutoM^{3}L does **not** decrease user workload.*

We performed single-sided t-tests to evaluate statistical significance. Specifically, we compared AutoM^{3}L and AutoGluon on the following variables: task execution time, the number of attempts, system usability, and perceived workload. See Appendix C.3 for details about the variables.

User Study Design. As depicted in Fig. 5, our user study's workflow unfolds in structured phases. Note that the user study has been reviewed by IRB and granted full approval. The study begins with the orientation phase where voluntary participants are acquainted with the objectives, underlying motivations, and procedural details of the user study. This phase is followed by a user background survey, which gleans insights into participants' professional roles, their prior exposure to technologies such as LLM and AutoML, and other pertinent details. The core segment of the study involves hands-on tasks that participants undertake in two distinct conditions: perform multimodal task AutoML with AutoGluon and with AutoM^{3}L. These tasks center around exploring the automation capabilities of the AutoML frameworks, as well as gauging the user-friendliness of their features

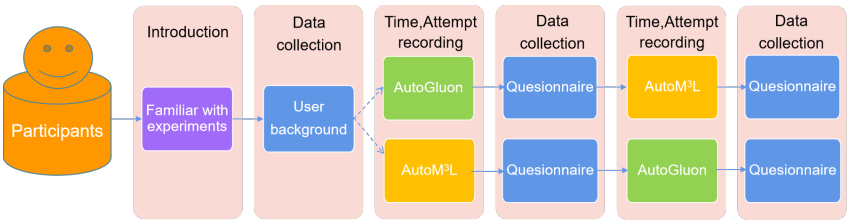

Figure 5: The workflow of the user study to measure the user-friendliness of the AutoM^{3}L.

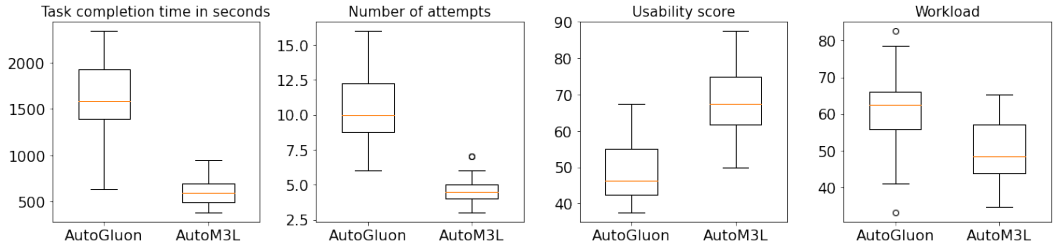

Figure 6: Boxplots displaying the distribution of the four variables collected in the user study.

such as hyperparameter optimization. Participants, guided by clear instructions, are tasked with constructing multimodal training pipelines employing certain models and defining specific hyperparameter optimization domains. To ensure a balanced perspective, participants are randomly split into two groups: the first interacts with AutoGluon, while the second delves into AutoM^{3}L. Upon task completion, the groups swap platforms. For a holistic understanding of user interactions, we meticulously track both the time taken by each participant for task execution and the number of attempts before the successful execution. The study culminates with a feedback session, where participants articulate their impressions regarding the usability and perceived workload of both AutoGluon and AutoM^{3}L via questionnaire. Their feedback and responses to the questionnaire, captured using Google Forms, form a crucial dataset for the subsequent hypothesis testing and analysis.

Results and Analysis of Hypothesis Testing. Our study cohort consisted of 20 diverse participants: 6 software developers, 10 AI researchers, and 4 students, which ensured a rich blend of perspectives of the involved users. The data we gathered spanned four variables, visualized in Fig. 6. To validate our hypotheses, we performed paired two-sample t-tests (essentially one-sample, one-sided t-tests on differences) for the aforementioned variables across two experimental conditions: AutoGluon and AutoM^{3}L. These tests were conducted at a significance level of 5%. The outcomes in Table 4 empower us to reject all the null hypotheses, underscoring the superior efficacy and user-friendliness of AutoM^{3}L. The success of AutoM^{3}L can be largely attributed to the interactive capabilities endowed by LLMs, which significantly reduce the learning curve and usage costs. Please refer to Appendix C.3 for detailed analysis.

Table 4: Hypothesis testing results from paired two-sample one-sided t-tests.

Hypothesis	T Test Statistic	P-value	Reject Hypothesis
H1	12.321	8.2×10^{-11}	Yes
H2	10.655	9.3×10^{-10}	Yes
H3	-5.780	1.0×10^{-5}	Yes
H4	3.949	4.3×10^{-4}	Yes

5 CONCLUSION

In this work, we introduce AutoM^{3}L, a novel LLM-powered Automated Multimodal Machine Learning framework. AutoM^{3}L explores automated pipeline construction, automated feature engineering, and automated hyperparameter optimization. This enables the realization of an end-to-end multimodal AutoML framework. Leveraging the exceptional capabilities of LLMs, AutoM^{3}L provides adaptable and accessible solutions for multimodal data tasks. It offers automation, interactivity, and user customization. Through extensive experiments and user studies, we demonstrate AutoM^{3}L's generality, effectiveness, and user-friendliness. This highlights its potential to transform multimodal AutoML. AutoM^{3}L marks a significant advance, offering enhanced multimodal machine learning across domains. One future direction is to encompass a diverse range of data modalities, spanning video, audio, and point clouds, among others. While we have currently addressed data imputation for tabular and textual formats, another future endeavors will integrate sophisticated image generation techniques to manage missing data in visual datasets. Such advancements will further solidify our standing in multimodal data analysis.

REFERENCES

John Brooke. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3):189–194, 1996.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Radwa Elshawi, Mohamed Maher, and Sherif Sakr. Automated machine learning: State-of-the-art and open challenges. *arXiv preprint arXiv:1906.02287*, 2019.

William A Falcon. Pytorch lightning. *GitHub*, 3, 2019.

Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. An open source automl benchmark. *arXiv preprint arXiv:1907.00909*, 2019.

Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pp. 139–183. Elsevier, 1988.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10. IEEE, 2015.

Ambika Kaul, Saket Maheshwary, and Vikram Pudi. Autolearn—automated feature generation and selection. In *2017 IEEE International Conference on data mining (ICDM)*, pp. 217–226. IEEE, 2017.

Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*, 2017.

Erin LeDell and Sebastien Poirier. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020. ICML, 2020.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, pp. 561–577, 2018.

OpenAI. Introducing chatgpt, 2022a. URL `https://openai.com/blog/chatgpt`.

OpenAI. New and improved embedding model, 2022b. URL `https://openai.com/blog/new-and-improved-embedding-model`.

OpenAI. Gpt-4 technical report, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Anton Vakhrushev, Alexander Ryzhkov, Maxim Savchenko, Dmitry Simakov, Rinchin Damdinov, and Alexander Tuzhilin. Lightautoml: Automl solution for a large financial services ecosystem. *arXiv preprint arXiv:2109.01528*, 2021.

Chi Wang, Qingyun Wu, Markus Weimer, and Erkang Zhu. Flaml: A fast and lightweight automl library. *Proceedings of Machine Learning and Systems*, 3:434–447, 2021.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hilde JP Weerts, Andreas C Mueller, and Joaquin Vanschoren. Importance of tuning hyperparameters of machine learning algorithms. *arXiv preprint arXiv:2007.07588*, 2020.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Peter Gam Jiaxi Cui circlestarzero Shiwen Ni Jaseon Quanta Qingxu Fu Siyuan Hou Yongle Luo, Rongsheng Wang. Chatpaper: Use llm to summarize papers. `https://github.com/kaixindelele/ChatPaper`, 2023.

A PROMPTS

Prompt 1: Full System Prompt For MI-LLM.

You are a helpful assistant that analyze data modalities in multimodal Auto-Machine learning task.
Your task is to analyze the data type of each column of the pandas.DataFrame tabular data.
Your answer must be in a strict JSON format: {"column name": "data type"}.
You can analyze the data type based on the corresponding column name,column data and the user instructions, which may contain the context of tasks/datasets, etc..
You should not omit any column of data in your answer.

Here are some examples for your reference:
Input: instructions:{data1_desc},Date:{data1_input}
Output: {data1_output}
Input: instructions:{data2_desc},Date:{data2_input}
Output: {data2_output}
Input: instructions:{data3_desc},Date:{data3_input}
Output: {data3_output}
Input: instructions:{data0_desc},Date:{data0_input}
Output:

Prompt 2: Full System Prompt For AFE-LLM$_{\text{filter}}$.

You are a helpful assistant that apply feature engineering, especially feature selection.
Given a set of features, you task is to filter out some features that are not relevant to a specific task.
You should filter out the features based on the feature names, feature type and user instrucions, which may contain the context of tasks/datasets, etc..
You cannot forge features that are not in the Input.
In particular, image features should be preserved.

Here are some examples for your reference:
Input: instructions:{data1_task}, features type:{data1_type1}, features:{data1_feat1}
Output: {data1_feat2}
Input: instructions:{data2_task}, features type:{data2_type1}, features:{data2_feat1}
Output: {data2_feat2}
Input: instructions:{data0_task}, features type:{data0_type1}, features:{data0_feat1}
Output:

Prompt 3: Full System Prompt For AFE-LLM$_{\text{imputed}}$.

You are a helpful assistant that apply feature engineering, especially data imputation.
Given a feature sequence, your task is to predict missing values in it. Missing values are represented by "???".
You should predict missing values based on other feature values in the sequence, and you can refer to user instructions, which may contrain context of the task/dataset, etc...
Your output format must be a certain element value, don't reply the reasoning process.

Here are some examples for your reference:
Input: instructions:{data1_task}, feature sequence:{data1_sequence}
Output: {data1_miss}
Input: instructions:{data2_task}, feature sequence:{data2_sequence}
Output: {data2_miss}
Input: instructions:{data3_task}, feature sequence:{data3_sequence}
Output: {data3_miss}
Input: instructions:{data0_task}, feature sequence:{data0_sequence}
Output:

Prompt 4: Full System Prompt For MS-LLM.

I am a deep learning software develop engineer, you're the code compiler, and we're working together on a multimodal Auto-Machine learning task.
Given the dataset description and user request ,Your task is to helps the user to select a suitable model.
You should focus more on the description of the model and find the model that has the most potential to solve requests and tasks.
Your answer must be in a strict JSON format: {"name": "model name", "reason": "your reasons to select the model"}. Please choose the most suitable model from: {model_cards}

User: Assume we have a dataset:{data_desc} and user request: {user_request},please select the most suitable model.
Answer is:

Prompt 5: Full System Prompt For PA-LLM(Data Processors Generation).

You are a helpful assistant that writes data processors code to load different types of data for multimodal Auto-Machine learning task.
Since different types of models need different data preprocessing, you task is to writes a function to return the corresponding data processors based on models' config.
Specifically, you do not need to define a data processor for the fusion model, and the label data processor is also required to provide label data for each model.
The function return must be in a strict dict format: {"data type": "data processor"}.
Please specify the library you imported in the code.

Here are some data processors code for you reference:

```
from multimodal.data import ImageProcessor
class ImageProcessor:
    def __init__(self,model_config):
    ...

from multimodal.data import TextProcessor
class TextProcessor:
    def __init__(self,model_config):
    ...

from multimodal.data import CategoricalProcessor
class CategoricalProcessor:
    def __init__(self,model_config):
    ...

from multimodal.data import NumericalProcessor
class NumericalProcessor:
    def __init__(self,model_config):
    ...

from multimodal.data import LabelProcessor
class LabelProcessor:
    def __init__(self,model_config):
    ...
...
```

Given some models' config as follow:
{configs}

Prompt 6: Full System Prompt For PA-LLM(Pipeline Assembly).

You are a helpful assistant that writes the Deep learning model code.
You task is to write a fusion model to fuse different base models' features.
Use # before every line except the python code.

Here are some model code for you reference:

```
from multimodal.models import CategoricalTransformer
class CategoricalTransformer(nn.Module):
    def __init__(self,model_config):
    ...

from multimodal.models import NumericalTransformer
class NumericalTransformer(nn.Module):
    def __init__(self,model_config):
    ...

from multimodal.models import TimmAutoModelForImagePrediction
class TimmAutoModelForImagePrediction(nn.Module):
    def __init__(self,model_config):
    ...

from multimodal.models import HFAutoModelForTextPrediction
class HFAutoModelForTextPrediction(nn.Module):
    def __init__(self,model_config):
    ...
...
```

Given some base models' config as follow:
{base_configs}
Give the fusion model config as follow:
{fusion_config}

You should then respond to me the code with:
1). Fusion technique should be learnable, MLP is recommended.
2). The fusion model structure should be defined as fusion_model and fusion_head,which output features and logits, respectively.
3). Base models instance should be defined in Fusion model Class.You should not change the value of the output of base model instances.
4). All base models have a uniform variable(self.out_features_dim) to represent the output features dimension.
5). Finding the maximum dimension of all base models' output features, and define learnable linear layers to adapt all base models' output features to the maximum dimension as the input of fusion_model. For example, if three models have feature dimensions are [512, 768, 64], it will linearly map all the features to dimension 768.
6). Output the logits,features,loss weights of fusion model and base models.The return must be in a JSON format: {model_name:{"logits":...,"features":...,"weight":...}}.
7). All the network layers and variable self.model_name,self.loss_weight should be defined in function __init__, not in function forward.
8). Some variables are not present in each model's config,you cannot use a variable that does not exist in the corresponding model config.
9). you should import the tools you used.

You should only respond in the format as described below :

```
Class Fusion:
    def __init__(self,...)
    ...
    def forward(self,batch)
    ...
    fusion_features = self.fusion_model(...)
    fusion_logits = self.fusion_head(fusion_features)
    ...
```

Prompt 7: Full System Prompt For Hyperparameter Description Generation.

You are a helpful assistant that adds descriptions for the parameters in the training config for machine learning task.
Your answer must be in a strict JSON format: {"hyperparameter name":"descriptions"}.
You should not mention the specific values in config in the description.

Given the model configs as follow: {configs}
Your answer:

Prompt 7: Full System Prompt For HPO-LLM.

You are a helpful assistant that infers the hyperparameters and their search ranges for hyperparameter optimization in machine learning task.
You can use the format:[value1,value2,value3,...,] to represent a discrete search range.
Your answer must be in a strict JSON format: {"hyperparameter_name":"search_range"}.
Here are some comments to help you understand the parameters better:
{self_desc}

Here are some things you need to focus on:
1).If the values in the search space are of type INT or FLOAT, then the search space needs to have at least 3 values.
2).The search ranges should refer to the original value of the config. The search ranges should include the original value of the config.
3).You should not output the hyperparameters don't need to optimize.
4).You cannot forge parameters that are not in the configuration file.
5).If the "checkpoint_name" is in config, only the "loss_weight" is taken.

Given the config as follow:
{config}
Given the user requirements:
{user}
Your answer:

B Structured Table Datasets

B.1 Dataset Details

Dataset Downloading Links. For the purpose of reproducibility, we provide the downloading link for each of the datasets used in this work.

- PetFinder.my-Adoption Prediction dataset (PAP):
  `https://www.kaggle.com/competitions/petfinder-adoption-prediction`
- PetFinder.my-Pawpularity Contest dataset (PPC)
  `https://www.kaggle.com/competitions/petfinder-pawpularity-score`
- Google Maps Restaurant Reviews dataset (GMRR):
  `https://www.kaggle.com/datasets/denizbilginn/google-maps-restaurant-reviews`
- Shopee-Price Match Guarantee dataset (SPMG):
  `https://www.kaggle.com/competitions/shopee-product-matching`

Table 5: Task and structure of each dataset

Dataset Name	#Train	#Test	Task	Metric	Prediction Target
PAP	11721	2931	multiclass	accuray	category of adoption speed
PPC	7929	1983	regression	R^2	appeal'rate
GMRR	880	220	multiclass	accuracy	rating category of restaurant
SPMG	5000	1000	retrieval	roc-acc	whether data pair is in same class

Table 6: Example of data in multimodal structured table dataset with text (name, description), numerical (age), categorical (gender), and image paths (images) columns. With these attributes, we want to predict how quickly the pet will be adopted (adoption_speed). We only display the partial columns for brevity.

name	age	gender	description	images	adoption
Coco	13	2	Hi, Coco is a rescued puppy from xthe streets, ...	images/640683dd9-1.jpg	0
Muffin	1	2	This is the puppy we adopted from Crystal, ...	images/e3935c62d-1.jpg	0
Usyang	4	1	Both of my kitten is so active and spoilt, ...	images/d33f713d0-1.jpg	1
...	...	...	...	...	...

Table 7: Example of data in multimodal structured table dataset with categorical attribute (Eyes, Face, Near, Blur) and corresponding photo paths (Images) of pets. With these attributes, we want to determine a pet photo's appeal (Pawpularity). We only display the partial columns for brevity.

Eyes	Face	Near	Blur	images	Pawpularity
1	1	1	0	train_images/ 0007de18844b0dbbb5e1f607da0606e0.jpg	63
1	1	0	0	train_images/ 0009c66b9439883ba2750fb825e1d7db.jpg	42
1	1	1	0	train_images/ 0013fd999caf9a3efe1352ca1b0d937e.jpg	28
...	...	...	...	...	...

C Questionnaire and Variables in User Study

C.1 User Background Survey Questionnaire

1. Age? *Single-choice question.*

◯ <18
◯ 18-24
◯ 25-34
◯ 35-44
◯ >44

2. Gender? *Single-choice question.*

◯ Male

Table 8: Example of data in multimodal structured table dataset with image (photo), text (business name, author name, text), numerical(rating). we want to predict which category(rating$_c ategory)theauthorisrating$.

business_ name	author_ name	text	photo	rating	rating_ category
Haci'nin Yeri - Yigit Lokantasi	Gulsum Akar	We went to Marmaris with ...	dataset/ taste/ hacinin_yeri_ gulsum_akar.png	5	taste
Haci'nin Yeri - Yigit Lokantasi	Oguzhan Cetin	During my holiday in Marmaris we ate ...	dataset/ menu/ hacinin_yeri_ oguzhan_cetin.png	4	menu
Pizza Fellas	Kadir Tasci	The ambiance of the place is ...	dataset/ indoor_atmosphere/ pizza_fellas_ kadir_tasci.png	5	indoor_ atmosphere
...	...	...	...	...	...

Table 9: Example of data in multimodal structured table dataset with image paths (image1, image2) and texts (title1, title2). we want to determine whether the image-text and image-text pair is in same class(p=1) or not. the original data give a image path, it's text description and corresponding class. For each item, we choose other item from same or different class with equal probability to form positive or negative pair.

image1	title1	image2	title2	p
f28094791c585c3f 1f7c0662e2cbecee .jpg	YANG YY 001 Air pump aerator baterai Yang	a4e379e2da3947ce d71630fbdda70c4b .jpg	Paket Super Kinclong Lengkap	0
1267eb326c6ad70a 32fb942b4834f818 .jpg	Promag Tablet 1 Box	2d8ca235317a263c aeb5432e57aeeff8 .jpg	Promag 1 Box isi 3 lembar	1
2d8ca235317a263c aeb5432e57aeeff8 .jpg	Promag 1 Box isi 3 lembar	088fec7809a7d809 73606507b123c66d .jpg	PAKET SHAMPO KUNTZE	0
...	...	...	...	...

◯ Female

3. What is your highest level of education? *Single-choice question.*

◯ High School or Below
◯ Bachelor's Degree
◯ Master's Degree
◯ Ph.D.
◯ Other: _____

4. What is your occupation? *Single-choice question.*

◯ Student
◯ Engineer
◯ Data Scientist/Analyst
◯ AI Algorithm Engineer
◯ Educator
◯ Doctor/Medical Professional

◯ Other: _____

5. Are you familiar with Python? *Single-choice question.*

   ◯ Yes
   ◯ No

6. Are you familiar with terminal operation? *Select only one bullet point.*

   ◯ Yes
   ◯ No

7. Do you have any experience with machine learning? *Select only one bullet point.*

   ◯ Yes, experienced
   ◯ Yes, some experience
   ◯ No, no experience

8. Have you used any AutoML tools or platforms before? *Select only one bullet point.*

   ◯ Yes, very familiar
   ◯ Yes, somewhat familiar
   ◯ No, not familiar

9. Are you familiar with the AutoGluon used in this experiment? *Select only one bullet point.*

   ◯ Yes
   ◯ No

10. Are you familiar with the Large language model? *Select only one bullet point.*

    ◯ Yes, very familiar
    ◯ Yes, somewhat familiar
    ◯ No, not familiar

11. Would you be willing to participate in this experiment? *Select only one bullet point.*

    ◯ Yes, I am willing to participate
    ◯ No, I am not willing to participate

12. What are your expectations for automated machine learning methods? *Select only one bullet point.*

    (a) _____

C.2 Questionnaire After Task Execution

1. How much time did it take in total to complete all the tasks? (in seconds) _____
2. How many script execution attempts did you make in total to complete the tasks?_____
3. I think that I would like to use this system frequently. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

4. I found the system unnecessarily complex. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

5. I thought the system was easy to use. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

6. I think that I would need the support of a technical person to be able to use this system. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

7. I found the various functions in this system were well integrated. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

8. I thought there was too much inconsistency in this system. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

9. I would imagine that most people would learn to use this system very quickly. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

10. I found the system very cumbersome to use. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

11. I felt very confident using the system. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

12. I needed to learn a lot of things before I could get going with this system. *Select only one bullet point.*

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

13. Mental Demand:How mentally demanding was the task? *Please assign a score between 1 and 20, where 1 = very low, and 20 = very high.*

_____

14. Physical Demand:How physically demanding was the task? *Please assign a score between 1 and 20, where 1 = very low, and 20 = very high.*

_____

15. Temporal Demand:How hurried or rushed was the pace of the task? *Please assign a score between 1 and 20, where 1 = very low, and 20 = very high.*

_____

16. Performance: How successful were you in accomplishing what you were asked to do? *Please assign a score between 1 and 20, where 1 = very low, and 20 = very high.*

_____

17. Effort:How hard did you have to work to accomplish your level of performance? *Please assign a score between 1 and 20, where 1 = very low, and 20 = very high.*

_____

18. Frustration:How insecure, discouraged, irritated, stressed and annoyed wereyou? *Please assign a score between 1 and 20, where 1 = very low, and 20 = very high.*

_____

19. Main source of workload? *Select only one bullet point.*
    ◯ Mental Demand ◯ Physical Demand
20. Main source of workload? *Select only one bullet point.*
    ◯ Temporal Demand ◯ Performance
21. Main source of workload? *Select only one bullet point.*
    ◯ Effort ◯ Frustration
22. Main source of workload? *Select only one bullet point.*
    ◯ Mental Demand ◯ Temporal Demand
23. Main source of workload? *Select only one bullet point.*
    ◯ Effort ◯ Physical Demand

24. Main source of workload? *Select only one bullet point.*
◯ Performance ◯ Frustration
25. Main source of workload? *Select only one bullet point.*
◯ Effort ◯ Mental Demand
26. Main source of workload? *Select only one bullet point.*
◯ Temporal Demand ◯ Frustration
27. Main source of workload? *Select only one bullet point.*
◯ Physical Demand ◯ Performance
28. Main source of workload? *Select only one bullet point.*
◯ Mental Demand ◯ Performance
29. Main source of workload? *Select only one bullet point.*
◯ Temporal Demand ◯ Effort
30. Main source of workload? *Select only one bullet point.*
◯ Frustration ◯ Physical Demand
31. Main source of workload? *Select only one bullet point.*
◯ Frustration ◯ Mental Demand
32. Main source of workload? *Select only one bullet point.*
◯ Frustration ◯ Temporal Demand
33. Main source of workload? *Select only one bullet point.*
◯ Performance ◯ Effort

C.3 User study Details

C.3.1 Definition and Calculation of Variables

We denote participant responses to the $\mathtt{i}^{\mathtt{th}}$ question in questionnaire C.2 as $\mathtt{s_i}$. Questions 1-18 are numerical variables, while the remaining are categorical.

Dependent Variables.

- **Task Execution Time**: Objective, continuous variable measuring the total time taken by participants to successfully complete the task. Derived directly from the response to question 1 in questionnaire C.2:

$$\mathtt{Time} = \mathtt{s_1}. \tag{2}$$

- **Number of Attempts**: Objective, continuous variable recording the total script execution attempts by participants to successfully complete the task. Derived directly from the response to question 2 in questionnaire C.2:

$$\mathtt{Attempts} = \mathtt{s_2}. \tag{3}$$

- **Usability Score**: The Usability Score is a subjective, continuous metric gauging the system's perceived usability. It is sourced from the System Usability Scale (SUS) survey questionnaire(Brooke, 1996), which comprises 10 questions. Each question offers five response choices from "Strongly Disagree" to "Strongly Agree", which are numerically scored from 1 to 5. Formally, the usability score is based on the responses to questions 3 through 12 in questionnaire C.2. To quantify usability, we apply the standard scoring system of the SUS to convert the scores for each participant on these questions into a new numerical format. Subsequently, we calculate the sum of these scores and multiply the result by 2.5. This step serves to reposition the original scores, which originally ranged from 0 to 40, into a revised scale spanning from 0 to 100. Although interpreted like percentiles, they aren't percentages. Higher scores signify better-perceived usability, which is mathematically defined as

$$\mathtt{Usability} = 2.5 \times \sum_{i=1}^{5} (\mathtt{s_{1+2i}} - 1) + (5 - \mathtt{s_{2+2i}}). \tag{4}$$

- **Workload Index**: This subjective, continuous variable assesses perceived mental workload and is derived from the NASA Task Load Index (NASA TLX) questionnaire (Hart & Staveland, 1988). Recognized for its comprehensive evaluation of mental workload, the NASA TLX divides workload into six categories: Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, and Frustration. Participants rate each category on a scale of 1 to 20 (questions 13 to 18). They also evaluate the significance of 15 pairs of these categories in shaping the overall workload (questions 19 to 33). The scale score for each dimension is calculated as $\mathtt{s_i} \times 5$. The weighted score $\mathtt{w_i}$ is determined based on the frequency of selection for each dimension as more important in questions 19 to 33, divided by 15. The overall workload score, ranging from 0 to 100, is then computed by summing the products of the scale score and weighted score for each dimension as follows:

$$\mathtt{Workload} = \sum_{i=13}^{18} \mathtt{w_i} \cdot (5 \cdot \mathtt{s_i}). \tag{5}$$

Independent Variables. The most direct independent variables stem from the differences in approaches between participants using $\mathtt{AutoM^3L}$ and AutoGluon when performing tasks. Furthermore, various independent variables have the potential to impact user outcomes, including:

- **Participant Background:** These categorical variables encompass background information about the participants, such as their professional roles, providing deeper insights into potential background knowledge, biases, or preferences that users may bring to task execution.
- **Familiarity with Technology:** These numerical variables represent each participant's familiarity with terminal operations, the Python programming language, LLM, and AutoML methods. Familiarity levels can potentially impact the ease with which participants complete AutoML tasks, thus influencing the final measurement outcomes.

C.3.2 Participant Recruitment.

We strategically recruited volunteers to participate in our user study, encompassing potential users of AutoML frameworks, including software developers, AI researchers, and students. Among AI researchers, we included individuals both familiar and unfamiliar with AutoML frameworks. We believe that this diverse group of participants provides a comprehensive evaluation of our $\mathtt{AutoM^3L}$, considering a range of backgrounds and expertise levels in AutoML methods.

C.3.3 User Study Analysis Process.

Collected Data. We collected both objective and subjective evaluations from each user regarding the systems, including task execution time, number of attempts, usability scores, and workload indices. Box plots for these four variables are presented individually in Fig 6. Each box plot displays the minimum value, first quartile (Q1), median (Q2), third quartile (Q3), and maximum value for these variables. The box represents the interquartile range (IQR) from Q1 to Q3, with a line inside indicating the median.

Normality Testing. To ensure the validity of our subsequent statistical analyses, we conducted a normality test on our data using Q-Q plots, as depicted in Fig 7 and Fig 8. The proximity of our data points to the theoretical quantile lines, along with the bell-shaped curve observed in the histograms, suggests that task completion time, the number of attempts, usability score and workload reasonably adhere to the assumption of normality.

Hypothesis Testing. We employed hypothesis testing to assess the statistical significance of the observed performance differences between the AutoGluon and $\mathtt{AutoM^3L}$ conditions. The differences we are analyzing, denoted as $\mathtt{d_i}$, were calculated by taking the AutoGluon measurements and subtracting the corresponding $\mathtt{AutoM^3L}$ measurements. Assuming the null hypothesis, both AutoGluon and $\mathtt{AutoM^3L}$ exert an equivalent impact. Consequently, these differences are expected to adhere to a distribution centered around zero, denoted as $\mu_\mathtt{d} = 0$. Our dataset for hypothesis testing comprises 20 samples, and we express the null and alternative hypotheses as follows:

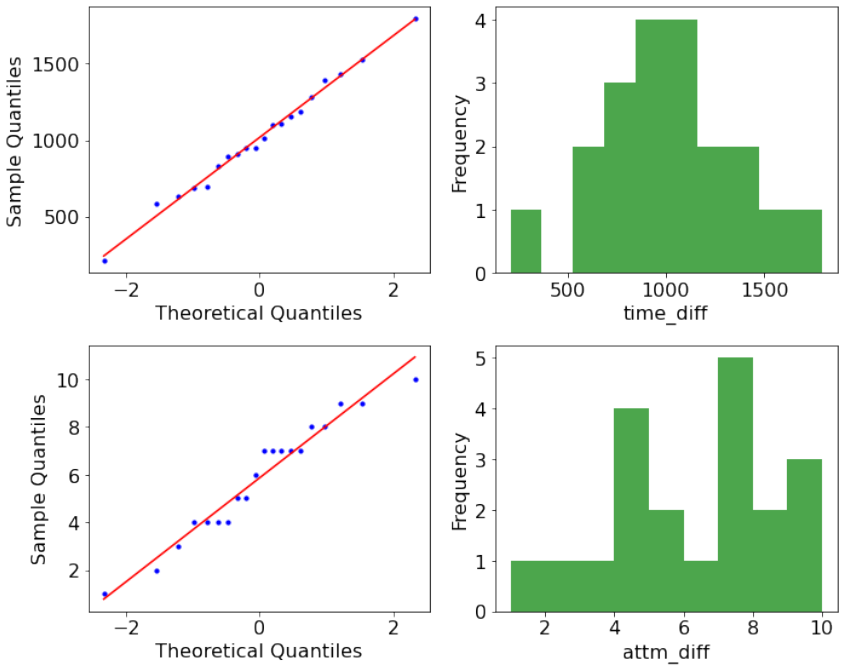

Figure 7: Normality Testing for task completion time and the number of attempts. The top row of panels present the Q-Q plot and histogram for task completion time, respectively. Similarly, the lower row of panels illustrate the Q-Q plot and histogram for the number of attempts.

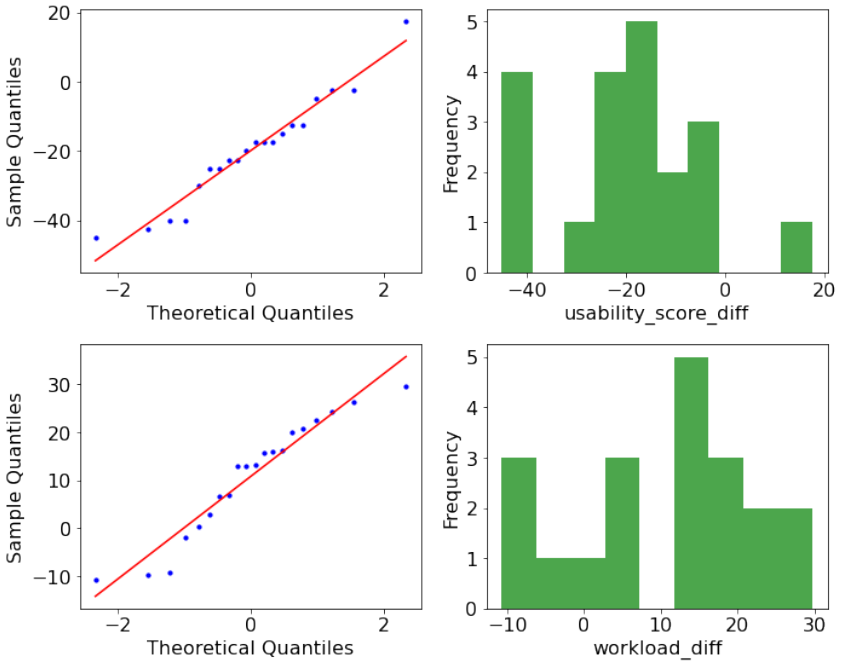

Figure 8: Normality Testing for system usability and workload. The top row of panels present the Q-Q plot and histogram for the usability, respectively. Similarly, the lower row of panels illustrate the Q-Q plot and histogram for the workload.

$$H_0 : \mu_d = 0 \quad \text{against} \quad H_1 : \mu_d > 0 \tag{6}$$

This applies to the testing of hypotheses H1, H2, and H4. In the case of testing H3, the alternative hypothesis is that $\mu_d < 0$. Here, $\overline{d}$ and s_d denote the sample mean and sample standard deviation of the observed differences, respectively. With these parameters in mind, the sampling distribution of the test statistic follows a t-distribution with degrees of freedom equal to n 1. Consequently, under the null hypothesis H_0,

$$\tau = \frac{\overline{d}}{s_d/\sqrt{n}} \sim t_{n-1} \tag{7}$$

D EXPERIMENT IMPLEMENTATION

Implementations for Quantitative Evaluations. In our quantitative assessment, we primarily relied on OpenAI's APIs: `gpt-4-0314`(OpenAI, 2023) for code generation, `gpt-3.5-turbo-0301`(OpenAI, 2022a) for text completion, and `text-embedding-ada-002`(OpenAI, 2022b) for text embedding. For all APIs, we set the temperature parameter to 0 to maximize determinism. The experiments utilized the PyTorch Lightning framework(Falcon, 2019) for model training, and Ray(Moritz et al., 2018) served as our tool for hyperparameter search. Given that the competition datasets' test sets were unlabeled, we opted for a stratified sampling approach, reserving 20% of the training set as a validation set for performance assessment. For the retrieval dataset, 20% of the IDs were randomly chosen, and we created matching pairs of positive and negative samples for validation. While we consistently used the same models in our experiments as those in the AutoGluon assessments for the same modality data, our emphasis on the model selection module was not solely on accuracy. Instead, we were driven by the goal of intelligently choosing models based on data modality and user-specific needs.

Implementations in User Study. For the user study, given the advanced capabilities of GPT-3.5, we chose to employ the `gpt-3.5-turbo-0301` API as the LLM backbone of `AutoM`^{3}`L`. Participants in the study were provided execution scripts for both AutoGluon and `AutoM`^{3}`L`, allowing them a comparative experience.