University of Padova

Department of Mathematics

Master Thesis in Data Science

Development of an AutoML web application for machine learning tasks

Supervisor
Professor Alberto Testolin
University of Padova

Master Candidate
Andrea Bello

Academic Year
2024-2025

In a world deeply entangled with AI, we must take the initiative and show everyone how to make good use of it.

"Unless someone like you cares a whole awful lot, nothing's going to get better. It's not."

-The Once-ler

Abstract

Automated Machine Learning, also known as AutoML, is a new and florid field of data science that aims at symplifying the process of designing, development and usage of typical machine learning models. Usually the tasks involved when model-building are performed by experts in the field of computer science, expecially due to the technicalities such as, but not only: data preprocessing, feature engineering, model selection, hyperparameter tuning, model evaluation, performance analysis and forecasting procedures. These are literal barriers that limit the accessibility for non-expert users. Exploiting AutoML systems aim to automate these processes, making machine learning available to all users, regarding of their knowledge. This thesis navigate the role of AutoML in "democratizing" machine learning, emphasizing on simplicity, usability and efficiency. It examines challenges usually associated with automating end-to-end machine learning pipelines. Additionally, this study highlights the integration of a chatbot assistant powered by a large language model (LLM) that guides the user in its journey as a machine learning expert. The research culminates in the development of a web application based on AutoML, designed to implement machine learning tasks for users with varying levels of expertise. This work is a contribution to the broader adoption of machine learning across diverse industries, as the applications are fair and diverse, regarding of the field of application.

Contents

Listing of figures

1 Introduction

Machine learning is seen as an ability to combine algorithms and data to generate useful, even interesting, predictions that companies, governments and society can use to solve problems, such as more accurate customer segmentation, predictive value of the life value of the customer, improved public policy, and autonomous driving. Machine learning is different from normal computer programming—the instruction is not explicitly encoded to tell the machine what to do—and it is different from AI because it does not make autonomous decisions. For people with machine learning fears, it feels like abstract magic, but in reality, it's just mixed mathematics and a process carefully trained that can benefit all of us. [1] Machine learning, even deep learning models, require domain experts to perform procedures such as data preprocessing, feature engineering, model selection, and hyperparameter tuning. These technical aspects account for most of the complexity of machine learning, which makes it difficult for non-experts to learn or use machine learning models effectively in their respective fields. However, since the application field of machine learning is growing further from tasks such as facial recognition, spam filtering, financial forecasts, and social media optimization, non-experts are not abandoning their interest, so today we are faced with the need to bridge this gap between non-experts and machine learning. [2]

An example of this challenge is the prediction of customer turnover in subscription-based business. Companies that provide online services such as streaming and cloud storage want to identify users who may cancel their subscriptions. To solve this problem through machine learning,

an expert programmer must make several key decisions. First, they have to determine relevant data such as user activity logs, billing history and customer support interactions. Next, they must prepare the data, handle the missing value, encode classification variables such as subscription types, and create new features such as average monthly usage or complaints frequency. Once the data are prepared, the experts must choose the appropriate model based on the task and the dataset used. The datasets must then be divided into training, validation, and test sets to ensure good generalization of the model. Hyperparameters such as learning rates must be adjusted using techniques like grid searches and Bayesian optimization. After training, the model must be evaluated using precision, recall, or other evaluation metrics. If the performance is not satisfactory, the expert may need to tweak the characteristics, adjust the preprocessing steps or select another model. Finally, if a satisfactory model is achieved, it must be implemented, integrated into the company's CRM system, and continuously monitored for changes in client behavior that might require re-training.

I had the opportunity to work for LoopAI Group, headquartered in New York, San Francisco and Milan, which aims to transform organizations into cognitive businesses and provide tools needed to meet the future demands of knowledge workers, and increase current capacity. [3] I was directly involved in the company and was able to be part of a team addressing a specific cognitive problem. In particular, the research focuses on developing a web application that integrates AutoML and LLM to enable non-experts to build and build easy-to-implement machine learning models. The application aims to streamline the entire ML workflow from data intake and preprocessing to model selection, training and evaluation. By integrating LLM-powered guided and natural language interfaces, users can interact with systems using simple queries, making complex ML tasks easier to access and intuitive. By reducing manual intervention in model selection and optimization, the web application is aimed at giving users in various industries (such as business analytics, health care, finance, and retail) the tools to obtain meaningful insights from data. The research framework of this project explores various AutoML frameworks, LLM-based assistance techniques and user interface design principles to create an effective and user-friendly ML platform. Through this research, I hope to contribute to efforts to make ML more inclusive, ensuring that non-expert users can use its potential without requiring extensive technical knowledge. The findings and methodologies discussed in this study can also provide useful information to organizations wishing to integrate AutoML and LLM into their workflows and ultimately promote innovation and efficiency in various areas.

2

Background

This chapter is divided into five parts as follows: Section 2.1 presents an overview of AutoML and describes its purpose and use. Section 2.2 discusses the major challenges associated with automatic machine learning pipelines, including computer constraints and the need for human expertise. Section 2.3 examines the role of the Large Language Model (LLM) in AutoML, particularly the role of automating. Section 2.4 shows some AutoML tools that have been tested. Section 2.5 provides a review of literature on frameworks that facilitate the integration of LLMs into machine learning processes.

2.1 AutoML Overview

Automated Machine Learning (AutoML) is the process of automating tasks to apply machine learning to real problems. [4] It is a combination of automation and ML. AutoML provides a variety of methods and processes to enable non-machine learning experts, thereby improving its efficiency and accelerating research on the subject. [5] Since these tasks are often more complex than those of non-ML experts, the rapid growth of machine learning applications has led to the need for inexpensive and expertly used machine learning methods. [6]

2.2 Challenges in Automating ML Pipelines

Despite its promises, automation of machine learning is not a simple task, because it often faces several challenges, both naturally encountered by experts, but also some new ones. The first major problem is the need for human expertise when performing tasks such as data preprocessing, feature construction and model optimization. This problem is also known as "development as a cottage industry", and highlights the tension between expert-led development and the fully autonomous utopian systems that can learn and improve themselves. When performing data preprocessing, I, for example, refer to the following tasks: data cleansing, normalization, feature discovery, and imbalanced data management. Data cleansing is the process of detecting and correcting corrupt or inaccurate data. Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature extraction, also known as dimensionality reduction, is the transformation of high-dimensional data into a meaningful representation of reduced dimensionality, which should have a dimensionality that corresponds to the intrinsic dimensionality of the data. [7] Only in recent years has machine learning achieved significant success in many fields, but this success crucially relies on human experts to perform the following tasks:

- Preprocessing and data clean-up.
- Manual features selection or controlled (i.e. grid search algorithms).
- Model selection (i.e. finding the family of models).
- Hyperparameters initialization and optimization.
- Designing neural networks (i.e. in case of deep learning).
- Postprocessing of machine learning models.
- Technical analysis of the results.

While AutoML aims to reduce the involvement of experts, it also requires technical knowledge in machine learning algorithms and system design to ensure that automated systems work correctly and effectively when faced with different specific tasks. In addition, challenges such as meta-learning and the allocation of computational resources complicate the automation process. With the term "Meta-learning" I refer to the process of learning from previous machine learning tasks to improve future performance, and it remains a difficult area to efficiently automate even today. Furthermore, when AutoML code is running and automatically selecting

the best models, there is a high computational cost, so that it can represent a computational burden in time (i.e. it takes hours or days to run) and space (i.e. the need for a huge amount of Ram or vram). Balance between these constraints and the demand for high-quality and scalable solutions is a constant challenge in this area.

2.3 LLMs in AutoML

Large Language Models, such as the Generation Pre-trained Transformer (GPT-3), have made significant progress in natural language processing (NLP) in recent years. These models are trained on massive amounts of text data and are capable of generating text similar to humans, answering questions and performing other language-related tasks with high precision. An important development in this field is the use of transformer architectures, which can be seen briefly in Figure 2.1, and the underlying mechanism of attention that have greatly improved the ability of linguistic models to deal with long-range dependency in natural language texts. More specifically, the transformer architecture uses the self-attention mechanism to determine the relevance of the different parts of the input when generating predictions. This allows the model to better understand the relationships between words in a sentence, regardless of their position. [8] Large Language Models (LLMs) are emerging as a powerful tool in the AutoML space because they can potentially enhance various stages of the machine learning pipeline. [9] The main idea is that if we use their natural language understanding and generation abilities, we can use an LLM as an assistant to automate tasks such as data preprocessing, feature engineering, model selection, and also to produce first interpretations of results. As LLMs evolve, they can play a crucial role in reducing barriers to entry for non-experts, further democratizing access to machine learning tools and accelerating research in this field.

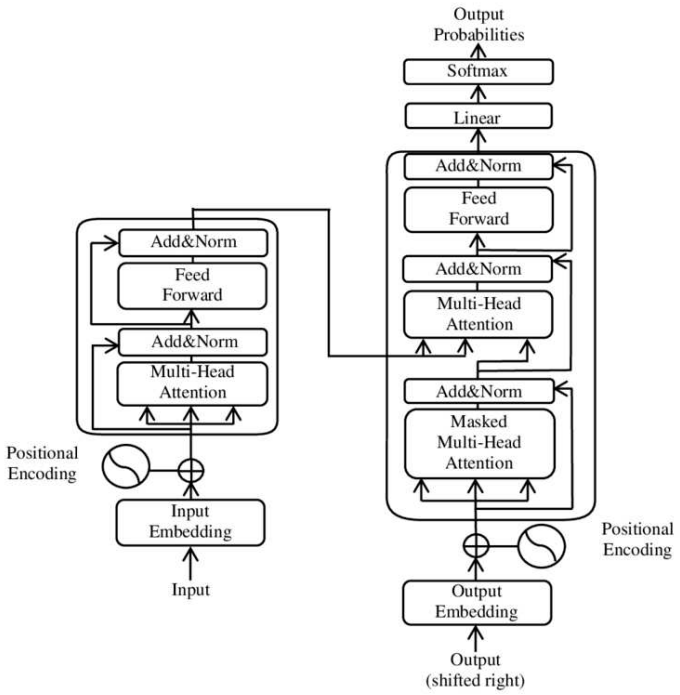


Figure 2.1: The Transformer - model architecture.

2.4 Tools tested

There are several AutoML tools available for my use cases, so I tested some of them to evaluate their effectiveness and suitability. Tools tested include: H2O.ai, AutoSklearn, mljar-supervised, AutoGluon and TransmogrifAI. The summary of the results is provided below.

2.4.1 H2O.ai

H2O.ai [10] is an AutoML platform that supports a wide range of machine learning models, including GLMs, XGBoost, random forests, deep learning, and stacked ensembles. Its AutoML feature automates model training and tuning according to a user-specified time limit. H2O.ai is mostly known for its scalability, allowing it to run seamlessly across distributed systems, but not only, its commercial product, Driverless AI, enables the automation of feature engineering, model validation, and even model interpretability. The major downside is that all this robustness and flexibility come with a cost as it can be resource-intensive and challenging to set up. On paper it seems more suitable for enterprise-grade applications as it comes with extensive

documentation and a great community support, but its complexity presents a barrier for users without deep machine learning expertise.

2.4.2 AutoSklearn

AutoSklearn is an AutoML library that automates the following procedures: model selection, hyperparameter optimization, and ensemble learning. It is known in the community because it mainly uses algorithms from scikit-learn, making it particularly suited for classification, regression, and ensemble tasks. It already uses meta-learning and it also works well for small and medium-sized datasets, but for large ones it needs the use of PoSH Auto-sklearn under rigid time limits .[11] One of the main strenghts of AutoSklearn, as previously mentioned, is its integration with scikit-learn, making it familiar and easy to use for those who have already used the libraries. Other cons are that its primary focus is on tabular data, and it is also resource-intensive, requiring significant computational power for large-scale experiments.

2.4.3 Mljar-supervised

Mljar-supervised offers an intuitive interface and extensive reporting features. It supports many different machine learning algorithms, going from the simple linear models, to also decision trees, and ensemble methods. [12] Mljar-supervised is particularly useful for small and medium-sized datasets, and it includes specific features for time series forecasting. It is user-friendly, so it is an excellent choice for non-expert users. It also provides interactive reports and emphasizes model interpretability. However, it is limited in scope, focusing primarily on traditional machine learning tasks and lacking support for deep learning or more complex use cases.

2.4.4 AutoGluon

The next tool presented is AutoGluon, and it's one of the easiest AutoML tool that supports all three data types: multimodal, tabular and time series. The main characteristic is that it uses ensemble methods to automatically combine multiple models, improving overall performance. This solution is particularly useful, as it is designed to deliver strong performances with close to no user input, but it is also very customizable, allowing users to personalize models and data types. AutoGluon excels in its ease of use, making it highly accessible for Python-based environments. [13] It is also highly versatile, handling a wide range of machine learning tasks such as, but not only: classification, regression and time series forecasting. It can become resource in-

tensive when training deep learning models, but it is a common problem to almost all AutoML tools.

2.4.5 TransmogrifAI

TransmogrifAI is an AutoML library written in Scala that runs on top of Apache Spark. It was developed with a focus on accelerating machine learning developer productivity through machine learning automation, and an API that enforces compile-time type-safety, modularity, and reuse. For context, an API is an Application Programming Interface that exposes data and functions of a software application to third-party users. [14] Originally it was a project within Salesforce, but it has been released under a BSD-3 open source license. One of its main features called "Transmogrification" is an automated feature engineering based on an extended set of feature data types. [15] This tool excels in handling structured data, while it performs poorly for unstructured data and tasks such as, but not only: natural language processing (NLP) and image processing. One major disadvantage is that its APIs are Scala-based, so it can be difficult for users not familiar with the language.

2.5 Literature Review

I am reviewing some key tools and frameworks that simplify and improve the development of applications using LLMs. I provide some information on Langchain, Semantic Kernel, Dust.tt and AutoGPT. All of these offer unique functions and components with the aim of simplifying the process of integrating and coordinating LLM-based applications. The main reason for the following information is due to the nature of the web application, which contains a chatbot powered by an LLM.

2.5.1 Langchain

LangChain is an open-source framework for developing applications based on LLMs. Supports two common programming languages, Python and JavaScript, enabling developers to work in different programming environments. [16] Langchain has a strong and active community that ensures regular updates and new features. I can describe Langchain architecture as composed of the following elements:

- Model I/O: This component enables the integration of LLMs by managing the formatting of inputs and outputs.

- Retrieval: Essential for Retrieval-Augmented Generation (RAG) applications, this component facilitates the loading of data, connection with vector databases, and transformation of documents to align with the needs of specific applications.
- Tools: These allow developers to integrate external services and applications, extending the functionality of their LLM systems.
- Agents: These guide the LLM through the application process by determining the next steps to take.
- Chains: These represent sequences of calls linking various components to create coherent and functional LLM applications.

2.5.2 Semantic Kernel

Semantic Kernel is an SDK that integrates LLMs such as OpenAI, Azure OpenAI and Hugging Face with conventional programming languages such as C#, Python and Java. Semantic Kernel does this by allowing the developer to define plugins that can be combined in just a few lines of code. [17] What makes the Semantic Kernel special, however, is its ability to automatically orchestrate plugins with AI. With semantic kernel planners, the developer can ask LLMs to create plans that reach a user's unique goal. Afterwards, Semantic Kernel will execute the user plan. The main components of Semantic Kernel are:

- Kernel: The central component of Semantic Kernel, it contains the necessary plugins and services to develop and execute AI applications.
- Planners: Special prompts within the kernel allow the generation of plans for task completion, helping agents determine the best sequence of actions.
- Plugins: These provide additional functionality, giving the LLM system unique capabilities using both code and prompts.
- Memories: Semantic Kernel includes a memory feature that stores context and embeddings, enabling the system to retain information across interactions and improve performance.

2.5.3 Dust.tt

Dust, in particular, is a platform that allows teams to create customized and secure AI assistants powered by LLMs. [18] These assistants integrate with company data sources to improve productivity and improve working processes in various domains. Key components of Dust.tt include:

- Flow-based Interface: A visual interface that helps developers design and manage complex data pipelines, ensuring data flows smoothly through different tasks and components.
- Templates: Pre-built templates for common LLM use cases, such as chatbots, text generation, and summarization, making it easier for developers to quickly implement standard functionalities.
- Transformers: A tool for preprocessing and transforming data before it is fed into the LLM, optimizing performance and results.
- Metrics and Monitoring: Provides real-time insights into the performance of LLM-based applications, allowing for efficient debugging and optimization.

Dust.tt's flow-based interface and monitoring tools make it a valuable platform for those looking to simplify the development and management of LLM workflows.

2.5.4 AutoGPT

AutoGPT is a powerful platform that allows the user to create, deploy and manage continuous artificial intelligence agents to automate complex workflows. [19] It is an open-source "AI agent" that, given an objective in natural language, tries to achieve it by decoupling it into subtasks and using the Internet and other tools in an automatic cycle. It uses OpenAI's GPT-4 or GPT-3.5 APIs and is one of the first examples of applications that use GPT-4 to perform autonomous tasks. [20] Key AutoGPT components include:

- Goal-Oriented Agents: Autonomous agents that use GPT-4 to accomplish user goals by decomposing them into manageable steps.
- Task Management: AutoGPT plans and executes tasks using self-prompting, allowing for iterative progress without human intervention.
- LLM-based Reasoning: The core GPT-4 model drives the agent's reasoning and decision-making process, enabling it to assess how to approach and complete each task.

- External Integrations: AutoGPT can connect with APIs, file systems, and web services to gather additional data or perform actions outside of its core model.
- Memory: Retains context across multiple iterations, improving the agent's ability to handle multi-step tasks autonomously.
- Multi-step Task Handling: AutoGPT excels in managing and executing multi-step processes autonomously, making it suitable for complex workflows.

AutoGPT's self-executing tasks and external integrations make it a powerful tool for automating workflows requiring continuous decision-making and data processing.

3

Development Process and System Architecture

The design of an efficient and easy to use system architecture is important when faced with the integration of an LLM and AutoML. Some challenges can arise from such implementations, mainly because I aim to create a framework that allows the user to interact with machine learning in the simplest way possible, while providing a solid system background automation. One of the main objectives is to balance user access and computational performance to ensure that users feel comfortable performing machine learning tasks and that results are reliable. The main question is whether it is realistically possible to create a modular system capable of achieving both.

3.1 User-Friendly Interface

The main interaction between the user and the application performing machine learning tasks is supported by an intuitive chatbot interface that acts as a natural assistant, allowing the user to talk and describe tasks using simple text. From the point of view of usability, the majority of the interaction performed by the user when navigating the application is known, as it follows patterns already present in everyday online interactions such as when shopping platforms or content management systems are accessed. In simpler terms, the user can navigate the site sim-

ply by filling out structured fields, moving between pages, registering, entering, uploading files and visualizing processed data sets. By supporting this approach, I aim to reduce the cognitive load requested by users, allowing them to focus on their tasks rather than being forced to learn a larger amount of information only to use the system. The first example for front-end design is created using Figma, and there are several examples listed here: The main work bench page with chatbot can be seen in Figure 3.1, and in Figure 3.2 and Figure 3.3 there are some examples of how results and/or insights can be displayed after model training.

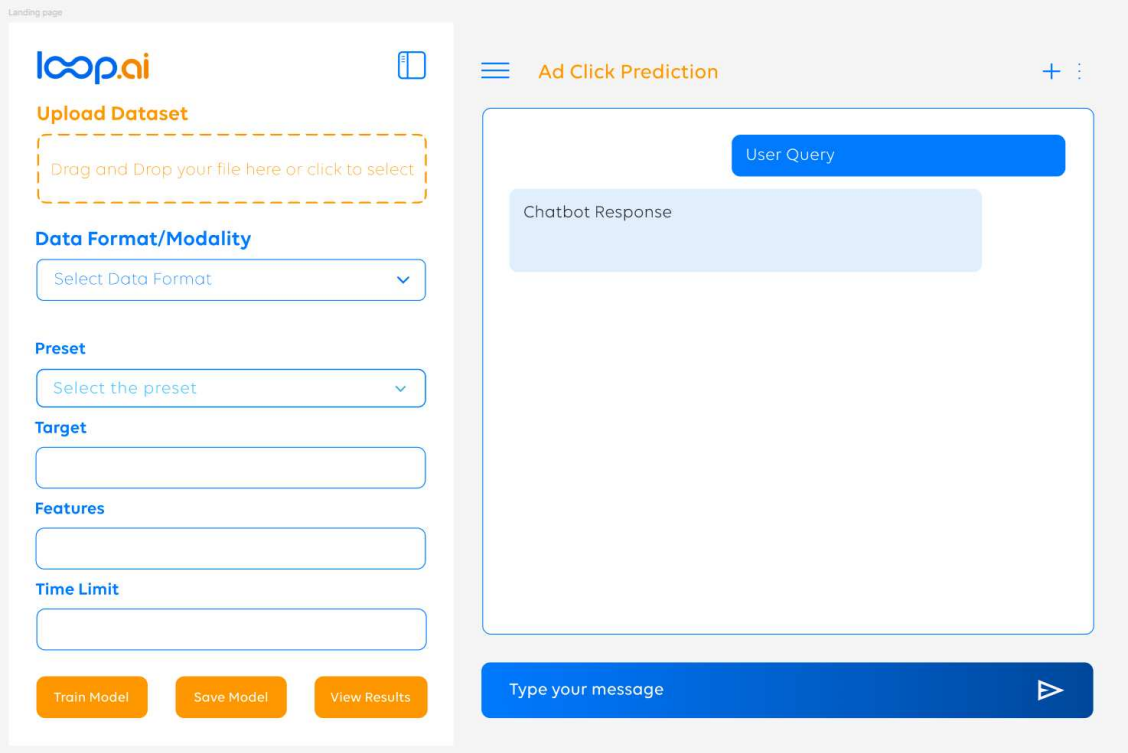


Figure 3.1: Sketch of workbench using Figma.

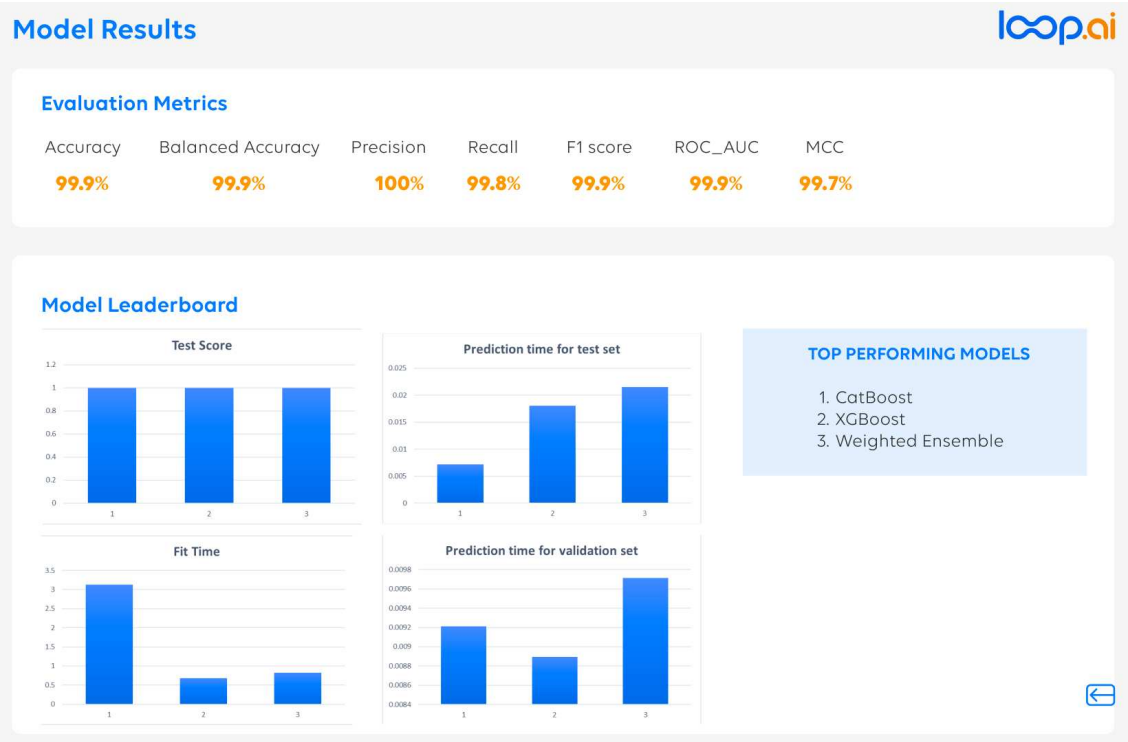
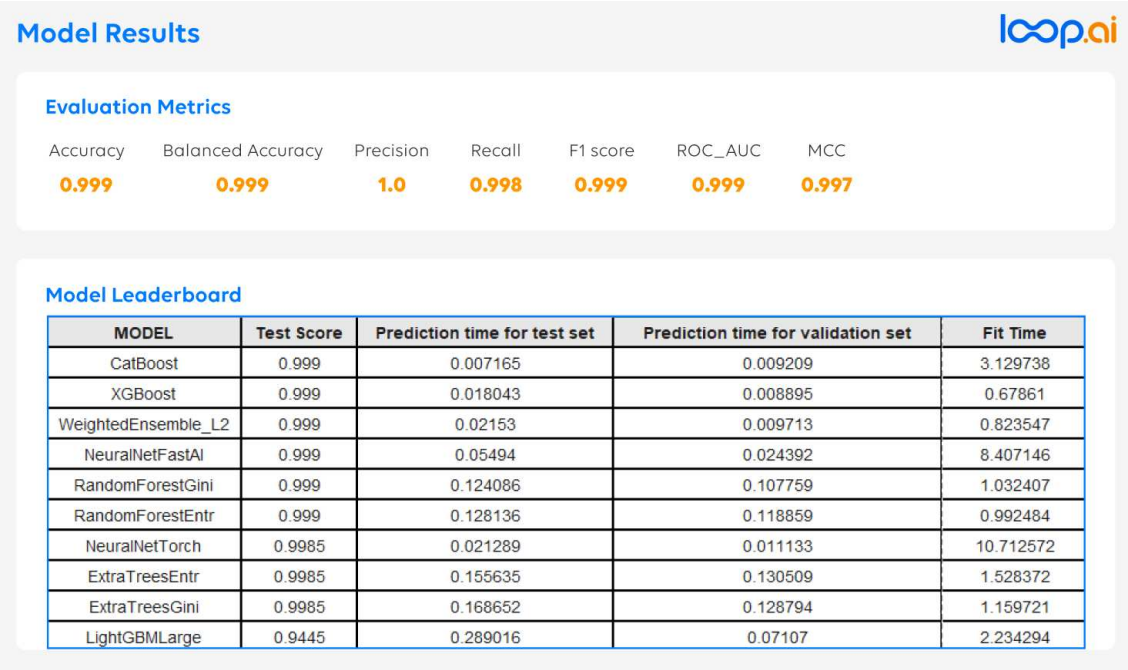


Figure 3.2: Sketch of results using Figma.



MODEL	Test Score	Prediction time for test set	Prediction time for validation set	Fit Time
CatBoost	0.999	0.007165	0.009209	3.129738
XGBoost	0.999	0.018043	0.008895	0.67861
WeightedEnsemble_L2	0.999	0.02153	0.009713	0.823547
NeuralNetFastAI	0.999	0.05494	0.024392	8.407146
RandomForestGini	0.999	0.124086	0.107759	1.032407
RandomForestEntr	0.999	0.128136	0.118859	0.992484
NeuralNetTorch	0.9985	0.021289	0.011133	10.712572
ExtraTreesEntr	0.9985	0.155635	0.130509	1.528372
ExtraTreesGini	0.9985	0.168652	0.128794	1.159721
LightGBMLarge	0.9445	0.289016	0.07107	2.234294

Figure 3.3: Sketch of metrics using Figma.

3.2 Backend for the AutoML Pipeline

The backend logic part of the system finds the most complex when handling the parameters and feeding them to AutoGluon. It is believed that the main backend logic scheme reduces complexity for users. What follows is the initial idea to identify the steps needed for backend logic:

- User Input Handling: The user interacts with the chatbot sending a text query.
- Task Identification: The LLM determines the task type (classification, regression or time series forecasting). If the task is not identified, then the system will request clarification.
- Parameters Extraction: The LLM identifies the dataset, the target variable and additional metrics (i.e. accuracy, precision, minimizing/maximizing F1-score etc.).
- Code Generation: The LLM generates the code to be executed by AutoGluon.
- Model Execution: The AutoGluon script is executed, performing automatic training, hyperparameter tuning, and evaluation.
- Results Presentation: Graphs, metrics and predictions are provided with proper readable format to the user.
- (Optional) Interpretation: The LLM provides insights into the model's performance and key findings.

From a more structural point of view, it is possible to see in Figure 3.4 the individual elements that enable the execution of the previous tasks. The green blocks identify the different inputs that come from the user, whether they are text queries or submitted parameters. Then, the yellow main function analyzes the query to LLM and the other files (i.e., CSV data set, image) to the AutoGluon part of the code. After this first step, LLM returns the information detected to the AutoGluon code (i.e. classification using tabular data, evaluation measurement derived from the request, etc.), then the AutoGluon code is generated, executed and the output returned with the appropriate formatting. There are some additional stages; in fact, LLM can output a text response to the user asking for more details, or it can output a simple response to the user's question if the user has not initiated the training phase.

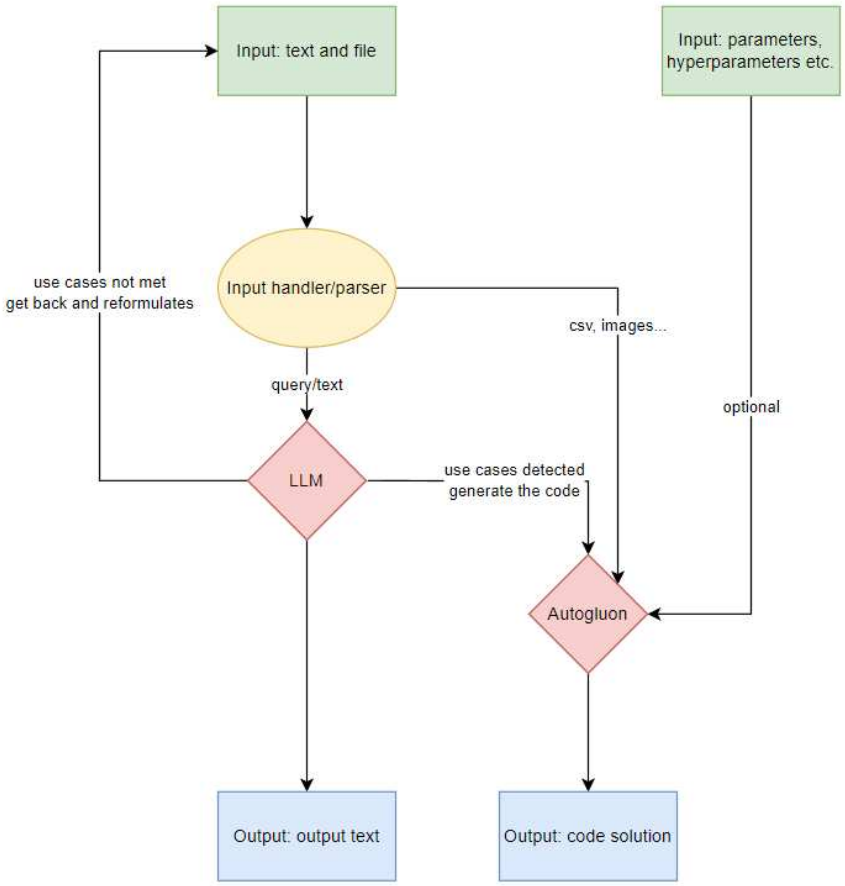


Figure 3.4: Sketch logic flow.

3.3 Structured Function Calls

Structured function calls, often referred to as Function Calling, are a powerful mechanism used in modern large language models (LLMs) to interact structuredly with external tools, APIs and services. For the implementation of GPT-4o, I use function calls because they provide a powerful and flexible way for OpenAI models to interface with code and external services. It offers two main use cases [21]:

- Fetching Data: Real time retrieval of information to be incorporated into the model's response. This is useful for searching knowledge bases and retrieving specific data from APIs (e.g. task and data format).

- Taking Action: Perform actions like submitting a form, calling APIs, modifying application state, or executing agentic workflow actions (such as handing off the conversation).

In simpler terms, instead of producing free-form text, the model returns a machine-readable function call, typically formatted as JSON, which can be processed and executed programmatically. In Figure 3.5, there are some examples of function calls used in this project, in which some parameters are always returned (e.g. task, data format, and evaluation metric), while all other parameters are identified and extracted only if present in the conversation with the user.

```
functions = [
    {
        "name": "extract_content",
        "description": "Extracts task, data_format, and eval_metric from user's query. If other parameters are present, then extract also those.",
        "parameters": {
            "type": "object",
            "properties": {
                "task": {
                    "type": "string",
                    "description": "The machine learning task among classification, regression or time_series_forecast."
                },
                "data_format": {
                    "type": "string",
                    "description": "The data format among tabular, multimodal or time_serifit_weighted_ensemblees"
                },
                "eval_metric": {
                    "type": "string",
                    "description": "The evaluation metric, for example accuracy, or some other else in the context of those allowed to be used inside AutoGluon code."
                ..........................
                "seed": {
                    "type": "string",
                    "description": "If specified, it's the parameter of the seed. Valid values are integers."
                },
                "clean_ckpts": {
                    "type": "string",
                    "description": "If specified, it's the parameter of the clean_ckpts. Valid values are true or false."
                }
            },
            "required": ["task", "data_format", "eval_metric"]
```

Figure 3.5: Example of Function calling used for parameters and metrics extraction.

The main advantages of using structured function calls are reliability and efficiency. By applying a defined structure, they help to reduce hallucinations in LLM responses and ensure that outputs match expected formats. This feature was the main reason I adopted the function call strategy to extract parameters and other useful metrics from queries without leaving space for hallucination output. Furthermore, this type of approach is especially valuable for applications such as chatbots, virtual assistants and AI-powered automation systems. Even if open-source models such as Llama are used, function calls are a suitable way to reliably identify and extract important information in a structured way. In fact, recent Llama models operate very similarly to OpenAI models, offering many advantages. [22] From the point of view of code, function call allows:

- Deterministic control: Transform natural text into a structured output through functions.
- Structured data extraction: Extract specific information from text efficiently.

- Development of custom functions: Connect the model to external APIs, databases, and internal tools.

3.4 Handler using gpt-4o

The `Handler` class is responsible for processing user requests to extract the essential machine learning parameters needed for model training. It uses a Large Language Model (LLM), especially gpt-4o, to interpret user inputs and to deduce key attributes such as machine learning tasks (classification, regression, or time series forecasts), data format (tabular, multimodal, or time series), and evaluation metrics (e.g. accuracy, balanced accuracy). The class interacts with OpenAI's API to analyze user-assisted queries, using structured function calls to extract additional parameters such as ensemble settings, hyperparameters and validation strategies. These extracted parameters are then stored in a JSON configuration file that serves as a basis for the training of automated models. In addition, a secondary class, `Handler_chat`, is implemented to facilitate user-assisted interaction, ensuring that the extracted parameters match the user's intention. Once the required parameters are identified, a dynamically generated Python script, which includes AutoGluon functions for the training of an automated model, is created. The script is then executed using `subprocess.run` and the results, including predictions, are displayed to the user. This approach allows an easy and user-friendly experience for individuals without previous knowledge in machine learning, as the system auto-identifies the best configurations and implements the training process with minimal manual intervention. Algorithm 3.1 shows the pseudo code of `Handler`.

Algorithm 3.1 Automated ML Pipeline with OpenAI and AutoGluon

```
Procedure ExtractParameters(usr_query, opt_parameters)
    Query sent to LLM
    Extract parameters like: task, data_format, eval_metric, others
    return parameters
End Procedure
Procedure ProcessDataset(ds_path)
    Load dataset from csv or xlsx
if error in loading
        Print error and exit
end if
End Procedure
if data_format is tabular
    Update config for tabular data
    Train AutoGluon TabularPredictor
else if data_format is multimodal
    Update config for multimodal data
    Train AutoGluon MultiModalPredictor
else if task is time-series forecasting
    Update config for time-series data
    Train AutoGluon TimeSeriesPredictor
end if
Generate and execute AutoGluon script
Print execution results
```

3.5 JSON parameters

The JSON configuration file can be seen in Algorithm 3.2 where key parameters are defined for different machine learning tasks and specifically for three data types: tabular, time series, and multimodal. The common section contains general settings, including a `time_limit` of 100 seconds for model training and an option to `save_space`, set to `true`. For tabular data, the primary target variable is specified during training (e.g. `class`), for example with `accuracy` as the evaluation metric. Hyperparameters for models such as `GBM`, `NN_TORCH`, `CAT`, and others are included. Additional settings, such as `fit_weighted_ensemble` and `auto_stack`, are enabled, with parameters for bagging (`num_bag_folds` set to 7) and stacking (`num_stack_levels` set to 1). A `holdout_frac` of 0.2 is used for validation, while options like `refit_full` and `keep_only_best` are also activated. For time series data, the target

variable is defined during training (e.g. `target`), with `MASE` as the starting evaluation metric. A `prediction_length` of 30 is specified, and it is possible to define models such as `DeepAR` and `Theta` in the configuration. In multimodal learning, the target variable is set during training (e.g. `AdoptionSpeed`), while `accuracy` is an example of evaluation metric that can be used. The `holdout_frac` is set to 0.3, with a random seed of 10 and a verbosity level of 1. This structured configuration provides the baseline parameters available at the beginning, but all the values are completely customizable by the user. The use of the JSON file allows flexible and optimized handling of different data formats, ensuring reliable settings for the three main cases.

Algorithm 3.2 JSON Configuration Pseudo-Code

```
Common Parameters:
    time_limit ← 100
    save_space ← True
Tabular Parameters:
    label ← "class"
    eval_metric ← "accuracy"
    hyperparameters ← {
        GBM: {
            extra_trees ← False
            learning_rate ← 0.02
            num_leaves ← 100
            feature_fraction ← 0.8
            min_data_in_leaf ← 4
        },
        NN_TORCH: {}, CAT: {}, XGB: {}, FASTAI: {},
        RF: {}, XT: {}, KNN: {}, LR: {}, AG_AUTOMM: {}
    }
    fit_weighted_ensemble ← True
    auto_stack ← True
    num_bag_folds ← 7
    num_bag_sets ← 3
    num_stack_levels ← 1
    holdout_frac ← 0.2
    refit_full ← True
    set_best_to_refit_full ← True
    keep_only_best ← True
Time Series Parameters:
    target ← "target"
    eval_metric ← "MASE"
    target_column ← ""
    prediction_length ← 30
    hyperparameters ← {
        DeepAR: {},
        Theta: {
            decomposition_type ← "additive"
            seasonal_period ← 1
        }
    }
Multimodal Parameters:
    label_column ← "AdoptionSpeed"
    eval_metric ← "accuracy"
    holdout_frac ← 0.3
    seed ← 10
    verbosity ← 1
```

3.6 Prompt Engineering for LLM Integration

Prompt engineering is a crucial step in ensuring that the LLM selected meets the expectations. In principle, prompt engineering is the process of structuring or creating instructions to produce the best possible output of a generative artificial intelligence (AI) model [23]. A prompt is a natural language text that describes the task that an AI should perform. A command for a text-to-text language model can be a query, a command, or a longer statement including context, instructions, and conversation history. Short-term engineering may involve phrasing an inquiry, specifying a style, selecting words and grammar [24], providing a relevant context, or describing a character for the AI to mimic. In communication with a text-to-image or text-to-audio model, a typical prompt is a description of the desired result, such as "a high-quality image of a horse-drawn astronaut" [25] or "Lo-fi slow BPM electro chill with organic samples" [26]. Introducing a text-to-image model may include adding, deleting, highlighting and reordering words to achieve a desired theme, style [27], layout, lighting [28], and aesthetics.

3.6.1 Why and how I perform prompt engineering

Prompt engineering is a necessity for AI engineers because it allows them to create better services, such as chatbots that can manage complex tasks like customer service. Today we have reached a point, in this world driven by big data, where training AI models can help deliver solutions much more efficiently without manually sorting large amounts of data. Proper prompt engineering can also help prevent or mitigate prompt injection attacks, i.e. hackers try to access and modify the logic behind the chatbot. The prompt engineering step took a long time to be completed for the web application, but some important instructions on how I was able to achieve the final behaviour are described here:

1. Making clear and short queries: Because generative AI is a deep learning model trained on data produced by humans and machines, it doesn't have the capability to sift through what I was communicating to understand the meaning. For example, instead of, "Try to identify the task and data format from the user's query" I obtained better results with, "Identify the task (among classification, regression, time series forecasting) and data format (among tabular, time series and multimodal) from the user's query."

2. Small variations are important: For each type of output, I experimented with generative AI by using different variations of the same request, by doing so I was able to understand whether I needed to include stuff like "don't use pleasantries". Instead after further test-

ing I found out that formulating the tone as "Try not to use too many pleasantries" was more in line with the expected behavior.

3. Include instructions: When I was able to get the output in the right format and tone, I began to set a limit on the number of words. The experimentation of different lengths allowed the model to be more speakative when responding, but also the limitation of the contextual dimension was important, as it allowed the model to "remember" different pieces of conversation.

4. Zero-shot prompting: This is the most direct and simplest method of prompt engineering in which a generative AI is simply given direct instruction or asked a question without being provided additional information. This is best used for relatively simple tasks rather than complex ones.

5. Few-shot prompting: This method involves supplying the generative AI with some examples to help guide its output. This method is more suitable for complex tasks than zero-shot prompting.

6. Chain-of-thought (CoT) prompting: This method helps improve an LLM's output by breaking down complex reasoning into intermediate steps, which can help the model produce more accurate results.

7. Prompt chaining: The prompter splits a complex task into smaller (and easier) subtasks, then uses the generative AI's outputs to accomplish the overarching task. This method can improve reliability and consistency for some of the most complicated tasks.

3.7 Database management with PostgreSQL

PostgreSQL is used as a relational database management system (RDBMS) to store and manage user interactions, datasets, machine learning models and training jobs. PostgreSQL is a free and open-source relational database management system that emphasizes extensibility and SQL compliance. PostgreSQL has atomic, consistent, isolated, durable (ACID) properties, automatic updating views, materialized views, triggers, foreign keys, and stored procedures. It supports all major operating systems, including Windows, Linux, macOS, FreeBSD and OpenBSD, and handles a variety of workloads ranging from single machines to data warehouses, data lakes, or web services with many simultaneous users. [29]

The database schema is designed with the following entities:

- Users: To store user credentials and metadata.
- Projects: Organizing datasets, models, and chat sessions under specific user-created projects.
- Datasets: To keep track of uploaded data files, including metadata and storage references.
- Models: To manage trained machine learning models with associated parameters.
- Training Jobs: To monitor the status and results of model training processes.
- Chat Sessions & Messages: To log user interactions with the chatbot for parameter setting and troubleshooting.

For the application, each entity is implemented as a separate table in PostgreSQL, with the relationship defined with foreign keys to connect to other entities. For example, the table `projects` contains a foreign key linking each project to a `user_id` in the table `users`, allowing the deletion of a project to also take actions in cascade with the associated entities contained. In addition, `datasets`, `models`, and `training_jobs` have foreign keys corresponding to `projects`, which ensures an appropriate and complete data organization. Indexes are used in frequently queried columns such as `username`, `email`, and `project_id` to optimize performance.

3.8 Framework development with Flask

Flask is a lightweight WSGI web application framework. It is designed to get started quickly and easy with the ability to scale to complex applications. [30] It is considered a Python-written micro-web framework and does not require specific tools or libraries. There is no database abstraction layer, form validation or other component where existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. [31] The Flask application is structured in a modular way, each module being represented as a folder. Each module contains the following key files:

- `routes.py`: Defines the HTTP routes for and associated request-handling logic to make the HTML pages work correctly.
- `__init__.py`: Initializes the module and registers it as part of the larger application.
- `forms.py` (if needed): Handles form definitions using Flask-WTF.

The main application directory includes:

- `modules.py`: Contains shared logic and helper functions used across different modules.
- `forms.py`: Defines global form structures (e.g. user authentication and project creation).
- `__init__.py`: Initializes the Flask application and registers Blueprints for modular integration.

To maintain a clean architecture, Flask's Blueprint feature is used to separate different functionalities into distinct modules. Each module represents a core functionality, such as:

- Authentication Module (**`auth`**): Manages user login, registration, and session handling.
- Project Creation (**`projects`**): Handles project creation, modification, and organization of datasets and models.
- Model Training (**`datasets`**): Manages model training.
- Prediction Page (**`models`**): Stores trained models, configurations, training results and it allows for model loading to be used for new predictions.

3.9 Final ChatBot powered by MistralAi-0.3v

The final chatbot is developed using the Mistral-7B-Instruct-v0.3 model, integrated into the Flask framework. The chatbot uses a large language model (LLM) based on a transformer to process user requests and extract relevant machine learning parameters. Implementation is designed for efficiency and accuracy, ensuring minimum latency in response and maintaining coherence in conversation history. Chat sessions are composed of sequential messages stored in PostgreSQL using an ORM model (`ChatMessage`). These messages are structured with user and assistant roles to manage the information submitted. The "prompt limit" limits the model from providing unnecessary suggestions, examples or explanations unless explicitly requested by the user, but also allows more colloquial style chat to support simplicity when explaining the user new technical problems. This ensures that the chatbot remains focused on providing support and guidance without deviating too much from the track, allowing for a bit of creativity when answering with examples or lists.

4
Data Types

This chapter is divided into three sections: section 6.1 which deals with tabular data and its applications. Section 6.2 describes the multimodal data and potential uses for different tasks. Section 6.3 contains the description and application of time series data for time series forecasting.

4.1 Tabular Data

The tabular is one of the most common data types and even if it sounds like a foreign term, there are countless occasions where someone has encountered it at least once, for example the following are all tabular data:

- Text emails are considered tabular data as they are made out of a list of messages where we can find attributes like: sender, subject line, main thread, body, and more.
- Music playlists can be tabulars as they contain different informations for each song like: song name, singer, duration, genre, and others.
- The everyday OS used to interact with a computer usually offers a tree like structure with folders. Each folder can contain multiple files, where each file has many attributes like: name, modification date, size, and others.

In statistics, tabular data refers to data organized in a table with rows and columns. Today, most Business Intelligence (BI) tools are able to generate such tables by automatically identify-

ing the relationships between database entries and structuring them accordingly. In the table, rows represent observations and columns represent attributes for these observations. [32] This structured format contrasts, for example, with visual data where both panels and graphics are used for better interpretation.

4.2 Multimodal Data

Multimodal data refers to data collected from different sources and formats and then combined for a more complete analysis. [33] The word *modality* simply means "the way or mode by which something happens, is experienced, or is captured." Therefore, multimodal means "experiencing or capturing something in several different ways". Humans experience the world in an inherently multimodal way. This is because usually all five senses are used to collect information about a particular situation and interpret it. In a similar way, multimodal data in AI involves the mixture of different data formats such as text, image, audio and video.

There are many examples of multimodal data and areas of application, here some of them are presented:

- Posts on social media usually combine images and text.
- Healthcare incorporates electronic health records (EHR), medical images like X-rays, radiology reports, and blood test results.
- AI-powered speech recognition systems use audio and textual data to improve intent identification among the two of them.
- Autonomous vehicles use a mix of camera footage, radar signals, and sensor data for navigation and safety.

Multimodal data enhances AI models by providing richer and more contextual information. Some benefits include:

- Improved Accuracy: Combining different data sources allows models to detect patterns that may be missed when analyzing a single data modality.
- Robustness to Noise: Models that leverage multiple data types can better handle noisy or missing information in one modality by relying on other available modalities.
- Deep Semantic Understanding: By linking multiple data types, models can extract deeper semantic relationships, enhancing applications such as visual question answering (VQA) and cross-modal retrieval.

- Expanding Applications: Multimodal learning has enabled advances in areas like medical image analysis, robotics, augmented reality, and large language models (LLMs) for text-to-image generation and other AI-driven tasks.

Multimodal datasets are particularly useful for computer vision applications. By integrating text, audio and depth information, these datasets improve tasks such as object detection, image classification and segmentation. In addition, they promote new applications such as:

- Enhancing large language models (LLMs) for human-like understanding.
- Improving AI-based diagnostics in medical imaging by combining radiology reports with scans.
- Enabling more immersive experiences in augmented reality and robotics.

4.3 Time Series Data

Time series data refer to a sequence of data points collected or recorded at consistent intervals over time. Unlike other types of data, time series data are uniquely characterized by their time order, and time itself is a critical variable. Analysis of these data, known as time series analysis, helps to uncover patterns, trends and dependencies within the data, providing valuable insights for various applications. In mathematics and statistics, a time series is defined as a series of data points indexed in time order, usually taken in consecutively equally spaced time intervals. Data can be discrete (e.g. daily stock prices) or continuous (e.g. temperature readings). A key feature of time series data is its dependence on historical values, which means that previous observations influence future observations. [34] Time series data are often visualized using a *run chart* chart, also known as a time line chart, which helps to identify trends, seasonality and cyclic patterns. To ensure reliability, time series analysis usually requires a large number of observations to reduce the impact of noise and accurately capture the underlying trends. Time series analysis is widely used in various areas where data fluctuates over time. Some common applications include:

- Finance: Stock price movements, interest rates, and automated trading algorithms.
- Economics: Industry forecasts, GDP growth trends, and inflation rates.

- Retail: Quarterly sales trends, demand forecasting, and customer purchasing behavior.
- Weather and Climate Science: Temperature readings, rainfall measurements, and climate change predictions.
- Medical and Health Sciences: Electrocardiogram (EKG) and electroencephalogram (EEG) monitoring for heart and brain activity analysis.
- Engineering and Signal Processing: Control system measurements and communications signal analysis.

One of the main advantages of time series analysis is its ability to predict future values based on historical data. Time series forecasting is an essential component of predictive analytics, enabling companies and researchers to anticipate future trends and make data-based decisions.

5

Use Cases and Applications

This chapter focuses on the presentation and description of the three specific use cases for which the web application is developed. Given that these examples are only representative examples used during development to test the correct functionality of the application, it does not exclude the use of the same with other datasets. In particular, the three datasets are used since each of them is a different type of data: tabular, multimodal, and time series. Through the tests with these datasets, I guarantee that the AutoML pipeline is effectively capable of handling the three data types.

5.1 Ad Click Prediction

The first use case is represented by a binary classification task whose goal is to predict whether a user will click on an online advertisement, depending on the position of the latter. In general, the purpose of this dataset is to monitor and determine user interactions with online ads based on specific factors such as demographics, browsing behaviour, ad display context and time of day. One of the main difficulties is that in order to achieve accurate predictions, it is necessary to clean up the data before applying machine learning models. Performing this task is usually difficult, especially due to the complexity and variability inherent in user interactions with online advertisements. The dataset can be useful for tasks such as advertising strategies, banner placements and data collection on user engagement with online ads. The dataset fields

are presented here:

- id: Identifier for each user.
- full_name: Anonymized user name formatted as ”UserX”.
- age: User’s age, ranging from 18 to 64 years.
- gender: Gender of the user (Male, Female, Non-Binary).
- device_type: Type of device used (Mobile, Desktop, Tablet).
- ad_position: Position of the ad on the webpage (Top, Side, Bottom).
- browsing_history: User’s browsing activity before seeing the ad (Shopping, News, Entertainment, Education, Social Media).
- time_of_day: Time when the ad was viewed (Morning, Afternoon, Evening, Night).
- click: Target variable (1 for a click, 0 for no click).

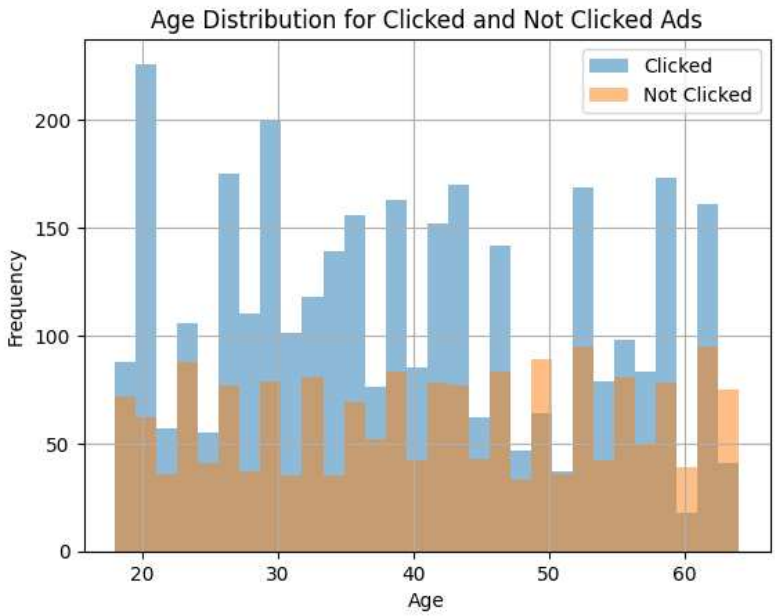


Figure 5.1: Distribution of ads clicked based on age.

5.2 Skin Cancer Detection

The second use case is a multi-label classification task aimed at diagnosing skin diseases based on dermatoscopic images. The set supports the training of machine learning models to classify images into several diagnostic categories. The dataset contains the following categories of skin diseases:

- Actinic keratoses and intraepithelial carcinoma / Bowen's disease (akiec)
- Basal cell carcinoma (bcc)
- Benign keratosis-like lesions (bkl)
- Dermatofibroma (df)
- Melanoma (mel)
- Melanocytic nevi (nv)
- Vascular lesions (vasc)

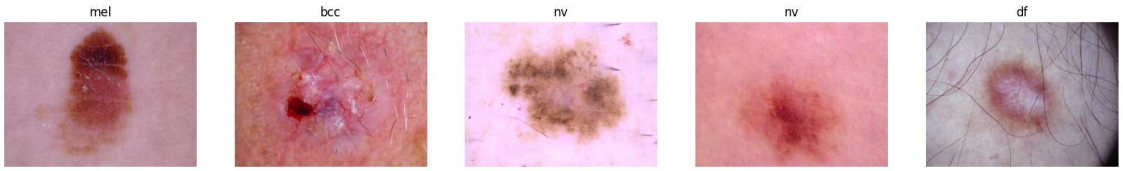


Figure 5.2: Examples of skin images from the Skin Cancer dataset.

The set includes 10,015 dermatoscopic images representing all major diagnostic categories for pigment lesions. These images were collected from different populations using different methods of acquisition to ensure diversity and robustness of training models. More than 50% of the lesions have been confirmed by histopathology, while the remaining cases depend on follow-up examinations, expert consensus, or confirmation of in vitro confocal microscopy. The dataset is divided into several files due to upload size limitations:

- HAM10000_images_part1.zip (5000 JPEG files)
- HAM10000_images_part2.zip (5015 JPEG files)

5.3 Sales Forecasting

The third use case is a time series forecast scenario, whose aim is to predict future furniture sales based on historical data. The challenge includes utilizing time series data from the superstore dataset to make accurate predictions for the next year. For a retail furniture store, forecasting future sales is crucial to avoid inventory problems such as over-stocking or under-stocking. By analyzing historical sales data, stores can optimize stock levels, improve customer experience, and ensure sustainability.

- Order Date: The date when the order was placed.
- Ship Date: The date when the order was shipped.
- Ship Mode: The shipping method used for delivery.
- City: The city where the order was placed.

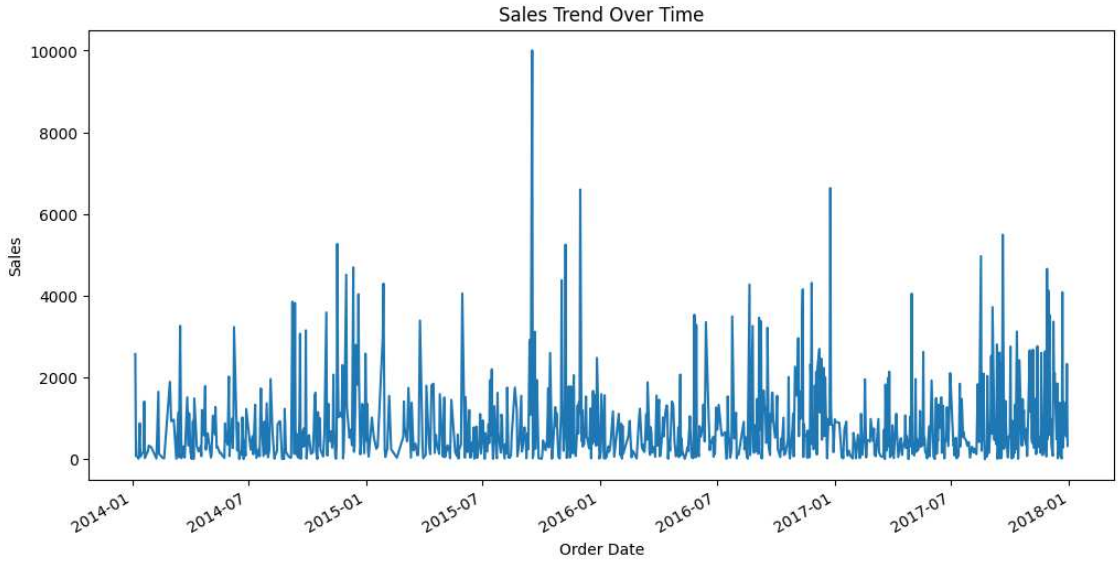


Figure 5.3: Depiction of trend for sales over time.

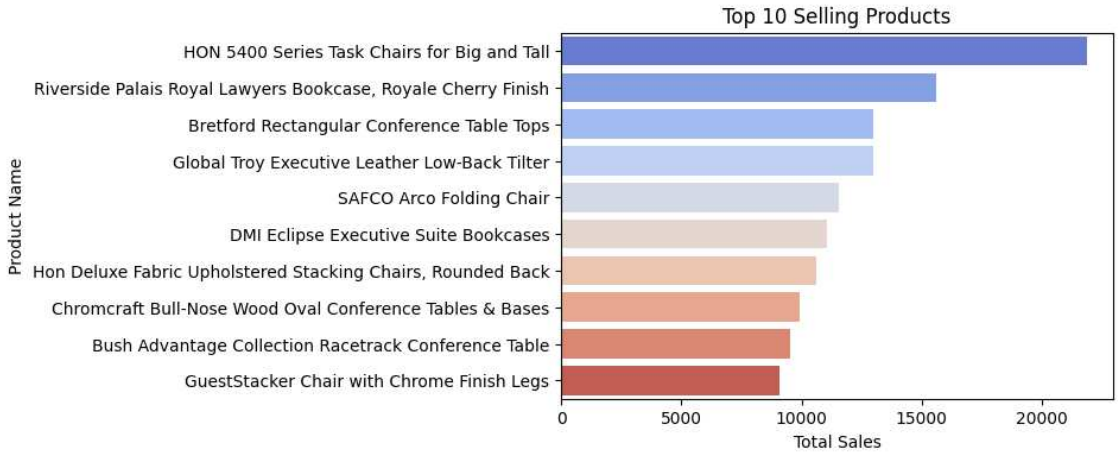


Figure 5.4: Example of top 10 products sold.

This dataset is used to develop models that accurately predict furniture sales for the next year. This can help retailers optimize stock management, minimize losses and improve overall operational efficiency. Machine learning models can be trained on this dataset to identify sales patterns and trends, which leads to better decision-making in retail operations.

6 Evaluation and Results

6.1 Performance Metrics

Model evaluation is the process that uses certain parameters to help analyze the general performance of a model. [35] It is an approach to assess and visualize if the model was able to learn the patterns or failed. Since modeling is a multi-step process, it is important to check whether the model is making future predictions and analyzing its shortcomings. In order to perform the model assessment, there are many metrics that can be used, but also the implementation of cross validation during training is considered a step of assessment. For assessing the performance of the model I use different metrics depending on the task performed, for example, accuracy and balanced accuracy are used for classification tasks in the case of tabular data, but only accuracy is available for assessing the performance of the model in case of multimodal data, whereas for time series forecasting I generally implement a mixture of mean absolute scaled error (MASE), mean absolute error (MAE) and mean absolute percentage error (MAPE), but many more metrics are available.

6.1.1 Metrics used for Classification

Accuracy measures the proportion of correctly classified instances among all instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6.1}$$

Precision quantifies the number of correctly predicted positive observations relative to the total predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{6.2}$$

Recall (or Sensitivity) measures the ability of the model to detect positive instances:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{6.3}$$

F1-score is the harmonic mean of precision and recall:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6.4}$$

6.1.2 Metrics used for Regression

R-squared (R^2) Score represents the proportion of variance explained by the model:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}. \tag{6.5}$$

Mean Squared Error (MSE) measures the average squared difference between actual and predicted values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{6.6}$$

Mean Absolute Scaled Error (MASE) is a measure of forecast accuracy:

$$\text{MASE} = \frac{\frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t|}{\frac{1}{T-1}\sum_{t=2}^{T}|y_t - y_{t-1}|}. \tag{6.7}$$

Mean Absolute Error (MAE) is the sum of absolute errors divided by the sample size:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{6.8}$$

Mean Absolute Percentage Error (MAPE) is defined as:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{6.9}$$

6.2 Evaluation of tabular datasets

This section presents an assessment of different machine learning models applied to different datasets, covering classification and regression tasks for the specific tabular format of the data. The following are the details of each set of data and the corresponding model performance metrics.

6.2.1 Email Spam Classification

This is a csv file containing 83446 e-mail records that are labelled as spam or non-spam. It was created by combining the TREC Public Spam Corpus and the Enron-Spam Dataset in 2007. [36] Here are some useful information about the database and the task to be performed:

- Columns: label, text
- Target Column: label ('1' - spam; '0' - ham)
- Task: Binary Classification

The results obtained can be seen in Figure 6.1 where it is possible to see that the model can achieve high values in terms of accuracy, precision, recall and F1 score, which means that the task is correctly captured by the model. Figure 6.2 shows the confusion matrix, indicating that only a total of 134 elements are misclassified.

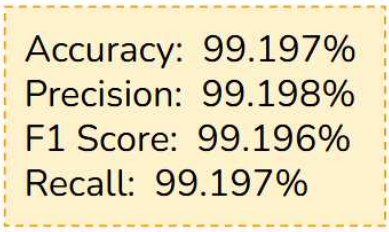


Figure 6.1: Model results for email spam classification.

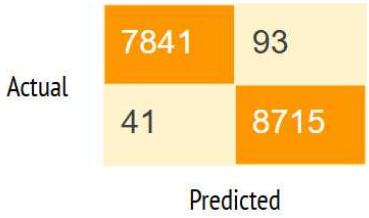


Figure 6.2: Confusion matrix for email spam classification.

6.2.2 Ad Click Prediction

The dataset provides insights into user behaviour and online advertising, especially in order to predict whether a user will click an online advertisement. It contains user demographic information, browsing habits and details related to advertising displays. [37] Useful information about the dataset and the tasks performed are listed here:

- Columns: id, full_name, age, gender, device_type, ad_position, browsing_history, time_of_day, click
- Target Column: click ('1' - click; '0' - no click)
- Task: Binary Classification

The results can be consulted by examining Figure 6.3 and Figure 6.4, where, mainly due to the small dimensionality of the dataset, the model is able to classify all 1700 samples with perfect conformity.

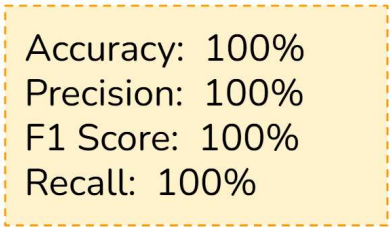


Figure 6.3: Model results for ad click classification.

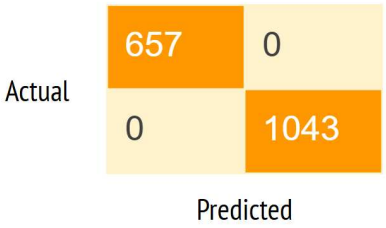


Figure 6.4: Confusion matrix for ad click classification.

6.2.3 Flipkart Reviews Sentiment Analysis

The database consists of reviews of the products customers have bought from Flipkart. The reviews are the experience of the product customer purchased through Flipkart and the rating of this product. [38] Some useful information about the database is listed here:

- Columns: Product Name, Review, Rating
- Target Column: Rating (1 to 5 stars)
- Task: Multiclass Classification

The model's performance is still discrete, as seen in Figure 6.5, and the values for accuracy, accuracy, recall, and F1-core are more than 80%. The values are a little lower than other examples because the dataset is quite small, in fact, in Figure 6.6 the confusion matrix shows that only a few values are generally misclassified, while there is a chunk of incorrectly classified samples represented by the number 48. Probably due to the lack of data, the model is not as good as in other cases.

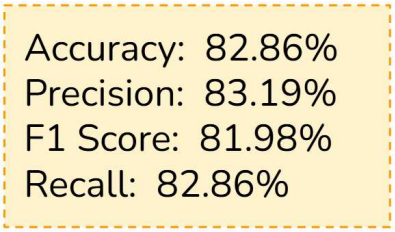


Figure 6.5: Model results for flipkart classification.

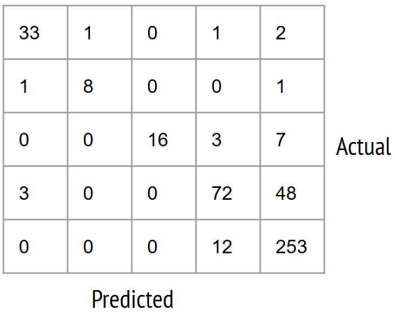


Figure 6.6: Confusion matrix for flipkart classification.

6.2.4 UK Key Stage Readability for English Texts

This dataset was generated by literature in the public domain of Project Gutenberg and reprinted by the UK key stages equivalents of the Lexile Reading Framework. The dataset contains a total of 20,000 lines distributed evenly through four educational phases. [39] Some useful information is listed here:

- Columns: text, linguistic features
- Target Column: UK_Key_Stage (Key Stage 2 (KS2), Key Stage 3 (KS3), Key Stage 4 (KS4), Key Stage 5 (KS5))
- Task: Multiclass Classification

It is also clear, also by looking at Figure 6.7, that the model is capable of reaching high values in terms of accuracy, precision, memory, and F1 score, which means that it captures most of the variability of the dataset consistently. Figure 6.8 also shows that only 51 samples were misclassified during the testing.

Accuracy: 98.40%
Precision: 98.50%
F1 Score: 98.40%
Recall: 98.40%

Figure 6.7: Model results for english texts classification.

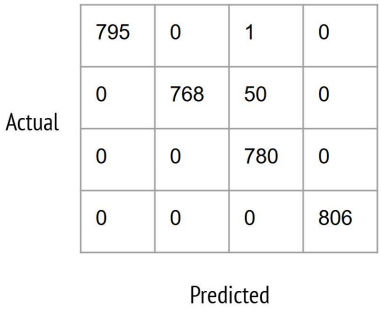


Figure 6.8: Confusion matrix for english texts classification.

6.2.5 Diabetes Prediction

The data were collected by the Iraqi society, since they were collected from the laboratory of the Medical City Hospital and (the specialized center for endocrinology and diabetes-Al-Kindy Teaching Hospital). Patients' files were extracted and the data extracted from them were entered into the database to build the diabetes dataset. [40] The data are composed of medical information and laboratory analysis, other information is presented here:

- Columns: No. of Patient, Sugar Level Blood, Age, Gender, Creatinine ratio (Cr), Body Mass Index (BMI), Urea, Cholesterol (Chol), Fasting lipid profile (Total, LDL, VLDL, Triglycerides (TG), HDL Cholesterol), HBA1C, Class
- Target Column: Class (Diabetic, Non-Diabetic, Predict-Diabetic)
- Task: Multiclass Classification

According to Figure 6.9, the results for all metrics are very high. The confusion matrix in Figure 6.10 confirms the high accuracy, accuracy, recall and F1 score values, showing that only two samples are incorrectly classified.

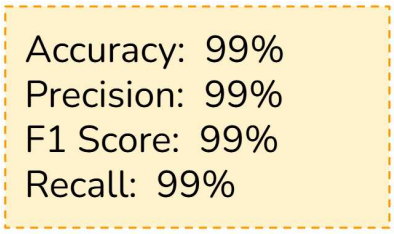


Figure 6.9: Model results for diabetes classification.

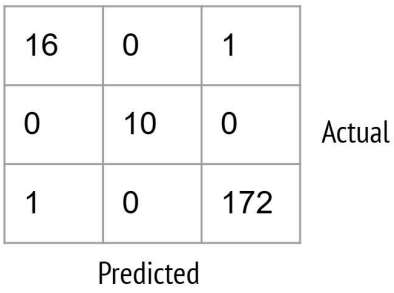


Figure 6.10: Confusion matrix diabetes classification.

6.2.6 Walmart Sales Prediction

This dataset contains sales data from one of the world's largest retailers. These data are useful because they can be used to test and evaluate if certain factors, such as fuel prices or air temperatures, affect sales of such large companies. [41] Here we list some useful information about the dataset and the tasks to be performed:

- Columns: Store, Date, Weekly_Sales, Holiday_Flag, Temperature, Fuel_Price, CPI, Unemployment
- Target Column: Weekly_Sales
- Task: Regression

The results shown in Figure 6.11 show a high value for the R^2 point, which means that the model is able to capture most of the variance, but also means that the model is too suitable for the data. Looking at the MSE of 4.14 billion means that the average square difference between actual and projected sales is high, which can be due to the high variation in sales data, but also most likely because the data has a large scale. The MAE of 39,021.83 means that, on average, the forecasts of the model are wrong by this value in the weekly sales per store. Knowing that

weekly sales usually have a value greater than one million, a MAE of about 40.000 is not bad at all.

```
R2 Score:  98.75%
Mean Squared Error:  4144722122.52
Mean Absolute Error:  39021.83
```

Figure 6.11: Model results for walmart regression.

6.2.7 Medical Insurance Cost Prediction

The medical insurance database includes various factors that influence medical expenditure. This datasets serve as a basis for training machine learning models capable of forecasting medical expenses for new insurers. Its goal is to lighten up the essential elements contributing to higher insurance costs and help companies to make more informed decisions regarding pricing and risk assessment. [42] Here we have some useful information:

- Columns: Age, Sex, BMI, Children, Smoker, Region, Charges
- Target Column: Charges
- Task: Regression

If we take a look at Figure 6.12, we can see that the model's performance in this dataset is strong, as the R^2 mark is high, as is the previous case of Walmart sales. The MSE is in the range of millions and could show large errors, but fortunately the MAE is only 780 and knowing that the value for the insurance price varies within the dataset from only one thousand to more than fifty thousand, this final error is not bad for high prices, but for low prices it is more harmful.

```
R2 Score:  97.8%
Mean Squared Error:  2995118.03
Mean Absolute Error: 780.05
```

Figure 6.12: Model results for medical regression.

6.2.8 Temperature and Ice Cream Sales

This dataset contains different temperatures at which ice cream has been sold and its corresponding income. It began as a Google Data Analytics project, but then some correlation patterns between temperature and ice cream sales revenue were detected. In simpler terms, higher temperatures usually mean more income, which is quite realistic, but the data were further expanded to perform a deeper analysis. [43] Here we list some useful information:

- Columns: Temperature, Ice Cream Profits
- Target Column: Ice Cream Profits
- Task: Regression

The results are quite good, in fact in Figure 6.13 one can see a high R^2 score that means a high adaption of the model to data variation, a MSE of 8.845 that is relatively low and a particularly good MAE of 2.385 that shows a very small error if the prices are taken into account from 13 to 80 dollars.

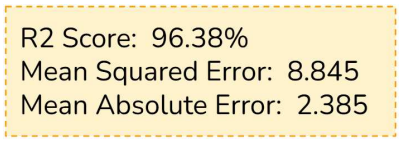


Figure 6.13: Model results for ice cream and temperature regression.

6.3 Evaluation of multimodal datasets

This section presents an assessment of different machine learning models applied to different datasets, covering classification and regression tasks for the specific multimodal format of data. The following are the details of each dataset and the corresponding model performance metrics.

6.3.1 Breast Cancer Detection

The dataset contains features calculated from a digitalized image of a fine needle aspirate (FNA) of breast mass. [44] They describe the features of the cell nuclei present in the image. Here are some useful information about the dataset:

- Columns: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal dimension
- Target Column: (M = malignant, B = benign)
- Task: Binary Classification

In Figure 6.14, it can be seen that overall performance in terms of accuracy, accuracy, memory and F1 score are harmful, which means that the model is able to accurately capture the characteristics of breast cancer. Also looking at the confusion matrix in Figure 6.15 there are only 3 erroneous sample classification, which again confirms the high capacity of the model, especially if one considers that it was only trained on around 600 samples.

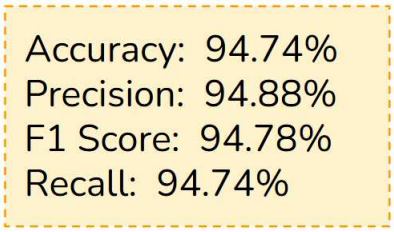


Figure 6.14: Model results for binary classification on breast cancer.

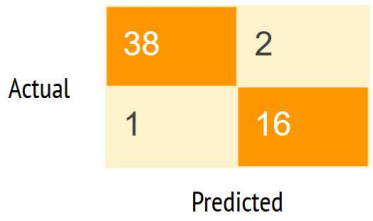


Figure 6.15: Confusion matrix of the results for breast cancer.

6.3.2 Wine Classification

In the dataset, the data are the result of a chemical analysis of wines grown in the same region of Italy, but derived from three different producers. The analysis determined the amounts of 13 components found in each of the three types of wines. [45] Here are some information about the dataset:

- Columns: Alcohol, Malic acid, Ash, Alkalinity of ash, Magnesium, Total phenols, Flavonoids, Non Flavonoid phenols, Proanthocyanidins, Color intensity, Proline.
- Target Column: Class (1, 2, 3)
- Task: Multiclass Classification

In Figure 6.16, overall accuracy, accuracy, memory, and F1 score are high, meaning that the model correctly captures the three different wine classes based on the characteristics contained in the dataset. Looking at the confusion matrix of Figure 6.17, it is clear that only one sample is wrongly classified while 17 are correctly classified in the appropriate classes.

Accuracy: 94.44%
Precision: 95.55%
F1 Score: 94.53%
Recall: 94.44%

Figure 6.16: Model results for multiclass classification on wine.

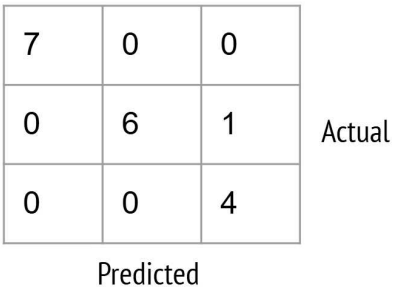


Figure 6.17: Confusion matrix of the results for wine.

6.4 Evaluation of time series datasets

This section presents an assessment of different machine learning models applied to different datasets in the performance of time series forecasting tasks. The following details of each dataset and the corresponding model performance metrics are given below.

6.4.1 M4 Hourly

The M4 dataset is a collection of 100,000 time series used for the fourth edition of the Makridakis Forecasting Competition. The M4 database consists of a series of annual, quarterly, monthly, and other (weekly, daily, and hourly) datasets, divided into training and testing sets. The M4 dataset was created by selecting a random sample of 100,000 time series from the ForeDeCk database. [46] Here I am listing some useful information about the dataset:

- Columns: item_id, timestamp, target
- Target Column: target
- Task: Time series forecast

The results can be seen in Figure 6.18 with a value of 0.855 for the MASE showing that the model can achieve greater accuracy compared to a naive approach, while MAE of only 0.1085 is a good result with respect to data scale, and finally MAPE below 1% shows that the model is very accurate with its forecasts.

MASE: 0.855
Mean Absolute Error: 0.1085
Mean Absolute Percentage Error: 0.0054

Figure 6.18: Model results for time series forecasting using M4 dataset.

6.4.2 Electric Production

This dataset contains electricity generation over time for a general application. It represents other similar datasets that can be used to forecast energy production or consumption based on time series forecasts. [47] Here I list some context information about the dataset:

- Columns: DATE, IPG2211A2N
- Target Column: IPG2211A2N (Electric production)
- Task: Time series forecast

In Figure 6.19, the results show that the MASE remains below 1, indicating that the model is better than a naive approach, while the MAE of 7.871 indicates that there is an average slight variation in the actual predicted values, which is acceptable, while the MAPE of 7.86% suggests that the predicted values differ slightly from the actual values. The model is quite good, and small errors could arise from things not captured such as seasonality, sudden fluctuations or external factors (e.g. economy, wheat, political changes).

MASE: 0.861
Mean Absolute Error: 7.871
Mean Absolute Percentage Error: 0.0786

Figure 6.19: Model results of forecasts for the energy production using the electric dataset.

6.4.3 Daily Minimum Temperatures

This dataset is similar to the previous in the sense that it serves as a sample of the similar dataset category that can be used to forecast daily temperatures when measured in a time series. [48] Here some information about the dataset is listed:

- Columns: Date, Daily minimum temperatures
- Target Column: Daily minimum temperatures

- Task: Time series forecast

Taking a look at the results of Figure 6.20, it is clear that the model performs better than a naive approach, as suggested by MASE, then the MAE of 1.99 is acceptable for most values, while it becomes harmful for smaller values, and finally the MAPE of 16% shows that, on average, forecasted values can vary by a different amount. While MAPE and MAE may not provide such good performance, it is expected that daily temperatures will vary significantly more than the stable process of energy production previously seen. Finally, I achieve good results for the majority of temperatures above 10 degrees both in terms of MAPE and MAE, while for smaller values close to 5, the performance is a little worse.

MASE: 0.726
Mean Absolute Error: 1.99
Mean Absolute Percentage Error: 0.160

Figure 6.20: Model results of forecasts for the daily temperatures.

6.4.4 Microsoft Stock

The dataset contains Microsoft stock information from April 2015 to April 2021, and the data was acquired using the command GOOGLEFINANCE'. It is a highly descriptive dataset that can be used to test the time series forecasting capabilities of different models. [49] Here some useful information is listed:

- Columns: Date, Open, High, Low, Close, Volume
- Target Column: Volume (Number of shares traded on that day)
- Task: Time series forecast

In Figure 6.21 the results indicate that the model is much better than the naive approach with a MASE of 0.659, while the MAPE of 23.5% is not very ideal, as it suggests that on average the prediction by this percentage deviates from the real values. Prediction of stocks is very difficult because fluctuations depend on many different factors, so for the time being, these results are sufficient for the tested model.

MASE: 0.659
Mean Absolute Percentage Error: 0.235

Figure 6.21: Model results of forecasts for the microsoft stocks.

6.4.5 Bitcoin Trade Trends

This dataset contains approximately the data of the last eight years of the most famous crypto "Bitcoin". It is a complex task to predict values of such cryptocurrencies and, like the previous Microsoft dataset, I expect the model to have some difficulties in correctly predicting values. [50] Here some information about the dataset are listed:

- Columns: Date, Open, High, Low, Close, Adj Close, Volume
- Target Column: Volume (volume of trade)
- Task: Time series forecast

The model is highly capable, as shown in Figure 6.22, with a MASE of only 0.546, while the MASE of around 20% indicates that the forecasts on average deviate by this percentage from the actual values. The problem with having such a MASE could come from some volatility problems of crypto currencies, in fact an error of 20.8% is not unusual for this type of data.

MASE: 0.546
Mean Absolute Percentage Error: 0.2080

Figure 6.22: Model results of forecasts for the bitcoin volumes.

6.4.6 Household Energy

The dataset contains data collected from an apartment in San Jose for a year plus. The data were collected with smart meters and shared by the energy company. [51] Here I present some information about the dataset:

- Columns: TYPE, DATE, START TIME, END TIME, USAGE, UNITS, COST, NOTES
- Target Column: USAGE
- Task: Time series forecast

Figure 6.23 suggests that the MASE of 0.637 below 1 significantly exceeds a naive approach, with a MAE of only 0.02 should be taken into account carefully because for many values it could be harmful, but with a MEAN of only 0.717% the model is proved to be powerful since, on average, the predicted values deviate by less than 1% from the actual values.

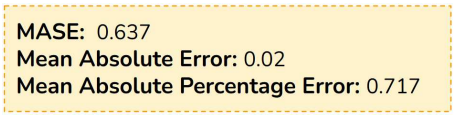


Figure 6.23: Model results of forecasts for house energy consumptions.

7 Conclusion

Finally, this thesis has explored the complex world of automatic machine learning and made it available to non-expert users. I was confronted with the challenges of three different data types and their correct handling, resulting in an almost fully automated application that allows the creation of machine learning models. I began with a discussion about the reasons for the need for machine learning models in modern times, focusing especially on existing technologies that implement a form of AutoML. Afterwards, I presented a thorough review of the specifications required for the application to meet certain general use cases, also discussing the technicality of each (i.e. table, time series, multimodal). The development was presented by describing and showing the different modules and their evolution during the process, especially with emphasis on handlers for the implementation of LLM and the techniques applied to make it work in the way I wanted. The final part showed some results and measurements obtained by testing the application with different data sets. Therefore, I can affirm that, through the application of advanced automated models using the AutoGluon libraries and a refined implementation of a Mistralai chatbot, I was able to design an adaptation system that proves to be a reliable and easy-to-use tool that empowers users to create and use machine learning models. Several improvements could still be made in the future. New types of data such as audio and video could be added to extend the information used by models, as well as capture features from a wider range of data. More implicit preprocessing methods could be applied to solve some data processing problems that occur when using very specific encoded data sets such as those that use latin-1" as encoding. Add the possibility for the user to be guided and manually perform

some data preprocessing such as deleting or adding columns/rows, changing names, handling missing values with input of mean/median/mode and similar. In addition, the inclusion of voice-to-text and text-to-speech functions when interacting with the chatbot would improve accessibility and provide a more user-friendly experience, especially for those who prefer audio communication or who have accessibility requirements. In addition to these improvements, further optimizations in model selection and hyperparameter adjustment could be introduced to increase the efficiency and effectiveness of the AutoML system. The expansion of the integration of more advanced natural language processing techniques (NLPs) could enable chatbots to better understand and guide users through the machine learning process, making modeling more intuitive. The inclusion of real-time feedback mechanisms to provide insight into model performance and potential improvements would be another valuable addition. Furthermore, taking into account the rapid evolution of deep learning and transformer-based models, future iterations of this application could include more complex architectures to improve predictions and recommendations. The system's adaptability could be improved by enabling continuous learning, where models update dynamically on the basis of new data without the need for a complete retraining process. This would make the system more robust and applicable to real-world, constantly changing data sets. Overall, although this paper has successfully established a functional and accessible AutoML solution, the potential for further improvements remains enormous. With the integration of new data modalities, the refining of preprocessing techniques, the improvement of user interaction through voice functions, and the use of more advanced machine learning strategies, the application could continue to evolve into an even more powerful tool to democratize AI and make machine learning accessible to all users.

References

[1] J. Bell, "What is machine learning?" in *Book Title Unknown*, S. Carta, Ed. Wiley, 2022, ch. 9.

[2] M. Awad and R. Khanna, "Machine learning in action: Examples," in *Efficient Learning Machines*. Springer, 2015, pp. 209–240.

[3] LoopAI Group. Our mission. (accessed: 11.02.2025). [Online]. Available: https://www.loop.ai/company

[4] T. Spears and K. Bondo Hansen, "The use and promises of machine learning in financial markets," in *The Oxford Handbook of the Sociology of Machine Learning*. Oxford University Press, Dec. 2023.

[5] F. Hutter, L. Kotthoff, and J. Vanschoren, *AutoML: Methods, Systems, Challenges*, 1st ed. Springer, 2019.

[6] Wikipedia Contributors. Automated machine learning. (accessed: 09.02.2025). [Online]. Available: https://en.wikipedia.org/wiki/Automated_machine_learning

[7] M. Kang and J. Tian, *Machine Learning: Data Pre-processing*, 1st ed. Wiley, 2018.

[8] E. Kasneci, K. Sessler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, T. Seidel, M. Stadler, J. Weller, J. Kuhn, and G. Kasneci, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[9] Y. Gu, H. You, J. Cao, and M. Yu, "Large language models for constructing and optimizing machine learning workflows: A survey," *arXiv preprint arXiv:2411.10478v1*, 2025.

[10] S. Ajgaonkar, *Practical Automated Machine Learning Using H2O.ai*, 1st ed. <packt>, 2022.

[11] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-sklearn 2.0: Hands-free automl via meta-learning," *Journal of Machine Learning Research*, vol. 23, no. 261, 2022, (accessed: 09.02.2025).

[12] B. Mota, P. Faria, and C. Ramos, "Automated machine learning and explainable artificial intelligence in predictive maintenance: An mljar framework review," in *Distributed Computing and Artificial Intelligence, 21st International Conference*, R. Chinthaginjala, P. Sitek, N. Min-Allah, K. Matsui, S. Ossowski, and S. Rodríguez, Eds. Cham: Springer Nature Switzerland, 2025, pp. 150–159.

[13] W. Qi, C. Xu, and X. Xu, "Autogluon: A revolutionary framework for landslide hazard analysis," *Natural Hazards Research*, vol. 1, no. 3, pp. 103–108, 2021.

[14] M. Bahrami, J. Park, L. Liu, and W.-P. Chen, "Api learning: Applying machine learning to manage the rise of api economy," in *Companion Proceedings of The Web Conference 2018 (WWW '18)*. ACM, 2018, pp. 151–154.

[15] L. Jansen, "Evaluating automl performance of sagemaker autopilot and transmogrifai," in *Proceedings of Hochschule Darmstadt - University of Applied Sciences*, Darmstadt, Germany, 2025.

[16] O. Topsakal and T. C. Akinci, "Creating large language model applications utilizing langchain: A primer on developing llm apps fast," in *Proceedings of the Computer Science Department, Florida Polytechnic University*, 2025.

[17] "Integrating ai services into semantic kernel: A case study on enhancing functionality with google palm and large language models," *Transactions on Open Source Software Projects*, vol. 1, no. 1, p. Article 2, 2024.

[18] dust. dust github repository. (accessed: 09.02.2025). [Online]. Available: https://github.com/dust-tt/dust

[19] H. Yang, S. Yue, and Y. He, "Auto-gpt for online decision making: Benchmarks and additional opinions," *arXiv preprint*, 2023.

[20] M. Fırat and S. Kuleli, "What if gpt-4 became autonomous: The auto-gpt project and use cases," *Journal of Emerging Computer Technologies*, vol. 3, no. 1, pp. 1–6, 2023.

[21] I. Abdelaziz, K. Basu, M. Agarwal, S. Kumaravel, M. Stallone, R. Panda, Y. Rizk, G. Bhargav, M. Crouse, C. Gunasekara, S. Ikbal, S. Joshi, H. Karanam, V. Kumar, A. Munawar, S. Neelam, D. Raghu, U. Sharma, A. M. Soria, D. Sreedhar, P. Venkateswaran, M. Unuvar, D. Cox, S. Roukos, L. Lastras, and P. Kapanipathi, "Granite-function calling model: Introducing function calling abilities via multi-task learning of granular tasks," *arXiv preprint*, 2024.

[22] W. Liu, X. Huang, X. Zeng, X. Hao, S. Yu, D. Li, S. Wang, W. Gan, Z. Liu, Y. Yu, Z. Wang, Y. Wang, W. Ning, Y. Hou, B. Wang, C. Wu, X. Wang, Y. Liu, Y. Wang, D. Tang, D. Tu, L. Shang, X. Jiang, R. Tang, D. Lian, Q. Liu, and E. Chen, "Toolace: Winning the points of llm function calling," *arXiv preprint*, 2024.

[23] T. F. Heston and C. Khun, "Prompt engineering in medical education," *International Medical Education*, vol. 2, no. 3, pp. 198–205, 2023.

[24] J. P. Wahle, T. Ruas, Y. Xu, and B. Gipp, "Paraphrase types elicit prompt engineering capabilities," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, 2024, pp. 11 004–11 033.

[25] W. D. Heaven, "This horse-riding astronaut is a milestone on ai's long road towards understanding," *MIT Technology Review*, 2022.

[26] K. Wiggers, "Meta open sources an ai-powered music generator," *TechCrunch*, 2023.

[27] M. Diab, J. Herrera, and B. Chernow, "Stable diffusion prompt book," PDF, October 28 2022.

[28] claid.ai, "How to write ai photoshoot prompts: A guide for better product photos," June 12 2023.

[29] B. Momjian, *PostgreSQL: Introduction and Concepts*. Boston: Addison-Wesley, 2004, second Printing, February 2001.

[30] Miguel Grindberg, *Flask Web Development*. O'reilly, 2018, second Edition, March 2018.

[31] Matt Copperwaite and Charles Leifer, *Learning Flask Framework*. Birmingham: PACKT, 2015, first Published, November 2015.

[32] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Journal of Artificial Intelligence Research*, 2021.

[33] K. Sharma and M. Giannakos, "Multimodal data capabilities for learning: What can multimodal data tell us about learning?" *British Journal of Educational Technology*, vol. 51, no. 5, p. 12993, 2020.

[34] T. chung Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, pp. 164–175, 2010.

[35] Y. Reich and S. Barai, "Evaluating machine learning models for engineering problems," *Artificial Intelligence in Engineering*, vol. 13, no. 3, pp. 301–317, 1999.

[36] P. Singhvi, "Spam email classification dataset," Available at Kaggle: https://www.kaggle.com/datasets/bayes2003/emails-for-spam-or-ham-classification-trec-2007, 2024.

[37] C. Marius, "Ad click prediction dataset," Available at Kaggle: https://www.kaggle.com/datasets/marius2303/ad-click-prediction-dataset, 2024.

[38] HarishEdison, "Flipkart reviews sentiment analysis," Available at Kaggle: https://www.kaggle.com/datasets/harishedison/flipkart-reviews-sentiment-analysis, 2022.

[39] J. J. Bird, "Uk key stage readability for english texts," Available at Kaggle: https://www.kaggle.com/datasets/birdy654/uk-key-stage-readability-for-english-texts, 2024.

[40] AravindPCoder, "Diabetes dataset," Available at Kaggle: https://www.kaggle.com/datasets/aravindpcoder/diabetes-dataset, 2024.

[41] Mikhail, "Walmart sales," Available at Kaggle: https://www.kaggle.com/datasets/mikhail1681/walmart-sales/data, 2024.

[42] M. R. Vyas, "Medical insurance cost prediction," Available at Kaggle: https://www.kaggle.com/datasets/rahulvyasm/medical-insurance-cost-prediction/data, 2024.

[43] rephy, "Temperature and ice cream sales," Available at Kaggle: https://www.kaggle.com/datasets/raphaelmanayon/temperature-and-ice-cream-sales, 2024.

[44] U. M. Learning and . collaborator, "Breast cancer wisconsin (diagnostic) data set," Available at Kaggle: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29, 2017.

[45] AravindPCoder, "Chemical analysis of wine," Available at Kaggle: https://www.kaggle.com/datasets/aravindpcoder/chemical-analysis-of-wine, 2024.

[46] S. Makridakis, "The m4 competition: Results, findings, conclusion and way forward," Available at PapersWithCode: https://paperswithcode.com/dataset/m4, 2018.

[47] ShenbagaKumarS, "Time series datasets," Available at Kaggle: https://www.kaggle.com/datasets/shenba/time-series-datasets?select=Electric_Production.csv, 2017.

[48] ——, "Time series datasets," Available at Kaggle: https://www.kaggle.com/datasets/shenba / time-series-datasets / data ? select = daily-minimum-temperatures-in-me . csv, 2017.

[49] V. V. Venkitesh, "Microsoft stock - time series analysis," Available at Kaggle: https://www.kaggle.com/datasets/vijayvvenkitesh/microsoft-stock-time-series-analysis, 2017.

[50] SJ, "Analyzing and predicting bitcoin pricing trend," Available at Kaggle: https://www.kaggle.com/datasets/surajjha101/analyzing-and-prediction-of-bitcoin-pricing, 2020.

[51] J. Gopinadhan, "House hold energy data - time series," Available at Kaggle: https://www.kaggle.com/datasets/jaganadhg/house-hold-energy-data, 2018.

Acknowledgments

This thesis was made possible thanks to LoopAI Group that allowed me to work as part of their development team, and thanks to Professor Alberto Testolin who agreed to be my supervisor. I want to thank them for the unique experience I had. My deepest thanks go to my partner for supporting me during these years and believing in me more than I did myself. I sincerely thank my family for the support and the opportunity to continue studying, and my friends who helped lighten the day. I am thankful to have all of you in my life.