

**LLMs in automated tabular data analysis**

Requirements Specification Document

Filip Kołodziejczyk, Jakub Świstak

Supervisor:

prof. dr hab. inż. Przemysław Biecek

version 1.4

16.10.2023

Table of Contents

[1 Abstract 3](#__RefHeading___Toc2796_2030408508)

[1.1 History of changes 3](#__RefHeading___Toc800_779315115)

[2 Vocabulary 4](#__RefHeading___Toc802_779315115)

[3 Specification 5](#__RefHeading___Toc6747_630044125)

[3.1 Executive summary 5](#__RefHeading___Toc6749_630044125)

[3.2 Functional requirements 5](#__RefHeading___Toc6751_630044125)

[3.3 Non-functional requirements 7](#__RefHeading___Toc6753_630044125)

[4 Project schedule 8](#__RefHeading___Toc6755_630044125)

[5 Risk Analysis 10](#__RefHeading___Toc6757_630044125)

[6 Bibliography 11](#__RefHeading___Toc6793_630044125)

# Abstract

This document is an initial document for the thesis project about the utilization of LLMs in automated tabular data analysis. The goals of the systems, as well as their general description and planned use cases, are discussed here. Use cases along with non-functional requirements and project schedules developed in this document will be key in sticking to well-defined project goals to avoid inadvertently deviating from the predetermined course of work. In the end, risk analysis will be provided, to identify all potential threats to the project’s success early and come up with a plan for risk management.

## History of changes

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Description** | **Version** |
| 14.10.2023 | Filip Kołodziejczyk | First version (Abstract, Vocabulary, Executive Summary, Bibliography) | 1.0 |
| 14.10.2023 | Jakub Świstak | Minor fixes of first version | 1.1 |
| 15.10.2023 | Filip Kołodziejczyk | Added Functional and Non-Functional requirements | 1.2 |
| 16.10.2023 | Jakub Świstak | Added Project Schedule and Risk Analysis | 1.3 |
| 16.10.2023 | Filip Kołodziejczyk | Final fixes and formatting adjustments | 1.4 |

# Vocabulary

**LLM** – Large Language Model, a type of language model that can perform various natural language processing (NLP) tasks. LLMs use deep learning algorithms, mainly transformers, and are trained using massive datasets. This enables them to recognize, summarize, translate, predict, and generate text and other forms of content [1].

**Tabular data analysis** – a statistical method of analyzing dataorganized in a table with rows and columns. The rows represent observations, while the columns represent attributes for that observation [2].

**Prompt engineering** – the process of refining prompts that a person can input into a generative artificial intelligence (AI) service to generate text or images. It enables direct interaction with the AI model using only plain language prompts. In the past, working with machine learning models typically required deep knowledge of datasets, statistics, and modeling techniques. Today, LLMs can be “programmed” in English, as well as other languages. Prompt engineering is an emerging field that requires creativity and attention to detail. It involves selecting the right words, phrases, symbols, and formats that guide the model in generating high-quality and relevant texts [3]. There are several types of prompt engineering, including:

* **Zero-shot (direct prompting):** providing a single prompt to the model without any additional examples or training data. The model is expected to generate a response based on the prompt alone.
* **One-, few-, and multi-shot (prompting with examples):** providing one or more examples along with the prompt to help the model generate a response. The number of examples can vary from one to many, depending on the complexity of the task.
* **Chain-of-thought prompting:** providing a sequence of prompts that guide the model through a series of related tasks or questions. The model generates responses to each prompt in the sequence, building on its previous responses to create a coherent output.
* **Zero-shot CoT:** generating a sequence of prompts that guide the model through a series of related tasks or questions, without any additional training data. The model generates responses to each prompt in the sequence, building on its previous responses to create a coherent output [4].

**Hallucination** - instances when the AI model “imagines” or “fabricate” information that does not directly correspond to the provided input [5].

**Jupyter Notebook** ­­– an open-source web application that allows users to create and share documents that contain live code, equations, narrative text, visualizations, interactive dashboards, and other media. It is a popular tool among data scientists and researchers for creating and sharing computational documents. Jupyter Notebook supports over 40 programming languages, including Python, R, Julia, and Scala [6].

**Containers** – lightweight, portable, and self-sufficient environments that can run on any machine with the open-source platform Docker installed. Containerization simplifies software development and innovation by enabling applications to exchange data and functionality easily and securely [7].

# Specification

## Executive summary

The work aims to develop a new system leveraging Large Language Models (LLMs) like GPT-4 [8] or LLaMA 2 [9] for the automated analysis of tabular data. The system will enable users to input a set of data for semi-automatic analysis, eliminating the need for specialized coding knowledge. The system will generate automatic reports summarizing the analysis and relevant plots. Thus, the target user of this system will be a user in need of tabular data analysis, while not having sufficient knowledge of programming or even statistical analysis of such datasets. This system will be able to be used in various contexts, such as business or scientific analysis.

## Functional requirements

The system in terms of functional requirements from the user’s perspective is a very simple one. He or she should be able to add the dataset to be analyzed. Then, for the selected dataset (the system will be able to keep multiple datasets for the user) analysis generation will be made. Such analysis, next to the internal algorithm managing the conversation flow and code executions (in internal system runtime), will engage the AI chatbot. Finally, the user will have the possibility to display and download generated analysis reports, as well as check directly how the conversation between the system and chatbot looked like. It is represented in Figure 1 as a use case diagram [10], with its detailed description in Table 1.

A screenshot of a diagram

Description automatically generated

Figure 1 Use case showing the user’s interaction with the system

Table 1 Description of use cases for actors and system

|  |  |  |  |
| --- | --- | --- | --- |
| Actor | Name | Description | System response |
| System user | Import dataset | Load dataset in one of the supported formats to the system | Webpage with all imported datasets till this moment. |
| Export dataset | Select and download previously loaded dataset to the system | Downloadable file with selected dataset in it. |
| Select current context | Select one of imported datasets as one to be used for analysis (also see the previous analysis if such was conducted) | The selected dataset’s name highlighted in system as current context. |
| Generate analysis (as active actor) | Run the algorithm which will attempt to generate the analysis for the selected dataset and generate the report | Response message if the analysis was successful. An error message if it was a failure, otherwise prompt to show analysis report. |
| Show analysis report | Display analysis report generated for selected dataset (if it was generated prior to that). | Jupyter Notebook read-only interface. |
| Download analysis report | Download analysis report generated for selected dataset (if it was generated prior to that). | Downloadable Jupyter Notebook file which can be edited later. |
| Show conversation history | Check the details of analysis report generation. | A website with log file covering the conversation between system and chatbot used for report generation. |
| LLM Chatbot | Generate analysis (as passive actor) | Conversate with the system, providing answers for prompts given by the algorithm. | Attempt to run the provided code snippet and generate new prompts to chat based on result. |

## Non-functional requirements

Besides functional requirements, the system should meet several other requirements to become an application usable in real tasks. Those requirements are represented in the form of a table split into a few categories. This is Table 2.

Table 2 Non-functional requirements of the system

|  |  |
| --- | --- |
| Requirements area | Description |
| Utility  ( *Usability* ) | The application is hosted in the cloud and accessible via a web browser only to minimize hardware and software requirements on the user’s side. |
| The application front end has a simple interface, possibly compliant with Web Content Accessibility Guidelines (WCAG) 2.1. |
| The system accepts the most popular dataset formats. |
| Reliability  ( *Reliability* ) | The system can work 24/7 except for patching windows. |
| The system must be able to continue its work after the error occurrence (but restart/refresh may be needed). |
| The application always returns the result of analysis (which may be an error), except for internal errors. It means it never enters infinite computation. |
| Performance  ( *Performance* ) | The application can handle multiple users at the same time. |
| Maintenance  ( *Supportability* ) | The system can be connected to more than one chatbot. |
| The application will keep old analyses accessible after new updates. Hence, the update won’t destroy previous actions in app. |
| The application provides logs from conversations with the chatbot for audit purposes. |

# Project Schedule

For the sake of well-organized and diligent work with the project, a sketch of already realized (for better clarity) and yet-to-be-done tasks was created in the form of a list (below). Those tasks were assigned to the project members and planned in time dimension using the Gantt diagram (Figure 2). It is expected that the tasks may slightly differ during the implementation process, as the architecture of the solution will mature over time. Nevertheless, it is desired to keep the deadlines for the emerging tasks by finding the most similar / correlating one from the below list and keeping its timeline.

Development environment setup

1. Create a Docker image for the code execution environment (Filip Kołodziejczyk)
2. Jupyter Notebook API integration (Filip Kołodziejczyk)
3. Deploy to the cloud environment (Filip Kołodziejczyk)

Conversational features

1. Implement a feature to add summarizations to ongoing conversations (Jakub Świstak)
2. Add an option to remove older messages and replace them with summaries (Jakub Świstak)
3. AI agent to critique the GPT response and achieve the conversation goal (Jakub Świstak)

LLM integration

1. Integrate with GPT-4 API (Jakub Świstak)
2. Integrate with LLaMA2 API (Jakub Świstak)

User Interface / Reporting

1. Implement a feature to visualize conversations in the UI (Filip Kołodziejczyk)
2. Add functionality to display executable code and generated plots in the conversation (Filip Kołodziejczyk)
3. Develop logic to automatically generate analysis reports (Filip Kołodziejczyk)
4. Create simple web app UI (Filip Kołodziejczyk, Jakub Świstak)

Comparative analysis

1. Implement different prompt techniques for LLM queries (Jakub Świstak)
2. Compare the accuracy of GPT-4 and LLaMA2 for specific tasks in tabular data analysis (Filip Kołodziejczyk, Jakub Świstak)

Evaluation

1. Prepare human feedback tests (Filip Kołodziejczyk)
2. Create automated tests (Jakub Świstak)
3. Develop metrics to measure the output quality and efficiency of the generated solutions (Filip Kołodziejczyk, Jakub Świstak)

Finalization

1. Generate a final comprehensive report summarizing evaluations, comparative analysis, and overall system performance (Filip Kołodziejczyk, Jakub Świstak)

A white grid with many colored lines

Description automatically generated with medium confidence

Figure 2 Project schedule, Gantt diagram

# Risk Analysis

Finally, we assess the risks which may harm the development and even finalization of the project. First, we take a general view of SWOT in the business aspect (less relevant in a research project like this) - Figure 3, and later go into detail through all potential risks that we may approach in the implementation process (more relevant).

A diagram of swot analysis

Description automatically generated

Figure 3 SWOT analysis

**Model Performance:**

* **Risk:** The performance of LLMs may not be sufficient for accurate tabular data analysis, especially for bigger datasets. (width and relations between tables?)
* **Mitigation:** Notify the user about the risks that come with automatic analysis.

**Data Leakage:**

* **Risk:** Transmitting data to the LLM API might expose sensitive information to the third-party vendor.
* **Mitigation:** We will have on-site models to be self-hosted (LLMa 2) to open-source model, and since GPT-4 is only available via API, we will use encryption. However, data will still be sent and accessible to OpenAI.

**Unauthorized Access:**

* **Risk:** A breach could expose both your data and generated reports.
* **Mitigation:** The app will be available to be served on the user's device so they can expose or limit access to the solution. The only thing unavailable for on-device access is the GPT-4 model.

**Erroneous Analysis:**

* **Risk:** LLMs could introduce bias in automated reports, leading to incorrect conclusions.
* **Mitigation:** Reports will be marked as auto-generated based on the AI responses. We will also implement a review stage where we will validate the generated reports.

**Credibility:**

* **Risk:** The bias might compromise the credibility of the system.
* **Mitigation:** Disclose the potential for bias upfront to the users.

**Prompt Size Limitation:**

* **Risk:** LLMs have a maximum token limit which could be exceeded by large datasets.
* **Mitigation:** Chunk data and analyze in parts, then aggregate.

**Data Ownership:**

* **Risk:** Users want to protect their data and do not want to share the specifics.
* **Mitigation:** Users will use their API key and self host the app to protect their data from third-party vendors (app admin), and if they wish, they can decide to host their open-source model.

**Inability to Generate or Interpret Images Using LLMs API:**

* **Risk:** LLM doesn’t support graphical data
* **Mitigation:** Before generating the plot, we will try to export the data in a text format available for LLM to interpret.

**Malicious Code Generation by LLMs**

* **Risk:** LLMs generale code that could be malicious or harmful when executed.
* **Mitigation:** Implement a safe separated environment to separate the user environment from the analysis environment.

**Risk of LLMs Generating False or Nonsensical Information (Hallucinations)**

* **Mitigation:** Implement a validation layer to cross-reference and mitigate the issue.

**Loops in Generation (Non-terminating Conversations)**

* **Mitigation:** Before every message, the user will be able to terminate the analysis.

# Bibliography

1. What is large language model (LLM)?, <https://www.elastic.co/what-is/large-language-models>
2. What is Tabular Data, <https://www.statology.org/tabular-data/>
3. What is Prompt Engineering? Explanation With Examples, <https://blog.enterprisedna.co/what-is-prompt-engineering/>
4. What is Prompt Engineering (section Exploration of Essential Prompt Engineering Techniques and Concepts), <https://www.promptengineering.org/what-is-prompt-engineering/>
5. Deepchecks Glossary LLMs Hallucinations, <https://deepchecks.com/glossary/llms-hallucinations/>
6. What is a Jupyter Notebook?, <https://www.databricks.com/glossary/jupyter-notebook>
7. What is containerization? <https://www.ibm.com/topics/containerization>
8. GPT-4 Technical Report, [**https://arxiv.org/abs/2303.08774**](https://arxiv.org/abs/2303.08774)
9. Llama 2: Open Foundation and Fine-Tuned Chat Models, <https://arxiv.org/abs/2307.09288>
10. The Use Case Diagram, UML @ Classroom (2014), Springer, <https://link.springer.com/book/10.1007/978-3-319-12742-2>