Warsaw University of Technology

group project

Post-completion Documentation

Authors:
Kołodziejczyk Filip, Świstak Jakub

Supervisor:
prof. dr hab. inż. Przemysław Biecek

Version: 1.2

January 14, 2024

Contents

1 History of Changes

Table 1 History of Changes

Date	Author	Description	Version
02.01.2023	Filip Kołodziejczyk	First version (Abstract, Table of Contents, Deployment Documentation, Installation Instruction)	1.0
02.01.2023	Filip Kołodziejczyk	Added Technical Documentation	1.1
02.01.202	Jakub Świstak	Added User's Manual	1.2

2 Abstract

This document details the achievements of the application development stage in the automated tabular data analysis project. It summarizes the software's technical specifications, highlighting the primary features and technology used. A key focus is on the notable differences between the initial project plans and the final software version, illustrating the evolution and adaptability of the project.

Additionally, the document includes a user manual catering to users with varied technical backgrounds. It also outlines the production deployment process, covering environment requirements, installation steps, and cloud hosting details.

3 Notable Changes

Most differences between the initial assumptions outlined in the General Design Document and Requirements Specification Document, and the final version of the project, stem from the limited development time and emerging needs during the project's progression. Adapting in an Agile manner, we prioritized aspects crucial to the value of our engineering work, focusing on the conversation handling and analysis rating, at the expense of user experience and commercially-ready features. The most notable changes include:

- User data is not persisted; each session is ephemeral. Users can work only with the currently loaded dataset and access the most recent analysis report from that session
- The application lacks user authentication, aligning with the ephemeral nature of each session.
- Users cannot review the assistants' conversation behind the analysis.
- The User Interface has been simplified in response to the above changes.

4 Deployment Documentation

This application offers flexible deployment options: it can be installed on a user's personal computer as a local solution, or it can be deployed on a server to serve multiple users. Leveraging software containerization, each component, except for those externally provided, is distributed as Docker images. This approach ensures multi-platform compatibility, portability, and ease of execution.

In both cases, configuration of the application can be done via environmental variables. Below is the example of such configuration:

```
RUNTIME=<selected runtime>
PROMPT=<selected prompting strategy>
PORT=<port to run web app on>

# python-ssh-runtime
# RUNTIME_HOST=<host of python-ssh>
# RUNTIME_PORT=<port of python-ssh>
# USERNAME=<username to access python-ssh>
# PASSWORD=<password to access python-ssh>

# jupyter-notebook-runtime
RUNTIME_HOST=<host of jupyter>
RUNTIME_PORT=<port of jupyter>
RUNTIME_USE_HTTPS=<true/false>
TOKEN=<token to access jupyter>

# apache-zeppelin-runtime
# RUNTIME_HOST=<host of zeppelin>
# RUNTIME_PORT=<port of zeppelin>
```

Please note that certain variables may be excluded depending on the selected runtime environment. Additionally, while not mandatory, it is advisable to thoroughly test each version of the application before deployment.

4.1 Local deployment

To deploy the application on a local computer, users need to download the project's repository from `https://github.com/jswistak/automatic-data-analysis`. It is important to ensure that all subsequent operations are performed in the root directory of the project, which must be extracted first if downloaded as an archive. The available Docker Compose profiles are 'jupyter-notebook' and 'python-ssh', which should be selected in accordance with the chosen environmental variables.

1. Verify the installation of the Docker engine by executing the command 'docker version'. If it's not installed, as indicated by an error message, follow the installation instructions at `https://docs.docker.com/engine/install/`.

2. Create and populate the '.env' file as outlined in the previous listing.
3. Execute the command 'docker compose –profile <profile> -d build' to build the Docker environment.
4. Start the application by running 'docker compose –profile <profile> -d up'.

Once these steps are completed, all relevant components will be accessible via the specified ports. To stop the running containers, use the command 'docker compose –profile <profile> -d down'.

4.2 Cloud deployment

The application is designed to efficiently support multiple, concurrent users. It operates using Docker Compose, which is akin to the local deployment process but necessitates certain modifications to the server's network settings to optimize performance and security. While local deployment is viable, deploying the software in a public cloud environment often presents a more scalable and robust solution. Accordingly, we have chosen Azure Cloud for hosting, utilizing Azure Container Applications due to their flexibility and reliability.

To facilitate deployment in this cloud environment, the following steps should be undertaken:

1. Create a new resource group or utilize an existing one.
2. Set up a Container App Environment and a Container Registry.
3. Upload the Docker images to the Container Registry.
4. Establish new Container Apps, with each one managing one of the uploaded images. Ensure that all variables are correctly set during this process.

By following these steps, the application will be securely and effectively deployed in the Azure Cloud environment. This method of deployment offers several advantages, including scalability, high availability, and reduced infrastructure management overhead. Additionally, cloud deployment facilitates easier updates and maintenance, ensuring the application remains up-to-date and secure.

Our application is available online, hosted in Azure, at the moment. For security reasons, the URL and credentials for our deployed application are not disclosed in this document.

5 Installation Instruction

As highlighted in the previous section, all components of the solution are containerized, eliminating the need for any installations from the user's perspective, aside from a modern web browser such as Google Chrome, as this is a web-based application. However, if one wishes to deploy the application locally, the Docker Engine will also be required. Detailed installation instructions for the Docker Engine can be found in the deployment section.

Additionally, for an optimal experience, users should ensure their web browser is up to date to fully support the application's features. Regular updates to the browser will help in maintaining security and performance standards. For those deploying locally, it's important to regularly update the Docker Engine to the latest version to ensure compatibility and security. Users should also have a basic understanding of Docker commands for managing the local deployment effectively.

6 Technical Documentation

The application is designed exclusively for use through its web-based Graphical User Interface (GUI). As such, there is no public API or protocol interfaces that users need to be aware of. The operational mechanics of the application and its feature set are comprehensively documented in the General Design Document. The majority of this content remains consistent with the original design, with exceptions noted in section 3, which outlines the notable changes made during development.

In terms of quality assurance, the application's codebase is thoroughly vetted through both unit and integration testing. The unit tests focus on individual components to ensure they function correctly in isolation, thereby guaranteeing the reliability of each part of the application. Integration tests, on the other hand, are crucial for confirming the seamless interaction between all components of the system. This includes testing the internal components, such as the runtime environment and the main application, as well as interactions with third-party services like the OpenAI API. These tests are instrumental in ensuring that the various parts of the application work together effectively and efficiently.

Furthermore, to complement these automated tests, acceptance testing was conducted manually. This phase involved real-world scenario testing to validate the overall functionality and usability of the application. Manual acceptance testing is essential for assessing the user experience, ensuring that the application not only meets the technical specifications but also delivers a user-friendly and intuitive interface.

Additionally, continuous integration and continuous deployment (CI/CD) practices were implemented to streamline the development process. These practices enable automatic testing and deployment of the application, ensuring that new changes are seamlessly integrated and that any issues are quickly identified and resolved. The

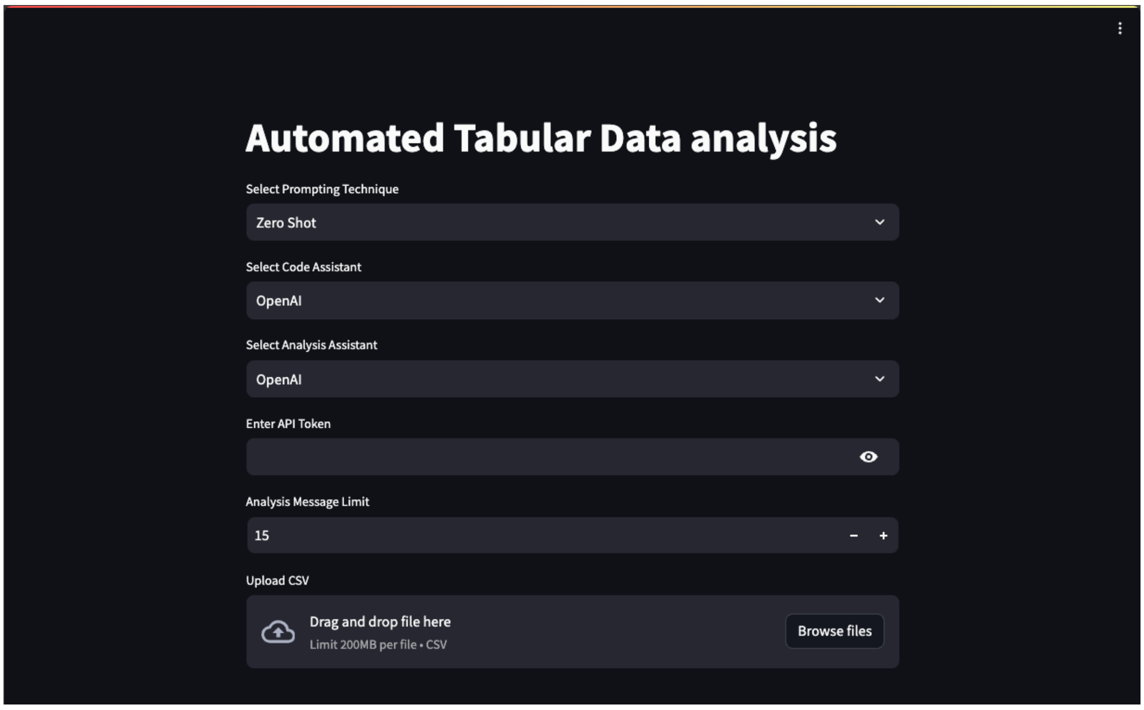
CI/CD pipeline significantly enhances the efficiency of the development process and maintains a high standard of quality in the application.

7 User's Manual

To utilize this application, users must possess the necessary key and/or credentials to access the API of the selected Large Language Model (LLM) assistants. Each assistant API has its unique setup process, so users should refer to the specific documentation for guidance on obtaining these credentials.

Step-by-Step User Guide:

1. Accessing the Web App: Navigate to the web app address using a web browser. Upon opening, the interface depicted in Figure 1 should be displayed.



Automated Tabular Data analysis

Select Prompting Technique: Zero Shot

Select Code Assistant: OpenAI

Select Analysis Assistant: OpenAI

Enter API Token

Analysis Message Limit: 15

Upload CSV

Drag and drop file here
Limit 200MB per file • CSV

Figure 1: Initial page

2. Setting Up the Analysis:
   (a) *Select Prompting Technique* - Field allows users to apply different techniques when prompting both idea and code assistants.
   (b) *Selecting Assistants and Entering API Token* - Choose the desired setup and enter the API Token for the LLM assistant.
   (c) *Setting Analysis Message Limit* - This option determines the length of the generated report.

(d) *Uploading Data File* - attach the CSV file containing the data to be analyzed. Once all fields are completed and the file is uploaded, an overview of the uploaded file and a `Analyze` button will appear, as shown in Figure 2.

(e) *Removing or Replacing Data* - To remove the added data, use the right-hand side cross. To replace it, simply choose a new file to be analyzed.

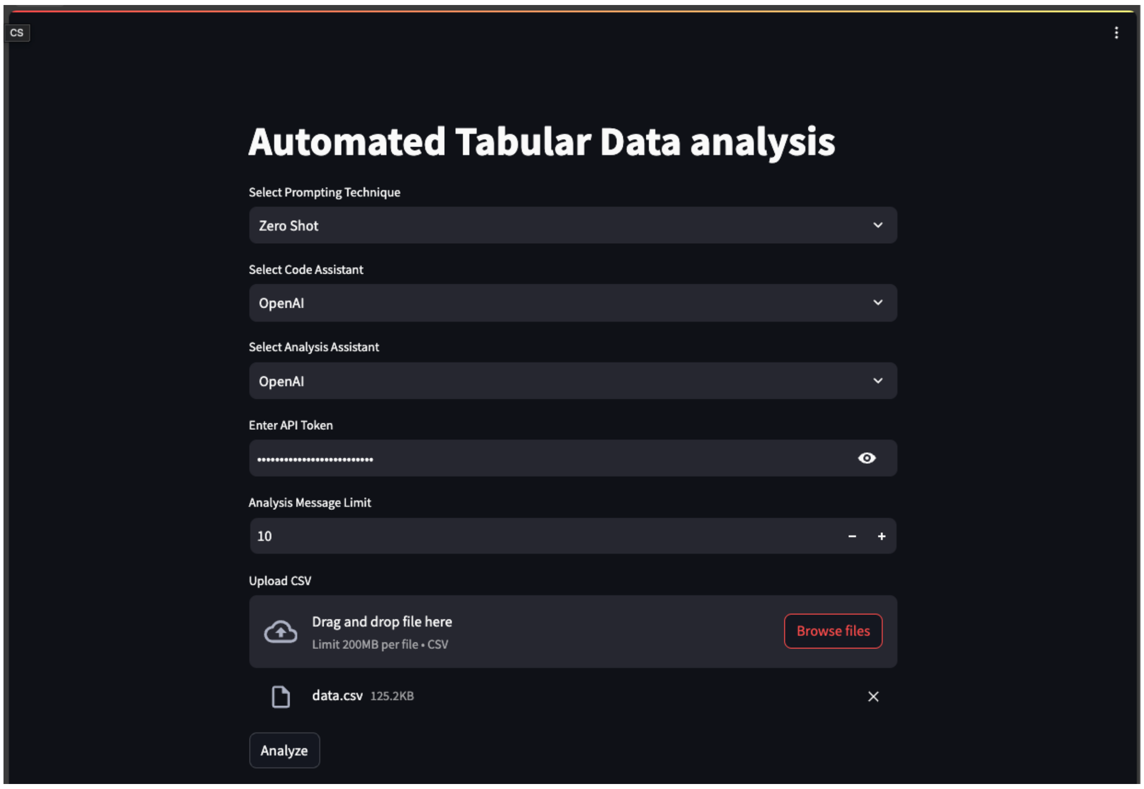
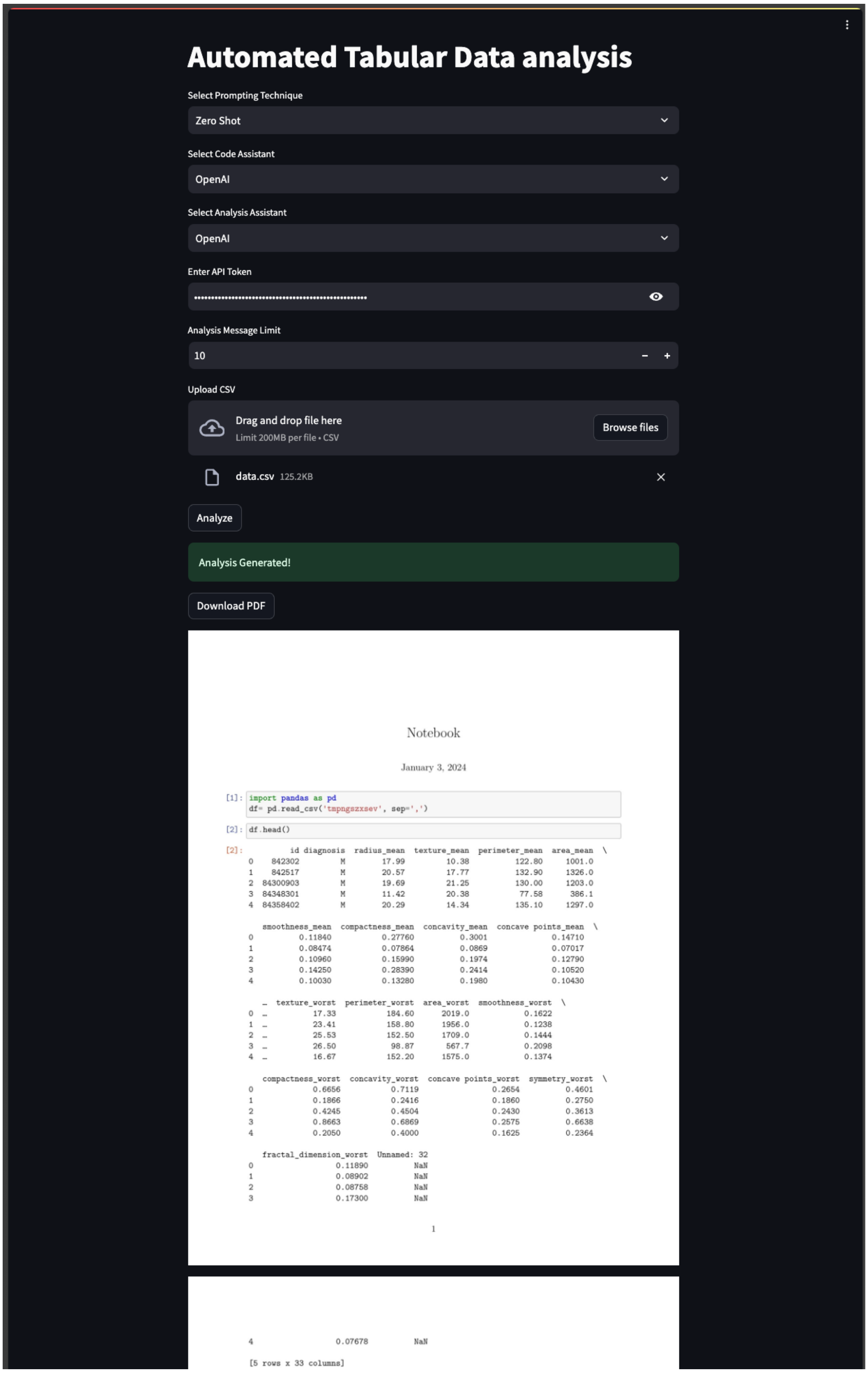


Figure 2: Set up completed, waiting for analysis

3. Whenever ready, user have to click `Analyze` button and wait for outcome.
4. If the analysis is successful, a report will be displayed in the bottom section of the page, as depicted in Figure 3. Users can download this report in PDF format by pressing the `Download PDF` button.
5. In case of failure, red box with error code will appear. Then, windows will look like on Fig. 4



Automated Tabular Data analysis

Select Prompting Technique: Zero Shot

Select Code Assistant: OpenAI

Select Analysis Assistant: OpenAI

Enter API Token: ••

Analysis Message Limit: 10

Upload CSV

Analysis Generated!

Notebook

January 3, 2024

```
[1]: import pandas as pd
     df= pd.read_csv('tmpngszxsev', sep=',')
```

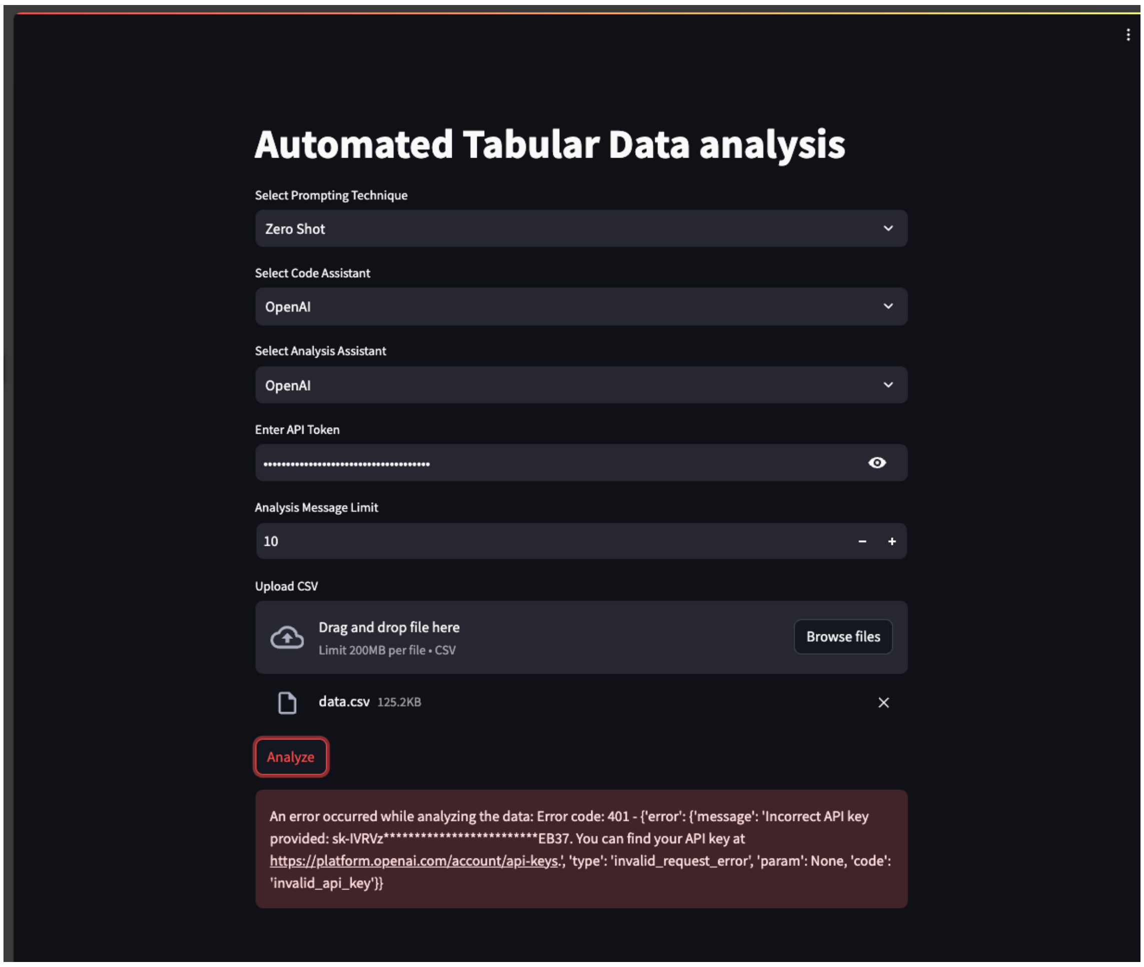
```
[2]: df.head()
```

```
[2]:          id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
     0    842302         M        17.99         10.38          122.80     1001.0
     1    842517         M        20.57         17.77          132.90     1326.0
     2  84300903         M        19.69         21.25          130.00     1203.0
     3  84348301         M        11.42         20.38           77.58      386.1
     4  84358402         M        20.29         14.34          135.10     1297.0

        smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
     0          0.11840           0.27760          0.3001              0.14710
     1          0.08474           0.07864          0.0869              0.07017
     2          0.10960           0.15990          0.1974              0.12790
     3          0.14250           0.28390          0.2414              0.10520
     4          0.10030           0.13280          0.1980              0.10430

        …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
     0  …          17.33           184.60      2019.0            0.1622
     1  …          23.41           158.80      1956.0            0.1238
     2  …          25.53           152.50      1709.0            0.1444
     3  …          26.50            98.87       567.7            0.2098
     4  …          16.67           152.20      1575.0            0.1374

        compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
     0             0.6656           0.7119                0.2654          0.4601
     1             0.1866           0.2416                0.1860          0.2750
     2             0.4245           0.4504                0.2430          0.3613
     3             0.8663           0.6869                0.2575          0.6638
     4             0.2050           0.4000                0.1625          0.2364

        fractal_dimension_worst  Unnamed: 32
     0                  0.11890          NaN
     1                  0.08902          NaN
     2                  0.08758          NaN
     3                  0.17300          NaN
```

1

```
     4                  0.07678          NaN

     [5 rows x 33 columns]
```

Figure 3: Analysis successful



Automated Tabular Data analysis

Select Prompting Technique: Zero Shot

Select Code Assistant: OpenAI

Select Analysis Assistant: OpenAI

Enter API Token: ••

Analysis Message Limit: 10

Upload CSV

```
An error occurred while analyzing the data: Error code: 401 - {'error': {'message': 'Incorrect API key provided: sk-IVRVz************************EB37. You can find your API key at https://platform.openai.com/account/api-keys.', 'type': 'invalid_request_error', 'param': None, 'code': 'invalid_api_key'}}
```

Figure 4: Analysis failure