Assignment 3

**Collaborative Development of Data**

**Explorer Web App**



Group 2

08.11.2024

Benedict Brunker (25551995)

Justin Govanxa (25153673)

Yuzhong Wang (13123157)

Britney Rosalia Odria (14131609)

<https://github.com/benedict-brunker/DSP_AT3_Group2>

94692 - Data Science Practice

Master of Data Science and Innovation

University of Technology of Sydney

Table of Contents

[1. Executive Summary 0](#_Toc182001283)

[2. Introduction 1](#_Toc182001284)

[2.1 Project Objectives 1](#_Toc182001285)

[2.2 Addressing Stakeholder Requirements 1](#_Toc182001286)

[3. Web App Presentation 2](#_Toc182001287)

[a. Purpose 2](#_Toc182001288)

[b. Functionality 2](#_Toc182001289)

[c. Setup 3](#_Toc182001290)

[d. Potential Users and Use Cases 3](#_Toc182001291)

[e. Limitations 4](#_Toc182001292)

[4. Reflecting On Building Data Product 5](#_Toc182001293)

[a. Importance 5](#_Toc182001294)

[b. Key Skills and Technologies 5](#_Toc182001295)

[c. Data Products in the Age of Artificial Intelligence 6](#_Toc182001296)

[5. Collaboration 7](#_Toc182001297)

[a. Individual Contributions 7](#_Toc182001298)

[b. Group Dynamic 8](#_Toc182001299)

[c. Ways of Working Together 9](#_Toc182001300)

[d. Issues Faced 9](#_Toc182001301)

[6. Conclusion 11](#_Toc182001302)

7. References

# 1. Executive Summary

Our project was to develop an interactive data explorer web application, designed to help users explore and analyse their datasets. It provides a simple, accessible tool for quickly obtaining a general overview of the user’s data and providing useful high-level analysis, allowing users to gain insights into their data.

Data exploration is important for many potential stakeholders, be they working professionals, students, or just amateur hobbyists. However, not all these stakeholders possess the technical prowess to conduct data exploration through special programming languages like Python. This app will provide non-specialists an easy and fast way to gain a high-level overview of their data, by rendering an interactive user interface for uploading and querying CSV files based on the Streamlit library.

The result is a functioning, fast and visually appealing web application that can be hosted locally or on the Streamlit cloud. Our workflow, code and project structure are documented on GitHub, which will allow other programmers to understand, use and adapt the source code to their needs.



# 2. Introduction

## 2.1 Project Objectives

The goal of this project is the collaborative development and deployment of a data explorer web application. The app will provide an accessible user-interface for uploading Comma-Separated Values (CSV) files and rendering useful summary information about the CSV based on user preferences.

Desired outcomes for our data application can be summarized thus:

* A User Interface (UI) accessible specialists and non-specialists alike.
* App is fast and seamless in operation.
* Aesthetically pleasing and elegant User Experience (UX).
* App is robust in handling a wide variety of CSV formats, fields and data types.
* Workflow and code are well documented through GitHub and Python files, allowing other programmers to understand, use and adapt the application’s underlying source code.

## 2.2 Addressing Stakeholder Requirements

We identify the following stakeholders as likely beneficiaries of the project:

**Business professionals** Our app will provide an accessible user experience for professionals who lack data manipulation expertise to be able to quickly and easily generate summaries of data files, facilitating a general exploration and overview of their data.

**Students** Similarly, our app will be useful to students not specializing in data manipulation but who need to work with data, providing a simple and accessible tool for summarizing and exploring their data.

**Programmers** Our application will be well documented and structured, defining new classes and functions that can be used, adapted, updated and improved by the wider Python programming community.



# 3. Web App Presentation

## a. Purpose

The primary purpose of this application is to provide users with a tool for conducting Exploratory Data Analysis (EDA) on datasets formatted as Comma-Separated Values (CSV) files. By uploading a CSV to our app, the user can quickly gain a general overview of the dataset, analyze numbers, text, and datetime columns, and render instructive visualizations of their data. This enables users to identify data issues, spot patterns, and prepare data effectively for further analysis.

## b. Functionality

The application is designed according to the following structure:

**app/** Contains source code for running the main architecture of the Streamlit app in *streamlit\_app.py,* and configures the folder as an importable Python package with an *\_\_init\_\_.py* file.

**tab\_df/** Contains files that read the uploaded CSV into a Pandas DataFrame (*logics.*py), and displays a dataset overview in the app, which includes dataset information such as row count, duplicate rows and details about each column (*display.py*).

**tab\_numeric/** Contains files for analyzing numerical columns of the uploaded CSV, displaying statistical summaries and a histogram for visual exploration of numeric data.

**tab\_text/** Contains files contains analysing text columns, which provides details on unique values, empty strings, and character formats, along with a frequency distribution chart.

**tab\_date/**  Contains files that handles datetime column analysis, showing information such as date range, weekday counts, and a histogram of the timeseries distribution.

**requirements.txt/** Contains the list of required packages to run the app

## c. Setup

1. Ensure Python 3.9 or higher is installed on your device. If not, install it from its official website:

[***Download Python | Python.org***](https://www.python.org/downloads/)

1. Open your terminal or command prompt. Navigate to the directory containing your application files, then run the following command to install the required libraries:

***pip install -r requirement.txt***

1. From the same directory, run the following command to launch the application:

***streamlit run app/streamlit\_app.py***

## d. Potential Users and Use Cases

The application is designed for use by anyone who needs to conduct high-level data exploration. This may include professionals, students or even amateur hobbyists. The app is designed so that no technical expertise in Python, R or any other data manipulation libraries is required.

The application can also be used data scientists and other data specialists to simplify and accelerate their exploratory data analysis on datasets. For instance, a data scientist is tasked with analyzing a large dataset to glean initial insights before developing their model. By uploading the dataset into the application, they can quickly examine the information of the dataset such as missing values, duplicates, data types, and column distributions. This will allow the data scientist to dig deeper into numeric, text, and datetime columns, helping them spot unusual patterns, outliers, and potential data quality issues early on. This initial EDA saves valuable time, enabling the data scientist to make informed decisions on data cleaning and transformation steps, ensuring a smoother modeling process down the line.

There is strong potential for future commercialization for this application through evolving into a data exploration platform. We can further improve this app by supporting different types of data file types, such as Excel and SQL, which would appeal to a broader range of data professionals. Also, adding cloud storage would allow teams to access and analyze data collaboratively from any location, enhancing productivity and facilitating seamless teamwork.

## e. Limitations

Currently, the application only supports CSV files, which may not be ideal for users who want to upload data in Excel, SQL, JSON or other formats. Also, the application does not support large datasets over 200MB. Even below the 200MB limit, the app can sometimes lag when dealing with large datasets. Furthermore, the visualizations in this application are as yet relatively basic, with limited options for customization and interaction. The current version of the application also struggles to handle files with .csv extension that use delimiters other than commas, such as semicolons (“;”) or bars (“|”). The application can sometimes struggle to correctly parse datetime columns with unusual formats. Future improvements could include support for more file types, better performance with large datasets and more detailed visualizations to make the tool even more flexible and powerful for data exploration. As this is the first version of the application, we hope in future to release v.2, which will deal with some of these problems. We invite and encourage users and programmers in the GitHub community to offer feedback and suggestions based on their use of the app.



# 4. Reflecting On Building Data Product

## a. Importance

Data scientists may become accomplished at analyzing, exploring and modelling data through programming languages like Python and R. However, these tools can often be inaccessible to non-specialists. Data scientists need not only to learn from data, but communicate their findings and capabilities to non-specialists who can make use of data insights. The ability to develop data products bridges the gap in technical competence between specialists and non-specialists. Through development of data products, the data scientist is able to leave non-specialist colleagues and other stakeholders with ongoing, dependable and accessible applications that allow them to unlock key insights that would otherwise remain obscure and impenetrable.

## b. Key Skills and Technologies

This project gave us an opportunity to learn and develop the core skills needed for data product development, including:

* **Programming** in Python, with a focus on data handling and app development. We honed our skills in core Python libraries like Pandas, Streamlit and Altair for data manipulation, web application and visualization respectively. We tested ourselves in building an end-to-end Python application structured as a cohesive package that can be used and adapted by fellow developers. We learned how to program and document core functionalities of Python like classes and functions.
* **Web Development** in Streamlit, which we used to create an interactive, accessible and aesthetically pleasing web interface.
* **Version Control** with Git allowed us to collaborate dynamically, keeping detailed records of our work. Git allows us to share our work with one another, experiment with each other’s code, and revert back to previous versions when mistakes are inevitably made.
* **Unix programming.**  The project developed our skills in working from the command-line interface (CLI), which provides a more flexible, exact and dynamic way of unlocking computing capabilities than can be achieved by using a computer’s Graphical User Interface (GUI) alone.
* **Data Visualization** with Python packages like Altair. The app includes interactive and aesthetically beautiful charts and graphs for easy data exploration.
* **Data Analysis**, by automating the exploration of numeric, text, and datetime columns, providing key insights for each.

## c. Data Products in the Age of Artificial Intelligence

While the current app focuses on EDA, integrating AI could take its capabilities further. For example:

* **Automated Insights**: AI could highlight unusual patterns, outliers, or trends without manual exploration. Retrieval Augmented Generation (RAG) can be used to customize Large Language Models (LLMs) on user input data, which allows these models to not only provide answers tailored to specific organizational information (which may be proprietary), but also to cite and document sources.
* **Predictive Suggestions**: Based on the data, the app could suggest potential next steps, like transformations or models to consider.
* **Natural Language Queries**: Users could interact with the app using simple questions.



# 5. Collaboration

## a. Individual Contributions

### Britney Odria

Britney contributed significantly by working on the tab\_df directory, which involved handling the information of user's chosen datasets, ensuring it was well-structured and accessible for analysis. Also, played a key role in discussing the design and functionality of the web application, focusing on how to use the application and how the application supports data scientists in performing exploratory data analysis. Additionally, worked in discussing the application development and purpose, and the application potential use for real life scenarios.

### Justin Govanxa

* Set up the project structure for tab\_num, including organizing and managing imports in \_\_init\_\_.py to ensure proper initialization and integration of all components.
* Developed the user interface using Streamlit in display.py, creating an intuitive and interactive experience for selecting and analyzing numeric columns, displaying data summaries, and rendering visualizations such as histograms.
* Implemented the core logic in logics.py for data processing and analysis, including creating the NumericColumn class. Developed methods for computing statistical summaries (mean, median, standard deviation, etc.), handling missing values, and generating visual representations of the data.
* Streamlined data analysis by integrating efficient data handling and visualization features, resulting in a user-friendly and comprehensive data exploration tool.

### Yuzhong Wang

* Design and implement text column analysis logic: the core code in logics.py is written, and the statistical functions of the text column are constructed based on TextColumn class. This includes statistics such as identifying text columns, counting the number of unique values, missing values, null values, whitespace only rows, all lowercase and all uppercase rows, letters and numbers only rows, and calculating modes. The Altair library implements a bar chart of the distribution of text values, allowing users to visually see how often each text value occurs. The most common values are extracted and their occurrence times and percentages are calculated, which is convenient for users to analyze the common patterns in text columns.
* Develop Streamlit front-end display logic: In display.py, design the display\_tab\_text\_content function, implement the user interface logic. Use Streamlit components, including selectbox, expander, and table, to enable users to select specific text columns and view detailed statistics. It integrates the column chart and frequency table generated by the TextColumn class, and ensures that these data are displayed in the appropriate interface position, improving the user experience of the application.
* Solve file upload and data load problems: Adjust the code structure to ensure that the data set can be correctly loaded into session\_state after file upload, avoiding NoneType errors. In the project's main function, the file upload and DataFrame passing logic has been improved to ensure smooth display of text column statistics in Streamlit applications.

### Benedict Brunker

* Engineered the display and operation of the *Datetime Series* tab of the application.
* Helped to improve CSV uploading features to handle a wider variety of formats.
* Set up initial GitHub repository for collaboration.
* Helped with communication and co-ordination between team-members.
* Helped with writing and editing of final report.

## b. Group Dynamic

Group members brought their own skills, interests and expertise to the project, each contributing their own part. Our work was divided up equally, with team members developing different aspects of the overall project architecture, and collaborating to ensure the parts would cohere into a seamless whole. With different schedules and commitments, group members contributed their work when they could, and others were able to pick up the slack whenever a member was weighed down by other commitments.

## c. Ways of Working Together

The workflow of the project was split into four components, with each group member handling a sub-directory of the application structure corresponding to one of the four tabs in the application. Group members volunteered for certain branches of the application development based on their interests and skills.

We made use of Git for version control, committing and merging our changes consistently to ensure inter-operability between different aspects of the application and a detailed record of the workflow history.

We communicated primarily through a WhatsApp group to co-ordinate our efforts, plan project timelines, and provide and seek feedback and advice from team-members. Communication was consistent and ongoing.

Team members were able to devote time to the project based on their existing commitments and schedules.

## d. Issues Faced

One challenge we faced was in moving towards testing and debugging the application. Because the different parts of the application operate together as a cohesive whole, we needed at least an early beta-version of the application to be completed before each team member could get to work on debugging, refining and improving their contributions.

Another challenge was in ensuring consistent Python environments amongst all our members so that there would be consistency in the approach. We solved this by specifying a consistent *requirements.txt* file which could be shared among all members through GitHub.

There were competing preferences with regard to software to be used for collaborating on the present report. We shared versions of the report through GitHub, but also collaborated on a shared Google Docs draft of the report so that we could update, contribute and share work quickly and easily without having to worry about Git commits.

Finally, we faced challenges with regard to team members commitments outside the project. Three of the four team-members were working on additional projects with an identical deadline. This problem could be addressed by team-members completing and contributing assigned work for the project ahead of time, while team members with freer schedules could contribute more in the project’s later stages.

In future, we would improve collaboration in the following ways:

* Regular team meetings, either face-to-face or virtual, to provide updates, ensure deadlines and targets are met, and offer and solicit feedback from group members.
* Agreement on preferred software at an early stage of the project, to eliminate ambiguities and clashes with software packages.
* Agreement at an early stage of the project on consistent conventions for coding, such as variable names, file formats and package versions. Agreement on consistent coding styles for aspects of the project that involved similar or overlapping classes and functions, and consistent styles for doc-strings and comments.



# 6. Conclusion

The project succeeded in its primary goal of developing a deployable data explorer web application which provides users an accessible, interactive and visually appealing interface for uploading CSV files and gaining a high-level overview of key characteristics of their dataset. We hope this application will be of use to professionals, students and even amateur hobbyists in easily exploring their data. The application is built primarily for non-specialists, allowing them to easily explore, visualize and draw insights from their data. However, the application can also be of use to trained data scientists, by allowing them a quicker and easier alternative to custom-coded Exploratory Data Analysis.

In future, we hope to develop and release version 2.0 of the application, with which we hope to expand the range of file formats able to be uploaded to the application, handle a wider variety of possible table formats, and expand the range of data exploration options available to the user, for example with a wider array of possible visualizations and other ways of exploring data relationships. We invite users of our application to submit feedback on their experience with the application, suggest possible improvements, and report problems. We also invite the wider community of programmers on GitHub to use and adapt our project code, report problems, provide suggestions and help us build out future versions.

We also hope to host our app on the cloud, allowing it to be accessed through the World Wide Web, which will save users needing to download the project files manually and serving the application from their local machine.



# **7. References**

Nantasenamat, C. (2024, April 24). Building a dashboard in Python using Streamlit. Streamlit. <https://blog.streamlit.io/crafting-a-dashboard-app-in-python-using-streamlit/#bonus-5-reminders-when-building-dashboards>

Easonlai. (n.d.). GitHub - easonlai/chat\_with\_csv\_streamlit\_with\_chart: In this repository, you will find an example code for creating an interactive chat experience that allows you to ask questions about your CSV data with chart visualization capabilities. GitHub. <https://github.com/easonlai/chat_with_csv_streamlit_with_chart>.