

Assignment

3

Collaborative Development of Data Explorer Web App

Group 27

08-11-2024

Aditya Kulkarni (25410263)

Archit Pradip Murgudkar (14190286)

Rohan Rocky Britto (24610990)

Vishal Kandikattu (25413586)

https://github.com/rohanbrit/dsp_at3_group27

94692 - Data Science Practise
Master of Data Science and Innovation
University of Technology of Sydney

Table of Contents

1.	Executive Summary	2
2.	Introduction	3
3.	Web App Presentation	5
a.	Purpose of the Application and Main Functionalities	5
b.	Setup and Launch Instructions	5
c.	Potential Users, Use Cases, and Benefits	6
d.	Limitations and Future Improvements	6
4.	Reflecting On Building Data Product	7
5.	Collaboration	8
a.	Individual Contributions	8
b.	Group Dynamic	14
c.	Ways of Working Together	14
d.	Issues Faced	15
6.	Conclusion	16
7.	References	17



1. Executive Summary

The growth of the data industry has made it possible to explore and analyze data in many different ways. With modern tools, building web applications has become easier, allowing people to make informed decisions based on trends in the data.

Streamlit and Altair are two popular Python libraries for creating data visualizations. They help make data analysis faster and more flexible, which is useful for data scientists who need to build web applications for quick insights.

The goal of this assessment is to use Streamlit and Altair to create a web app for exploratory data analysis (EDA). This app will help users understand the data better and fix any issues it may have. The app aims to be a helpful tool for data scientists, giving them a clear understanding of datasets.



2. Introduction

This project focuses on creating a web application (using a code skeleton provided in the Assignment brief) (Kabir, n.d.) specifically designed to support data scientists. The application will allow them to input datasets and receive a detailed analysis in return. By processing and examining the data, the app will make it easier for data scientists to interpret patterns, trends, and significant details in the data. The goal is to create a tool that not only speeds up the data analysis process but also provides clearer, more accurate insights.

This will help data scientists and application developers make well-informed decisions, solve complex problems, and draw meaningful conclusions based on the data they are working with.

Data Scientists


- A way to upload CSV files and connect to databases.
- Tools to help understand data, like summaries and visualizations (e.g., charts and tables).
- Options to explore different types of data (numbers, text, dates) to find useful insights.
- An easy-to-use interface with clear options to filter data and focus on specific columns.

Application Developers

- Clear task assignments and a well-organized code structure to make collaboration easier.
- A private GitHub repository to share code and track changes as a group.
- A process to review and merge code without issues.
- Clear assignment instructions and support from lecturers when needed.
- A final report that documents the project structure and collaboration process.

The main parts of building this interactive web app are:

1. **Learning Streamlit and Altair:** Understanding these libraries to create and visualize the app.
2. **CSV Upload Menu:** Creating a simple menu where users can upload their CSV files.
3. **Four Analysis Tabs:**

- 
- Data Frame Analysis: Converting the CSV file to a DataFrame and showing a summary of the dataset.
 - Numeric Series Analysis: Helping users explore and understand the numbers in each numeric column.
 - Text Series Analysis: Allowing users to analyse text data and see useful results.
 - Datetime Series Analysis: Letting users view and analyse data within specific timeframes.

Private GitHub Repository: Setting up a private GitHub for the team to merge their code to the main branch.



3. Web App Presentation

a. Purpose of the Application and Main Functionalities

This application, developed with Streamlit, enables users to conduct exploratory data analysis on CSV files interactively. Its objective is to provide users with insights into dataset structure, statistical details, and distribution of values. It contains four main sections:

1. File Upload and DataFrame Overview:

- a. Users can upload a CSV file and view an overview, including the number of rows, columns, duplicated rows, and rows with missing values.
- b. A table displays column data types, memory usage, and interactive exploration of the dataset based on user-defined display options (head, tail, sample).

2. Numeric Series Analysis:

- a. Allows users to select any numeric column for further analysis, showing statistics like unique value count, missing values, zero/negative occurrences, mean, median, and standard deviation.
- b. Features an interactive histogram, box plot and scatter plot of values and a frequency table of the top 20 most common values.

3. Text Series Analysis:

- a. Detects text columns and displays information such as unique value count, missing rows, empty and whitespace-only rows, mode value, and count of alphabetic, numeric, lowercase, and uppercase-only rows.
- b. Includes an interactive bar chart for value occurrences and a table listing the top 20 values by frequency.

4. Datetime Series Analysis:

- a. Detects datetime columns (or prompts conversion of text columns). It provides insights like unique dates, weekend occurrences, future dates, and counts of historical dates like 1900-01-01.
- b. An interactive histogram and top 20 frequent dates table are also displayed.

b. Setup and Launch Instructions

1. **Install Requirements:** Ensure Python 3.9+ is installed on the system and run the following command to install all package dependencies:

```
pip3 install -r requirements.txt
```

2. **Run the Application:** Navigate to the app's directory and run the following command:



```
streamlit run .\app\streamlit_app.py
```

3. Access the app at <http://localhost:8501> in your browser.

c. Potential Users, Use Cases, and Benefits

This app is designed for data scientists, data analysts, researchers, business professionals, and educators needing quick, no-code data insights. Example use cases include:

- **Business Analysts:** Quickly examine sales or customer data.
- **Data scientists/Data analysts/Researchers:** Perform preliminary analysis on data.
- **Students:** Learn data analysis concepts hands-on.

Benefits include efficiency, accessibility for non-technical users, and support for faster, data-driven decisions. With further development, this tool could serve industry-specific needs, offering specialized analyses for fields like finance or healthcare. A hosted, subscription-based version could allow for secure, scalable use by multiple users.

d. Limitations and Future Improvements

Limitations: Currently, the app has limited customization for complex analyses, faces performance issues with large datasets, and could benefit from enhanced error handling.

Improvements: Adding machine learning models, richer visualizations, and user authentication could increase functionality and user reach.



4. Reflecting On Building Data Product

Developing data products is a vital skill for data scientists, transforming their work from isolated analyses to impactful, user-friendly tools that support business growth and automated decision-making. This capability is increasingly essential for career progression, as it enables data scientists to not only provide insights but also embed those insights into applications that non-technical stakeholders can access and utilize. Key skills include programming in languages like Python and SQL, statistical analysis, visualization with tools like Streamlit or Tableau, and the integration of machine learning models. These competencies empower data scientists to create diverse products, such as recommendation systems, predictive models, anomaly detectors, and interactive dashboards, adding value across industries.

With the rapid advancements in AI, data products are evolving into smarter, more adaptive tools capable of real-time data processing, personalization, and even autonomous decision-making, reshaping business and consumer experiences. This trend opens opportunities for innovative products tailored to both internal and commercial needs, such as dynamic customer segmentation tools, AI-driven financial forecasting applications, and interactive platforms for business intelligence. By developing data products, data scientists not only enhance their technical and collaborative capabilities but also play a direct role in advancing organizational goals, creating a pathway to career growth and positioning themselves at the forefront of AI-driven innovation.



5. Collaboration

a. Individual Contributions

In our 4-member team, each member took responsibility for a unique section of the interactive web application, contributing specific expertise to create a comprehensive data analysis tool.

Aditya Kulkarni (25410263):

Role: Developer of the Overall Dataframe Analysis Tab (tab_df)

Aditya was responsible for creating and implementing the DataFrame tab in the CSV Explorer web application. His contributions focused on both the user interface (UI) design and the backend logic to display and analyze the dataset in an interactive way.

- Designing the User Interface

Aditya's first task was to design the user interface for the DataFrame tab. He created a section to show overall dataset information, such as the number of rows, columns, duplicated rows, and rows with missing values. This gives users an immediate understanding of the dataset's size and quality. He also added a table displaying column names, their data types (e.g., text, numeric, date), and their memory usage. This helped users understand the structure of the data.

To make the application interactive, Aditya included a slider that allows users to select how many rows to display (between 5 and 50). Additionally, he implemented a radio button for users to choose how the rows should be displayed: either the first rows (head), last rows (tail), or a random sample. These features provided users with flexibility in exploring the dataset.

- Implementing the Logic

In the backend, Aditya worked on the logic that powers the DataFrame tab. He used the ``Dataset`` class from ``tab_df/logics.py`` to manage the dataset and calculate important statistics, like the number of duplicates and missing values. By saving the dataset in Streamlit's session state, Aditya ensured that the data remained accessible across the app's various sections.

- Testing and Debugging

After building the UI and logic, Aditya thoroughly tested the features to ensure everything worked as expected. He verified that dataset information, such as rows and columns, was accurate and that the interactive elements, like the slider and radio button, functioned properly.

Overall, Aditya played a crucial role in making the DataFrame tab user-friendly and dynamic. His work on the interface and backend logic allowed users to easily explore and understand their datasets, making it a core feature of the app.

Archit Pradip Murgudkar (14190286):

Role: Developer of the Numeric Analysis Tab (tab_numeric)

Primary Responsibility:

Archit was responsible for designing and implementing the Numeric Analysis tab, one of the core functionalities of the project. This section provided users with a comprehensive analysis of numeric data columns in their dataset. Below is a detailed description of Archit's tasks, responsibilities, and achievements.

- Key Tasks and Responsibilities:
- Development of the NumericColumn Class:

Objective: Create a class to manage the analysis of numeric columns within a DataFrame.

Implementation:

- Developed methods to compute key statistical metrics such as unique values, missing values, zero counts, mean, standard deviation, minimum, maximum, and median.
- Ensured that these methods were efficient and could handle large datasets without performance degradation.
- Incorporated error handling and data validation to prevent issues when processing columns with missing or non-numeric data.

Creation of Visualization Methods:

- Interactive Histogram: Implemented a method to generate a histogram using Altair, providing users with an interactive visualization of the distribution of values in a numeric column.
- Box Plot (set_boxplot() Method): Added a method to create an Altair-based box plot that allows users to view the distribution, quartiles, and outliers of a selected numeric column. Designed the plot to be interactive and customizable for a more engaging user experience.
- Scatter Plot (set_scatterplot() Method): Implemented a method to generate scatter plots for comparing the selected numeric column with another column chosen by the user. Ensured

that the scatter plot was interactive, with tooltips for detailed data inspection, enhancing the user's ability to identify correlations or trends in the dataset.

Integration with Streamlit (`display_tab_num_content()` Function):

Objective: Connect the numeric analysis functionalities with the Streamlit app interface.

Implementation:

- Enhanced the `display_tab_num_content()` function to display interactive visualizations including histograms, box plots, and scatter plots.
- Utilized Streamlit widgets to allow users to choose columns for visualization dynamically.
- Ensured smooth integration of visualizations with the existing Streamlit application structure, allowing users to interact with plots and explore different numeric columns easily.

Comprehensive Data Exploration Features:

- Value Frequency Analysis: Developed a method to calculate the frequency of unique values in the selected column and display the top results in a tabular format.
- Statistics Summary: Implemented functionality to compile all calculated statistics into a summary DataFrame that is displayed in the Streamlit app for quick review.

Testing and Optimization:

- Performance Checks: Ran extensive tests on different datasets to ensure that the methods in `NumericColumn` handled edge cases such as columns with missing values, zeros, or non-numeric data types. Optimized code for faster execution and better memory handling.
- User Feedback and Iteration: Incorporated feedback from peers and initial users to refine the visualizations and make the UI more intuitive.

Notable Achievements:

- Enhanced User Engagement: The addition of interactive box plots and scatter plots significantly improved user engagement by making data exploration more dynamic and insightful.
- Modular and Reusable Code: The design of the `NumericColumn` class was modular, allowing future developers to extend the class easily or repurpose the visualization methods for other projects.
- Robust Error Handling: Implemented robust error-checking mechanisms to handle issues like missing data or invalid column selections gracefully, improving the reliability of the app.

- **Comprehensive Documentation:** Provided detailed documentation and docstrings for each method to ensure that other team members and future developers could understand and use the code efficiently.

Summary of Archit's Contributions:

Archit's work on the Numeric Analysis tab was crucial for the overall functionality and effectiveness of the Streamlit app. By implementing advanced numeric analysis features and interactive visualizations, Archit transformed the tab into a comprehensive tool for exploring numeric data. The integration of the NumericColumn class with the Streamlit interface allowed users to interact with and analyze their data in a meaningful way. These contributions not only added depth to the application but also set a solid foundation for future enhancements and scalability.

Rohan Rocky Britto (24610990):

Role: Developer of the Text Analysis Tab (tab_text)

- **Primary Responsibility:**

Rohan was responsible for developing the Text Analysis section of the application, ensuring functionality to analyze text columns for unique values, missing data, character patterns, and statistical insights.

- **Key Tasks and Responsibilities:**
 - **Design and Implementation of Text Analysis:** Built functionality to detect key features in text columns, including identifying unique values, counting rows with missing or whitespace-only entries, and calculating character-based insights like uppercase, lowercase, alphabet-only, and digit-only rows.
 - **Data Visualization for Text Patterns:** Integrated Altair charts to dynamically visualize the distribution and frequency of text values, enabling users to better interpret the patterns within text data.
 - **Collaboration and Code Management:** Set up the project's GitHub repository, enabling organized version control and ensuring easy access for all team members to share updates and improvements.
 - **Peer Review:** Performed peer review of code to ensure consistency and accuracy of code.
- **Notable Achievements:**

Successfully designed the interactive text analysis features to ensure comprehensive and accurate insights. Facilitated smoother collaboration by creating and maintaining the GitHub repository, which served as the central codebase for our project.

- Summary of Rohan's Contributions:

In this project, I focused on developing the Text Analysis section, bringing together my technical and analytical skills to create a robust and interactive solution for analyzing text-based data. Beyond coding, I contributed equally to our report, documenting the Text Analysis functionality and assisting in presenting our collective findings. Setting up the GitHub repository and code peer reviewing was also a critical contribution, helping our team coordinate efficiently and integrate each section smoothly into the final application. Through my contributions, I aimed to enhance the application's ability to deliver deep insights and value in exploratory text data analysis.

Vishal Kandikattu (25413586):

Role: Developer of the Date Analysis Tab (tab_date)

Vishal's responsibility mainly involves designing the user interface and implementing the logic for displaying information related to date columns in the selected dataset.

1. Text Column Tab (display.py):

Vishal created the user interface elements for the date column analysis. The main components he worked on include:

- **Select Box:** A drop-down menu that lets users choose a date column from the dataset to analyze.
- **Expandable Container:** A section that expands to show the results of the date column analysis.
- **Summary Table:** A table that presents key details, such as:
 - The number of unique date values.
 - The number of missing values in the date column.
 - The count of weekend and weekday dates.
 - Dates that appear in the future.
 - The minimum and maximum dates.
 - The number of rows with specific dates like '1900-01-01' and '1970-01-01'.
- **Bar Chart:** A visual representation of the count of each unique date value in the selected column.
- **Most Frequent Values:** Displays the most common dates in the column, along with their counts and percentages.

2. Logic Implementation (logics.py):

Vishal was responsible for the backend logic that handles the date column analysis. His tasks involved:

- Creating an Instance of DateColumn: He used the DateColumn class from `tab_date/logics.py` to manage the selected date column effectively.
- Date Column Detection: He created a method to detect and list date columns present in the uploaded dataset.
- Extracting and Analyzing Date Data: Vishal wrote code to compute various statistics about the selected date column, such as counting unique values, handling missing data, and identifying future dates.
- Visual Representation: He built a bar chart using Altair to show the frequency of each unique date value.
- Frequent Values Data: Vishal constructed a data frame that presents the most frequently occurring date values, including their counts and percentages.

3. Testing and Debugging:

Vishal thoroughly tested the functionality of the date column analysis to ensure the accuracy of the computed statistics and visualizations. He focused on making sure the bar chart was correctly displayed and the correct most frequent date values were shown.

4. Documentation:

Vishal documented the `display_tab_date_content` function. He clearly explained its purpose, the parameters it takes, and what it returns. This documentation helps future developers understand how the function works and how to use it in the project.

5. Challenges:

Vishal faced some challenges while working on this feature. One of the main issues was converting the date values into the `int64` format to plot the bar charts. He also worked on identifying the datetime columns in the dataset, which required careful handling of different formats.

Overall, Vishal made significant contributions to the CSV Explorer web app project, particularly in the development of the date column analysis feature. He was responsible for designing the user interface, including a select box for users to choose a date column, and an expandable container to display analysis results. Vishal implemented the core logic to detect and manage date columns, calculate various statistics, and generate visualizations like bar charts. He also thoroughly tested and debugged the app to ensure accuracy. Despite facing challenges, such as converting date values and identifying datetime columns, Vishal's work helped make the app functional and user-friendly.

b. Group Dynamic

Our team exhibited a highly collaborative and supportive group dynamic, which was crucial for the successful completion of our Streamlit web application project. We communicated primarily through MS Teams, where we used audio and video calls with screen-sharing capabilities. This allowed us to have real-time discussions and adjust instantly, improving the speed and clarity of our communication.

Each team member shared their strengths during our initial meetings, enabling us to assign tasks that aligned with individual expertise while also encouraging everyone to explore and learn new skills. This approach of aligning tasks with strengths created a balanced workload distribution, and it allowed us to work more efficiently.

Continuous feedback was a key practice we adopted. We frequently checked in with each other, shared updates, and provided feedback on each other's work. This open feedback loop fostered an environment of trust and accountability, ensuring that any issues were promptly addressed and adjustments were made to stay aligned with our project goals.

c. Ways of Working Together

We adopted a bi-weekly meeting schedule to discuss our progress, set goals for the upcoming weeks, and address any challenges. These meetings were structured to cover each team member's progress, share feedback, and make decisions collectively. Between meetings, we stayed in touch through MS Teams chats to address any immediate queries and track ongoing work.

For project management, we used GitHub for version control and code management. Visual Studio Code served as our primary development environment, and GitHub Desktop simplified the process of managing and pushing our code to the repository. Our team used GitHub's branching feature for a smoother code integration process, allowing each member to work on their assigned parts without affecting others. We merged our code into the main branch only after review, ensuring consistency and quality.

Our project management methodology emphasized agility. We used MS Teams for daily discussions on any blockers, upcoming tasks, and timelines. In addition to our bi-weekly meetings, we also had quick ad hoc meetings whenever there were changes in scope or new requirements. This agile and adaptive approach allowed us to make iterative progress on the project, ensuring alignment with the project requirements while allowing flexibility to incorporate improvements and resolve issues as they arose.



d. Issues Faced

1. User Interface (UI) Design

- Challenge: Designing an intuitive and user-friendly interface that accommodated various functionalities while maintaining simplicity was a major task.
- Resolution: The team held brainstorming sessions and collected feedback to iterate on the UI design, focusing on clarity and ease of navigation. Mock-ups were developed before final implementation.
- Lesson Learned: Engaging all team members in web design planning proved beneficial. Having multiple perspectives enriched the process, resulting in a more cohesive and user-centered product.

2. Coordination and Version Control

- Challenge: Coordinating development efforts, particularly with multiple team members working on different aspects of the application simultaneously, led to conflicts in version control.
- Resolution: The team standardized the workflow by using branching strategies in Git. Regular team check-ins helped sync progress and manage dependencies.
- Lesson Learned: Establishing clear version control guidelines and maintaining a well-documented workflow is essential for seamless collaboration in group projects.



6. Conclusion

a. Key Findings and Outcomes

- The CSV Explorer web app successfully achieved its objective of allowing users to perform comprehensive exploratory data analysis. It facilitated insights into the dataset through well-structured tabs that provided general overview statistics, numeric analysis, text analysis, and datetime analysis.
- The app's interactive visualizations, supported by Altair, significantly enhanced the user's ability to interpret data visually. The tool also catered to different user needs by enabling flexible data display through filters and options.

b. Project Success and Meeting Requirements

- The project met all specified requirements, including interactive exploration of data with user-friendly components like sliders, selection boxes, and radio buttons.
- Feedback from peer reviews suggested that the app was highly intuitive and informative, demonstrating that the project goals aligned well with user expectations.

c. Future Work and Recommendations

- Future Enhancements: Expanding the app's functionality to support additional file formats and more complex EDA tools.
- Recommendations: Implementing real-time collaboration features for teams working with the same dataset and introducing more robust machine learning-based data cleaning tools.
- Next Steps: Conduct further user testing to refine the interface and incorporate more customization options for analysis reports.

In conclusion, the collaborative development of the CSV Explorer web app provided valuable insights into handling real-world data analysis challenges through interactive and responsive design. Lessons learned around effective team coordination, modular code design, and user-centric features will inform future projects, enhancing overall efficiency and project success.



7. References

Kabir, M. (n.d.). Collaborative Development of Data Explorer Web App. Retrieved October 25, 2024, from <https://canvas.uts.edu.au/courses/33018/assignments/200404>

