Exploratory Data Analysis Tool

BITE497J – Project I

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Department of Information Technology

by

APPANA VENKATA RAJU SAI

21BIT0589

Under the guidance of

Prof: Dr. Ranichandra C

School of Computer Science Engineering and Information Systems

VIT, Vellore

November 2024

Exploratory Data Analysis Tool

BITE497J – Project I

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Department of Information Technology

by

APPANA VENKATA RAJU SAI

21BIT0589

Under the guidance of

Prof: Dr. Ranichandra C

School of Computer Science Engineering and Information Systems

VIT, Vellore

November 2024

DECLARATION

We hereby declare that the BITE497J – Project I thesis entitled **"EXPLORATORY DATA ANALYSIS TOOL"** submitted by **Appana Venkata Raju Sai(21BIT0589)** for the award of the degree of *Bachelor of Technology in Department of Information Technology, School of Computer Science Engineering and Information Systems* to VIT is a record of bonafide work carried out by me under the supervision of **Dr. Ranichandra C, Associate Professor Grade 2, SCORE, VIT, Vellore.**

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place: Vellore

Date: 19th Nov 2024

Candidate: Appana Venkata Raju Sai

CERTIFICATE

This is to certify that the BITE497J – Project I thesis entitled **"EXPLORATORY DATA ANALYSIS TOOL"** submitted by **Appana Venkata Raju Sai(21BIT0589)** School of Computer Science Engineering and Information Systems, VIT, for the award of the degree of *Bachelor of Technology in Department of Information Technology, School of Computer Science Engineering and Information Systems*, is a record of bonafide work carried out by him under my supervision during the period, 15. 07. 2024 to 30.11.2024, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.

Place : Vellore

Date : 19th Nov 2024

Signature of the Company Guide

Signature of the VIT-SCORE - Guide

Internal Examiner **External Examiner**

Head of Department

Department of Information Technology

ACKNOWLEDGEMENT

It is my pleasure to express with a deep sense of gratitude to my **BITE497 - Project I** guide **Dr. Ranichandra C, Associate Professor Garde 2,** School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore for **her** constant guidance, continual encouragement, in my endeavor. Our association with **her** is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and an expert in the field of **Data Analytics.**

"I would like to express my heartfelt gratitude to Honorable Chancellor **Dr. G Viswanathan**; respected Vice Presidents **Mr. Sankar Viswanathan**, **Dr. Sekar Viswanathan**, Vice Chancellor **Dr. V. S. Kanchana Bhaaskaran**; Pro-Vice Chancellor **Dr. Partha Sharathi Mallick**; and Registrar **Dr. Jayabarathi T**.

My whole-hearted thanks to Dean **Dr. Sumathy S**, School of Computer Science Engineering and Information Systems, Head, Department of Information Technology, **Dr. Prabhavathy P**, Information Technology Project Coordinator **Dr. Sweta Bhattacharya** & **Dr. Praveen Kumar Reddy**, SCORE School Project Coordinator **Dr. Srinivas Koppu**, all faculty, staff and members working as limbs of our university for their continuous guidance throughout my course of study in unlimited ways

It is indeed a pleasure to thank my parents and friends who persuaded and encouraged me to take up and complete our **Exploratory Data Analysis Tool** successfully. Last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of the Exploratory Data Analysis Tool.

Place: Vellore
Date: 19-11-2024

Appana Venkata Raju Sai

Executive Summary

This capstone project presents a web application designed to simplify Exploratory Data Analysis through a user-friendly interface. Developed using Streamlit, this tool allows users to upload datasets (CSV and XLSX) and generate a range of dynamic visualizations, including histograms, scatter plots, box plots, and heatmaps, among others. This application supports data analysts by enabling quick exploration of data trends, distributions, and relationships, ultimately uncovering actionable insights.

Key features include dataset upload functionality, allowing seamless analysis of custom datasets across domains, and interactive visualizations that make data insights more accessible. Additionally, customization options empower users to personalize visualizations, with settings for colour schemes and downloadable plots, enhancing clarity and presentation flexibility.

By automating the EDA process, this application streamlines workflows, saving time and reducing reliance on complex coding. Designed for data science and analytics professionals, it meets the need for accessible, interactive tools that facilitate data-driven decisions. The platform bridges the gap between raw data and meaningful analysis, offering a practical solution to enhance the efficiency of data exploration.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

EDA	Exploratory Data Analysis
CSV	Comma Separated Values
XML	Extensible Markup Language
Pandas	Python Data Analysis Library
XLSX	Excel Open XML Spreadsheet
NumPy	Numerical Python
PEP	Python Enhancement Proposal
AWS	Amazon Web Services
UI	User Interface
VIT	Vellore Institute of Technology

CHAPTER 1
INTRODUCTION

1.1 Objective

The primary objective of this project is to revolutionize the way professionals approach Exploratory Data Analysis (EDA) through the development of a sophisticated yet user-friendly web application. At its core, the platform aims to transform complex data analysis tasks into intuitive, streamlined processes that can be executed with minimal technical expertise. By leveraging modern web technologies and advanced visualization techniques, the application serves as a bridge between raw data and actionable insights, making data analysis accessible to users across various skill levels and professional backgrounds.

The platform's comprehensive visualization capabilities stand at the forefront of its design objectives. Users can generate and manipulate a wide array of visual representations, from basic histograms and scatter plots to complex correlation matrices and interactive heatmaps. This visual-first approach enables analysts to quickly identify patterns, anomalies, and relationships within their datasets, significantly reducing the time traditionally required for initial data exploration. The application's automated analysis features further enhance this efficiency by handling routine tasks such as data cleaning, type inference, and basic statistical calculations automatically.

Beyond its technical capabilities, the platform serves as an educational tool, helping users develop a deeper understanding of statistical concepts and data analysis techniques through hands-on interaction. The interactive nature of the visualizations, combined with intuitive controls and real-time updates, creates an environment conducive to learning and experimentation. This educational aspect makes the tool particularly valuable in academic settings and professional training contexts, where understanding the principles of data analysis is as important as generating results.

1.2 Motivation

The motivation behind this project stems from a growing recognition of the challenges faced by modern organizations in extracting meaningful insights from their data. Traditional approaches to data analysis often require extensive programming knowledge and familiarity with complex statistical tools, creating a significant barrier to entry for many potential users. This technical hurdle not only limits the pool of individuals who can effectively analyze data but also slows down the decision-making process in organizations that increasingly rely on data-driven insights.

In today's fast-paced business environment, the ability to quickly explore and understand data has become crucial for maintaining competitive advantage. However, existing tools often require significant setup time, specialized training, and substantial financial investment. Our platform addresses these challenges by providing an immediately accessible, web-based solution that combines powerful analysis capabilities with an intuitive user interface. The motivation extends beyond mere technical efficiency; it encompasses a vision of democratizing data analysis and enabling professionals across different domains to leverage the power of their data effectively.

The rapid growth in data availability and the increasing emphasis on data-driven decision-making across industries has created an urgent need for more accessible analysis tools. Organizations are generating and collecting more data than ever before, but the gap between data collection and data understanding continues to widen. This project is motivated by the opportunity to bridge this gap, providing a platform that transforms raw data into meaningful insights without requiring extensive technical expertise or resource investment.

1.3 Background

The landscape of data analysis has evolved significantly over the past decades, moving from manual calculations and basic statistical software to sophisticated analytical platforms. Traditional methods of conducting Exploratory Data Analysis often involved writing extensive code in languages like R or Python, requiring users to possess both programming skills and statistical knowledge. This historical context has shaped the

development of modern analysis tools, highlighting the need for more accessible alternatives that maintain analytical rigor while reducing technical complexity.

Recent technological advancements, particularly in web technologies and visualization libraries, have created new opportunities for developing interactive, browser-based analysis tools. The emergence of frameworks like Streamlit, combined with powerful data processing libraries such as Pandas and NumPy, has made it possible to create sophisticated analysis platforms that run entirely in the web browser. These technological developments form the foundation of our project, enabling us to deliver professional-grade analysis capabilities through an accessible, user-friendly interface.

The current data analysis ecosystem is characterized by a diverse range of tools, from traditional statistical packages to modern business intelligence platforms. However, many of these solutions either prioritize power over accessibility or simplicity over functionality. Our project builds upon this background by recognizing the need for a balanced approach that combines robust analytical capabilities with intuitive user experience. The platform leverages modern web technologies and visualization techniques while incorporating best practices from traditional statistical analysis methods.

Looking toward the future, the field of data analysis continues to evolve with emerging technologies and changing user expectations. Machine learning integration, advanced visualization techniques, and automated insight generation are becoming increasingly important. Our platform is designed with this future in mind, incorporating a flexible architecture that can adapt to new technologies and methodologies as they emerge. The background of this project reflects not only current best practices in data analysis but also anticipates future developments in the field, ensuring the platform remains relevant and valuable as technology continues to advance.

This comprehensive background underscores the significance of developing an accessible, powerful EDA platform that can serve the needs of modern organizations while preparing them for future challenges in data analysis. By understanding the historical context, current landscape, and future trends, we have created a solution that addresses immediate needs while maintaining the flexibility to evolve with the field.

1.4 Literature Review

The academic performance of students, particularly in challenging core subjects like Mathematics and Reasoning, remains a critical area of concern, as highlighted in various studies. Recent advancements in machine learning and data mining techniques offer promising solutions for analysing educational datasets and predicting student success. [1] Kumar (2021) conducted a survey leveraging the student performance dataset from the UCI Machine Learning Repository, utilising Python and Jupyter Notebook for exploratory data analysis and grade prediction. This work demonstrates the potential of machine learning models to extract meaningful insights from raw data, providing a data-driven approach to understanding and improving academic outcomes. The study underscores the importance of technological intervention in identifying key factors influencing student performance and devising strategies to address learning challenges.

Various studies highlight the potential of data analysis and machine learning in addressing domain-specific challenges. For instance, Kumar (2021) utilized machine learning techniques to predict student performance, demonstrating how exploratory data analysis can uncover meaningful insights from educational datasets. Similarly, Da Silva et al. [16] introduced *Dominoes*, an interactive data exploration tool designed for project managers and developers to analyze software relationships. Unlike traditional approaches that rely on post hoc analyses or complex scripts, *Dominoes* enables users to interactively investigate project data, visualize intermediate results, and save exploration paths for reuse. Through scenario-based evaluations, the tool achieved an 86% success rate, highlighting its effectiveness in answering project-specific questions within constrained timeframes. Together, these works illustrate the growing role of advanced data analysis tools and machine learning in providing actionable insights across diverse domains, including education and software engineering.

Hassan-Montero et al. [17] introduced *SCImago Graphica*, a no-code tool designed to simplify the creation of complex data visualizations through drag-and-drop interactions. By allowing users to bind data variables to various encoding channels and adjust their settings, the tool generates interactive graphical displays that are suitable for both exploratory data analysis and data communication. The study evaluates the tool's expressiveness and ease of use through diverse examples and a visualization

catalog, demonstrating its capability to quickly and efficiently produce a wide range of data visualizations. *SCImago Graphica* addresses the long-standing challenge of combining high expressive power with user-friendliness, making it a valuable resource for data exploration and presentation.

Palocsay et al. [18] proposed using Microsoft Excel as an introductory tool for business intelligence (BI) and decision support system (DSS) applications, emphasizing its accessibility and versatility for managers and business students. The study highlights three key capabilities of Excel: manipulating records as a database, creating PivotTables® and PivotCharts® for data analysis, and importing data as an automation container. These foundational skills enable users to leverage Excel's potential for exploratory data analysis and support information discovery in business contexts. By demonstrating how Excel can serve as a gateway to advanced BI tools, the article underscores its importance in fostering data-driven decision-making across organizations.

Mackeprang et al. [19] introduced *Kaleidoscope*, an exploratory data analytics tool designed to assist in the critical and complex task of evaluating and selecting ideas during collaborative ideation. By integrating automatic computational methods with human sensemaking, *Kaleidoscope* allows users to explore and annotate ideas interactively. The tool’s design principles emphasize semantic technologies to support intuitive data exploration. Based on qualitative feedback on a prototype, the study identified potential enhancements and outlined directions for future development. This work highlights the importance of combining computational efficiency with human judgment to ensure valuable ideas are not overlooked in ideation processes.

Furcila [20] proposed *InTool Explorer*, an innovative interactive exploratory data analysis tool designed to address the complexities inherent in studying Alzheimer's disease (AD). AD is characterized by progressive cognitive decline and diverse neuropathological changes that vary across brain regions and between patients, making data analysis challenging. *InTool Explorer* integrates multidisciplinary data, including histological, neuropsychological, and clinical variables, into a unified platform. In a study using data from 11 AD patients, the tool enabled the identification of regional differences in tau and amyloid protein expression, as well as patient-specific characteristics related to disease progression. This interactive tool demonstrates the

potential to uncover novel insights beyond conventional statistical analysis, improving neuroscientific research productivity and enhancing understanding of AD.

Pitroda [21] introduced *QuickViz*, an interactive web application tool designed to automate the Exploratory Data Analysis (EDA) phase of the data science lifecycle. Data science has become integral across industries, requiring efficient methods for deriving insights from complex datasets. EDA, a crucial early phase, helps identify irregularities and uncover patterns in data but can be time-intensive when performed manually. *QuickViz* simplifies this process by offering a user-friendly, no-code platform that allows users to upload datasets and explore descriptive and graphical EDA techniques. The tool's web-based dashboard facilitates the visualization of findings, making it accessible to stakeholders without prior programming knowledge. By conserving time and enhancing accessibility, *QuickViz* empowers users to derive insights efficiently and present them effectively.

Otero-Escobar and Velasco-Ramírez [22] emphasized the pivotal role of Exploratory Data Analysis (EDA) in educational data mining, highlighting its application in analyzing academic achievement among high school students in Veracruz, Mexico. By utilizing tools like Google Colaboratory and Python libraries such as Pandas, Numpy, Matplotlib, and Seaborn, the study identified key variables influencing academic performance while ensuring data integrity. EDA techniques, including histograms, provided insights into data dispersion, concentration, and outliers, offering actionable recommendations for adjusting teaching strategies and educational tools. This research underscores how EDA not only enhances understanding of student data but also supports interventions aimed at reducing dropout rates and improving academic outcomes.

Chen et al. [23] introduced WhatsNext, an interactive notebook framework designed to enhance exploratory data analysis (EDA) for users with limited coding expertise. The framework aims to support low-code visual data exploration by integrating a recommendation panel in notebook cells, suggesting potential next-step exploration questions or actions. Additionally, WhatsNext features a dynamic tree visualization, allowing users to easily trace analytic dependencies between notebook cells. This approach addresses the challenge of disorganized data analysis workflows in traditional computational notebooks, offering a structured, insight-driven

environment for EDA, which enhances both the accessibility and effectiveness of data exploration.

In an era marked by rapid technological advancements, individuals strive to keep their knowledge and skills up-to-date, often at the expense of their health and well-being. While physical fitness is often considered a priority for maintaining a healthy lifestyle, eating habits—the cornerstone of nourishment—are frequently neglected. According to RamyaSri et al. [9], food provides the essential proteins and minerals necessary to sustain productivity and overall health. Despite this, global obesity rates have risen by 27.5% over the past 33 years, highlighting the widespread disconnect between perceived and actual healthy eating practices. This study investigates generational views on healthy lifestyles by analyzing daily eating habits and individuals' perceptions of their health, aiming to uncover the gap between self-reported and actual dietary behaviours.

Exploratory Data Analysis (EDA) plays a pivotal role in understanding and preparing data for predictive modeling. Deming et al. [24] elaborate on the advantages of EDA, emphasizing its utility in preprocessing data and setting a foundation for statistical and machine learning models. By illustrating how predictive modeling techniques vary in performance when applied to the same dataset, the study underscores the importance of proper data exploration and validation. Through detailed discussions on data comprehension, preparation, and visualization, this work highlights EDA's significance in ensuring robust analytical outcomes and forming the basis for effective business analytics.

Exploratory Data Analysis (EDA) is a cornerstone of the data preparation and cleaning process, often consuming a significant portion of the entire statistical analysis workflow. Rahmany et al. [25] emphasize the iterative and dynamic nature of EDA, highlighting its role in uncovering patterns, detecting anomalies, testing hypotheses, and validating assumptions through descriptive statistics and graphical tools. The study also underscores the accessibility of EDA, which allows both statisticians and non-statisticians to derive actionable insights from data. Despite its longstanding presence in data analysis, EDA remains a critical area of research, with continuous advancements and applications reinforcing its importance in modern data science practices.

The analysis of resistomes in complex microbial communities, such as the human microbiome, has been significantly advanced by whole metagenomic sequencing. However, despite the development of numerous pipelines for data processing and annotation, the lack of user-friendly tools for visual, statistical, and functional analysis remains a key bottleneck. Dhariwal et al. [26] address this gap with ResistoXplorer, a web-based tool designed to simplify high-throughput analysis of resistome data. ResistoXplorer integrates statistical and visualization advancements with functional annotations to streamline the exploration of antimicrobial resistance data. It features three modules: the *Antimicrobial Resistance Gene Table* for profiling and comparative analysis, the *Integration* module for exploratory analysis of resistome and microbiome profiles, and the *Antimicrobial Resistance Gene List* for network visual analytics to identify associations between resistance genes and microbial hosts. This tool lowers the technical barrier for resistome analysis, facilitating biological insights and driving progress in antimicrobial resistance research.

1.5 Rationale

The project stems from the growing importance of exploratory data analysis in deriving insights from datasets. In today's data-driven world, understanding patterns, trends, and anomalies is critical for making informed decisions in various domains such as business, healthcare, education, and technology. However, performing EDA often requires proficiency in coding, data manipulation, and visualization tools, which can pose a barrier for non-technical users. This project seeks to address this gap by providing a user-friendly web-based application that streamlines the EDA process.

The primary goal of the application is to simplify the data analysis process while still providing powerful visualization tools. By leveraging the capabilities of libraries like Plotly, Pandas, and NumPy, the project enables users to upload datasets, perform missing value and datatype analysis, and generate insightful visualizations such as histograms, bar charts, scatter plots, and heatmaps. The application also allows for the analysis of correlations among numerical features, making it a comprehensive tool for data exploration. Its accessibility, through support for common file formats like CSV and XLSX, makes it adaptable to various use cases, including business analytics, research, and education. Additionally, the dynamic navigation system ensures that users can seamlessly switch between different pages to perform diverse types of analysis.

The project aims to cater to business analysts, researchers, and data enthusiasts who require a quick, accessible solution for data exploration, without needing to delve into complex coding or use specialized software. It empowers users to derive valuable insights from datasets quickly, making it a useful tool for informed decision-making.

1.6 Gaps Identified

A key gap identified in the industry is the lack of easy-to-use, interactive EDA tools that do not require coding skills. While there are powerful data science platforms and programming languages such as Python and R, these require a high level of expertise to leverage effectively. Most commercial or open-source EDA tools are complex, requiring users to understand programming languages or statistical concepts. As a result, businesses or researchers without technical teams often struggle to perform initial data exploration. Tools like Streamlit and Plotly, which provide an accessible interface for non-technical users, are still relatively rare, creating a gap in the market for more intuitive data exploration tools.

Many existing visualization tools lack the level of customization required for users to tailor charts and graphs to their specific needs. While many tools offer basic charts such as histograms and pie charts, users often cannot adjust finer details such as colors, and axis scaling. This can result in visualizations that do not meet the specific requirements of users or organizations. In many professional settings, visualizations must adhere to strict branding guidelines or be customizable to specific data nuances. The lack of these options is a key gap in many existing tools, limiting their overall utility for professional reporting and presentations.

The use of data analysis tools on mobile devices is another gap in the industry. Many data exploration platforms are optimized for desktop use, but with the rise of mobile-first usage and remote working, the ability to access these tools on mobile devices is becoming increasingly important. Existing solutions often do not provide responsive or mobile-optimized versions of their platforms, making them difficult to use on smaller screens. Given the increasing reliance on mobile devices for various business functions, optimizing EDA tools for mobile access is an essential gap that needs to be addressed in the market.

CHAPTER 2

PROJECT DESCRIPTION AND GOALS

2.1 Project Overview

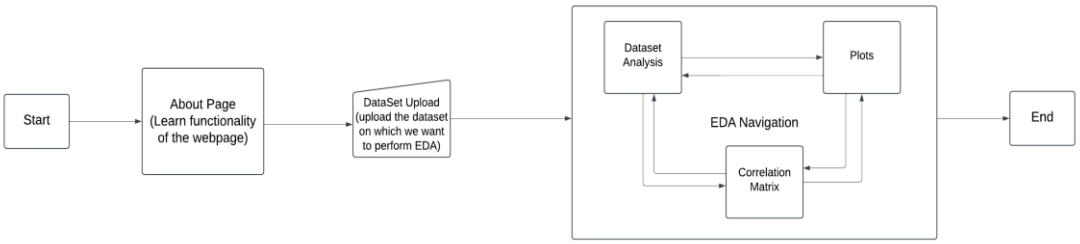

Fig. 2.1 Flowchart of the project

This project focuses on developing an advanced yet user-friendly web application designed to revolutionize the Exploratory Data Analysis (EDA) process for data analysts and professionals. Built using Streamlit, a powerful Python-based framework, the application delivers a seamless experience for users to upload datasets, generate meaningful visualizations, and explore data trends with minimal effort. The platform combines sophisticated analysis capabilities with an intuitive interface, making advanced data analysis accessible to users regardless of their programming expertise.

The application's architecture leverages the robust capabilities of Python's data science ecosystem, incorporating libraries such as Pandas for data manipulation, Plotly for interactive visualizations, and NumPy for numerical computations. This technological foundation ensures reliable performance while handling diverse datasets and complex analytical requirements. The platform supports various file formats, including CSV and XLSX, with built-in validation and error handling to ensure data integrity throughout the analysis process.

The application supports a comprehensive suite of visualization tools commonly used in EDA, each serving specific analytical purposes:

- Histograms: Enable users to understand the distribution of numerical data, identify patterns, and assess data normality. Users can adjust bin sizes, add density curves, and compare multiple distributions simultaneously.

- Scatter Plots: Facilitate the identification of relationships and correlations between variables. The plots support features such as trend line fitting, confidence intervals, and colour coding by categorical variables to enhance pattern recognition.
- Box Plots: Provide robust tools for detecting outliers and examining data spread. These visualizations include detailed statistical summaries, customizable whisker ranges, and the ability to compare multiple variables side by side.
- Heatmaps: Allow users to explore correlation matrices and visualize relationships between multiple variables simultaneously. The heatmaps feature customizable colour schemes, interactive cell information, and hierarchical clustering options.

2.2 Key Goals

1. Develop a User-Friendly Interface: The project aims to create a simple, intuitive interface that allows users to perform EDA without writing code. The design emphasizes clear navigation, logical workflow progression, and consistent interaction patterns, making the application accessible to a broader audience while maintaining powerful analytical capabilities.

2. Provide Dynamic, Interactive Visualizations: Users can interact with their data in real time, adjusting various parameters such as colour schemes, plot types, and statistical measures. This interactivity enhances data exploration by allowing immediate visualization of different aspects of the dataset, supporting deeper understanding and insight generation.

3. Automate Key EDA Tasks: The application streamlines the data analysis process by automating the generation of essential visualizations and statistical summaries. This automation reduces manual effort, minimizes potential errors, and allows users to focus on interpreting results rather than technical implementation.

4. Improve Data Exploration: By providing comprehensive visualization tools and automated analysis features, the application enables users to quickly explore relationships, trends, and distributions within their datasets. This efficiency in data exploration supports faster, more informed decision-making processes.

2.3 Objective of the study

The primary objective of this study is to design and develop a comprehensive, interactive web-based tool that transforms the EDA process. The application aims to bridge the gap between data analysis capabilities and user accessibility, providing a platform that supports sophisticated analysis without requiring extensive technical expertise.

The tool emphasizes flexibility and customization, allowing users to upload their own datasets and generate various types of visualizations ranging from basic histograms to complex correlation matrices. Through automated visualization generation and intuitive customization options, the application seeks to enhance both the efficiency and depth of data analysis, particularly in industries where rapid insight generation is crucial.

2.4 Overall Impact

The project's impact extends beyond mere technical innovation, fundamentally transforming how organizations approach data analysis. In the traditional EDA workflow, analysts spend significant time on coding and technical implementation, often at the expense of actual data interpretation and insight generation. This application shifts the focus back to analysis and decision-making by automating routine tasks and providing intuitive visualization tools.

Key areas where this project will have an impact include:

1) Increased Efficiency:
   The application dramatically reduces the time required for EDA by automating visualization generation and providing instant feedback on data characteristics. This efficiency gain allows analysts to explore multiple hypotheses quickly and iterate through different analytical approaches without technical delays. The streamlined workflow supports faster decision-making cycles and more comprehensive data exploration.
2) Democratization of Data Analysis:
   By removing technical barriers to EDA, the application makes sophisticated data analysis accessible to a broader range of professionals. This democratization of

data analysis capabilities enables organizations to foster more data-driven decision-making across different departments and roles. From financial analysts examining market trends to healthcare professionals analyzing patient data, the tool's flexibility and ease of use support diverse analytical needs across industries.The platform's impact on organizational efficiency and analytical capabilities positions it as a valuable tool for modern data-driven operations. By combining powerful analysis features with user-friendly design, the application supports both individual analytical excellence and broader organizational data literacy initiatives.

CHAPTER 3
TECHNICAL SPECIFICATIONS

3.1 Development Environment

The application was developed using a carefully chosen stack of Python-based technologies and frameworks that align with modern software development practices. The goal is to create an exploratory data analysis (EDA) tool that is reliable, user-friendly, and capable of handling diverse datasets effectively.

The development process emphasizes:

- Scalability: Ensuring the application can accommodate increased data volumes and user interactions.
- Maintainability: Using modular design principles to facilitate future enhancements and bug fixes.
- Performance Optimization: Leveraging efficient libraries to process large datasets quickly.
- User-Centric Design: Building intuitive interfaces for analysts, data scientists, and non-technical users.

3.2 Core Technologies

The application utilizes a variety of libraries and frameworks that enhance its functionality, efficiency, and interactivity

- Python 3.9+:

  Python serves as the backbone of the application due to its versatility and extensive library ecosystem. With support for multiple paradigms (object-oriented, procedural, and functional programming), Python provides the flexibility required for efficient data manipulation, statistical analysis, and web application development.
- Streamlit 1.2+:

  This cutting-edge framework is employed for developing the web-based interface. Streamlit simplifies the process of building interactive applications by enabling rapid prototyping and integration of widgets, making it ideal for

creating data-driven dashboards. Its ability to display visualizations dynamically makes it a perfect choice for an EDA tool.

- Pandas 1.4+:
  Pandas powers the application's data manipulation capabilities. Its high-performance data structures, such as Data Frames, enable efficient handling of structured data, including cleaning, preprocessing, and summarization.
- NumPy 1.21+:
  NumPy is used for numerical computing and supports fast array operations, which are essential for data transformation and mathematical computations.
- Plotly 5.5+:
  The application integrates Plotly to produce visually appealing, interactive, and customizable plots. With its support for 3D visualizations and advanced charting capabilities, Plotly enhances the user experience by enabling dynamic exploration of data.

3.3 Visualization Techniques

To empower users with a variety of insights, the application employs several visualization techniques:

- Histograms:
  These are used for analyzing the distribution of data. They allow users to visualize frequency distributions and identify patterns such as skewness or modality in datasets.
- Scatter Plots:
  Scatter plots help examine relationships between variables, making them ideal for understanding correlations or identifying clusters.
- Box Plots:
  These are employed to detect outliers and understand data variability. Box plots provide a concise summary of data distributions, including quartiles and median values.
- Heatmaps:
  Heatmaps are essential for visualizing correlation matrices, enabling users to identify strong or weak relationships between variables in the dataset.

- Line Graphs:
  Line graphs track trends and patterns over time, offering insights into temporal changes within continuous data.

- Pivot Charts:
  Pivot charts provide a powerful way to summarize and compare large datasets. With customizable grouping and filtering options, they enable dynamic and flexible data analysis.

3.4 User Interaction

The application prioritizes user experience by incorporating intuitive interactive widgets, including:

- Real-time customization of visualization parameters such as color schemes, axis selections, and chart types.
- Dynamic filtering and data selection to narrow down analyses to specific subsets of data.
- Options to export visualizations for presentations or reports, ensuring seamless communication of insights.

These features empower users to tailor the analysis and visualizations to their specific needs, making the tool suitable for both novice and advanced users.

3.5 Hosting and Deployment:

The application is designed to support deployment on various platforms, ensuring accessibility and scalability:

- Streamlit Cloud:
  Provides a straightforward hosting solution with minimal setup, making it ideal for rapid deployment.

- AWS (Amazon Web Services):
  Enables robust hosting with advanced features such as load balancing, auto-scaling, and secure storage. AWS also provides integration with other cloud services, ensuring seamless deployment in enterprise environments.

CHAPTER 4
DESIGN APPROACH AND DETAILS

4.1 Design Approach / Materials & Methods

The design approach for this Streamlit web application centers on creating an intuitive, efficient, and scalable platform for Exploratory Data Analysis (EDA). Our methodology emphasizes accessibility for users across varying levels of technical expertise while maintaining robust analytical capabilities. This section details the comprehensive framework, technical specifications, and implementation strategies employed in developing the application.

Table 4.1 Design Components and Their Purposes

Component	**Method/Technology used**	**Purpose**
User Interface	Streamlit framework	Provides a clean and intuitive interface for seamless navigation and EDA.
Data Handling	Pandas,Numpy	Handles data upload, processing, and validation efficiently.
Visualization	Plotly	Generates interactive and dynamic charts for data analysis.
Interactivity	Streamlit widgets (sliders,dropdowns,checkboxes)	Enables users to customize visualizations in real time.
Performance	Streamlit caching	Improves efficiency by reducing redundant computations.
Deployment	Streamlit cloud	Ensures scalability and accessibility for diverse user groups.
Architectural Design	Modular components(UI, Backend logic, visualization Engine)	Organizes application layers for maintainability and scalability.
Customization	Color selection, plot types	Enhances usability and adaptability to user preferences.

Data Privacy	Secure handling of uploaded datasets	Prevents unauthorized access to sensitive user data.

4.2 User-Centered Design

The application's design philosophy prioritizes user experience through an intuitive interface that guides users naturally through the data analysis process. The interface features a clean, logical layout with a side menu for straightforward navigation between different functionalities. This design choice allows users to maintain context while exploring various analytical features.

Implemented a progressive disclosure framework where complex features are presented gradually to prevent overwhelming users. This approach begins with basic analysis options at the top level and provides clear pathways to more advanced functionalities through expandable sections. Contextual help tooltips and step-by-step guidance support users through multi-stage analyses.

Real-time feedback forms a crucial component of the user experience. The interface provides immediate visual updates as users interact with the system, including progress indicators for long-running operations and clear error notifications. This responsive design ensures users remain informed and engaged throughout their analysis process.

4.3 Data Handling

The application's data handling capabilities focus on providing robust and flexible import options while ensuring data integrity. Primary support includes:

- CSV and XLSX file formats through Pandas integration
- Automatic data type detection and validation
- Column name verification and standardization
- Missing value identification

4.4 Visualization Techniques

The visualization system leverages Plotly's graph objects to create interactive, publication-quality charts. This framework provides a rich set of visualization options while maintaining high performance and responsiveness. The core visualization capabilities include:

4.4.1 Distribution Analysis

The application offers comprehensive tools for understanding data distributions through histograms with density estimation, kernel density plots, and Q-Q plots for normality assessment. Users can customize bin sizes and orientations to best represent their data.

4.4.2 Categorical Analysis

For categorical data exploration, the system provides multiple visualization options including box plots, with statistical overlays, and bar charts. These visualizations support various aggregation methods and can handle multiple categorical levels.

4.4.3 Interactive Features

Each visualization includes a robust set of interactive controls:

- Zoom capabilities with mouse wheel support
- Pan functionality with click-and-drag
- Multiple selection modes (Box, Lasso)
- Customizable tooltips
- Export options

4.4.4 Correlation Analysis

The application implements an interactive correlation matrix visualization using heatmaps with multiple colour scheme options. This feature allows users to quickly identify relationships between variables and explore patterns in their data.

4.5 Performance optimization

Performance optimization has been implemented through the strategic use of Streamlit's caching mechanisms and lazy loading techniques. The @st.cache decorator stores

computed results to avoid redundant calculations, significantly improving response times for repeated operations. Resource-intensive operations are handled through lazy evaluation, processing data in chunks when necessary to maintain responsiveness.

4.6 Deployment and Hosting

The deployment architecture is designed to ensure scalability and reliability in a cloud environment. The application is primarily deployed on Streamlit Cloud, with additional configuration options for deployment on alternative platforms such as Heroku or AWS Elastic Beanstalk.

The cloud infrastructure implements automatic scaling based on user load, with careful attention to memory optimization for concurrent users. Database connection pooling and cache management across instances ensure consistent performance under varying load conditions.

4.7 Future Considerations and Technical Dependencies

The system architecture has been designed with future extensibility in mind, allowing for seamless integration of additional features and capabilities. Key areas for future enhancement include machine learning integration, advanced data preprocessing options, and enhanced collaborative features.

This comprehensive approach ensures a robust, user-friendly, and scalable application that meets the needs of both novice and advanced users while maintaining high performance and reliability standards. The system's modular design allows for continuous improvement and adaptation to evolving user needs and technological capabilities.

4.8 Architectural Design

The architecture of the Streamlit-based web app is divided into three key components as in Fig 4.1 User Interface, Backend Logic, and the Visualization Engine. Each of these components plays a critical role in the functionality, data processing, and visualization capabilities of the application.

4.8.1 User Interface

The user interface is built using Streamlit, allowing seamless interaction between the user and the application. It consists of the following pages and features:

- Page Navigation:

  The application implements a navigation system through Streamlit's native sidebar component, ensuring users can efficiently move between different functional areas. This navigation design maintains a consistent layout across sessions while maximizing available screen space for data visualization and analysis. The sidebar provides persistent access to all major features, enabling users to seamlessly transition between different functionalities. This intuitive navigation structure significantly enhances the user experience by reducing cognitive load and maintaining a clear sense of location within the application.

- About Page:

  The About page serves as a comprehensive introduction and documentation hub for users, providing detailed insights into the application's capabilities and functionality.

- Upload Page:

  The Upload page implements a file handling system that streamlines the data import process while maintaining robust validation checks. Users can import data through CSV and XLSX formats, with support for both traditional file selection and convenient drag-and-drop functionality. Upon upload, the system automatically performs data type detection and validation, presenting users with a preview of the first several rows to verify proper data import. This immediate feedback mechanism, combined with clear file size and format constraints, ensures users can quickly identify and resolve any import issues before proceeding with their analysis. The interface's intuitive design makes the data upload process straightforward while maintaining the necessary technical rigor for data integrity.

- Dataset Analysis Page:

  This section includes:

- Missing Value Analysis: Interactive visualization of missing data patterns and percentage calculations for missing values per column
- Data Type Analysis: Displays the distribution of data types (numerical, categorical) for each column, helping users understand the structure of their dataset and also the count of each data type.

- Plot Analysis Page:

  This page is divided into two categories:
  - Categorical Data:

    Users can generate visualizations like Pie Charts, Bar Charts, Box Plots, Line Graphs, and Pivot Charts to analyze categorical data.
  - Numerical Data:

    Visualizations such as Histograms, Box Plots, Line Graphs, Scatter Plots, and Pivot Charts allow users to explore numerical data.

- Correlation Matrix Page:

  Users can generate a Heatmap to visualize correlations between different numerical attributes, providing deeper insights into relationships within the data.

4.8.2 Backend Logic and Processing System

The backend logic forms the core processing foundation of the application, orchestrating data management, analysis, and transformation operations. This component employs a modular architecture built on robust Python libraries to ensure efficient and reliable data processing capabilities.

The system's core processing infrastructure is built upon industry-standard Python libraries such as Numpy and Pandas are used for data manipulation and basic operations on the dataset.

- File Handler:

  Handles the reading and processing of the uploaded CSV/XLSX files and ensures they are properly loaded into the system.

- Data Processor:

  Responsible for handling the dataset once it's uploaded, including analyzing missing values, detecting data types, and performing other preliminary processing.
- Analytics Generator:

  Generates the necessary data transformations and calculations to be used in the visualizations.

4.8.3 Visualization Engine

This component generates the graphical representations of the dataset, leveraging Plotly for creating interactive and dynamic visualizations:

- Plot Generator:

  This module creates all the visualizations (such as histograms, scatter plots, pivot charts, etc.) by fetching the processed data from the backend.
- Interactive Controls:

  Users can interact with the visualizations by selecting different attributes for plotting and customizing color schemes, making the app highly flexible and user-friendly.
- Download Option:

  Allows users to download the visualizations for offline analysis or reporting purposes.

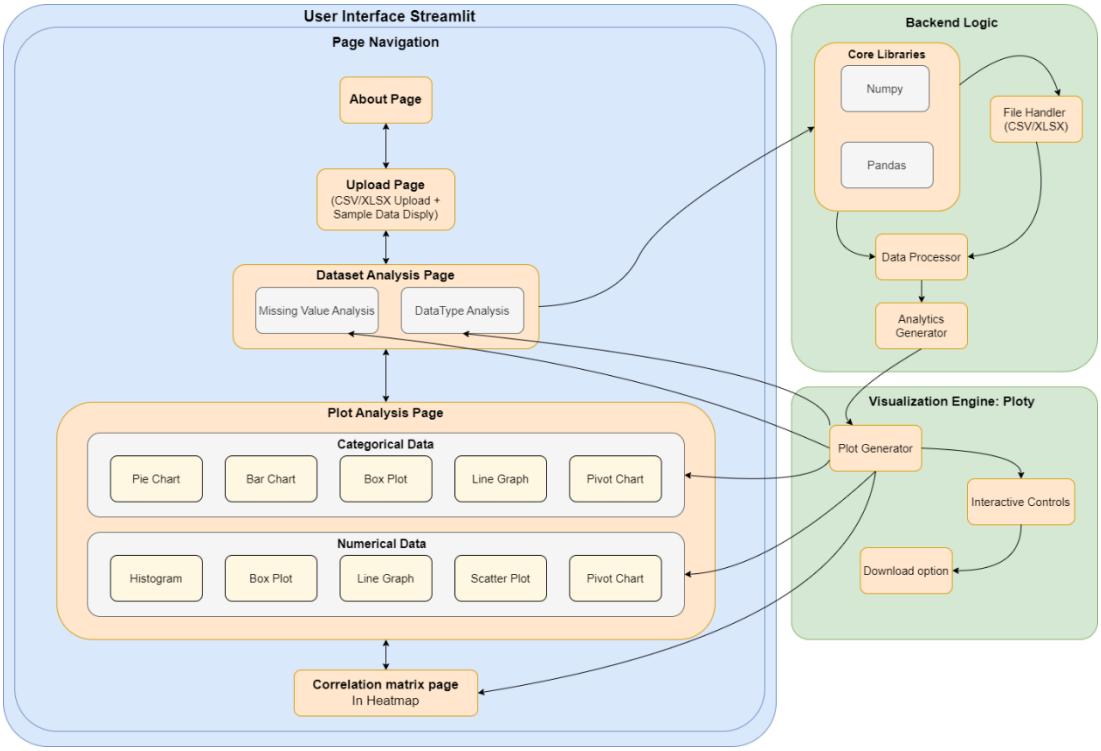

Fig. 4.1 Architectural Design of the Streamlit Webapp

4.9 Codes and Standards

The application development follows a comprehensive set of industry standards and best practices to ensure code quality, maintainability, and security. These standards form the foundation of our development process and guide all aspects of the application's implementation

4.9.1 Code Quality and Style:

The codebase strictly adheres to Python's PEP 8 style guidelines to maintain consistency and readability across the project. This includes:

- Structural formatting:
  - Consistent indentation (4 spaces)
  - Limiting line length to 79 characters
  - Proper naming conventions for variables, functions, and classes
- Type Hinting: Implementation of Python-type hints to improve code readability and catch type-related errors early.

4.9.2 Version Control and Collaboration:

- Git Workflow: Git is utilized for version control, enabling collaborative development, tracking changes, and managing code revisions efficiently.
- Documentation: Inline comments for complex algorithms or business logic.

4.9.3 Data Privacy Standards

The application complies with data privacy standards, ensuring secure handling of user-uploaded datasets. This includes implementing appropriate measures to prevent unauthorized access to sensitive data.

4.9.4 Responsive Design Principles

The UI is designed using responsive web design principles, ensuring the application performs well across various devices and screen sizes. Flexible grid layouts that adapt to different screen sizes and responsive components that maintain functionality across devices.

4.10 Constraints, Alternatives, and Trade-offs

This section outlines the constraints encountered during the development of the application, the alternatives considered for various components, and the trade-offs made to achieve a balance between functionality, usability, and performance.

4.10.1 Constraints

Several constraints were faced during the design and development of the application:

- Performance Limitations:
  Handling large datasets posed challenges, particularly for real-time interactive visualizations. Computationally intensive operations could lead to slower application response times or crashes. To address this, techniques like data sampling and caching were implemented. Data sampling reduced the processing load, while caching avoided redundant computations by storing intermediate results.
- Mobile Usability:
  Designing the application for smaller screens was another challenge. Complex visualizations and controls are often difficult to adapt for mobile devices. This issue was mitigated by implementing a responsive design that dynamically adjusts layouts based on screen size. Simplified views and restricted advanced

features on mobile devices ensured usability without compromising essential functionality.

4.10.2 Alternatives

To ensure the application was built on the most suitable tools and technologies, various alternatives were evaluated:

- Framework selection:
  - Dash and Flask with a JavaScript front-end were considered. Dash offers robust visualization capabilities, while Flask provides full customization. However, these frameworks required more development time and advanced technical expertise.
  - Streamlit was ultimately chosen for its simplicity, rapid development capabilities, and built-in interactive features, making it the best fit for a data-centric application with tight development timelines.
- Visualization Libraries:
  - Libraries like Bokeh and Altair were explored for creating interactive visualizations. While both have their strengths in flexibility and aesthetics, Plotly was selected due to its advanced interactivity, extensive community support, and compatibility with Streamlit.
- Hosting and Deployment Platforms:
  - Platforms like Heroku and AWS were considered for deploying the application. Streamlit Cloud emerged as the preferred choice due to its seamless integration with Streamlit applications, cost-effectiveness, and ability to handle moderate user concurrency with minimal setup.

4.10.3 Trade-offs Made

Developing the application required careful consideration of trade-offs to ensure a balance between competing priorities:

- Interactivity vs. Performance:
  Enhancing interactivity, such as enabling real-time updates or allowing complex visualizations, often comes at the cost of application performance. To maintain stability and responsiveness, caching strategies were implemented, and certain features were simplified when working with larger datasets.

- Feature Set vs. Usability:
  Adding more features increases the application's functionality but can overwhelm users, especially non-technical ones. To address this, a user-centric approach was adopted to prioritize essential features, ensuring the application remained intuitive while still being versatile.

- Development Time vs. Customization:
  Using Streamlit accelerated development but limited the level of customization compared to frameworks like Flask. This trade-off was acceptable given the project timeline, as Streamlit offered sufficient flexibility and a quick development cycle.

- Visualization Aesthetics vs. Performance:
  While visually appealing charts enhance the user experience, they can require significant computational resources, especially with large datasets. Users were given options to customize visualizations, allowing them to balance aesthetics with performance based on their needs.

CHAPTER 5

SCHEDULE, TASKS AND MILESTONES

Table 5.1 Project Schedule and Milestones

Week	**Tasks**	**Milestones**
1	Initial project planning, setup development environment (Python, Streamlit, Pandas, Plotly, Base64), design architecture, document features, discuss risks and challenges.	Finalized project scope, configured environment, completed architecture diagram, and defined features.
2	Design app skeleton with placeholders for key features, finalize architecture diagram, test small datasets for file upload.	App skeleton created, dataset upload functionality tested with small datasets, architecture diagram updated.
3	Implement dataset upload functionality (CSV, XLSX), display a sample of dataset based on row range input, test with small datasets.	Dataset upload and row display feature fully functional, system tested with small datasets, user-friendly input fields created.
4	Automate datatype classification (numerical, categorical), visualize data distribution with bar/pie charts, refine UI.	Datatype analysis feature completed, visual summaries of data structure created, UI refined for better user interaction.
5	Implement basic graph plotting (scatter, line, box), add color customization options, test features with various datasets.	Graph plotting functionality implemented, customization options (color) added, successful tests for different datasets.
6	Extend graph plotting functionality	Advanced plotting tools like

	with pivot charts, refine UI for smooth selection of attributes and graph types, test with large datasets.	pivot charts added, UI finalized for easy attribute/graph selection, successful tests with complex datasets.
7	Implement correlation analysis (numerical relationships), visualize correlation matrix (heatmap/table), test accuracy of results.	Correlation analysis functionality added, heatmap and table visualizations working accurately with different datasets.
8	Collect user feedback on correlation feature, optimize heatmap clarity, document the functionality, and prepare user guide.	Final adjustments made based on user feedback, correlation visualizations improved, documentation prepared.
9	Conduct end-to-end testing of all features (upload, analysis, plotting), identify and debug performance issues, optimize app for large datasets.	Successful end-to-end testing, app optimized for performance, all bugs resolved and features working seamlessly.
10	Finalize app UI, optimize performance, ensure all functionalities are stable with various datasets.	All features working smoothly, app's performance optimized for efficiency.
11	Review the app with the project guide, make necessary adjustments, deploy the app on a cloud platform (Streamlit Cloud/Heroku).	App reviewed and approved, deployed on cloud platform, accessible for end users.
12	Prepare and finalize project documentation, presentation materials (user guides, feature descriptions), conduct final project review.	Documentation and presentation materials completed, project final review done, app deployed and ready for demonstration.

Week 1: Initial Planning and Development Setup

Tasks:

- Meet with the project team and guide to finalize the scope, timeline, and deliverables for the project, ensuring clear understanding and alignment among all members.
- Set up the development environment by installing essential tools like Python, Streamlit, Pandas, Plotly, and Base64, and verify that all installations work correctly.
- Brainstorm and document the app's core features, ensuring they align with the project goals and cover user requirements.
- Discuss potential challenges, such as handling large datasets or ensuring smooth visualizations, and strategize ways to mitigate them effectively.

Milestones:

- Project scope and goals are clearly defined, with well-outlined responsibilities for each team member.
- Development environment is fully configured and thoroughly tested to avoid delays during implementation.

Week 2: Initial Architecture and Skeleton Design

Tasks:

- Design the initial architecture diagram outlining app components, data flow, and interactions, incorporating feedback from the team to ensure a comprehensive design.
- Create a functional skeleton of the app, including placeholders for critical components such as dataset upload, datatype analysis, and visualization options, laying a foundation for further development.
- Hold a follow-up meeting to assess progress, gather feedback, and ensure all members understand the workflow for subsequent weeks.

Milestones:

- Architecture diagram is completed, providing a clear development roadmap.
- The app skeleton is designed with flexibility to accommodate additional features and refinements during later stages.

Week 3: Dataset Upload Feature Development

Tasks:

- Develop seamless functionality for uploading datasets in CSV and XLSX formats, ensuring compatibility with commonly used data file types.
- Test the upload feature with small datasets of varying structures to confirm robustness and error handling.
- Add meaningful error messages for unsupported file formats or missing data, enhancing the user experience.

Milestones:

- Dataset upload functionality is fully operational, handling small datasets effectively.
- Error messages and validations ensure users are guided during file uploads.

Week 4: Row Range Display Feature

Tasks:

- Implement an intuitive feature to allow users to select and display specific rows of the dataset based on a defined range.
- Add user-friendly input options, such as sliders or number boxes, to enhance interactivity and usability.
- Test the feature with medium datasets to ensure it works efficiently under various scenarios and dataset sizes.

Milestones:

- Users can easily view dataset samples by specifying row ranges, ensuring better data exploration.
- Thorough testing confirms the stability of the feature with different dataset structures.

Week 5: Datatype Analysis (Part 1)

Tasks:

- Implement a system to automatically classify dataset columns into numerical, categorical, or other datatypes, ensuring accurate classification even for complex datasets.
- Display a concise summary of datatype counts to provide users with a clear overview of the dataset's structure.

- Test the classification logic with diverse datasets, focusing on edge cases to ensure reliability.

Milestones:

- Datatype classification is complete and functions accurately across different datasets.
- Dataset summaries provide meaningful insights, aiding users in understanding the dataset composition.

Week 6: Datatype Visualization (Part 2)

Tasks:

- Create visual representations of datatype distributions using charts like bar graphs and pie charts, ensuring they are clear and informative.
- Conduct tests with large datasets to verify the visualization feature's performance and scalability.
- Refine the UI layout to ensure that datatype summaries and visualizations are presented in a user-friendly manner.

Milestones:

- Visualization of datatype distributions enhances user understanding and exploration of the dataset.
- UI improvements ensure that datatype information is easily accessible and visually appealing.

Week 7: Basic Graph Plotting and Customization

Tasks:

- Implement features for plotting basic graphs like line, scatter, and box plots, allowing users to analyze numerical relationships and trends effectively.
- Introduce customization options for graphs, including the ability to modify colors and styles, providing a personalized user experience.
- Test the graph plotting and customization features with various datasets to ensure they are intuitive and error-free.

Milestones:

- Basic plotting features are functional, reliable, and offer flexibility through customization options.

Week 8: Advanced Graph Plotting and UI Enhancements

Tasks:

- Extend the graph plotting functionality with advanced visualizations such as pivot charts, enabling data aggregation and comparison.
- Enhance the UI to make it intuitive for selecting attributes, choosing chart types, and customizing graphs.
- Conduct rigorous testing of advanced visualizations with datasets of varying sizes and complexities.

Milestones:

- Advanced graph plotting features provide users with powerful tools for data exploration.
- UI is intuitive, making the app user-friendly for both novice and experienced users.

Week 9: Correlation Analysis Development

Tasks:

- Implement correlation analysis to identify and visualize relationships between numerical attributes, offering insights into data patterns.
- Develop clear and interactive visualizations for the correlation matrix, including a heatmap and tabular format.
- Perform thorough testing to confirm the accuracy and reliability of correlation results.

Milestones:

- Correlation analysis is complete, offering both detailed insights and visualizations for effective data exploration.

Week 10: Refining Correlation Analysis and User Feedback

Tasks:

- Refine correlation visualizations, ensuring the heatmap is visually appealing and the matrix is easy to interpret.
- Gather feedback from test users on the correlation analysis feature, focusing on usability and clarity.
- Update the feature based on feedback and document its functionality with examples for user guidance.

Milestones:

- Heatmaps are polished and enhance the interpretability of correlation results.
- Feedback-driven refinements make the feature more user-friendly and practical.

Week 11: End-to-End Testing and Debugging

Tasks:

- Conduct rigorous end-to-end testing of all implemented features to ensure they function seamlessly together.
- Identify and resolve any issues, focusing on performance bottlenecks, UI glitches, or incorrect analysis outputs.
- Optimize app performance to handle large datasets without lag, ensuring responsiveness under heavy loads.

Milestones:

- All features pass comprehensive testing, with bugs and performance issues resolved.
- The app is optimized for efficient operation across various dataset sizes.

Week 12: Final Review and Deployment

Tasks:

- Conduct a final review with the guide and team, addressing any last-minute concerns or suggestions.
- Deploy the app on a cloud platform like Streamlit Cloud or Heroku, ensuring accessibility and scalability.
- Prepare detailed project documentation and demonstration materials, including user guides, feature descriptions, screenshots, and examples.

Milestones:

- The app is finalized, deployed, and accessible to end users.
- Documentation and presentation materials are polished and ready for the final project review.

This detailed timeline provides a structured development plan to ensure that the project is completed efficiently within 8 weeks, with clear tasks and milestones for each phase.

CHAPTER 6
DISSERTATION DEMONSTRATION

6.1 Overview of the Web App

This web app is a versatile, user-friendly platform designed for interactive data visualization and EDA. It allows users to upload datasets and explore them through dynamic and customizable visualizations. The app is particularly useful for data analysts and researchers who require quick insights from their data without needing extensive programming knowledge. The app was built using Python, Pandas, Plotly, and Streamlit, making it both robust and flexible as in Fig 6.1.

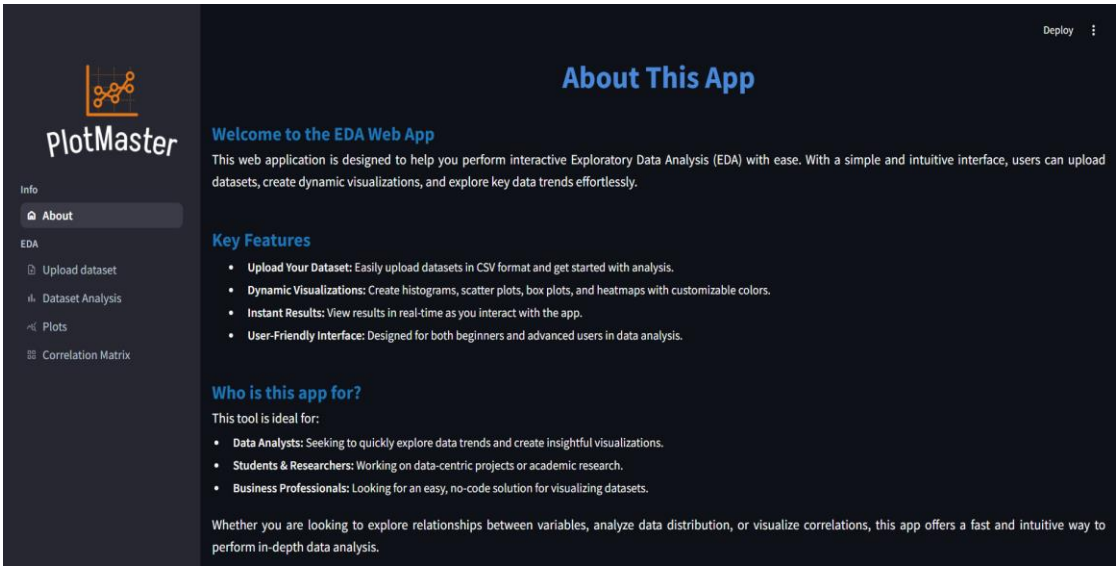

Fig. 6.1 Overview of the App

6.2 Features and Functionalities

The app comes equipped with various features aimed at simplifying the EDA process:

- Dataset Upload: Users can easily upload datasets in CSV or XLSX format for immediate analysis as shown in Fig 6.2.
- Specifying rows: After uploading the dataset the user can select a specific number of row ranges to display in the web app as in Fig 6.3.

Fig. 6.2 Uploading the dataset

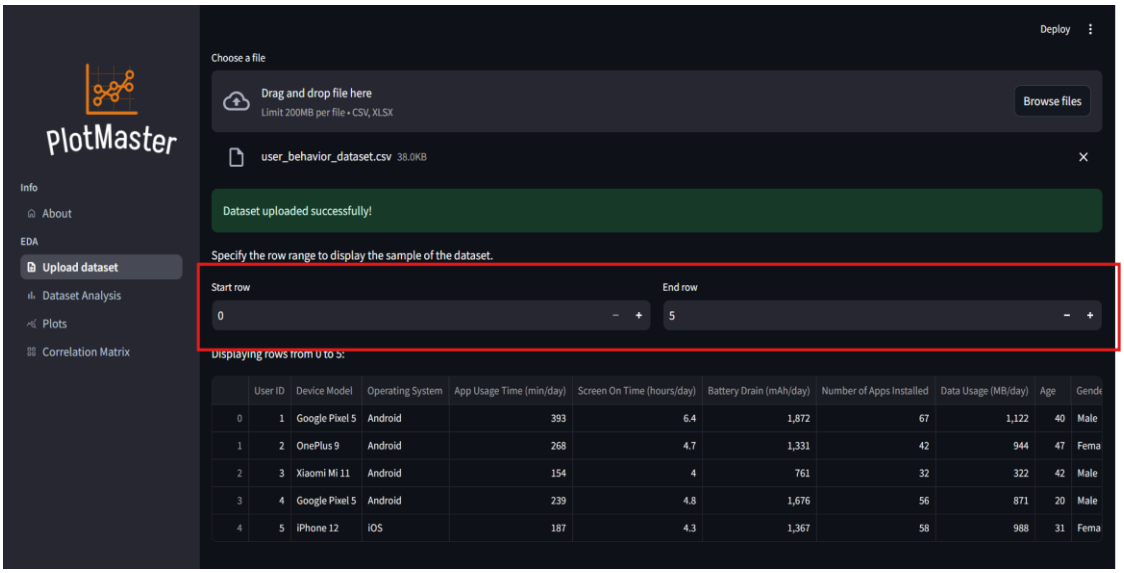

Fig. 6.3 Displaying the row range of the dataset

The end row number should always be greater the start row number, then only we can view the number of rows in the dataset.

- Visualizations**:** Users can generate dynamic histograms, scatter plots, box plots, and heatmaps, with the ability to filter data interactively and adjust colour schemes for better analysis as in Fig. 6.4

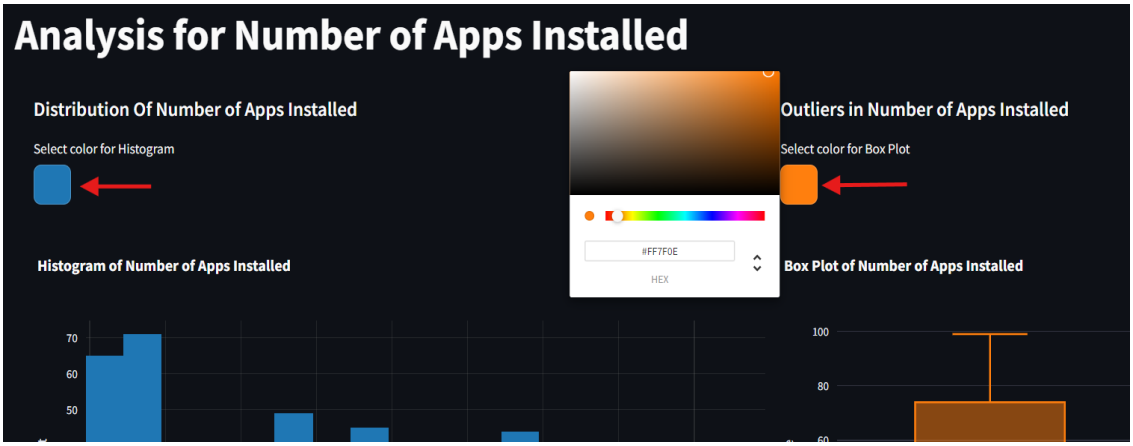

Fig. 6.4 Adjusting color schemes

- Dataset Analysis: The user can see the distribution of numerical and categorical features available in the dataset and also the count of each datatype as in Fig. 6.5.

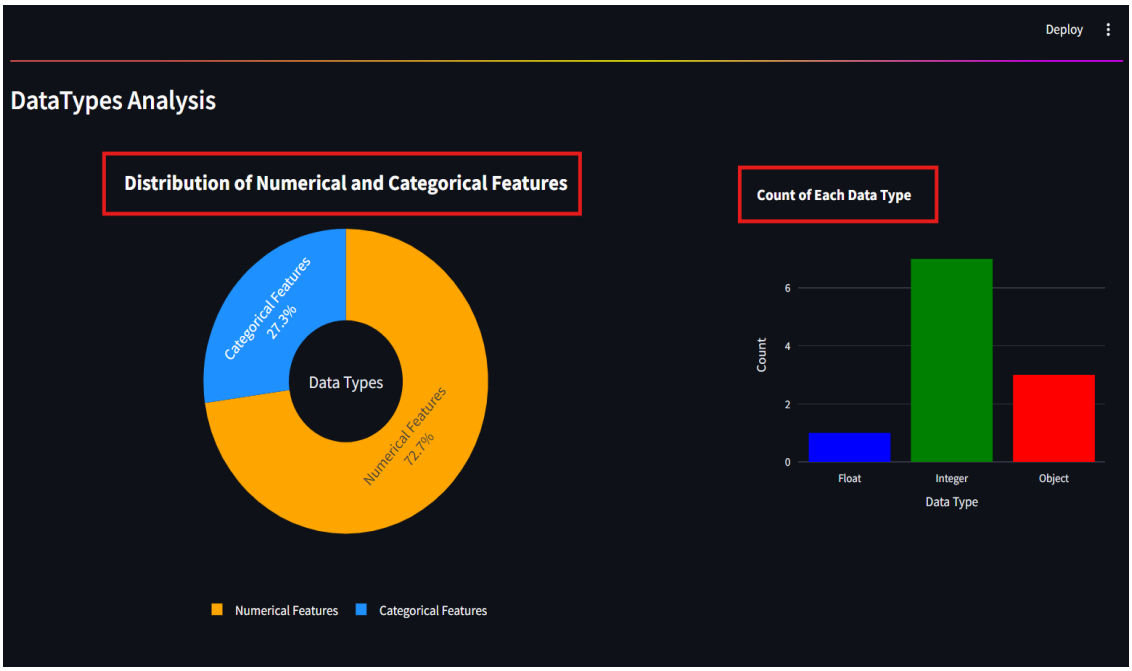

Fig. 6.5 Datatype Analysis

- Customization: It offers real-time customization of visualizations, including axis adjustments, chart types, and colour settings. The users can also customize the attributes to be displayed in the plots section and the selected attributes appear accordingly as shown in Fig. 6.6.

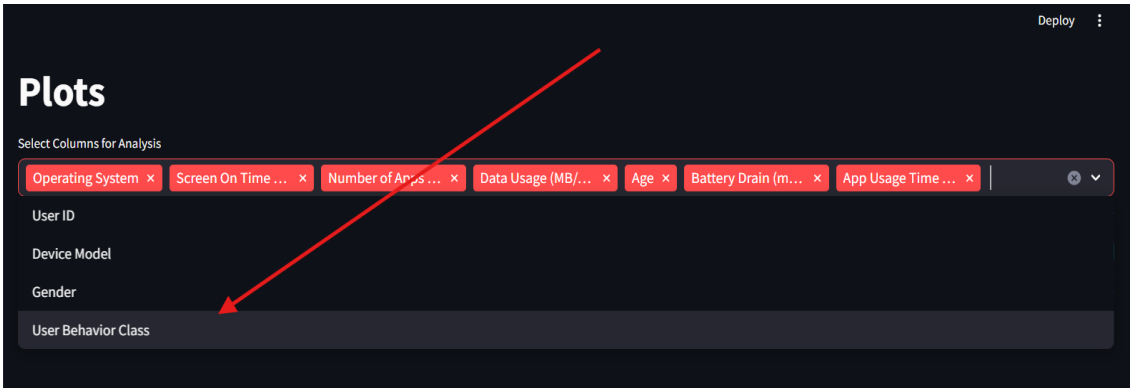

Fig. 6.6 Attribute Selection

- Data Interaction: Users can interact with data through sliders and dropdowns, providing an engaging way to explore trends and patterns.
- Analysis of the attributes: After attribute selection, the user can see the charts and graphs of the particular attribute as in Fig. 6.7.

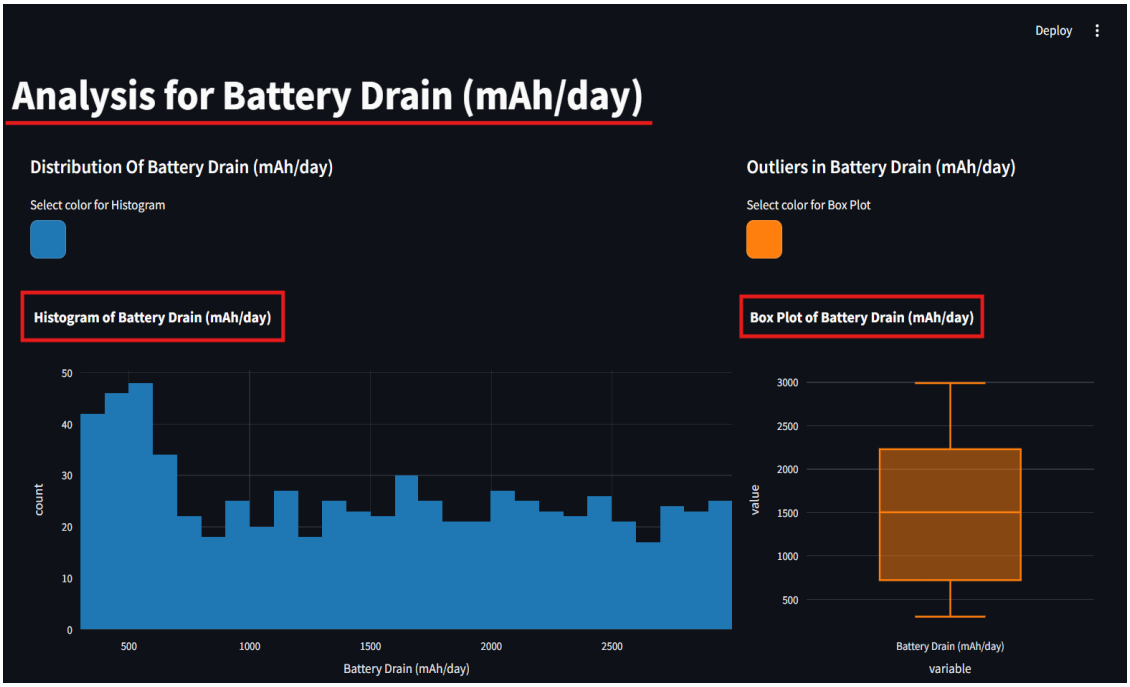

Fig. 6.7 Chart of the chosen attribute

- Graph Selection: Scatterplot, pivot chart, line graph and box plots are included in the graphs. Based on the selected attribute and graph, we can display that particular graph with customizable colors as in Fig. 6.8.

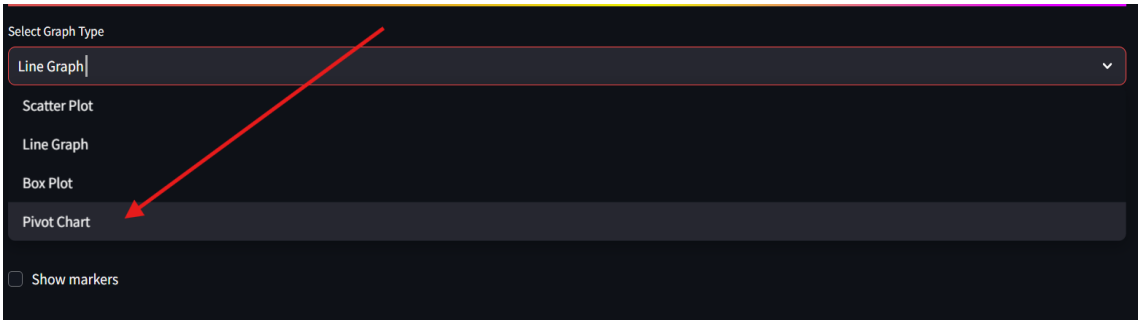

Fig. 6.8 Graph selection

We can also select the Y-axis and column for hue to display the graph.

- Scatter Plot: Fig. 6.9 shows the scatter plot between the data usage installed and the number of apps per day from the user behaviour dataset.

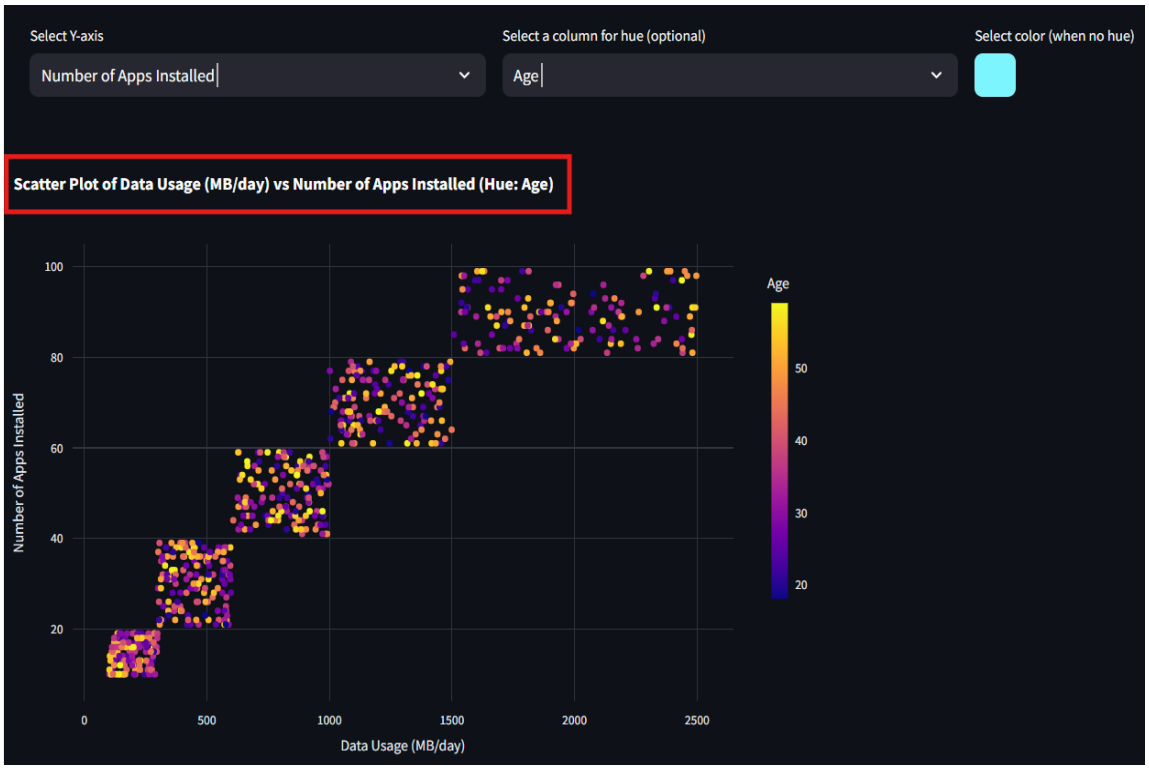

Fig. 6.9 Scatter Plot

- Line Graph: Fig. 6.10 shows the line graph between device model and battery drain of the user as per the user behaviour dataset.

Fig. 6.10 Line Graph

- Box Plot: Fig. 6.11 shows the box plot between the Operating system of the mobile device and data usage as per the user behaviour dataset.

Fig. 6.11 Box Plot

- Pivot chart: Fig. 6.12 shows the pivot chart between Device model and the mean of app usage time as per the user data behaviour dataset.

Fig. 6.12 pivot Chart

- Correlation Matrix: Users can select specific attributes for correlation analysis and customize the color palette used to display the correlation matrix as in Fig. 6.13, allowing for a more tailored and visually appealing representation of data relationships.

Fig. 6.13 Correlation Matrix

6.3 User Interface (UI)

The app boasts a clean, intuitive UI that is easy to navigate for both technical and non-technical users. Key elements of the UI include:

- Simple Navigation Bar: Allows users to upload datasets, access visualization tools, and view data statistics with minimal clicks.
- Interactive Widgets: Dropdowns, sliders, and buttons that control the visualizations in real-time, providing an interactive experience.
- Visualization Panel: Displays charts and graphs generated from user-selected data, making it easy to switch between different types of visual representations.

6.4 Exploratory Data Analysis Workflow

The EDA workflow within the app is designed to guide users through the data exploration process seamlessly:

- Step 1: Dataset Upload: Users upload their dataset in CSV or XLSX format.
- Step 2: Data Exploration: Basic data statistics are automatically displayed, including mean, median, and standard deviation.
- Step 3: Feature Selection: Choose features/variables for analysis and filter data using dropdowns and sliders.

- Step 4: Visualization: Generate dynamic visualizations to explore relationships and trends within the dataset.
- Step 5: Export Results: Download cleaned data or visualizations for further analysis or reporting.

6.5 Technical Aspects

- Backend: Python 3.8 and Pandas for data manipulation and cleaning, Matplotlib and Plotly for generating visualizations.
- Frontend: Streamlit framework that ensures smooth interactivity and real-time feedback for users.
- Libraries Used:
    - Pandas & NumPy for data handling.
    - Matplotlib & Plotly for creating interactive graphs and charts.
    - Streamlit for the user interface and dynamic components.
- Scalability: The app can handle medium-sized datasets with ease and can be deployed on cloud platforms like Streamlit Cloud or Heroku or AWS for greater scalability.

6.6 Demonstration of Use Case

Use Case: Analyzing a User Behavior Dataset for Mobile Device Usage Patterns
This outlines a practical use case where the developed application is utilized to analyze a user behavior dataset, shedding light on mobile device usage patterns. The dataset includes features such as mobile device model, app usage time, screen-on time, battery drain, the number of apps installed, age, and gender. Below is a step-by-step demonstration of the analysis process:

Step 1: Upload the Dataset
The first step involves uploading a user behavior dataset into the application. The dataset must contain relevant features such as mobile device model, app usage time, screen-on time, battery drain, the number of apps installed, age, and gender. The application supports file formats like CSV or Excel, and the uploaded dataset is previewed for user confirmation (see Figure 6.14). Once the dataset is uploaded

successfully, users can begin exploring the data using interactive visualizations and analysis tools.

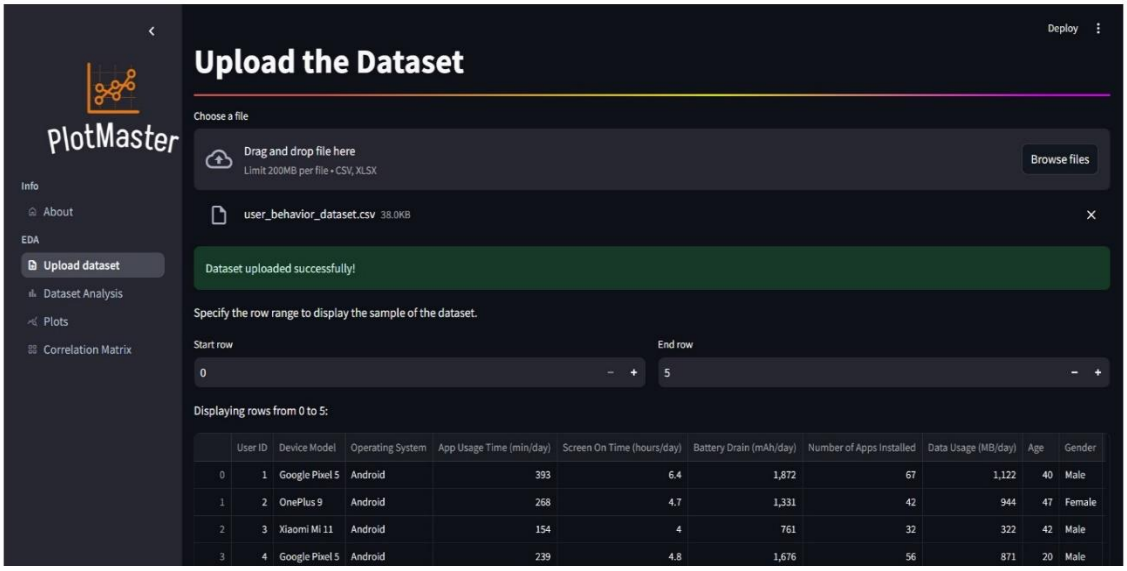

Fig. 6.14 Uploading the dataset

Step 2: Visualize App Usage Time Across Device Models

Using the app's histogram feature, users can analyze the distribution of app usage time across different device models. This helps identify which devices exhibit higher app engagement. For instance, a histogram could reveal that users of certain high-end models spend significantly more time using apps compared to budget-friendly devices. Such insights provide valuable information about device-specific engagement patterns and potential areas for targeted optimizations. Fig. 6.15 illustrates this distribution, showcasing how the application visualizes numerical and categorical features interactively.

Step 3: Analyze Screen-On Time vs. Battery Drain

A scatter plot is employed to analyze the relationship between screen-on time and battery drain. This visualization is crucial for identifying devices or usage patterns that contribute to quicker battery depletion. For example, the scatter plot might highlight clusters of devices where prolonged screen-on time correlates with a sharp increase in battery consumption. These findings can guide manufacturers or app developers to optimize battery usage for specific scenarios. Fig. 6.16 demonstrates such an analysis, showcasing trends and outliers effectively.

Step 4: Correlation Analysis with Heatmaps

To uncover deeper insights, heatmaps are used to detect correlations between demographic features (age, gender) and mobile usage behavior. For instance, the analysis could reveal that younger users tend to have more apps installed and spend longer durations on their devices, while older users may prioritize fewer, essential apps. Heatmaps also highlight the relationships between app usage time and other features like device models or battery drain, as shown in Fig. 6.17. This step enables a comprehensive understanding of how demographic factors influence device usage patterns.

Step 5: Customize Visualizations for Specific Insights

The app allows users to customize visualizations with different color schemes and grouping options. For example, trends across various age groups or device models can be highlighted with tailored color palettes. This customization improves the clarity of insights and makes the visualizations more suitable for presentations or reports. Once the analysis is complete, users can download the generated insights and visualizations in standard formats for further research or reporting.

This use case demonstrates how the app can effectively analyse user behaviour data, allowing businesses or researchers to gain insights into mobile usage trends and optimize user experience or resource management.

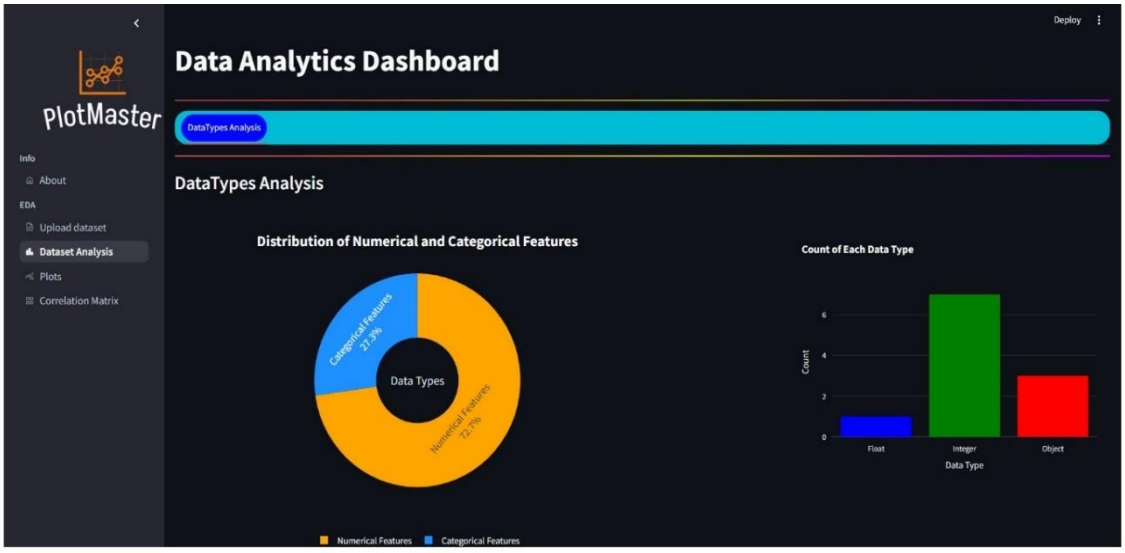

Fig. 6.15 Dashboard of the Uploaded dataset

Fig. 6.16 Scatter plot of the uploaded dataset

Fig. 6.17 Correlation matrix of the uploaded dataset

CHAPTER 7
COST ANALYSIS / RESULT & DISCUSSION

7.1 Cost Analysis

We conducted a comprehensive cost analysis, estimating potential expenses associated with dataset acquisition, software tools, and hosting services. The project development primarily relied on open-source software tools such as Python, Streamlit, Pandas, and Plotly, which are available for free. Similarly, publicly available datasets, sourced from platforms like Kaggle, incurred no cost. As a result, no significant expenses have been incurred so far during the development phase.

However, as the project expands and moves into more complex stages, potential costs could arise from using cloud services for hosting and managing datasets. For instance, Streamlit Cloud, AWS, or Heroku may be required for scalable deployment, enabling the app to handle larger datasets and higher user traffic. Additionally, if real-time data processing or advanced computational resources are needed, costs associated with cloud computing infrastructure (such as AWS EC2 instances or Google Cloud) may emerge.

7.2 Results

In the data exploration phase, the uploaded dataset was thoroughly analyzed for data types and distributions, ensuring a solid foundation for subsequent data analysis tasks. The app's ability to handle both numerical and categorical data allowed for a deeper understanding of dataset composition, as it identified outliers, missing values, and inconsistencies. The distribution visualizations provided by histograms and box plots enabled users to assess the spread of the data, identify skewness, and examine the presence of outliers across various variables.

The implementation of scatter plots and line graphs facilitated the examination of relationships and trends between variables, enabling users to identify correlations and dependencies. These visualizations proved useful for identifying anomalies and trends that were not immediately apparent through raw data inspection. Furthermore,

the ability to overlay multiple variables on a single graph enhanced comparative analysis, providing more comprehensive insights.

The heatmaps provided an intuitive way to understand correlations between multiple variables, visualizing both strong and weak correlations with ease. This feature, combined with the correlation matrix, offered users a clear and efficient method to discover which variables had the greatest impact on one another, supporting more focused and targeted analysis.

Additionally, the integration of pivot charts significantly improved data exploration by enabling users to group, filter, and summarize datasets interactively. This functionality allowed users to derive insights from aggregated data, facilitating decision-making processes that rely on high-level summaries, such as identifying overall trends in the dataset uploaded.

The app's customizability, such as the ability to choose specific colour schemes, enhanced the clarity and impact of visualizations. This feature enabled users to personalize their analysis environment, making it easier to highlight key findings and improve report generation. User-friendly controls and dropdown menus, ensured smooth navigation and ease of use, allowing users to generate customized visual outputs without extensive technical expertise.

Moreover, the app was tested on datasets of varying sizes, and it demonstrated the ability to maintain performance and responsiveness. Users were able to perform real-time adjustments to visualizations without experiencing significant lag or delays, highlighting the robustness of the app in practical use cases.

In conclusion, the app provided a comprehensive and flexible platform for data exploration and visualization, empowering users to perform thorough EDA with ease. The wide range of visual tools, combined with intuitive controls and customization features, ensured that users could uncover meaningful insights from their datasets efficiently and effectively.

7.3 Discussion

The project successfully delivered a highly functional Streamlit-based web application, providing a versatile toolkit for EDA. One of the project's standout features was its ability to allow users to select and filter specific attributes for targeted analysis. This enhanced the app's flexibility, giving users the capability to focus on particular aspects of the data based on their analytical needs.

A key strength of the application is its user-friendly interface, designed to be accessible to users with varying levels of technical expertise. Both experienced data analysts and non-technical users can easily navigate through the app's features, perform detailed analysis, and visualize their datasets without requiring programming skills. Features like pivot charts further simplified the process of summarizing and comparing datasets, helping users concentrate on extracting meaningful insights without unnecessary complexity.

The customization options available in the app, particularly the ability to personalize visualizations with different colour schemes, significantly improved usability. By allowing users to tailor the appearance of charts and graphs, the app ensures that visualizations are both engaging and easy to interpret, catering to different use cases and preferences.

In conclusion, the project met its objectives by providing an efficient, adaptable, and easy-to-use tool for EDA. While no significant costs have been incurred thus far, future expenses for cloud hosting and scaling should be anticipated as the project evolves. The app's flexible features and intuitive design make it a valuable resource for data analysts, offering a comprehensive solution for exploring and interpreting datasets effectively.

CHAPTER 8

SUMMARY

The Streamlit web app for interactive data visualization and EDA is designed to provide data analysts with a powerful, efficient, and user-friendly tool for in-depth exploration and visualization of datasets. The app allows users to easily upload datasets, perform essential tasks, and generate a wide range of visualizations, such as histograms, scatter plots, box plots, heatmaps and others. With added features like customizable colour schemes and interactive real-time controls, the app enhances the analytical process, making it easier to identify trends and patterns within the data. This customization empowers users to adapt visualizations to suit specific analytical needs, making the tool versatile for various industries and use cases.

The primary goal of this project is to streamline the EDA process, enabling users to quickly extract meaningful insights without needing extensive technical expertise in programming or data science. By automating parts of the EDA workflow and providing an intuitive interface, the app reduces the complexity typically associated with manual data analysis. Built using open-source technologies such as Python, Pandas, and Streamlit, the app is highly accessible and free from licensing costs. Its design allows for the handling of diverse datasets, ensuring that users can work with data ranging from small to medium-sized datasets without significant performance issues.

The app allows users to upload datasets and seamlessly generate a variety of visualizations, including histograms, scatter plots, box plots, line graphs, heatmaps and others. These visual tools provide a clear view of trends, distributions, and relationships within the data. Additionally, the application offers customizable features, such as colour schemes and graph types, enhancing the flexibility and adaptability of the platform to meet specific analytical needs. By simplifying the EDA process and automating parts of the workflow, the application reduces the time and effort typically required for manual analysis, allowing users to focus on gaining meaningful insights. Overall, the project successfully delivers an efficient and interactive platform for conducting in-depth data exploration and visualization.

CHAPTER 9
REFERENCES

[1.] Kumar, K. (2021). Exploratory Data Analysis for Predicting Student's Grades. In: Sharma, T.K., Ahn, C.W., Verma, O.P., Panigrahi, B.K. (eds) Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, vol 1381. Springer, Singapore https://doi.org/10.1007/978-981-16-1696-9_33

[2.] Patrick König, Sebastian Beier, Martin Mascher, Nils Stein, Matthias Lange, Uwe Scholz, DivBrowse—interactive visualization and exploratory data analysis of variant call matrices, *GigaScience*, Volume 12, 2023, giad025 https://doi.org/10.1093/gigascience/giad025

[3.] W. Kong, L. He and H. Wang, "Exploratory Data Analysis of Human Activity Recognition Based on Smart Phone," in *IEEE Access*, vol. 9, pp. 73355-73364, 2021 https://doi.org/10.1109/ACCESS.2021.3079434

[4.] DiCerbo, K.E. *et al.* (2015). An Application of Exploratory Data Analysis in the Development of Game-Based Assessments. In: Loh, C., Sheng, Y., Ifenthaler, D. (eds) Serious Games Analytics. Advances in Game-Based Learning. Springer, Cham. https://doi.org/10.1007/978-3-319-05834-4_14

[5.] Langer, Tristan, and Tobias Meisen. 2021. "System Design to Utilize Domain Expertise for Visual Exploratory Data Analysis" *Information* 12, no. 4: 140. https://doi.org/10.3390/info12040140

[6.] Eken, S. An exploratory teaching program in big data analysis for undergraduate students. *J Ambient Intell Human Comput* **11**, 4285–4304 (2020). https://doi.org/10.1007/s12652-020-02447-4

[7.] Lynn, T.; Rosati, P.; Nair, B.; Mac and Bhaird, C. An Exploratory Data Analysis of the #Crowdfunding Network on Twitter. *J. Open Innov. Technol. Mark. Complex.* **2020**, *6*, 80. https://doi.org/10.3390/joitmc6030080

[8.] Mohamudally, N., Armoogum, S. (2020). Smart City Mobile Apps Exploratory

Data Analysis: Mauritius Case. In: Ben Ahmed, M., Boudhir, A., Santos, D., El Aroussi, M., Karas, İ. (eds) Innovations in Smart Cities Applications Edition 3. SCA 2019. Lecture Notes in Intelligent Transportation and Infrastructure. Springer, Cham. https://doi.org/10.1007/978-3-030-37629-1_4

[9.] R. RamyaSri, S. IshaSanjida, D. Parasa and S. Bano, "Food Survey using Exploratory Data Analysis," *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, Jaipur, India, 2019, pp. https://doi.org/10.1109/ICCT46177.2019.8969016

[10.] Taboada, Guillermo L., and Liangxiu Han. 2020. "Exploratory Data Analysis and Data Envelopment Analysis of Urban Rail Transit" *Electronics* 9, no. 8: 1270.

[11.] H. Pitroda, "A Proposal of an Interactive Web Application Tool QuickViz: To Automate Exploratory Data Analysis," *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*, Mumbai, India, 2022, pp. https://doi.org/10.1109/I2CT54291.2022.9824068

[12.] Seo, J., & Gordish-Dressman, H. (2007). Exploratory Data Analysis With Categorical Variables: An Improved Rank-by-Feature Framework and a Case Study. *International Journal of Human–Computer Interaction*, *23*(3), 287–314. https://doi.org/10.1080/10447310701702519

[13.] Pérez-Verdejo, J.M., Piña-García, C.A., Ojeda, M.M. *et al.* The rhythm of Mexico: an exploratory data analysis of Spotify's top 50. *J Comput Soc Sc* **4**, 147–161 (2021). https://doi.org/10.1007/s42001-020-00070-z

[14.] Santhana Lakshmi, V., Vijaya, M.S. (2023). An Exploratory Data Analysis on Air Quality Data of Trivandrum. In: Joshi, A., Mahmud, M., Ragel, R.G. (eds) Information and Communication Technology for Competitive Strategies (ICTCS 2022). ICTCS 2022. https://doi.org/10.1007/978-981-19-9638-2_68

[15.] Vignesh, K., Nagaraj, P. (2022). Analysing the Nutritional Facts in Mc. Donald's Menu Items Using Exploratory Data Analysis in R. In: Balas, V.E., Sinha, G.R., Agarwal, B., Sharma, T.K., Dadheech, P., Mahrishi, M. (eds) Emerging Technologies in Computer Engineering: Cognitive Computing and Intelligent IoT. ICETCE 2022. Communications in Computer and Information Science, vol 1591. Springer, Cham. https://doi.org/10.1007/978-3-031-07012-9_48

[16.] Da Silva, J. R., Junior, Campagna, D. P., Clua, E., Sarma, A., & Murta, L. (2020). Dominoes: An Interactive Exploratory Data Analysis Tool for Software Relationships. *IEEE Transactions on Software Engineering*, *48*(2), 377–396. https://doi.org/10.1109/tse.2020.2988241

[17.] Hassan-Montero, Y., De-Moya-Anegón, F., & Guerrero-Bote, V. P. (2022). SCImago Graphica: a new tool for exploring and visually communicating data. *El Profesional De La Informacion*. https://doi.org/10.3145/epi.2022.sep.02

[18.] Palocsay, S. W., Markham, I. S., & Markham, S. E. (2009). Utilizing and teaching data tools in Excel for exploratory analysis. *Journal of Business Research*, *63*(2), 191–206. https://doi.org/10.1016/j.jbusres.2009.03.008

[19.] Mackeprang, M., Strama, J., Schneider, G., Kuhnz, P., Benjamin, J. J., & Müller-Birn, C. (2018). Kaleidoscope. *Adjunct Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 75–77. https://doi.org/10.1145/3266037.3266106

[20.] Furcila, D. (2017). [P2-426]: INTOOL EXPLORER: A NEW INTERACTIVE EXPLORATORY DATA ANALYSIS TOOL IN ALZHEIMER's DISEASE. *Alzheimer S & Dementia*, *13*(7S_Part_16). https://doi.org/10.1016/j.jalz.2017.06.1082

[21.] Pitroda, H. (2022). A Proposal of an Interactive Web Application Tool QuickViz: To Automate Exploratory Data Analysis. *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*. https://doi.org/10.1109/i2ct54291.2022.9824068

[22.] Otero-Escobar, A. D., & Velasco-Ramírez, M. L. (2023). Study on Exploratory Data Analysis Applied to Education. *Study on Exploratory Data Analysis Applied to Education*. https://doi.org/10.1109/icev59168.2023.10329702

[23.] Chen, C., Hoffswell, J., Guo, S., Rossi, R., Chan, Y., Du, F., Koh, E., & Liu, Z. (2023). WhatsNext: Guidance-enriched Exploratory Data Analysis with Interactive, Low-Code Notebooks. In *IEEE Symposium on Visual Languages and Human Centric Computing (VL/HCC)* (Vol. 15, pp. 209–214). https://doi.org/10.1109/vl-hcc57772.2023.00033

[24.] Deming, C., Dekkati, S., & Desamsetti, H. (2018). Exploratory Data Analysis and Visualization for Business Analytics. Asian Journal of Applied Science and Engineering, 7(1), 93–100. https://doi.org/10.18034/ajase.v7i1.53

[25.] Rahmany, M., Zin, A. M., & Sundararajan, E. A. (2020). COMPARING TOOLS PROVIDED BY PYTHON AND R FOR EXPLORATORY DATA ANALYSIS. *IJISCS (International Journal of Information System and Computer Science)*, *4*(3), 131. https://doi.org/10.56327/ijiscs.v4i3.933

[26.] Dhariwal, A., Junges, R., Chen, T., & Petersen, F. C. (2021). ResistoXplorer: a web-based tool for visual, statistical and exploratory data analysis of resistome data. *NAR Genomics and Bioinformatics*, *3*(1). https://doi.org/10.1093/nargab/lqab018

APPENDIX A

This appendix contains the code snippets utilized in the development of the project. The code snippets showcase key functionalities, including library imports, application initialization, and visualization setup**.**

A.1: Library Imports

The following code snippet demonstrates the essential Python libraries imported for data manipulation, visualization, and application development using Streamlit.

```
# Importing essential libraries
import streamlit as st        # Framework for creating interactive web applications
import pandas as pd           # Data manipulation and analysis
import numpy as np            # Numerical computations and array operations
import base64                 # Encoding and decoding base64 binary data
import plotly.graph_objects as go  # Advanced visualizations using Plotly
import plotly.express as px   # Simplified interface for Plotly visualizations
```

This snippet sets the foundation for the application's core functionalities by importing the required libraries, ensuring the app can handle data manipulation, user interaction, and dynamic visualizations seamlessly.

A.2: Page Configuration and Logo Integration

The following code configures the Streamlit application for optimal usability and embeds a custom logo in the sidebar:

```
# Set page config first - MUST be the first Streamlit command
st.set_page_config(
    page_icon="assets/logo.png",
    page_title="PlotMaster",
    layout="wide",  # Ensures wide layout
    initial_sidebar_state="expanded"  # Sidebar is expanded by default
)

def add_logo(logo_path):
    """Read and return a resized logo"""
    with open(logo_path, "rb") as f:
```

```
        data = base64.b64encode(f.read()).decode()

    logo_html = f'''
        <style>
            [data-testid="stSidebarNav"] {{
                background-image: url("{data}");
                background-repeat: no-repeat;
                background-position: 60px 0px;
                background-size: 180px auto;
                padding-top: 140px;
            }}
        </style>
    '''

    st.markdown(logo_html, unsafe_allow_html=True)

# Add the logo
add_logo("assets/plotmaster-high-resolution-logo-transparent.png")
```

The st.set_page_config function customizes the app's layout, sidebar state, and branding by setting the page icon and title. The add_logo function embeds a logo in the sidebar using Base64 encoding, ensuring it adapts dynamically to the layout.

A.3: Multi-Page Navigation

The following code sets up navigation across multiple pages in the application, facilitating user access to various features:

```
# Define your pages
about_page = st.Page(
    page="views/about.py",
    title="About",
    icon=":material/home:",
    default=True
)

eda_1_page = st.Page(
    page="views/upload.py",
    title="Upload dataset",
    icon=":material/upload_file:"
)
```

```
eda_2_page = st.Page(
    page="views/datatype.py",
    title="Dataset Analysis",
    icon=":material/bar_chart:"
)

eda_3_page = st.Page(
    page="views/plots.py",
    title="Plots",
    icon=":material/query_stats:"
)

eda_4_page = st.Page(
    page="views/matrix.py",
    title="Correlation Matrix",
    icon=":material/grid_view:"
)

# Set up navigation
pg = st.navigation(
    {
        "Info": [about_page],
        "EDA": [eda_1_page, eda_2_page, eda_3_page, eda_4_page],
    }
)

pg.run()
```

- Pages are defined using the st.page() method, with attributes for title, icon, and source file.
- The st.navigation function organizes pages into logical groups (*Info* and *EDA*) for intuitive navigation.
- The pg.run() command initiates the navigation framework, ensuring smooth transitions between pages.

This implementation ensures a streamlined and professional user interface, laying the foundation for an interactive Exploratory Data Analysis (EDA) tool.

A.4: Session State Management for File Uploads

This snippet ensures that the application properly manages the session state for an uploaded file. It initializes the uploaded_file key in the session state if it is not already present, allowing consistent access to the uploaded file throughout the application.

```
# Check if 'uploaded_file' is already in session state
if 'uploaded_file' not in st.session_state:
    st.session_state['uploaded_file'] = None
```

Purpose:

Streamlit's session state provides a mechanism to persist variables across user interactions. The above code snippet ensures that the uploaded_file key exists in the session state and initializes it with a value of None if it is absent.

Functionality:

- When a file is uploaded, it can be stored in the st.session_state['uploaded_file'] variable.
- If the user interacts with other elements, the session state retains the file's reference, avoiding the need for re-uploading.

Advantages:

- Enhances user experience by maintaining continuity across app interactions.
- Simplifies application logic by ensuring consistent variable initialization.

This is particularly useful in scenarios where users upload datasets or files for analysis. By storing the uploaded file in session state, the application avoids loss of data due to app reloads or navigation between pages.

A.5: Custom Horizontal Line with Gradient Styling

This snippet defines a function to create a visually appealing horizontal line with gradient styling using HTML and CSS. The line can be customized by specifying its width percentage and is displayed using Streamlit's Markdown feature.

```
def Line_Break(width):
        line_code=f"""

            <hr style="border: none; height: 2px;width: {width}%;
background: linear-gradient(90deg, rgba(216,82,82,1) 13%, rgba(237,242,6,1)
57%, rgba(226,0,255,1) 93%); margin: 0 auto;" />
```

```
            """
        st.markdown(line_code,unsafe_allow_html=True)

Line_Break(100)
```

This snippet is particularly useful in Streamlit applications where dividing content sections with aesthetic elements enhances the presentation and user experience.

A.6: File Uploader with Session State Management

This snippet demonstrates the implementation of a file uploader widget in Streamlit, ensuring that users can upload a dataset (CSV or Excel format) and the application remembers the uploaded file and its processed data using Streamlit's session state. This prevents the need for re-uploading the file during subsequent interactions.

```
# File uploader widget
uploaded_file = st.file_uploader("Choose a file", type=["csv", "xlsx"])

# If a new file is uploaded, process and store it
if uploaded_file is not None:
    # Store the uploaded file in session state
    st.session_state['uploaded_file'] = uploaded_file

    # Read the uploaded file based on file type
    if uploaded_file.name.endswith(".csv"):
        df = pd.read_csv(uploaded_file)
    elif uploaded_file.name.endswith(".xlsx"):
        df = pd.read_excel(uploaded_file)

    # Store the DataFrame in session state
    st.session_state['uploaded_data'] = df
    st.success("Dataset uploaded successfully!")

# If a dataset is already in session state, use it without requiring re-
upload
elif st.session_state['uploaded_file'] is not None:
    uploaded_file = st.session_state['uploaded_file']

    # Check if the dataset is already stored in session state
    if 'uploaded_data' in st.session_state:
        df = st.session_state['uploaded_data']
```

```
        st.success("Using previously uploaded dataset!")
    else:
        # Process the file if 'uploaded_data' is missing for some reason
        if uploaded_file.name.endswith(".csv"):
            df = pd.read_csv(uploaded_file)
        elif uploaded_file.name.endswith(".xlsx"):
            df = pd.read_excel(uploaded_file)
        # Store the DataFrame in session state
        st.session_state['uploaded_data'] = df
else:
    st.warning("Please upload a dataset to proceed.")

# Now 'df' contains the DataFrame for further use, whether from a fresh
upload or session state.
```

Enables the application to upload and manage datasets while maintaining state across different user interactions. The st.file_uploader widget allows users to upload a file, restricted to CSV and Excel formats. Uploaded files are processed into a Pandas DataFrame (df) and stored in st.session_state. If a file is already stored in the session state, the application reuses it without requiring a new upload.

Differentiates processing logic based on file extensions (.csv or .xlsx). Ensures continuity by retaining file and data across user interactions. Provides success or warning messages to guide the user. This snippet is ideal for applications requiring repeated interaction with uploaded datasets, such as exploratory data analysis tools or dashboards.

A.7: Interactive Column Selection and Plotting

This code snippet demonstrates how to implement an interactive column selection and dynamic plotting feature in Streamlit. It enables users to select columns for analysis and renders appropriate plots based on the datatype of the selected columns.

```
if 'uploaded_data' in st.session_state:
    st.title('Plots')

    # Initialize session state for selected columns if it doesn't exist
    if 'selected_columns_state' not in st.session_state:
        st.session_state.selected_columns_state = []

    # Function to update session state
```

```
    def update_selected_columns():
        st.session_state.selected_columns_state =
st.session_state.selected_columns_widget

    # Column selection widget
    selected_columns = st.multiselect(
        "Select Columns for Analysis",
        options=data.columns.tolist(),
        default=st.session_state.selected_columns_state,
        key='selected_columns_widget',
        on_change=update_selected_columns
    )

    # Display tabs and generate plots for selected columns
    if selected_columns:
        Line_Break(100)  # Visual separator
        tabs = st.tabs(selected_columns)
        for tab, column in zip(tabs, selected_columns):
            with tab:
                colmuns_datatype = check_datatype(data, column)

                if colmuns_datatype == "object":
                    categorical_variable(data, [column])
                elif colmuns_datatype in ["float64", "int64"]:
                    numarical_Features(data, [column])

else:
    st.warning("Upload the dataset to view the plots")
```

Purpose:

This snippet is designed to allow users to interactively select dataset columns for analysis. By dynamically categorizing columns into numerical and categorical types, the code ensures that users can explore their data effectively through appropriate visualizations.

Functionality:

The code begins by verifying the existence of an uploaded dataset in Streamlit's session state. If no dataset is available, a warning message prompts the user to upload one. Once a dataset is present, the code initializes a session state variable (selected_columns_state) to store user selections persistently. A multi-select widget then allows users to choose

specific columns for analysis. Based on these selections, the application dynamically creates tabs, each corresponding to a column, and categorizes the column for either categorical or numerical analysis.

State Persistence:

The snippet leverages Streamlit's session state to retain user interactions across application reloads or updates. This ensures that previously selected columns remain available, providing a seamless user experience. By using the on_change parameter, any updates to the selection are immediately reflected in the session state.

Dynamic Tabs:

To enhance organization, tabs are generated dynamically for each selected column. This structure allows users to switch between columns effortlessly, with each tab displaying specific visualizations or data summaries for the associated column.

Datatype-Specific Analysis:

The code includes logic to determine the datatype of each selected column. Categorical columns invoke the categorical_variable function, generating insights such as frequency distributions. Numerical columns, identified as either float64 or int64, call the numarical_Features function to produce relevant statistics or visualizations.

Advantages:

This feature improves user experience by providing an intuitive interface for exploring dataset columns. By automatically tailoring the analysis to the datatype of each column, it ensures that the displayed insights are always relevant. Additionally, dynamic tabs keep the interface clean and user-friendly, even when multiple columns are selected.

This approach is particularly useful in exploratory data analysis applications. It empowers users to examine individual dataset columns in depth, enabling a better understanding of their data and facilitating data-driven decision-making.