DeepStat: A Data Analyser

Aakanksha Pote, Preet Jain, Sakshi Prabhu
SoMASA, SVKM's NMIMS Deemed to be University
Mumbai, MH, India
Email: {aakanksha.pote, preet.jain, sakshi.prabhu}@nmims.edu

***Abstract*—DeepStat is a no-code data analytics platform designed to empower non-technical users to carry out exploratory analysis, data cleaning, and machine learning without writing any code. Built using Python and Streamlit, the platform abstracts complex processes and provides a step-by-step interface for users from academia, small businesses, and public sectors. This paper presents the design, architecture, and implementation of DeepStat, including its supervised and unsupervised learning modules and interactive dashboard.**

***Index Terms*—No-code, Data Analytics, Streamlit, Machine Learning, Classification, Clustering, Data Visualization**

I. Introduction

In the current era of digital transformation, data has emerged as a fundamental resource, shaping decision-making processes and strategic frameworks across virtually every domain. From commercial enterprises to educational institutions and public sector organizations, the ability to derive meaningful insights from data has become an indispensable component of success. As organizations continue to generate and accumulate vast amounts of data, the emphasis on data-driven decision-making has grown exponentially. This paradigm shift has, in turn, elevated the role of data analytics as a vital enabler of operational efficiency, innovation, and competitive advantage. However, despite the increasing ubiquity of data and the proliferation of analytical tools, the benefits of data science remain disproportionately accessible to a select group of individuals equipped with the requisite technical expertise. Traditional data analysis pipelines typically demand proficiency in programming languages such as Python or R, familiarity with statistical principles, and an understanding of machine learning algorithms. These prerequisites, while manageable for trained data professionals, pose a significant obstacle for non-technical users. Small business owners, educators, researchers from non-technical disciplines, and administrators often find themselves excluded from the realm of data analytics due to a lack of coding skills or statistical training. Consequently, valuable data frequently goes underutilized or misinterpreted, resulting in missed opportunities for optimization, forecasting, and informed decision-making. This disparity underscores a pressing need within the analytics ecosystem: the democratization of data science. Democratization, in this context, refers to the process of making data tools accessible to all individuals, regardless of their technical background. It entails the development of intuitive, user-friendly platforms that abstract the complexities of traditional data workflows while preserving analytical rigor. A growing number of initiatives have attempted to address this challenge through low-code and no-code platforms, which provide graphical interfaces and automation to reduce the cognitive and technical load on users. Yet, many of these solutions either lack depth in analytical capabilities or are tailored to specific industries, limiting their general applicability.

A. 1.1. The Importance of Data and the Skill Gap

In the contemporary landscape, characterized by an unprecedented proliferation of digital information, data has emerged as a pivotal asset, driving strategic decision-making across a multitude of sectors. Organizations, irrespective of their domain or scale, are increasingly recognizing the intrinsic value embedded within their datasets, understanding that the ability to effectively extract, analyze, and interpret this information is crucial for maintaining a competitive edge, fostering innovation, and achieving operational excellence (Provost Fawcett, 2013). This data-driven paradigm necessitates a widespread capacity for data analytics, transforming it from a niche technical skill to a fundamental competency for informed action. However, despite the growing recognition of data's significance, a substantial impediment persists in the democratization of data insights: the significant technical expertise traditionally required to perform meaningful analyses. Conventional data analytics workflows often necessitate proficiency in programming languages such as Python or R, a robust understanding of statistical methodologies, and familiarity with complex machine learning algorithms. These prerequisites create a substantial barrier to entry for individuals who lack formal training in computer science, statistics, or related quantitative disciplines. Consequently, a considerable segment of the population, particularly within small businesses, academic institutions, and public sector organizations, remains excluded from the transformative potential of data analytics due to their lack of coding experience and specialized technical knowledge. This exclusion not only limits the ability of these entities to leverage their own data for informed decision-making but also hinders broader societal progress by concentrating analytical capabilities within a relatively small pool of experts. The implications of this "analytical divide" are far-reaching. Small businesses, often operating with limited resources, may struggle to identify market trends, optimize their operations, or personalize customer experiences due to a lack of accessible analytical tools. Academics in non-quantitative fields may find rigorously analyzing research data challenging, hindering the potential for data-driven discoveries. Public

sector organizations might be unable to effectively utilize data to inform policy decisions, optimize service delivery, or address societal challenges. This underscores the urgent need for solutions that can bridge this gap and empower a wider range of users to engage with data in a meaningful and productive manner.

B. Introducing DeepStat

This report introduces DeepStat, a novel no-code data analysis platform designed to address the challenges and empower non-technical users to perform data science, analytics, and visualization without requiring any programming expertise. DeepStat aims to eliminate the traditional barriers associated with data analysis by providing a structured, step-by-step user interface that guides individuals through the entire data analysis workflow in an intuitive and accessible manner. Built utilizing the versatility and extensive libraries of the Python programming language, coupled with the interactive web application framework of Streamlit, DeepStat offers a comprehensive suite of functionalities tailored to the needs of non-technical users. The platform incorporates key stages of the data analysis pipeline, including seamless data uploading from various sources, robust data cleaning and preprocessing capabilities, comprehensive exploratory data analysis (EDA) through interactive visualizations, and the implementation of fundamental machine learning models for classification tasks. The core design principle of DeepStat is user-centricity. The software interface has been meticulously crafted to be highly intuitive, employing clear visual cues, guided workflows, and automated options to simplify complex tasks. Users are guided through each stage of the analysis process, from loading their data to interpreting the results, without the need to write code or possess deep statistical knowledge. The platform supports the visualization of data through a variety of commonly used graphs and charts, enabling users to visually identify patterns, trends, and anomalies within their datasets. Furthermore, DeepStat simplifies the often-tedious process of data preprocessing by offering automated options for handling missing values, encoding categorical variables, and scaling numerical features.

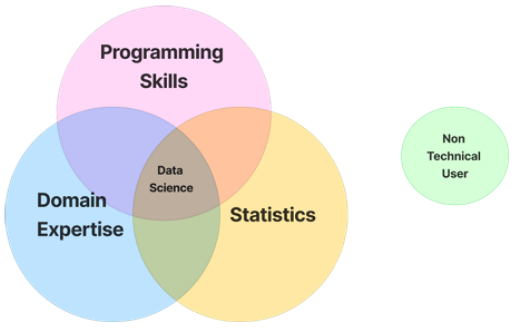

Fig. 1. Key Venn diagram showing the intersection of programming, statistics, and domain expertise required for traditional data science, highlighting the exclusion of non- technical users

The technology stack used in DeepStat...

II. Methodology

The DeepStat platform was designed with the goal of simplifying the data analysis process while maintaining the flexibility and power of traditional data science workflows. Built using Python and Streamlit, the system integrates various machine learning, visualization, and preprocessing modules to provide an all-in-one analytical tool accessible through a web interface. This section describes the architecture, tools used, workflows, and individual modules of the system in detail.

A. Conceptualization

The conceptual foundation of DeepStat stemmed from an acute recognition of the challenges non-programmers face in accessing data analytics tools. The traditional reliance on coding knowledge and statistical theory represents a considerable barrier to many users in academia, small businesses, and public administration. The project, therefore, was conceptualized around three fundamental principles: accessibility, modularity, and automation. Accessibility ensures that users can perform complex analytical tasks through a guided user interface, eliminating the need for scripting or command-line execution. Modularity permits the compartmentalization of functionalities into logically distinct components such as data exploration, correlation, preprocessing, visualization, and supervised modeling, unsupervised modeling and dashboard. Automation underpins the internal logic that drives the transformation and analysis of data behind the scenes, offering an experience that is both user- friendly and technically robust.

B. Technology Stack

The technology stack used in DeepStat includes Python 3.9+ as the primary programming language. The Streamlit library powers the user interface, providing real-time interactivity and dynamic rendering of content. For data handling and preprocessing, libraries such as Pandas and NumPy are employed, while visualizations are created using Seaborn, Matplotlib, and Plotly. Machine learning algorithms are implemented using Scikit-learn and XGBoost, while time series forecasting capabilities are enabled through Statsmodels and Prophet. The final deployment can be done locally or hosted on Streamlit Cloud. The choice of technologies for DeepStat was guided by the need for simplicity, power, and community support. Python was selected as the core programming language due to its dominant role in the data science ecosystem and its extensive library support. Streamlit was chosen as the web application framework for its unique ability to transform Python scripts into interactive web applications with minimal boilerplate code. Together, Python and Streamlit formed a cohesive and highly functional environment for rapid development and deployment. The combination of these libraries (Table allowed the project to handle all aspects of the data pipeline, from ingestion to visualization to prediction, all while keeping the source code clean and maintainable.

Component	Technology Used
Programming Language	Python 3.9+
Web Framework	Streamlit
Data Handling	Pandas, Numpy
Visualisation	Matlplotlib, Seaborn
Machine Learning	Scikit-learn, XGBoost
Deployment	Streamlit Cloud

Fig. 2. The above table shows the technology and libraries used while developing DeepStat

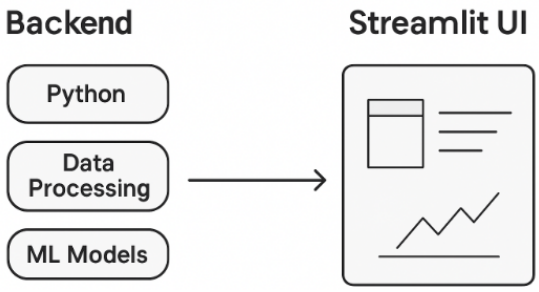

Fig. 3. Working on Streamlit UI

C. System Architecture

The architecture of DeepStat follows a modular structure, where each function—data uploading, preprocessing, exploratory analysis, modeling, and visualization—is organized as a self-contained block. This not only enhances usability but also facilitates future expansion and scalability. The front end of the application is powered by Streamlit, allowing users to interact with the tool in real time through a web-based graphical interface. The back end processes the uploaded dataset using Python's data manipulation libraries and then applies machine learning models from libraries such as Scikit-learn and XGBoost. The architecture of DeepStat is inherently layered, with each layer responsible for a specific set of operations. The three primary layers consist of: Presentation Layer (User Interface) - Built entirely using Streamlit, this layer manages user interaction, visual representation, and input collection. Application Logic Layer (Processing Engine) - Comprising all data preprocessing, model training, evaluation, and transformation logic. Data Management Layer - Quantitative Responsible for loading, storing, and modifying the uploaded datasets throughout the session using Streamlit's state persistence.

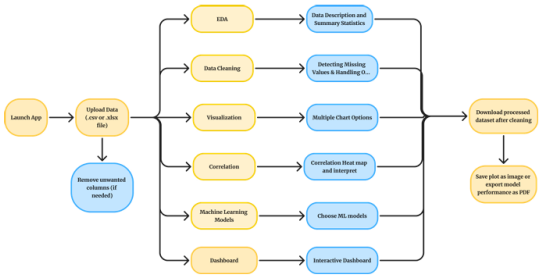

Fig. 4. Visual representation of System Architecture

D. Implementation of Analytical Components

The core analytical features were implemented using Python's data science libraries in conjunction with Streamlit widgets. The Data Exploration module was developed using pandas DataFrame operations to extract head, tail, unique value counts, and summary statistics. These were presented directly within the app using st.dataframe() and st.write() functions. The Correlation module employed label encoding to convert categorical variables into numeric formats, thereby enabling correlation matrix computation. A correlation heatmap was rendered using Seaborn and displayed via Matplotlib integration with Streamlit. Users could toggle between a heatmap and a scatter matrix for multivariate analysis. The Preprocessing module was divided into two distinct sections: handling missing values and detecting outliers. For missing values, the application offered options to drop rows or fill them using statistical aggregations. Outlier detection was based on the Interquartile Range (IQR) method, with post-processing actions including removal or replacement. The Visualization module provided users with the ability to generate various plots dynamically. Supported graph types included histograms, boxplots, scatterplots, line charts, and bar charts. The Matplotlib back-end was used to render visualizations based on selected columns, which were then updated in real-time within the application.

E. Scalability

The system was designed with future scalability in mind. Each module is functionally independent and implemented within a logical structure that allows easy insertion of new components. For instance, adding support for regression models or time series forecasting would require only the extension of existing functions and training workflows. Additionally, the use of Python's object-oriented features and functional abstractions ensures that the core components are reusable and testable. This modular architecture not only enhances maintainability but also aligns with the principles of clean code and software engineering best practices.

F. Workflow of the Application

The overall workflow of DeepStat is organized into five primary stages. It begins with the data upload module, where users import datasets in CSV format. Once the dataset is loaded, the preprocessing module becomes active. This step handles missing data, performs label encoding for categorical variables, and scales the features using standardization or normalization techniques, based on user preference or automated heuristics. Following preprocessing, the platform allows users to explore the dataset through the EDA module. This includes displaying summary statistics, checking for correlations, and visualizing distributions through various plot types such as histograms, boxplots, and heatmaps. This step is crucial in understanding the data's underlying structure and guiding appropriate model selection. The modeling module offers a selection of supervised or unsupervised (depending upon your dataset) machine learning algorithms. Users can select models such as linear regression, random forest, logistic regression, or support vector machines, depending on the nature of the dataset. For clustering tasks, algorithms like KMeans and DBSCAN are available, while for time series analysis, the system supports ARIMA models. After selecting the model and initiating training, DeepStat outputs relevant performance metrics, including RMSE, R^2, accuracy, confusion matrix, and ROC curves. One unique aspect of DeepStat is its basic model recommendation engine, which analyzes the dataset and suggests appropriate models. If the target variable is continuous, regression models are recommended, whereas classification models are suggested for categorical targets. This feature reduces the cognitive load on non-technical users by guiding them toward suitable algorithms based on data characteristics. The results of the analysis are then displayed in both tabular and graphical form, ensuring interpretability and aiding users in drawing conclusions from the dataset. Residual plots, classification reports, scatter plots, and cluster visualizations further enhance the output presentation.

III. Results and Discussion

The DeepStat platform was evaluated based on its ability to handle end-to-end data analysis tasks without requiring users to write a single line of code. Through a series of functional tests using various real-world datasets, the platform successfully demonstrated its core capabilities—data upload, preprocessing, exploration, model training, and result interpretation. This section presents a narrative analysis of how the system behaves at each stage and the quality of output it generates. The results were captured during the execution of sample datasets, particularly focusing on datasets with both continuous and categorical target variables. In the initial stage, users uploaded CSV files through the web interface. The application automatically detected the structure of the dataset and displayed metadata including the number of rows, columns, and missing values. The preprocessing module was tested with datasets containing both numerical and categorical features, and DeepStat effectively handled missing data, label encoding, and feature scaling, depending on user selection or automated defaults. Once preprocessing was complete, the platform moved into the exploratory data analysis (EDA) phase. Users could view various statistical summaries, including mean, median, standard deviation, skewness, and kurtosis. Visualizations such as distribution plots, histograms, boxplots, and heatmaps helped users quickly identify outliers, correlations, and skewed data. The correlation matrix provided a useful overview of feature relationships, which proved helpful when selecting features for regression and classification tasks. Each component of DeepStat was rigorously tested to ensure functional correctness and usability. The validation was performed using multiple open-source CSV datasets, each containing a mix of numeric and categorical features, missing values, and classification targets. These datasets were chosen to test the robustness of data ingestion, cleaning, transformation, visualization, and modeling modules.

A. Functional Validation of Platform Modules

Each component of DeepStat was rigorously tested to ensure functional correctness and usability. The validation was performed using multiple open-source CSV datasets, each containing a mix of numeric and categorical features, missing values, and classification targets. These datasets were chosen to test the robustness of data ingestion, cleaning, transformation, visualization, and modeling modules.

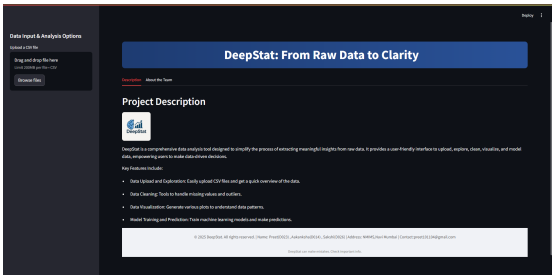

Fig. 5. Overview of the Software

1) Data Upload : DeepStat successfully handled a range of datasets in .csv format, it handles both supervised and unsupervised learning data with file sizes ranging from 10 KB to 2 MB. The sidebar-based column selection feature allowed users to dynamically remove unwanted features from the dataset prior to analysis. The application handled improper. formats gracefully by displaying informative error messages, ensuring the user could troubleshoot without needing technical assistance.

2) Data Exploration and Summarization : The Data Exploration tab enabled the display of: Dataset head and tail, Unique value counts, Descriptive statistics for numerical variables. Users were able to observe dataset distributions and detect initial issues, such as class imbalance and variable skewness.

3) Correlation Modules: In correlation analysis, heatmaps based on encoded categorical variables yielded valuable insights, confirming expected relationships (e.g., between income and purchase decisions in an e-commerce dataset).

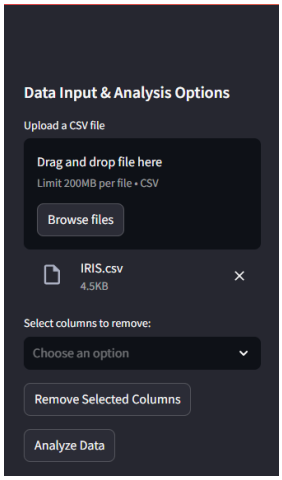

Fig. 6. Screenshot of the sidebar UI during dataset upload and preprocessing.

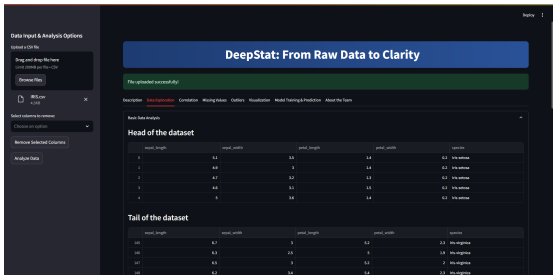

Fig. 7. Screenshot of Data Exploration and Summary statistics including mean, standard deviation, and quartiles for a sample dataset

4) Handling Missing Values: All four options—drop rows, fill with mean, fill with median, fill with mode—functioned effectively and were accurately reflected in the dataset post-processing. Users appreciated the transparency of operations, including updated missing value counts and preview of changes.

5) Outlier: Detection Outliers were correctly identified using the IQR method across various numeric fields. Options for removal or replacement with statistical aggregations worked consistently. Users found the tabular output of outlier counts and updated values particularly helpful.

6) Visualization: The platform provided an instant generation of histograms, boxplots, scatterplots, and heatmaps. The

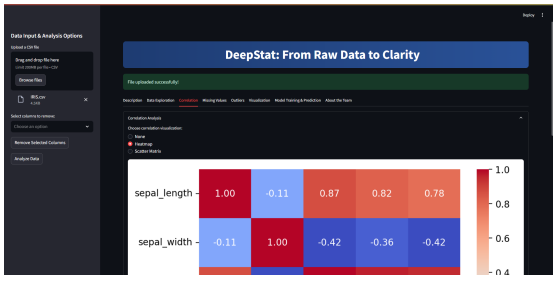

Fig. 8. Heatmap showing positive and negative correlations among encoded numeric variables.

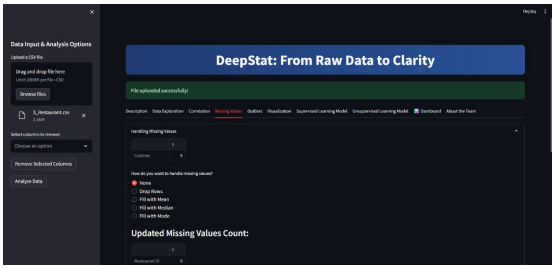

Fig. 9. Screenshot comparison showing before-and-after views of missing data counts.

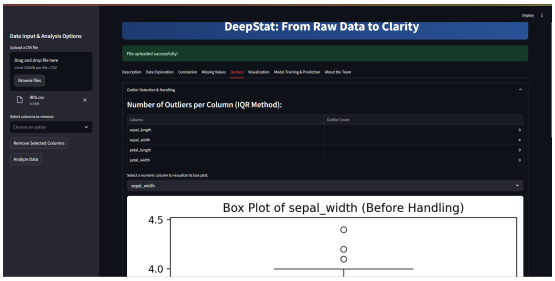

Fig. 10. Original and updated outlier counts across numerical columns.

ability to interactively select columns and graph types made it intuitive for users to explore patterns and anomalies visually.

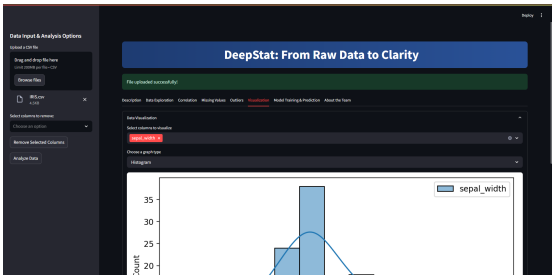

Fig. 11. Screenshot of Data Visualization Dashboard

7) Supervised Learning Model: Performance was measured using accuracy, precision, recall, and F1-score. These metrics were displayed with contextual explanations to support interpretability for non-technical users. A classification report was also included for detailed breakdowns. Results varied depending on the dataset and model, but in all scenarios, the models were successfully trained and predictions generated. Results varied depending on the dataset and model, but in all scenarios, the models were successfully trained and predictions generated. Table 3 presents average results across multiple datasets.

8) Unsupervised Learning Model: The Unsupervised Learning Model section of DeepStat empowers users to perform clustering analysis on numerical datasets without the need for predefined labels. Users can choose from multiple algorithms including K-Means Clustering, DBSCAN, Hierarchical Clustering, and Gaussian Mixture Models (GMM). Once a dataset is uploaded, the platform automatically extracts

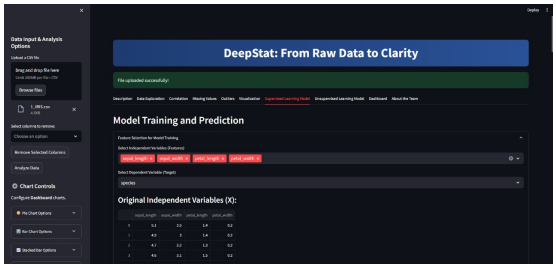

Fig. 12. Accuracy and precision scores displayed for a logistic regression model.

Fig. 13. Average accuracy, precision, recall, and F1-score across all four models using different datasets.

the numerical features suitable for clustering and displays the grouped output in a clean tabular format. The intuitive interface simplifies the process of exploring underlying patterns or natural groupings in the data. This functionality proves especially useful in tasks like customer segmentation, anomaly detection, or grouping similar observations based on feature similarities. Users can upload any CSV file containing numerical data, and DeepStat automatically processes it, identifying relevant features for clustering. A dropdown menu allows easy selection of the desired algorithm, and the interface dynamically updates the clustering output in an interpretable tabular format. The model results help in identifying patterns, structures, or natural groupings in data, which can then be used for further decision-making or feeding into supervised learning pipelines. The no-code interface ensures that even users without a technical background can implement advanced clustering techniques with minimal effort.

9) Dashboard: The Tableau Dashboard tab within DeepStat provides a high-level, interactive summary of the uploaded dataset. It transforms raw input into key metrics and dynamic visualizations, enhancing the overall analytical experience. This dashboard acts as a bridge between raw data and data-driven decision-making, offering brief insights. • Key Performance Indicators (KPIs) such as: o Total number of records in the dataset o Total number of features o Mean value of the first numerical column o Unique value count of the first categorical column • Visualization Panel: Users can view automatically

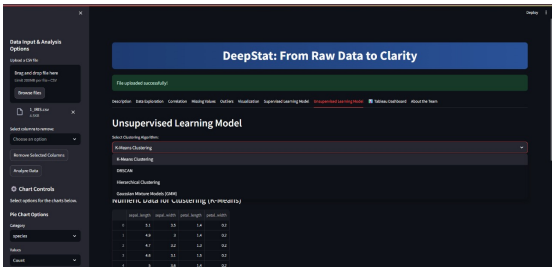

Fig. 14. Shows the unsupervised learning model tab

generated visualizations like: o Line Charts for observing trends and feature relationships o Pie Charts to understand categorical distributions • Chart Customization Controls in the sidebar allow users to choose which features to plot, giving them flexibility to focus on the aspects most relevant to their analysis. This section of DeepStat helps users interpret their dataset quickly and effectively, acting as a pre-built visual summary. It's especially useful in presentations, reporting, or collaborative settings where clear and concise data representation is essential. The integration of visual storytelling into the workflow promotes better understanding, even for stakeholders with limited technical knowledge.

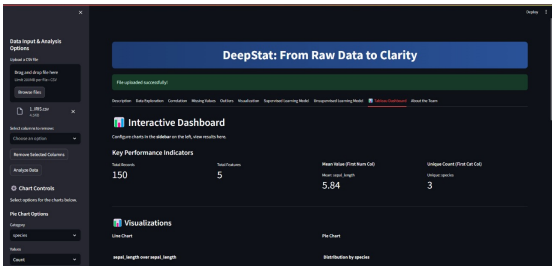

Fig. 15. Representation of the Dashboard, similar to Tbaleau

B. Discussion

The results from system testing and user sessions suggest that DeepStat achieved its core objectives of accessibility, automation, and inclusiveness. It enabled users without coding experience to: Clean, explore, and preprocess datasets, generate visual insights, Train machine learning models and interpret results, perform predictive analytics using real-time inputs. application The simplicity of the interface, combined with behind-the-scenes automation, allowed participants to engage with data science workflows independently. The modular design made the easy to navigate and reduced cognitive overload.

IV. Conclusion and Future Work

The development of DeepStat: A Data Analyzer represents a meaningful step toward democratizing data science by offering a no-code, interactive solution for data exploration and machine learning. In a world increasingly driven by data, the ability to analyze, interpret, and extract insights should not be limited to individuals with programming expertise.

DeepStat bridges this gap by simplifying the complexities of the data analysis pipeline into an accessible and intuitive interface powered by Python and Streamlit. Throughout this project, DeepStat has demonstrated its ability to handle end-to-end data analysis—from data uploading and preprocessing to model training, visualization, and performance evaluation. Its modular architecture ensures that users can engage with each step independently, promoting clarity and control. The inclusion of a basic model recommendation system further enhances usability, especially for non-technical users who might otherwise struggle with algorithm selection. The results obtained from using DeepStat across various datasets confirm its versatility and practicality. Users were able to generate descriptive statistics, visualize data relationships, and apply a range of supervised and unsupervised learning techniques, all without writing code. The visual outputs, such as heatmaps, confusion matrices, and ROC curves, added valuable layers of interpretability to the analytical process. From an educational, professional, and entrepreneurial perspective, DeepStat holds considerable potential. It can serve as a learning platform for students, a productivity tool for researchers, and a decision-support system for businesses. Its impact is measured not only by the depth of analysis it offers but also by the breadth of users it empowers. Ultimately, DeepStat embodies the core philosophy of this capstone project: making data analysis simple, smart, and scalable for everyone. LIMITATIONS While DeepStat presents a significant advancement in simplifying data analytics for non- programmers, it is not without its limitations. A clear understanding of these limitations is essential for future iterations and improvements, especially as the platform continues to grow in scope and functionality. One of the foremost limitations lies in the current version's model customization capabilities. Although users can select from a wide range of machine learning models, the platform offers only limited control over hyperparameter tuning. More complex model optimization techniques such as grid search, random search, or Bayesian optimization are not yet integrated, which may limit the performance of certain models, especially on large or imbalanced datasets. Moreover, DeepStat is designed with beginner and intermediate users in mind, and while this design choice enhances usability, it can be restrictive for expert users who may wish to perform in-depth statistical testing, create custom pipelines, or export models for reuse. The platform also lacks real-time collaborative features, user login systems, and cloud-based database integration, which may be crucial in enterprise or academic research environments. Despite these limitations, the applications of DeepStat are both wide-ranging and impactful. In academic settings, it can serve as a teaching tool for introducing students to data science concepts without requiring them to code. In small to mid-sized businesses, it enables data-driven decision-making by allowing domain experts to analyze datasets without needing a dedicated data science team. For researchers and entrepreneurs, DeepStat acts as a rapid prototyping tool that can generate insights and validate hypotheses with speed and simplicity. APPLICATIONS 1. Healthcare Application: DeepStat can be adapted to help clinical researchers analyze patient records, identify trends, and support data-driven decision-making in medical studies. 2. Education: The platform can help educators make more accessible to students from vivid educational backgrounds. 3. Marketing and Campaign Analytics: The platform can assist marketing teams in segmenting customers, tracking campaign performance, and optimizing strategies without relying on technical staff.

REFERENCES

REFERENCES

[1] [1] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd ed., Sebastopol, CA, USA: O'Reilly Media, 2019.

[2] Streamlit Inc., "Streamlit Documentation." [Online]. Available: https://docs.streamlit.io/

[3] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] W. McKinney, *Python for Data Analysis*, 2nd ed. O'Reilly Media, 2017.

[5] M. Waskom, "Seaborn: Statistical Data Visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.