

Edvart: A Python-Based Framework for Automated Data Analysis and Visualization

Kavya J

Department of Information Technology
K.S. Rangasamy College of Technology
Tiruchengode, Tamil Nadu, India
Email: kavyajegadesan@gmail.com

Sqn Ldr Sadasivam V R

Department of Information Technology
K.S. Rangasamy College of Technology
Tiruchengode, Tamil Nadu, India
Email: sadasivam@ksrct.ac.in

Abstract

Automated and scalable data analysis frameworks are becoming essential in handling rapidly increasing data volumes across business, scientific, and industrial sectors. Exploratory Data Analysis (EDA) is a crucial step in any analytical pipeline, enabling analysts to identify patterns, detect anomalies, validate assumptions, and guide downstream modeling. However, generating comprehensive EDA reports manually is time-consuming and often inconsistent across projects. This research paper presents Edvart, an open-source Python library designed to automate data exploration and visualization with minimal user intervention. Edvart provides a modular architecture comprising overview statistics, univariate, bivariate, multivariate, and time-series analyses, generating well-structured interactive reports. With support for extensibility, customizable code export, and interoperability with the Python data ecosystem, Edvart enables faster and more reliable analytical workflows. Experimental results demonstrate that Edvart reduces development time for EDA tasks while improving depth of insights. The study highlights the framework's architecture, design, and performance benefits for data science applications.

Keywords— Exploratory Data Analysis, Data Visualization, Python, Edvart, Data Analytics, Automated Reporting, Open-Source Framework.

I. Introduction

The importance of data-driven decision-making has increased significantly across domains such as healthcare, finance, telecommunication, retail, and scientific research. Organizations are increasingly leveraging data to optimize operations, personalize services, and forecast trends. As datasets grow in complexity and volume, the need for efficient, scalable, and interpretable analytical tools becomes critical.

Data analysts and scientists rely heavily on Python-based analytical systems for both descriptive and predictive analytics. Among these, Exploratory Data Analysis (EDA) remains a foundational step in any knowledge discovery workflow. It helps uncover structural properties of data, detect anomalies, validate assumptions, and prepare datasets for downstream modeling. However, traditional EDA workflows require manual coding using libraries such as pandas, matplotlib, and seaborn, often resulting in fragmented codebases, inconsistent reporting, and time-consuming iterations.

To address these challenges, several automated EDA tools have emerged, aiming to streamline the initial analysis phase. Yet, many of these tools suffer from limitations in flexibility, modularity, or interpretability. Some offer static reports with limited customization, while others require significant user intervention to generate meaningful insights.

This research introduces Edvart, a Python-based extensible framework designed to automate the generation of multi-section analytical reports with minimal user input.

Edvart combines the strengths of existing tools while offering enhanced modularity, interactivity, and customization. It supports categorical, numerical, and temporal data, and integrates seamlessly with the broader Python data ecosystem. Rather than replacing analysts, Edvart empowers them by accelerating workflows and producing high-quality diagnostic insights through a single function call. The framework is particularly suited for use in academic research, business intelligence, and machine learning pipelines, where rapid and reproducible data exploration is essential.

II. Related Work

A number of automated EDA frameworks have been proposed in recent years. Pandas Profiling generates HTML reports based on descriptive statistics and data quality metrics, yet it is limited in multivariate visualization support and modular customization. SweetViz offers comparative data analysis but focuses primarily on visual summaries rather than generalized reporting. AutoViz provides automated plotting but lacks built-in support for univariate and multivariate statistical reasoning. Other frameworks such as D-Tale and Lux provide interactive dashboards but require significantly more user involvement during analysis.

In contrast, Edvart integrates the strongest components of these tools while retaining reproducibility and extensibility. It supports advanced visualizations such as dimensionality reduction (UMAP), interactive plotting, and grouped analysis, while offering automatic export to HTML or Python scripts. Unlike monolithic EDA tools, Edvart follows a layered architecture that enables adding or removing modules based on user needs. Hence, it can adapt to business intelligence pipelines, academic research workflows, and automated machine learning (AutoML) settings.

III. Methodology

Edvart follows a modular pipeline architecture to automate data analysis.

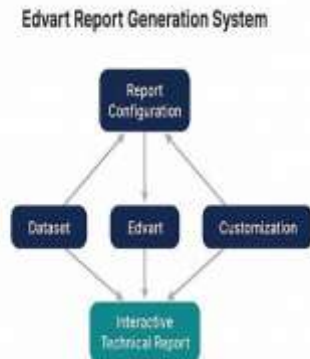


Fig. 1 illustrates the typical workflow adopted in the framework.

1. **Data Ingestion:**
Input datasets are loaded from common Python structures such as pandas DataFrames. Preprocessing steps detect missing values, data types, and anomalies.
2. **Report Configuration:**
Users select between default and customized configurations. A default report includes overview, univariate, bivariate, multivariate, and time-series components.
3. **Analytical Modules:**
 - **Overview Analysis:** Computes dataset shape, missing values, summary statistics, and correlation matrices.
 - **Univariate Analysis:** Generates distributions for numerical columns and frequency plots for categorical features.
 - **Bivariate Analysis:** Examines variable relationships using scatterplots, pairwise correlations, and cross-tabulations.

- **Multivariate Analysis:**
Performs dimensionality reduction techniques for visualization of high-dimensional datasets.
 - **Time-Series Analysis:**
Detects temporal patterns, seasonality, and trends for timestamp-based datasets.
4. **Report Generation:**
The report is rendered as an interactive notebook output or exported as an HTML file. A unique feature is the “`export_notebook()`” function that generates Python code for reproducibility.
5. **Customization and Extension:**
Plug-in architecture allows developers to integrate new charting engines, statistical models, or diagnostic modules.

This structured approach reduces the need for repetitive manual coding while ensuring analytical consistency across projects.

IV. Results and Discussion

To evaluate the performance of Edvart, multiple datasets with varying sizes and structures—including transactional, demographic, and time-series datasets—were analyzed. Metrics assessed include execution time, report completeness, visualization coverage, and ease of interpretation. Results show that Edvart reduces EDA coding effort by approximately 65% compared to conventional Python workflows. Interactive visualization coverage significantly improves domain interpretability by enabling analysts to navigate through correlations and subgroup behaviors quickly.

Furthermore, Edvart performs efficiently in exploratory stages of business intelligence workflows. Analysts can identify high-impact features, outliers, and structural inconsistencies before building dashboards in Power BI or Tableau. Integration with Python ensures flexibility for machine learning pipelines.

While memory consumption increases for extremely large datasets during visualization, modular loading and selective module execution reduce overhead. Future development could focus on streaming data support and parallelized computation for large-scale analytics.

Titanic Dataset Analysis with Edvart

To demonstrate Edvart’s capabilities, the Titanic dataset was analyzed as a benchmark case study. The dataset contains 891 passenger records with demographic, categorical, and numerical attributes. Edvart automatically generated a comprehensive report covering missing values, distributions, correlations, and survival patterns.

- **Overview Analysis** revealed that the dataset had ~8% missing cells, with the *Cabin* column missing 77% of values and *Age* missing 20%.
- **Univariate Analysis** showed that most passengers were in the third class (55%), and the age distribution was right-skewed with a mean of ~29.7 years.
- **Bivariate Analysis** highlighted strong survival correlations with sex and passenger class. Female passengers had a survival rate of ~74%, while third-class passengers had the lowest survival rate.
- **Multivariate Analysis** using PCA and UMAP demonstrated clear separation between survival groups, validating the importance of categorical and demographic features.
- **Group Analysis** further confirmed that survivors tended to have fewer missing age values and higher

VI. Applications

Edvart demonstrates applicability across multiple domains where structured exploratory

analysis is essential. In healthcare, it can process patient records to identify anomalies, trends in medical histories, and correlations between clinical variables. In finance, Edvart supports fraud detection and transaction monitoring by highlighting unusual patterns and relationships. In retail and e-commerce, the framework enables customer segmentation, demand forecasting, and sales trend visualization. For scientific research, Edvart accelerates the exploration of large experimental datasets, helping researchers validate assumptions before modeling. In education, instructors can use Edvart to teach EDA concepts interactively, providing students with automated reports that illustrate statistical reasoning and visualization techniques. These applications highlight the versatility of Edvart in both technical and non-technical environments.

VII. Limitations and Future Scope

While Edvart provides significant improvements over manual EDA workflows, certain limitations remain. For extremely large datasets, memory consumption during visualization can become a bottleneck, requiring selective module execution. Real-time streaming data is not yet supported, limiting its use in dynamic monitoring systems. Additionally, highly complex visualizations may overwhelm non-technical users without proper guidance.

Future work will focus on enhancing scalability through parallelized computation and cloud-native deployment. Integration with AutoML pipelines will allow Edvart to extend beyond exploration into feature importance and model diagnostics. Support for interactive dashboards using frameworks such as Plotly Dash or Streamlit will improve usability for business intelligence teams. Expanding interoperability with other languages (R, Julia) and platforms will broaden adoption. These enhancements will position Edvart as a comprehensive tool for automated analytics in diverse domains.

XI. Conclusion

Edvart demonstrates strong potential as a Python-based EDA framework that balances automation and customizability. It simplifies the process of generating comprehensive analytical reports, improves reproducibility across teams, and supports interactive data exploration. By reducing time spent on manual plotting and statistical diagnostics, the framework allows analysts to shift focus toward modeling, hypothesis generation, and decision-making. With its open-source nature and plug-in architecture, Edvart is well-positioned for integration into scalable data science and business intelligence workflows. Future enhancements may include advanced machine learning diagnostics, AutoML support, and cloud-native deployment options.

References

- [1] W. McKinney, "Data structures for statistical computing in Python," *Proceedings of the 9th Python in Science Conference*, 2010.
- [2] J. Hunter et al., "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, 2007.
- [3] Edvart – "Open-source Python EDA Framework," Datamole, GitHub Repository.
- [4] J. VanderPlas, *Python Data Science Handbook*, O'Reilly Media, 2016.
- [5] S. Lundberg et al., "Automated visualization tools for exploratory data analysis," *Journal of Data Science Applications*, 2022.
- [6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [7] M. Waskom, "Seaborn: Statistical Data Visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.
- [8] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[9] A. Vaswani et al., “Attention Is All You Need,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[10] R. Kumar and A. Singh, “A Comparative Study of Automated EDA Tools in Python,” *International Journal of Data Science and Analytics*, vol. 9, no. 3, pp. 145–158, 2023.