International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

DataCraft

Prof. Milind Kulkarni[1], Vishal Mahajan[2], Harsh Mahale[3], Pratik Mehetre[4], Rashmit Mhatre[5], Jay Patil[6]

[1,2,3,4,5,6]Department of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, 411037, Maharashtra, India

Abstract:
This paper presents a web-based framework for automated machine learning analysis and recommendation that streamlines the data science workflow. The system automatically performs data analysis, preprocessing, and model recommendation through an intuitive interface. The framework integrates dataset characterization, intelligent preprocessing suggestions, and context-aware model selection based on data characteristics and problem type. Results demonstrate the system's effectiveness in reducing the time and expertise required for preliminary data analysis while maintaining analytical rigor. The framework's web-based interface makes it accessible to both novice and experienced practitioners, contributing to the democratization of machine learning practices.

Keywords: automated machine learning, data preprocessing, model recommendation, web-based framework

I. INTRODUCTION

The exponential growth in data generation and the increasing demand for machine learning solutions has created a significant need for automated tools that can streamline the data analysis and model selection process. While machine learning has become instrumental in various domains, the complexity of data preprocessing, feature engineering, and model selection often presents substantial barriers to entry for newcomers and challenges for experienced practitioners.

This research introduces a comprehensive web-based framework that addresses these challenges through automation and intelligent recommendation systems. Our system combines modern web technologies with advanced machine learning capabilities, utilizing Flask for backend processing, while the frontend employs HTML5, CSS3, and JavaScript to create an intuitive and responsive user interface. This architectural choice ensures accessibility across different platforms while maintaining robust processing capabilities.

The framework follows a systematic workflow where users begin by uploading their dataset through a user-friendly interface. Upon upload, the system automatically performs comprehensive statistical analysis, detecting data quality issues such as missing values and outliers, while generating interactive visualizations for better data understanding. The preprocessing pipeline then offers context-aware recommendations and handles common data issues automatically, including feature encoding and scaling. Based on the dataset characteristics and problem type (classification or regression), the system recommends suitable machine learning models, ranking them according to their appropriateness for the specific dataset.

Our research makes significant contributions by introducing a novel approach to automated machine learning workflow management through a web-based interface. The system demonstrates the effectiveness of combining modern web technologies with machine learning automation, establishing a framework that reduces the time and expertise required for preliminary data analysis while maintaining analytical rigor. The system automatically generates detailed reports containing visualizations, preprocessing steps, and model recommendations, providing users with actionable insights for their machine learning projects. The result is a practical solution that serves both novice and experienced practitioners, contributing to the democratization of machine learning practices.

II. Literature Review

[1] The paper presents comprehensive data preprocessing strategies for supervised learning, focusing on steps like instance selection, handling missing data, feature selection, and normalization to enhance model performance and reduce data complexity. Both studies highlight the critical role of preprocessing in improving data quality and model outcomes.

[2] The paper introduces Atlantic, an open-source Python framework designed to automate data preprocessing for supervised machine learning tasks. It integrates customizable mechanisms like datetime feature engineering, automated feature selection using H2O AutoML, categorical encoding, and advanced null imputation methods. The framework uses tree-based model ensembles to optimize preprocessing steps, ensuring the selection of effective methods that enhance model performance. Atlantic aims to reduce manual effort, minimize human bias, and provide a standardized approach for preprocessing, facilitating efficient and high-quality machine learning workflows.

[3] This paper highlights the importance of automating data analysis to enhance productivity and reduce errors, focusing on Python due to its extensive library support. It evaluates popular libraries like Pandas, Matplotlib, Seaborn, Scikit-learn, and TensorFlow, comparing usability, efficiency, and accuracy across tasks and industries. The study concludes that selecting the appropriate library based on user expertise and task type is critical for effective and accurate automated data analysis.

[4] This paper proposes an automatic data visualization system using machine learning, incorporating a meta-level feature engineering process to recommend appropriate visualizations. It designs meta-features to evaluate the significance of visualization results and constructs a recommendation model, tested on datasets from UCI ML Repository, Data.world, and R. The study finds that a decision tree-based model delivers the best performance for visualization recommendations.

[5] The paper "A Simple Approach for Selecting the Best Machine Learning Algorithm" discusses the strengths and limitations of various supervised learning algorithms, such as linear regression, decision trees, random forests, SVMs, and neural networks. It provides a comparative analysis to guide beginners in choosing the most suitable algorithm based on dataset characteristics, including linearity, noise, dimensionality, and preprocessing requirements. The paper proposes a streamlined approach for algorithm selection by analyzing training data and evaluating accuracy, thus simplifying the traditionally time-consuming trial-and-error process.

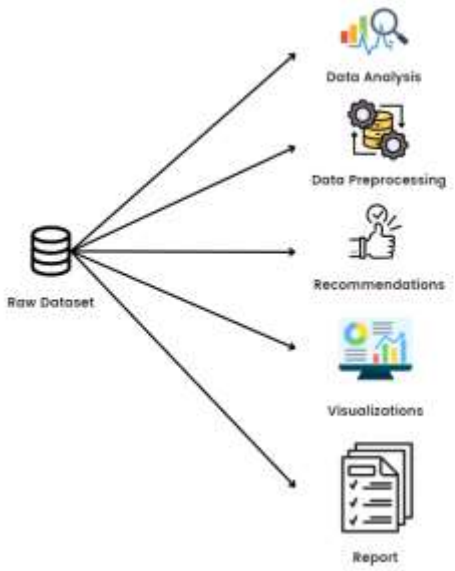
[6] AutoML focuses on automating key aspects of the machine learning workflow, including model selection, hyperparameter optimization, and feature engineering. This reduces manual effort, accelerates development, and improves accessibility for non-experts. It remains an evolving research field with significant potential for advancing ML efficiency and scalability.

[7] This paper examines Automated Machine Learning (AutoML) with a focus on model selection, a critical part of the machine learning pipeline. It explores various AutoML methods, including Combined Algorithm Selection and Hyperparameter (CASH) optimization, meta-learning, and Bayesian optimization, highlighting their advantages and use cases. The study concludes that while AutoML enhances efficiency and accessibility in machine learning, the choice of method depends on the specific task and remains an evolving field.

[8] This study introduces a Python-based Auto-preprocessing architecture for Machine Learning that automates and simplifies data preprocessing tasks while providing interactive, data-driven support. It identifies data issues, visualizes them for users, and recommends optimal preprocessing techniques after evaluating advanced methods. Tested on ten diverse datasets, the approach not only streamlines the process but also enhances model performance compared to manually preprocessed data.

[9] This paper introduces a procedure for analyzing unbalanced datasets commonly found in software benchmarking and productivity assessment. It identifies challenges like concealed and spurious factor impacts due to dataset imbalance and proposes a method using forward pass residual analysis to address these issues. The approach is demonstrated on artificial datasets and applied to real-world examples, emphasizing simplicity, flexibility in handling diverse factor types, and avoiding over analysis.

III. METHODOLOGY



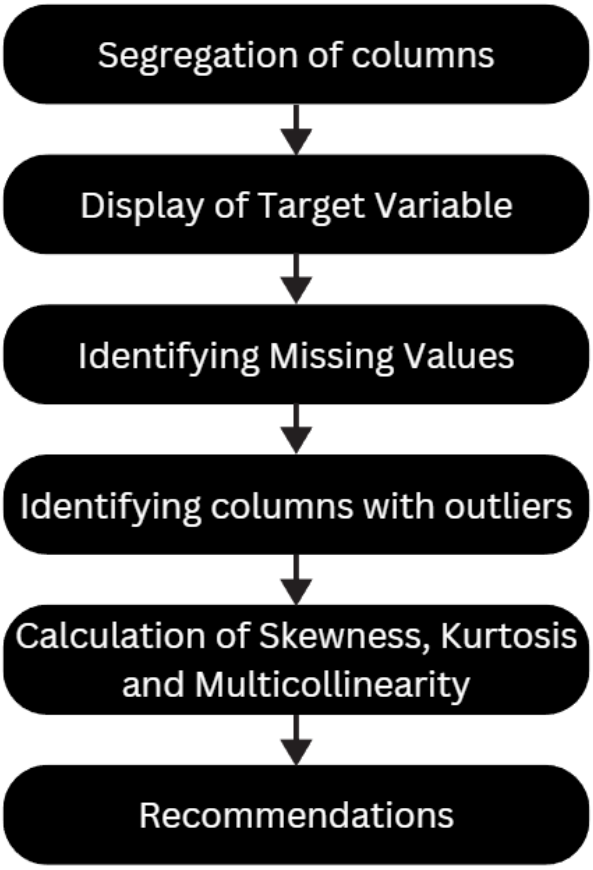
In this research project, a comprehensive machine learning data analysis pipeline was developed to transform raw datasets into actionable insights. The process begins with users uploading a CSV file through a web interface, which then undergoes a systematic workflow. Each uploaded dataset is automatically subjected to a multi-stage analysis pipeline involving dataset examination, preprocessing, feature engineering, model recommendation, and visualization generation. This approach enables automated, intelligent data exploration and model selection tailored to the specific characteristics of the input dataset.

The methodology encompasses five key stages:

1. Data Analysis
2. Data Preprocessing

3. Model Recommendation
4. Visualization Generation
5. Report Generation

Data Analysis:



Here is a detailed explanation of each component in the flowchart for analyzing a dataset:

Segregation of columns:
This step involves separating the columns in the dataset into different categories, such as numerical and categorical columns.

Display of Target Variable:
Target variable is treated as the last feature(column) of the dataset.

$$\text{Target Type}=\begin{cases}\text{"categorical"}, & if\ y.dtype == 'object' \\ \text{"numerical"}, & otherwise\end{cases}$$

Identifying Missing Values:
This step involves identifying the number of missing values (NaN or null) in each column of the dataset.

Identifying columns with outliers:
Outliers are detected using the 5th percentile and 95th percentile:
P5,P95=np.percentile(df[column],[5,95])

A column is flagged if any value lies outside:
Outliers={x|x < P5 or x > P95}

Calculation of Skewness, Kurtosis, and Multicollinearity:
Skewness:

$$S = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{3/2}}$$

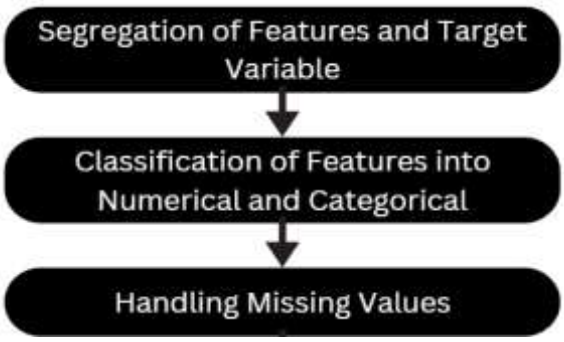
Kurtosis:

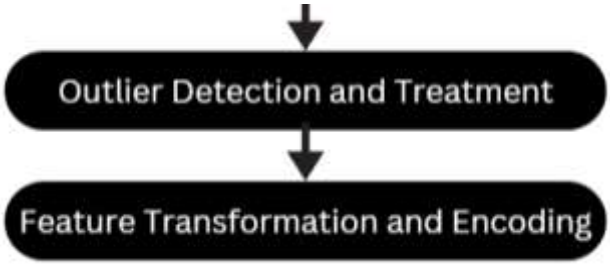
$$K = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{2}} - 3$$

Multicollinearity:
Multicollinearity is identified using the correlation matrix :
C=df[numeric_cols].corr()
Highly correlated pairs (|r|>0.8) are flagged:
rij>0.8

Recommendations:
Recommendations are derived based on:
Target Type:
Categorical: Suggest classification models.
Numerical: Suggest regression models.
Missing Values:
If missing values≤10%,suggest Median Imputation. Otherwise, Mean or Mode Imputation.
Outliers: Suggest techniques like trimming, Winsorization, or transformation.
Skewness: If |S|>0.5, recommend transformations such as log, square root, or Box-Cox.
Multicollinearity: Suggest feature selection or regularization.

Data Preprocessing:





Here is a detailed explanation of each component in the flowchart for preprocessing of dataset:

Segregation of Features and Target Variable:
The dataset is divided into input features (X) and the target variable (y), ensuring irrelevant columns are excluded. This allows for focused preprocessing and modeling.
Excluded columns consists features like id's, user id's, contacts, address, postal code's.

Classification of Features:
The input features are classified into numerical and categorical types to recommend type-specific preprocessing techniques.

Handling Missing Values:
Missing values are identified and imputed based on the feature type and distribution:
For numerical features:
Median imputation is applied if the feature is skewed (|S|>0.5):

$$S = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{3/2}}$$

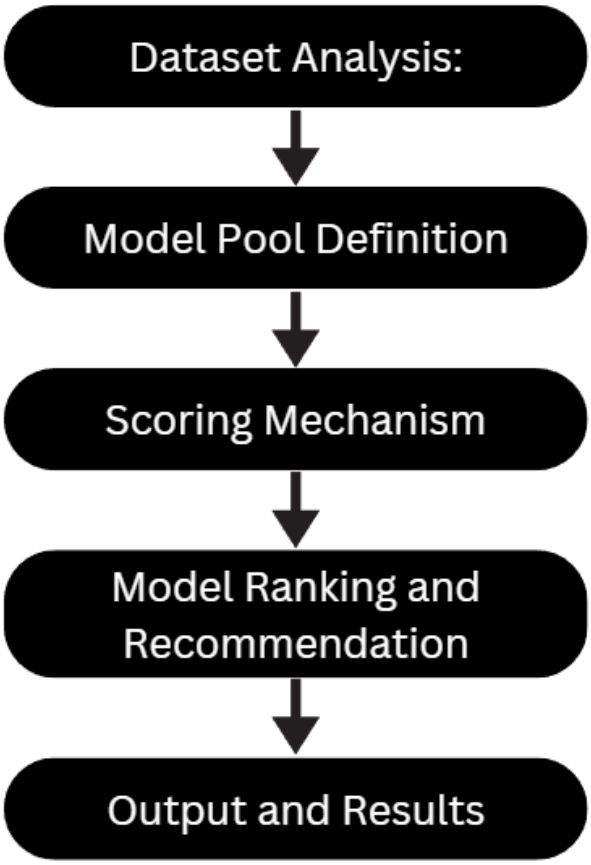
Mean imputation is applied for normally distributed features (|S|≤0.5|).
For categorical features, **mode imputation** is used to replace missing values with the most frequent category.

Outlier Detection and Treatment:
Outliers in numerical features are detected using the Interquartile Range (IQR) and treated using Winsorization to reduce their influence on the model:
A value is considered an outlier if:
$x < (Q1 - 1.5 \times IQR)$ or $x > (Q3 + 1.5 \times IQR)$
If the outliers exist in a particular column then winsorization is recommended for that column.

Feature Transformation and Encoding:
Standardization: Numerical features are scaled to ensure consistency in magnitude:
$z = (x-\mu)/\sigma$
One-Hot Encoding: Categorical features are transformed into binary columns for compatibility with machine learning algorithms.

Target Encoding: If the target variable is categorical, it is encoded into numerical values to facilitate classification tasks.
According to the conditions satisfied, particular preprocessing steps are recommended.

Model Recommendation:



Here is a detailed explanation of each component in the flowchart for model recommendation:

Dataset Analysis:
Input dataset and target column. Initial determination of whether the problem is classification or regression based on the data type of the target variable. Assessment of dataset characteristics, including sample size, number of features, linearity, presence of outliers, multicollinearity, and categorical features.

Model Pool Definition:
Creation of two model pools: one for classification and one for regression. Inclusion of model descriptions to specify their strengths and typical applications.
Classification pool:
{Logistic Regression, KNeighbors Classifier(KNN), Decision Tree Classifier, Random Forest Classifier, Support Vector Classifier(SVC), Gaussian Naïve Bayes, Gradient Boosting Classifier}
Regression Pool:
{Linear Regression, Ridge, Lasso, ElasticNet, Support Vector Regressor(SVR), Decision Tree Regressor, Random Forest Regressor, Gradient Boosting Regressor}

Scoring Mechanism:

The scoring mechanism evaluates and ranks machine learning models based on the dataset's characteristics to recommend the most suitable ones. Here's how it works:

1. Dataset Characteristics:

Number of Samples (n_samples):

If n_samples > 10,000, complex models like Random Forest Classifier, Random Forest Regressor, Gradient Boosting Classifier, Gradient Boosting Regressor receive **+2 points**. Simpler models like Logistic Regression, Linear Regression, Ridge, Lasso, Elastic Net, Support Vector Classifier, and Support Vector Regressor receive **+1 point**.

Number of Features (n_features):

If n_features > 100, models such as Random Forest Classifier, Random Forest Regressor, Gradient Boosting Classifier, and Gradient Boosting Regressor score **+2 points** for handling high-dimensional data.

2. Task-Specific Scoring:

For Regression Models:

Linear Relationships:

Models like Linear Regression, Ridge, Lasso, and Elastic Net score **+3 points** for datasets with clear linear relationships.

Non-linear models like Random Forest Regressor, Gradient Boosting Regressor, Support Vector Regressor score **+2 points** for non-linear relationships.

Multicollinearity:

Regularized models like Ridge, Lasso, and Elastic Net receive **+2 points** for handling multicollinearity effectively.

Outliers:

Robust models such as Random Forest Regressor, Gradient Boosting Regressor, and Support Vector Regressor score **+2 points**.

Linear models like Linear Regression, Ridge, Lasso, and Elastic Net lose **-1 point** when outliers are present.

For Classification Models:

Linearity:

Models like Logistic Regression score **+3 points** for datasets with linearly separable classes.

Complex models like Random Forest Classifier, Gradient Boosting Classifier, Support Vector Classifier score **+2 points** for non-linear relationships.

Multicollinearity:

Models like Logistic Regression lose **-1 point** for multicollinearity.

Non-linear models like Random Forest Classifier and Gradient Boosting Classifier gain **+1 point**.

Outliers:

Robust models such as Random Forest Classifier, Gradient Boosting Classifier, and Support Vector Classifier score **+2 points**.

Linear models like Logistic Regression lose **-1 point** when outliers are present.

3. Additional Characteristics:

Categorical Features:

Models like Decision Tree Classifier, Decision Tree Regressor, Random Forest Classifier, and Random Forest Regressor gain **+2 points** for handling categorical data natively.

High Dimensionality:
Models like Random Forest Classifier, Random Forest Regressor, Gradient Boosting Classifier, Gradient Boosting Regressor receive **+2 points** for datasets with many features.

Model Ranking and Recommendation:
Sorting models based on total scores assigned during the scoring process. Selecting the top-performing models for the final recommendation.

Output and Results:
Return the top 4 recommended models along with descriptions and their suitability based on the dataset's properties.

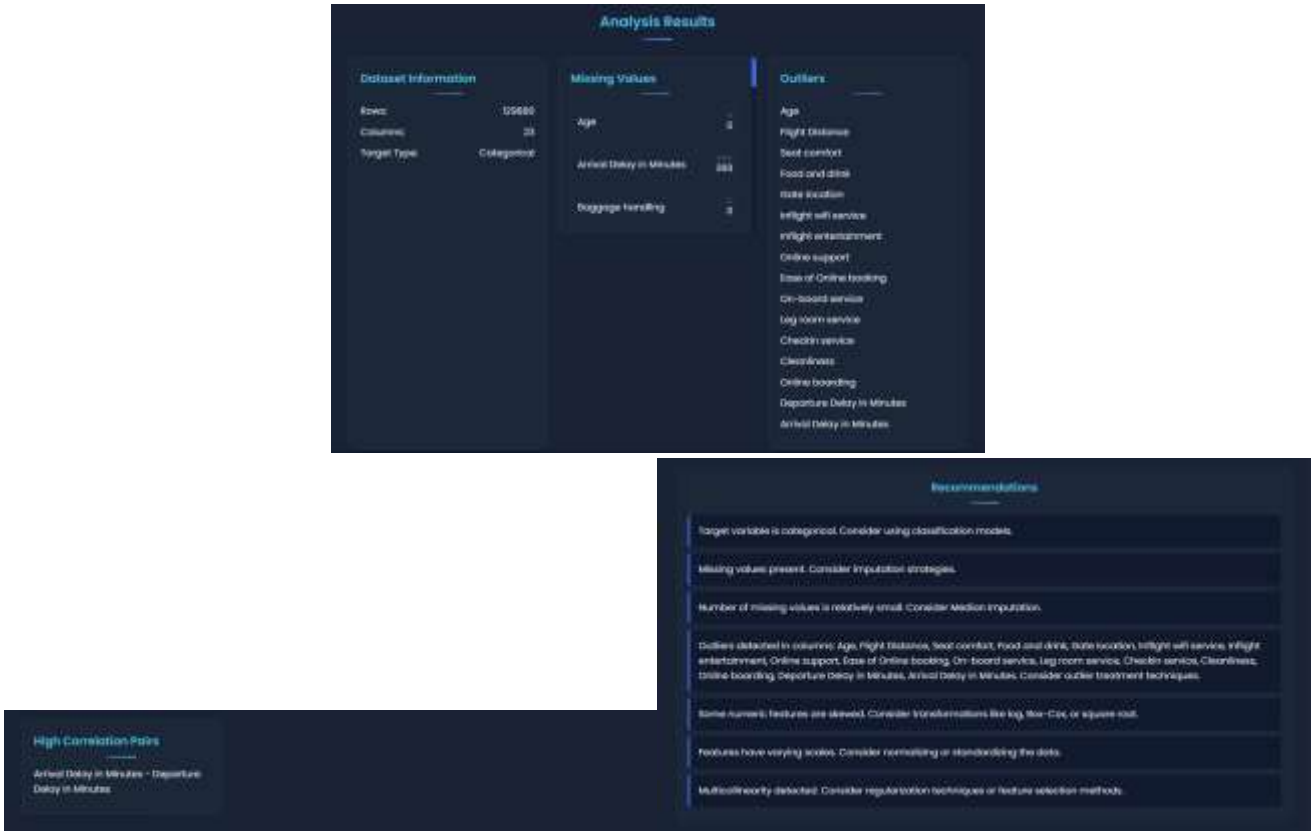
Visualization Generation:
Visualization generation begins by filtering the dataset to exclude unnecessary or irrelevant columns based on predefined criteria. This step ensures that only meaningful features are considered for visualization, enhancing the clarity and relevance of the insights derived. The visualizations primarily focus on both numerical and categorical columns, with tailored approaches for each data type.
For numerical columns, the process includes generating Correlation Heatmaps and Distribution Plots. The correlation heatmap highlights the relationships between numerical features by calculating and visualizing the correlation matrix. This provides insights into how strongly features are related to each other, identifying patterns such as multicollinearity or feature dependencies. Additionally, Distribution Plots are created for up to five numerical columns, accompanied by marginal box plots. These plots offer a detailed view of the data's spread, central tendency, and outliers, helping to identify skewness, variability, or anomalies within individual features.
For categorical columns, Bar Plots are used to represent the frequency distribution of unique categories. These plots are generated for up to five categorical features, showing the count of occurrences for each category. This visualization provides a clear understanding of the prevalence of different categories in the dataset, which is essential for feature analysis and preprocessing.
All visualizations are dynamically generated and structured in a serialized format, such as JSON, to facilitate their integration into dashboards or other reporting tools. This ensures that the visual outputs are both interactive and easily shareable, making them valuable for exploratory data analysis and decision-making processes.

Report Generation:
Generating an HTML report involves creating a dynamic report template that summarizes key aspects of the dataset and its analysis, while incorporating visualizations and preprocessing details. The report is structured into several sections, including dataset information, missing values, outlier columns, high correlation pairs, and analysis recommendations. Each of these sections is populated with relevant data, such as the number of rows and columns, missing value counts, and identified correlations, which are derived from the `analysis_data` object.
Additionally, the methodology incorporates visualizations, such as distribution plots and heatmaps, using the Plotly library to embed interactive charts within the HTML. These visualizations are rendered using data from the `analysis_data.get('visualizations', [])`. If preprocessing data is provided, the report also

includes a section for preprocessing steps and recommended machine learning models, each accompanied by a progress bar indicating the model's suitability based on the computed score. The HTML template is populated dynamically using Jinja templating and returns a complete, styled, and interactive report that can be viewed in a browser.

IV. RESULTS AND DISCUSSION

A user can upload a raw dataset and the format should be '.csv'.



Analysis Results:

The system first displays the dataset's basic information, including the number of rows and columns. It assumes the last column as the target variable and identifies its data type.

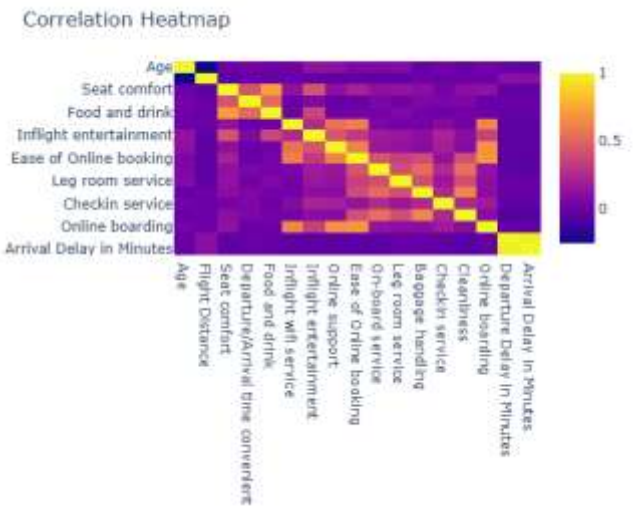
Next, it provides details on missing values, showing the count of missing entries for each column in the dataset.

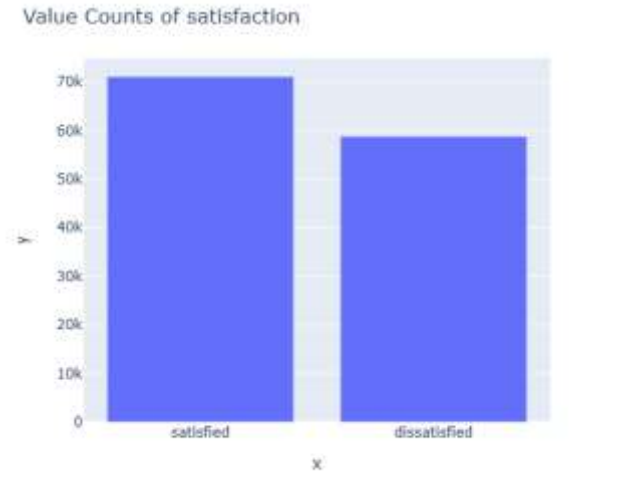
Columns containing outliers are then highlighted, followed by identifying highly correlated feature pairs with a correlation coefficient greater than 0.8.

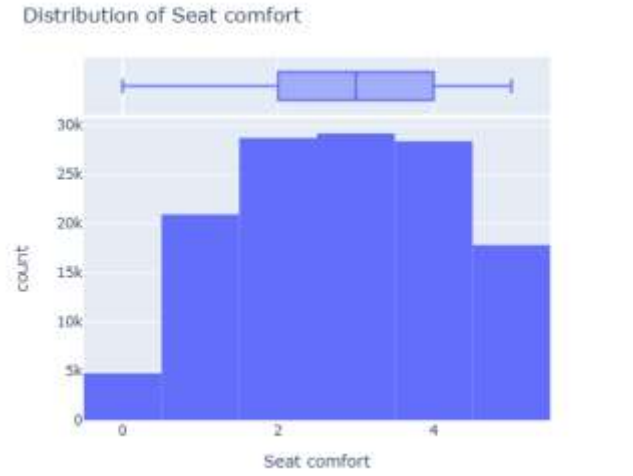
Finally, the system offers general recommendations, such as:

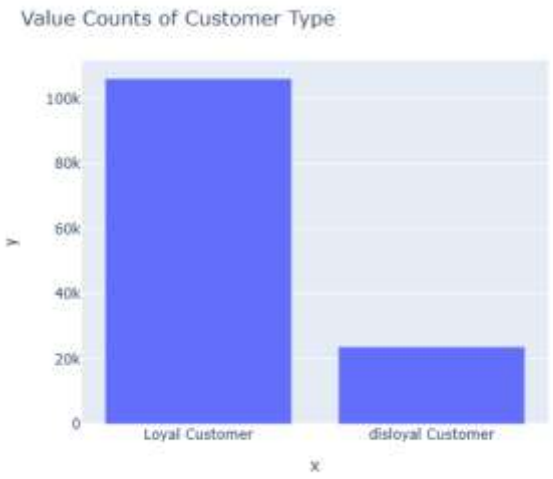
"The target variable is categorical. Consider using classification models.", "Missing values are present. Consider applying imputation strategies.", etc.

Visualizations:









The following visualizations are generated by the function:
Correlation Heatmap: Displays the correlation matrix of numerical columns, highlighting relationships between features.
Distribution Plots: Shows histograms with box plots for the distribution of up to five numerical columns.
Bar Plots for Categorical Columns: Illustrates the value counts for up to five categorical columns, providing insights into the distribution of categorical features.

The user can select the target variable for having model recommendation.

The recommended preprocessing steps can include:

Impute Missing Values:
Use **median imputation** for columns with missing values, particularly for numerical variables with a skewed distribution, to maintain data integrity.
Use **mean imputation** for columns with missing values in normally distributed numerical variables to preserve overall data consistency.
Use **mode imputation** for categorical columns or numerical variables with repeated values to retain the most frequent category or value.
Handle Outliers: Winsorize outliers in numerical columns to minimize their influence on the model and improve stability.
Feature Scaling: Scale numerical columns to standardize feature magnitudes, ensuring compatibility with algorithms sensitive to scale.

Encoding Categorical Variables: Apply one-hot encoding or label encoding to transform categorical columns into numerical representations suited for model requirements.

Model Recommendation:

The system recommends models based on a comprehensive scoring mechanism. This scoring is determined by evaluating dataset characteristics, task-specific requirements, and additional features like complexity and scalability. Each model is ranked according to its suitability for the given dataset, ensuring optimal performance for the intended task. Recommendations include both ensemble and non-linear models, highlighting their advantages and specific use cases for the dataset in question.
Then the user can generate and download the report.

V. FUTURE SCOPE

The current implementation of this automated machine learning analysis and recommendation system lays the groundwork for several promising future enhancements. The system can be expanded to incorporate advanced data preprocessing techniques that return fully optimized datasets, coupled with an intelligent model suggestion system that not only recommends but also ranks models based on their predicted accuracy for the specific use case. Implementation of automated cross-validation and hyperparameter tuning mechanisms would significantly enhance model performance and reliability. The integration of artificial intelligence, particularly reinforcement learning algorithms, could revolutionize the preprocessing and model selection process by learning from historical data and adapting recommendations accordingly. To address modern computing needs, the system could be enhanced to support real-time data processing capabilities, enabling dynamic analysis and model updates for streaming data applications. Additionally, implementing a comprehensive user feedback system would create a continuous improvement loop, where the system learns from user interactions and outcomes to refine its recommendations and improve overall efficiency. These enhancements would transform the current system into a more robust, adaptive, and intelligent platform for automated machine learning workflows.

VI. CONCLUSION

The developed automated machine learning analysis and recommendation system represents a significant advancement in simplifying complex data science workflows through intelligent automation. By providing comprehensive data preprocessing and model recommendations based on dataset characteristics, the system transforms the traditionally complex machine learning pipeline into an intuitive, user-friendly

experience. The interactive web application enables users to effortlessly upload datasets, receive detailed insights, and obtain tailored recommendations for data handling and model selection. Through AI-driven automation, the system effectively streamlines intricate tasks that typically require extensive domain expertise, thereby improving both efficiency and analytical accuracy. The framework's modular design and intelligent algorithms not only address current challenges in machine learning workflows but also lay a strong foundation for future enhancements that will make the system increasingly robust and adaptable across diverse dataset types. Ultimately, this research contributes to democratizing machine learning by reducing barriers to entry and providing accessible, intelligent tools for data scientists and researchers at all levels of expertise.

VII. REFERENCES

1. Kotsiantis, Sotiris & Kanellopoulos, Dimitris & Pintelas, P.. (2006). Data Preprocessing for Supervised Learning. International Journal of Computer Science. 1. 111-117.
2. Luís Santos, Luís Ferreira,Atlantic—Automated data preprocessing framework for supervised machine learning, Software Impacts, Volume 17, 2023, 100532, ISSN 2665-9638.
3. P. Bhardwaj, C. Choudhury and P. Batra, "Automating Data Analysis with Python: A Comparative Study of Popular Libraries and their Application," *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, Tashkent, Uzbekistan, 2023, pp. 1243-1248, doi: 10.1109/ICTACS59847.2023.10390032.
4. H. -W. Choi, S. -Y. Shin and H. -J. Kim, "Machine Learning-based Automated Data Visualization: A Meta-feature Engineering Approach," *2019 8th International Conference on Innovation, Communication and Engineering (ICICE)*, Zhengzhou, China, 2019, pp. 107-109, doi: 10.1109/ICICE49024.2019.9117537.
5. R M, Achshah & Prakash, Dr. (2021). A Simple Approach for Selecting the Best Machine Learning Algorithm. International Journal of Scientific and Engineering Research. 12. 902-909.
6. Nagarajah, Thiloshon & Poravi, Guhanathan. (2019). A Review on Automated Machine Learning (AutoML) Systems. 1-6. 10.1109/I2CT45611.2019.9033810.
7. Jabour, Joseph & Henslee, Althea & Dozier, Haley & Shukla, Indu & Hansen, Brandon & Salhi, Abderahim & Dettwiller, Ian. (2023). Automated Machine Learning Model Selection Analysis. 2702-2704. 10.1109/CSCE60160.2023.00435.
8. M. Bilal, G. Ali, M. W. Iqbal, M. Anwar, M. S. A. Malik and R. A. Kadir, "Auto-Prep: Efficient and Automated Data Preprocessing Pipeline," in *IEEE Access*, vol. 10, pp. 107764-107784, 2022, doi: 10.1109/ACCESS.2022.3198662.
9. B. Kitchenham, "A procedure for analyzing unbalanced datasets," in *IEEE Transactions on Software Engineering*, vol. 24, no. 4, pp. 278-301, April 1998, doi: 10.1109/32.677185.