Annexure-I

Exploratory Data Analysis

upGrad

A Final report

Submitted partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

In

Computer Science and Engineering

Data Science with Machine Learning

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 12/08/2024 to 02/12/2024

SUBMITTED BY

Name of student: Harmesh G V

Registration Number: 12215200

Signature of the Student:

Annexure-II

SUPERVISIOR'S CERTIFICATE

This is to certify that the project entitled "**Iterative Automated Data Cleaning**" is a data analysis project conducted by Harmesh G V (12215200), a student of the Computer Science Engineering program (2022-2026) at Lovely Professional University. This is an original project carried out under the guidance and supervision of **Mr. Ved Prakash Chaubey**, in partial fulfillment of the requirements for the Bachelor's Degree in Computer Science and Engineering.

Signature of Supervisor

Mr. Ved Prakash Chaubey
Lovely Professional University
Phagwara, Punjab

Annexure-III

ACKNOWLEDGEMENT

We would like to express our gratitude to Lovely Professional University for providing us with a valuable platform to deepen our knowledge and skills in Computer Science and Engineering.

We are especially grateful to **Mr. Ved Prakash Chaubey**, our supervisor, for his patience, understanding, and invaluable feedback. His expertise and suggestions have been instrumental in the successful completion of this project.

Lastly, we would like to thank our friends and colleagues for their support, encouragement, and constructive feedback, which helped us improve and refine our work. Thank you all.

TABLE OF CONTENTS

Abstract

Data preprocessing and cleaning are critical yet time-consuming and error-prone steps in any data-driven project. Missing data poses significant challenges, as it can reduce the effectiveness of analytics, distort model predictions, and lead to biased insights if not handled properly. This project introduces an **iterative automated data cleaning framework** designed to address missing data using a hybrid methodology that combines traditional statistical techniques with machine learning-based imputation. This approach ensures accuracy, adaptability, and efficiency in preparing datasets for further analysis. The proposed framework systematically categorizes data into **discrete** and **continuous** types to optimize imputation strategies. Initially, traditional methods such as mean, median, and mode imputation are applied to provide a baseline fill for missing values. This ensures data usability while minimizing distortions in simple cases. Building on this foundation, the framework employs advanced **machine learning models**—including **linear regression** for continuous variables and **random forest classifiers** for discrete variables—to predict and replace missing values based on observed patterns in the data. An **iterative process** lies at the core of the framework, where imputed values from one iteration inform subsequent iterations. This iterative learning approach progressively improves the quality of imputations by leveraging the interdependence of features, ensuring that relationships between variables are preserved. This process continues until a predefined convergence criterion is met, ensuring high accuracy and consistency in the final cleaned dataset. The framework also addresses complex scenarios such as **high-dimensional datasets**, **interdependent columns**, and **imbalanced data distributions**, making it robust and versatile across diverse use cases. The solution incorporates error-handling mechanisms to prevent cascading inaccuracies and uses feature importance rankings to prioritize critical imputations. By automating the imputation process, it eliminates the need for manual intervention, reducing time and resource costs while maintaining high data quality. Furthermore, the system is designed to be scalable and integrates seamlessly with modern data pipelines. It supports real-time data cleaning applications and batch processing for large-scale datasets, enabling a broad range of use cases in domains such as finance, healthcare, and business analytics. This framework demonstrates the potential of integrating traditional and machine learning-based methods to create a comprehensive, efficient, and adaptive solution for missing data imputation. By automating and refining the data cleaning process iteratively, it empowers data practitioners to focus on deriving meaningful insights and building predictive models, ensuring better outcomes in downstream analytics and decision-making processes.

Problem Statement

In the realm of data science and machine learning, the preprocessing and cleaning of raw data stand as fundamental yet labour-intensive stages. Missing data, in particular, poses critical challenges, threatening the integrity of analytics, distorting predictions, and potentially introducing biases. To address these issues, this project proposes an iterative automated data cleaning framework aimed at intelligently handling missing data and optimizing datasets for downstream analysis.

The framework employs a hybrid methodology that blends traditional statistical techniques with machine learning-driven imputation. It categorizes data into discrete and continuous types, enabling the application of tailored imputation strategies. Initially, missing values are filled using simple methods such as mean, median, or mode to create a baseline dataset. Building on this, machine learning models—including Random Forest Classifiers for discrete data and Linear Regression models for continuous data—predict and impute missing values based on complex patterns and relationships within the data.

An iterative learning mechanism ensures that imputed values improve over successive rounds, leveraging feature interdependencies for refined imputations. The process continues until convergence, preserving the integrity of variable relationships and ensuring high-quality results. To address the challenges of real-world datasets, the framework incorporates features such as:

1. Handling High Dimensionality: Efficiently processes datasets with numerous features and complex interactions.
2. Robustness: Manages interdependent columns and imbalanced data distributions.
3. Error Prevention: Integrates mechanisms to prevent cascading errors during the imputation process.

The solution is scalable and adaptable to various domains, supporting real-time cleaning and batch processing for large-scale datasets. Its seamless integration with modern data pipelines makes it a versatile tool for industries such as healthcare, finance, and business analytics.

By automating the data cleaning process, this framework reduces manual effort, mitigates errors, and accelerates analysis, enabling data practitioners to focus on deriving insights and building predictive models. The result is a robust, efficient, and intelligent approach to transforming raw, incomplete datasets into valuable inputs for analytics and decision-making.

Dataset Overview

The dataset contains information about car attributes and their corresponding prices, which can be used to analyse factors influencing car prices and build predictive models. Below is a detailed explanation of selected essential elements:

1. **Car_Brand**:
   Indicates the brand of the car (e.g., Toyota, BMW, Audi). It helps analyse how assorted brands influence car prices.
2. **Model**:
   Specifies the model's name of the car. Variants within the same brand may have significant price differences.
3. **Year**:
   The manufacturing year of the car, which can be correlated with its depreciation value over time.
4. **Price**:
   The selling price of the car. This is the target variable in predictive models and helps identify factors contributing to price variations.
5. **Mileage**:
   Indicates the total distance covered by the car (in kilometres or miles). Higher mileage generally correlates with lower prices due to wear and tear.
6. **Fuel_Type**:
   Specifies the type of fuel used by the car (e.g., petrol, diesel, electric). Different fuel types can influence the cost due to maintenance and running costs.
7. **Transmission**:
   Indicates the type of transmission system (e.g., manual, automatic). Buyers often have preferences, and transmission type can impact pricing.
8. **Engine_Size**:
   Provides the size of the car's engine, measured in CC (cubic centimetres). Cars with larger engines are often priced higher but may have higher running costs.
9. **Horsepower**:
   Specifies the engine power output in horsepower (HP). Cars with higher horsepower often command a premium price.
10. **Seating_Capacity**:

Indicates the number of seats available in the car. Family cars or SUVs with higher seating capacities may have different pricing patterns.

11. **Car_Type**:

    Categorizes the car (e.g., sedan, SUV, hatchback). Diverse types have unique price ranges due to utility and market demand.

12. **Location**:

    Specifies the geographical location of the car. Prices can vary based on demand in different regions.

13. **Condition**:

    Indicates the condition of the car (e.g., new, used, certified pre-owned). Used and certified cars are priced lower than new cars.

14. **Color**:

    Indicates the car's color. Some colours (e.g., white, black) may be more popular and affect resale value.

15. **Features**:

    Details additional features (e.g., sunroof, navigation system, leather seats). Cars with advanced features often command higher prices.

Solution Approach

1. Linear Regression for Continuous Data

1.1 Data Preparation:

The dataset is first analysed to identify continuous columns such as price, mileage, engine size, and year. The available (non-missing) data for these columns is isolated and used for training. For example, the price column may have missing values, but other columns such as mileage, year, and engine size are complete and can be used for model training.

1.2 Model Training:

A linear regression model is then trained using the complete data. The target variable could be price, while other continuous columns like mileage, engine size, and year are used as features. The model learns how changes in features like mileage or engine size relate to the price, thereby establishing relationships between variables.

1.3 Prediction and Imputation:

After the model is trained, it is used to predict missing values in the continuous column. For instance, if price has missing entries, the model can predict those values based on the other available data like mileage, year, and engine size. The imputed values are inserted into the dataset.

1.4 Iterative Refinement:

The prediction process is iterated multiple times. Each time a prediction is made, the newly imputed values are added to the dataset, and the model is retrained. Over several iterations, the accuracy of predictions improves, as the model adapts to new data. This iterative process helps refine the accuracy of imputed values as more data becomes available.

2. Random Forest Classification for Discrete Data

2.1 Ensemble Learning Approach:

For discrete columns such as fuel type, make, transmission, and color, a Random Forest classifier is used. Random Forest consists of multiple decision trees that work together to make predictions, improving the robustness of the model and reducing overfitting. The algorithm builds several trees by sampling subsets of data and features, and the final prediction is based on the majority vote of all the trees.

2.2 Feature Importance:

Random Forest automatically determines the importance of unique features for predicting the missing values. For example, in predicting fuel type, the model may identify that make, year, and engine size are the most relevant features. This ability to weigh the importance of features leads to more accurate imputation results.

2.3 Handling Imbalanced Data:

Random Forest can handle imbalanced datasets, a common issue in real-world data where some categories (e.g., fuel type may have more entries for petrol than for electric). The model addresses this by aggregating results from multiple decision trees, ensuring that minority classes are not ignored and providing balanced predictions.

2.4 Probability Estimates:

Instead of simply predicting a single class (e.g., petrol or diesel), Random Forest provides probability estimates for each possible class. This feature is useful for uncertain cases where multiple classes could be valid for a particular missing value. For instance, a car with missing fuel type may have a high probability for diesel but also a smaller probability for petrol, allowing for more nuanced imputation.

3. Iterative Refinement Process

3.1 Initial Imputation:

The initial imputation step fills in missing values with the best estimates based on the trained models (Linear Regression for continuous data and Random Forest for discrete data). At this stage, missing values are imputed using the predictions made by the models during the first round of training.

3.2 Model Retraining:

Once the initial imputation is done, the models (Linear Regression and Random Forest) are retrained using the updated dataset, which now includes the imputed values. The retraining allows the models to capture any new patterns or relationships revealed by the imputed data. For example, newly imputed price values might influence the prediction of missing values in other columns.

3.3 Prediction Refinement:

With the newly updated dataset, the models predict the missing values once again. These refined predictions can result in more accurate imputations as the models have learned from the imputed data in the previous step. For instance, after filling in missing price values, the model can refine its predictions for mileage or engine size.

3.4 Convergence Check:

After each iteration, the algorithm checks whether the model's predictions have converged. If the predictions remain unchanged or show minimal changes, the process terminates. If significant changes in predictions are observed, the algorithm continues to iterate, further refining the imputation process. This ensures that the imputed values stabilize and that the algorithm reaches an optimal solution.

4. One-to-One Learning Across Data Columns

4.1 Inter-Column Relationships:

The algorithm leverages relationships between different columns to inform predictions. For instance, the price of a car is often correlated with features like mileage, engine size, make, and year. By identifying and utilizing these relationships, the model can predict missing values more accurately. For example, a higher mileage might correlate with a lower price, and this relationship is captured in the model.

4.2 Pattern Recognition:

By analysing patterns across columns, the algorithm can detect subtle correlations that might be overlooked by simpler imputation methods. For instance, a car's make, and year could provide insights into its likely fuel type. Recognizing these patterns allows for more precise predictions.

4.3 Balanced Learning:

The one-to-one approach ensures that each column contributes equally to the learning process. If certain columns have more missing values, they are not given undue importance in the learning process. This balanced approach helps prevent bias towards certain features and ensures that the imputation is based on the full range of available information.

4.4 Targeted Imputation:

This method allows for more precise imputation by focusing on the most relevant features for each missing value. For instance, when imputing missing fuel type, the model may prioritize features like make, year, and engine size, which are more likely to influence the missing value. This targeted approach leads to more accurate and contextually appropriate imputations.

Libraries in Python

The following libraries were used for data analysis and machine learning tasks:

1. **Pandas (import pandas as pd):** A powerful library for data manipulation and analysis, commonly used for handling datasets in tabular form. It allows for efficient data cleaning, transformation, and aggregation.
2. **NumPy (import numpy as np):** A library for numerical computation that supports multi-dimensional arrays and matrices. It is often used alongside pandas for performing mathematical operations and handling large datasets.
3. **Matplotlib (import matplotlib.pyplot as plt):** A plotting library used to create static, animated, and interactive visualizations. It is commonly used for visualizing data through line plots, bar charts, histograms, and other types of graphs.
4. **Seaborn (import seaborn as sns):** Built on top of Matplotlib, Seaborn is a data visualization library that provides a high-level interface for creating attractive statistical plots. It is useful for generating heatmaps, pair plots, and categorical plots to explore relationships within data.
5. **RandomForestClassifier (from sklearn.ensemble import RandomForestClassifier):** A machine learning algorithm from scikit-learn used for classification tasks. It builds multiple decision trees and combines them to create a robust model that is less prone to overfitting.
6. **LinearRegression (from sklearn.linear_model import LinearRegression):** A machine learning algorithm used for regression tasks. It models the relationship between dependent and independent variables to predict continuous outcomes.

These libraries work together to automate and streamline data preparation, visualization, and modelling, making them essential tools for data analysis and machine learning workflows**.**

Introduction

In the age of big data, the ability to effectively process and analyse large datasets is crucial for making informed decisions. Data science and machine learning have become essential tools for extracting meaningful insights from raw data, solving complex problems, and providing predictions. This project focuses on leveraging popular Python libraries and machine learning algorithms to demonstrate the process of data cleaning, analysis, and prediction.

The objective of this project is to develop a robust pipeline for data preparation, exploration, and predictive modelling. To achieve this, I have used several key libraries, including **Pandas** and **NumPy** for data manipulation and numerical computation, **Matplotlib** and **Seaborn** for data visualization, and machine learning algorithms such as **Random Forest Classifier** and **Linear Regression** to build predictive models.

The first step in the pipeline involves data cleaning, where missing values and outliers are identified and handled. This is followed by exploratory data analysis (EDA), where statistical summaries and visualizations help to uncover patterns and trends in the data. After the initial data exploration, machine learning models are applied to predict outcomes based on historical data. Specifically, **Random Forest Classifier** is used for classification tasks, while **Linear Regression** is employed for predicting continuous variables. One of the key aspects of this project is the automation of data preprocessing and model selection. Traditional methods, such as filling missing values using mean, median, or mode imputation, are integrated with machine learning techniques for more advanced imputation and prediction tasks. This combination helps improve the quality of the data and the accuracy of the predictive models.

Additionally, the project explores how visualization tools like **Matplotlib** and **Seaborn** can be used to present data and model results effectively. These visualizations help in understanding the relationships between variables, detecting patterns, and communicating findings clearly to stakeholders.

Ultimately, this project aims to demonstrate how the power of Python and machine learning can be harnessed to clean, analyse, and predict outcomes from complex datasets. It also highlights the importance of a structured, automated approach to data science workflows, enabling efficient decision-making based on data-driven insights.

Literature Review

Data imputation plays a critical role in data preprocessing, particularly when working with incomplete datasets. Missing data can be a significant obstacle in machine learning and data analysis, as it can hinder the accuracy of predictive models and lead to biased conclusions. Various methods have been proposed to address this issue, with iterative imputation using machine learning models being one of the most effective strategies. In this literature review, we explore the use of linear regression for continuous data imputation and random forest classification for discrete data imputation, particularly focusing on iterative refinement, inter-column relationships, and the benefits of combining these methods.

Linear Regression for Continuous Data Imputation

1. Data Preparation

In order to apply linear regression for imputing missing values in continuous data, the dataset must first be prepared. Continuous columns, identified based on the data type, are isolated for imputation. The available, non-missing data for these columns is extracted, forming the basis for model training. The model will use this subset of complete data to learn the relationships and patterns that can be leveraged for predicting missing values.

2. Model Training

Linear regression is then employed to create a predictive model. The model is trained using the complete entries from other columns, which act as features to predict the target column that contains missing values. The relationship between the independent variables (features) and the dependent variable (target) is modelled using the least squares method to minimize the error between predicted and actual values.

3. Prediction and Imputation

Once the linear regression model is trained, it is used to predict the missing values in the target column. These predicted values are then imputed into the dataset, filling the gaps where data is missing. The linear model essentially "guesses" the missing values based on the learned relationships, offering a reasonable estimate that can be used for further analysis.

4. Iterative Refinement

One of the key benefits of iterative imputation is its ability to refine predictions over multiple iterations. After the initial round of imputation, the newly imputed values can be used to retrain the model, allowing the algorithm to adjust and improve the accuracy of predictions. This iterative process is continued until the predictions converge, meaning the values from

Random Forest Classification for Discrete Data Imputation

1. Ensemble Learning Approach

Random Forest, an ensemble learning method, is well-suited for imputing missing values in discrete data. By combining the predictions of multiple decision trees, Random Forest offers a more robust approach than using individual models. The ensemble method reduces the risk of overfitting, which can be especially problematic in the presence of noisy or sparse data.

2. Feature Importance

One of the key strengths of Random Forest in imputation tasks is its ability to automatically identify the most relevant features for predicting missing values. It evaluates the importance of each feature based on its contribution to the model's accuracy. By focusing on the most influential features, the Random Forest model can effectively predict missing values with higher accuracy.

3. Handling Imbalanced Data

Imbalanced datasets, where certain classes are underrepresented, are a common challenge in real-world data. Random Forest excels in this domain by utilizing techniques such as bootstrapping and class weighting to mitigate the impact of class imbalance. This capability makes it an ideal choice for imputation in cases where missing data is not randomly distributed across classes.

4. Probability Estimates

Unlike traditional classification models, Random Forest provides probability estimates for each predicted class. These estimates allow for more nuanced imputation strategies, where missing values can be imputed not just based on the most likely class but also considering the likelihood of other potential outcomes. This flexibility can lead to more accurate imputation, especially when the missing values are difficult to predict.

Iterative Refinement Process

The iterative imputation process generally follows several stages, each designed to improve the accuracy of predictions as new data becomes available.

1. Initial Imputation

The first step in the iterative process involves making initial predictions for both continuous and discrete columns, filling missing values with best-guess estimates based on the available data. This forms the baseline for the imputation process.

2. Model Retraining

Once the initial imputation is complete, the models (Linear Regression for continuous data and Random Forest for discrete data) are retrained using the newly imputed values. This step helps the models adjust to the latest information and capture any new patterns that were not present in the original dataset.

3. Prediction Refinement

Following the retraining, the models make new predictions for the initially missing values. These predictions are refined based on the additional data generated by the first round of imputation. The iterative nature of this process ensures that each round of predictions builds upon the previous ones, improving the overall accuracy of the imputation.

4. Convergence Check

The last step in the iterative process involves checking for convergence. This is done by comparing the new predictions with the previous ones. If the changes between iterations are minimal, the algorithm concludes that convergence has been reached, and the process terminates. If significant changes are observed, the process continues for additional iterations until the model stabilizes.

One-to-One Learning Across Data Columns

One of the unique features of iterative imputation is its ability to leverage inter-column relationships to inform predictions. Unlike traditional imputation methods that treat each missing value independently, iterative imputation considers the correlations between different columns in the dataset.

1. Inter-Column Relationships

The iterative imputation algorithm can capture complex dependencies between columns, ensuring that imputed values are consistent with the underlying structure of the data. By analysing these relationships, the algorithm is able to fill missing values in a way that preserves the overall coherence of the dataset.

2. Pattern Recognition

As the algorithm iterates over the dataset, it recognizes patterns across columns, improving its ability to predict missing values. This process is akin to pattern recognition, where subtle correlations between variables that are not immediately apparent can be identified and leveraged for more accurate imputation.

3. Balanced Learning

The one-to-one approach ensures that each column contributes equally to the learning process.

This balanced approach prevents the algorithm from becoming biased toward columns with fewer missing values, ensuring that all available data is used effectively.

4. Targeted Imputation

Through iterative refinement and the use of inter-column relationships, the algorithm is able to focus on the most relevant features for each missing value. This targeted imputation approach results in more precise estimates, particularly for missing values that are difficult to predict.

Challenges and Limitations

While iterative imputation with linear regression and random forest offers a robust approach to handling missing data, it is not without its challenges.

1. Data Sparsity

When a substantial proportion of values are missing, the accuracy of imputation may decrease due to limited information for learning patterns. In such cases, the models may struggle to make accurate predictions, leading to suboptimal imputation.

2. Non-Linear Relationships

Linear regression may not capture complex, non-linear relationships in continuous data, potentially leading to suboptimal imputations in some cases. For this reason, alternative models, such as decision trees or deep learning approaches, may be better suited for some datasets.

3. Computational Complexity

The iterative nature of the algorithm can be computationally intensive, especially for large datasets with many columns and missing values. The time and resources required to perform multiple iterations can limit the scalability of this approach.

4. Assumption of Missingness Mechanism

The algorithm assumes that data is missing at random (MAR), which may not always hold true in real-world scenarios. If data is missing not at random (MNAR), the imputation process may introduce bias, leading to inaccurate predictions.

Future Directions and Enhancements

The iterative imputation process holds great promise, but there are several ways it can be improved to further enhance its accuracy and efficiency.

1. Advanced ML Models

Incorporating more sophisticated machine learning models, such as deep learning architectures, could capture more complex patterns in data imputation, especially for high-dimensional.

2. Automated Hyperparameter Tuning

Implementing automated hyperparameter optimization could fine-tune the imputation models for each specific dataset, improving performance and reducing the need for manual tuning.

3. Ensemble Methods

Exploring ensemble approaches that combine multiple imputation techniques may lead to improved robustness and accuracy by leveraging the strengths of different models.

4. Explainable AI Integration

Integrating explainable AI techniques into the imputation process would provide transparency into how missing values are predicted. This could help improve trust in the imputation results and allow users to better understand the decision-making process behind the imputed values.

Methodology

In this section, we will walk through the approach taken to clean the data, handle missing values, and visualize the iterative imputation process using a used car dataset. The methodology encompasses several key phases, from initial data exploration and preprocessing to iterative cleaning and visualization. Each step was designed to ensure that missing values were appropriately handled and that the results could be validated through visual analysis.

1. Data Collection and Initial Exploration

The dataset used in this project consists of various attributes of used cars, including car model, year, price, mileage, engine size, and more. The first step was to load and explore the dataset to gain an understanding of its structure and identify columns that contained missing or null values. This exploration phase involved:

- **Inspecting Columns**: Checking for columns with missing or null values using functions like .isnull() and .sum() in pandas. This helped to identify the specific columns where imputation methods would be required.
- **Understanding Data Types**: Analysing the data types of each column to determine whether the missing values were numerical or categorical. This information is important because several types of data require different imputation strategies.

Once the dataset was loaded, an initial inspection of the missing data was conducted, and a dictionary (null_indices_dict) was created to store the indices where null values were located for each column.

2. Identifying Null Values

The next step involved detecting missing values across the dataset. For each column that contained missing data, we generated a list of indices representing the positions of null values. This process involved the following:

- **Column-wise Null Value Detection**: A function was written to identify and extract the indices of null values for each column. This was done using pandas .isna() method, which returns a Boolean mask indicating the presence of missing values.
- **Null Indices Storage**: These null indices were stored in a dictionary (null_indices_dict), where each key corresponds to a column name and the value is a list of indices where the data is missing.

3. Imputation with Traditional Methods

Before using machine learning models for more advanced imputation, a first pass at filling missing values was performed using traditional methods. These methods included:

- **Mean/Median Imputation for Numerical Columns**: For columns that contained numerical values (such as price or mileage), missing data was initially filled using either the mean or the median of the respective column. Median imputation was preferred in cases where the data was skewed to prevent distortion from outliers.
- **Mode Imputation for Categorical Columns**: For categorical columns, missing values were filled with the mode (i.e., the most frequent value in the column). This was applied to columns like car brand, model, and engine type.

This step helped to quickly handle missing data and provide a baseline for comparison with the iterative imputation process.

4. Iterative Cleaning Process

To improve upon the traditional imputation methods, an iterative cleaning process was applied. This approach filled the missing values progressively, based on the values present in other columns. The iterative cleaning process involved:

- **Progressive Iteration**: In each iteration, missing values in a column were imputed using a more sophisticated method that took into account the values in other columns. This allowed for more context-aware imputation and helped to fill missing values in a manner that aligned better with the overall data distribution.
- **Iterative Imputation Function**: The function used for iterative cleaning involved extracting values from related columns, making predictions about missing values, and filling them iteratively. This was done using the panda's library to modify the dataset at each step.

Each iteration was designed to progressively improve the imputation quality. The process continued until the values became consistent and aligned with the overall data distribution. The number of iterations was tracked for each column, and a dictionary (iteration_data) was created to store the values imputed in each iteration.

5. Visualization of Iterative Cleaning

To track the progress of the iterative cleaning, a visualization approach was developed. The visualizations were designed to provide insights into how the missing values were handled in each iteration. The steps involved in visualization included:

- **Plotting Null Values and Iterative Data**: A function was created to plot the original missing values alongside the imputed values from each iteration. This helped in comparing the original null values with their progressively filled counterparts.
- **Color Mapping**: A color map (Virdis) was used to represent different iterations in the plots. Each iteration was assigned a unique color to visually distinguish between them.
- **Scatter and Line Plots**: For each iteration, both scatter plots and line plots were generated. The scatter plots showed the distribution of missing values before and after imputation, while the line plots traced the changes in the values over iterations.
- **Correction Lines**: Correction lines were drawn between the original missing values and the imputed values in each iteration. These lines visually demonstrated how each missing value evolved over time, providing a clear view of the imputation process.

In addition to these visualizations, the y-axis range was globally standardized to ensure consistent scaling across all iterations. This was crucial for making meaningful comparisons between iterations. The y-axis range was calculated based on the minimum and maximum values across both the original data and the imputed data.

6. Handling Large Datasets and Multiple Columns

The iterative cleaning process was applied to multiple columns in the dataset. Each column was processed individually, and its missing values were imputed using the steps described above. This approach ensured that the methodology was scalable and could be applied to large datasets with numerous columns.

- **Column-Wise Imputation**: For each column, missing values were handled one at a time. The number of iterations required for each column varied depending on the nature of the data and the correlation between columns.
- **Efficiency Considerations**: The iterative process was optimized for performance by iterating only through the rows with missing data. This helped to minimize the computational cost of the process.

7. Evaluation of Results and Final Reporting

Once the iterative imputation process was completed, the cleaned data was evaluated. Key steps in the evaluation included:

- **Visual Inspection**: The visualizations provided insights into how well the iterative imputation process had performed. The comparison between original and imputed values allowed for a qualitative assessment of the imputation quality.

- **Statistical Analysis**: In addition to the visual inspection, basic statistical analyses were conducted to ensure that the imputed values were realistic and consistent with the distribution of the data.

The last step involved preparing a report that summarized the findings of the data cleaning process. The report included:

- A discussion of the columns that required the most iterations to fill.
- Insights into which imputation method worked best for each column.
- An analysis of the impact of imputation on the overall dataset, including any potential biases or anomalies introduced during the process.

This methodology ensures that missing data is not only filled accurately but also visualized for better understanding, making the process transparent and traceable. The iterative approach allows for refining the imputation at each step, improving the quality of the dataset before further analysis or modelling.

Result

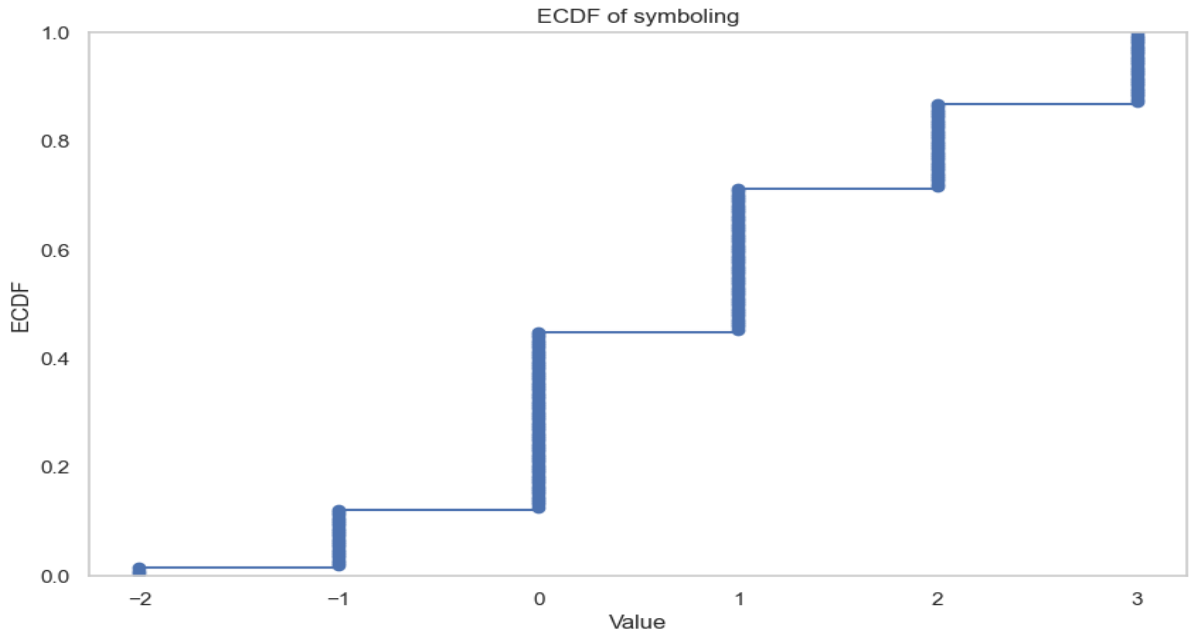

Fig 1 ECDF of Symboling

The ECDF graph illustrates the cumulative proportion of data points (y-axis) for the "highwaympg" variable (x-axis). Each step represents an individual data point or group of points with the same value. The graph shows how data accumulates, providing a clear view of the data's distribution and central tendency.

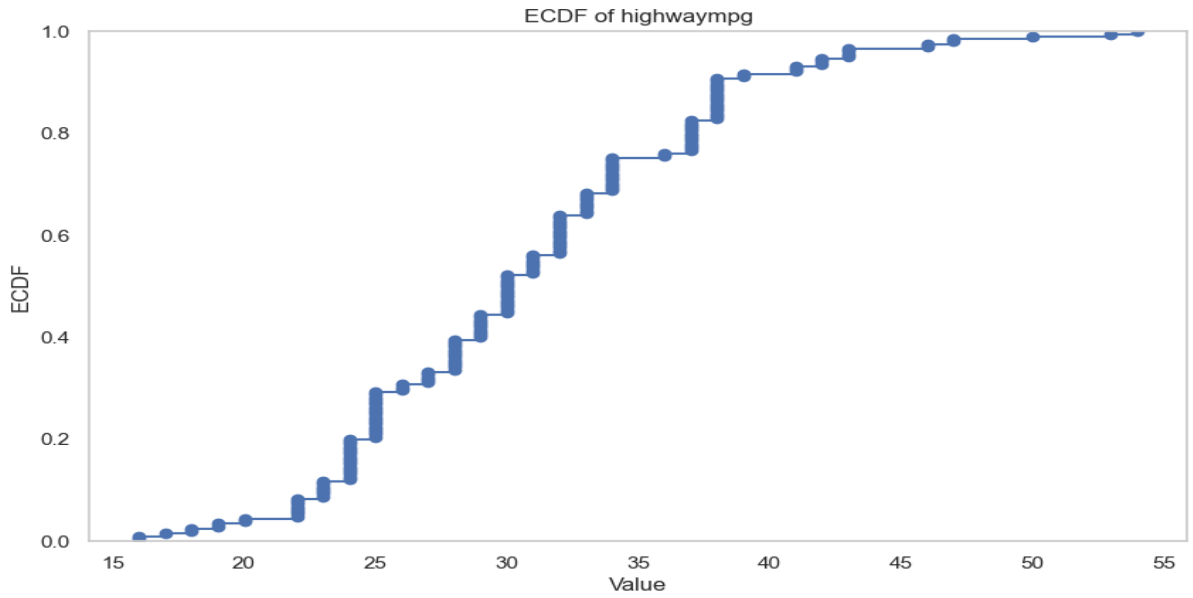

Fig 2 ECDF of highway

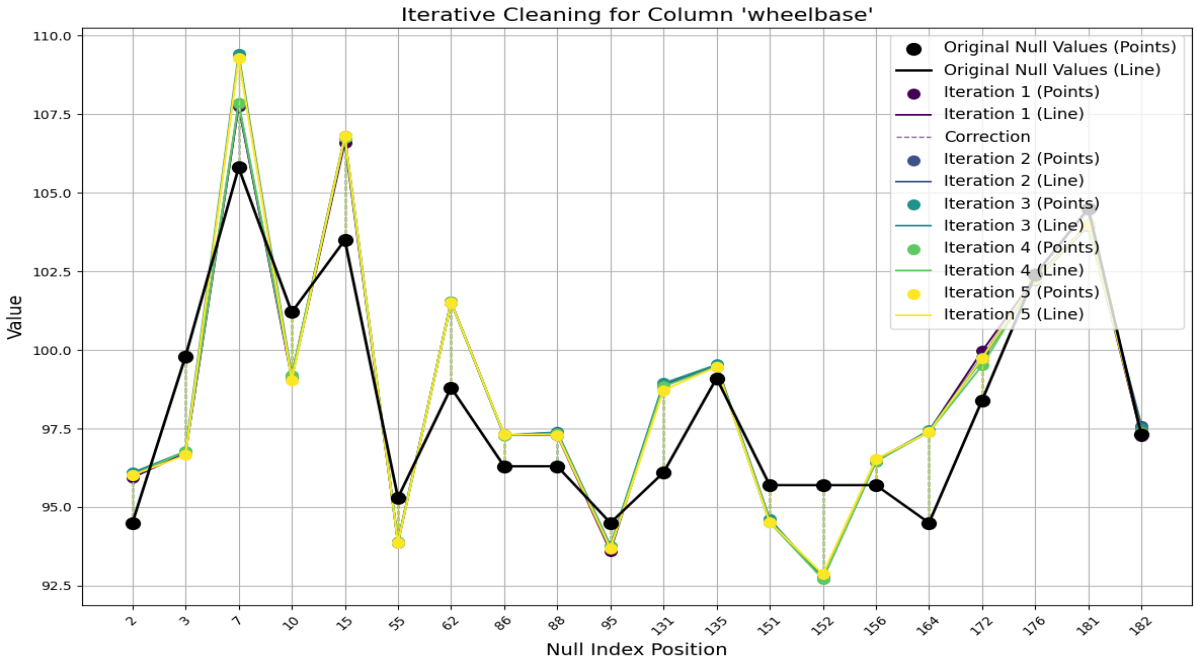

Fig 3 Iterative Cleaning

The graph illustrates the iterative cleaning process for the "wheelbase" column, where missing (null) values, represented by black points and lines, are corrected over five iterations. Each iteration adjusts the null values, shown as points and lines in different colors, progressively refining their alignment with the data's overall trend. This visualization highlights how iterative imputation improves the treatment of missing values with each step.

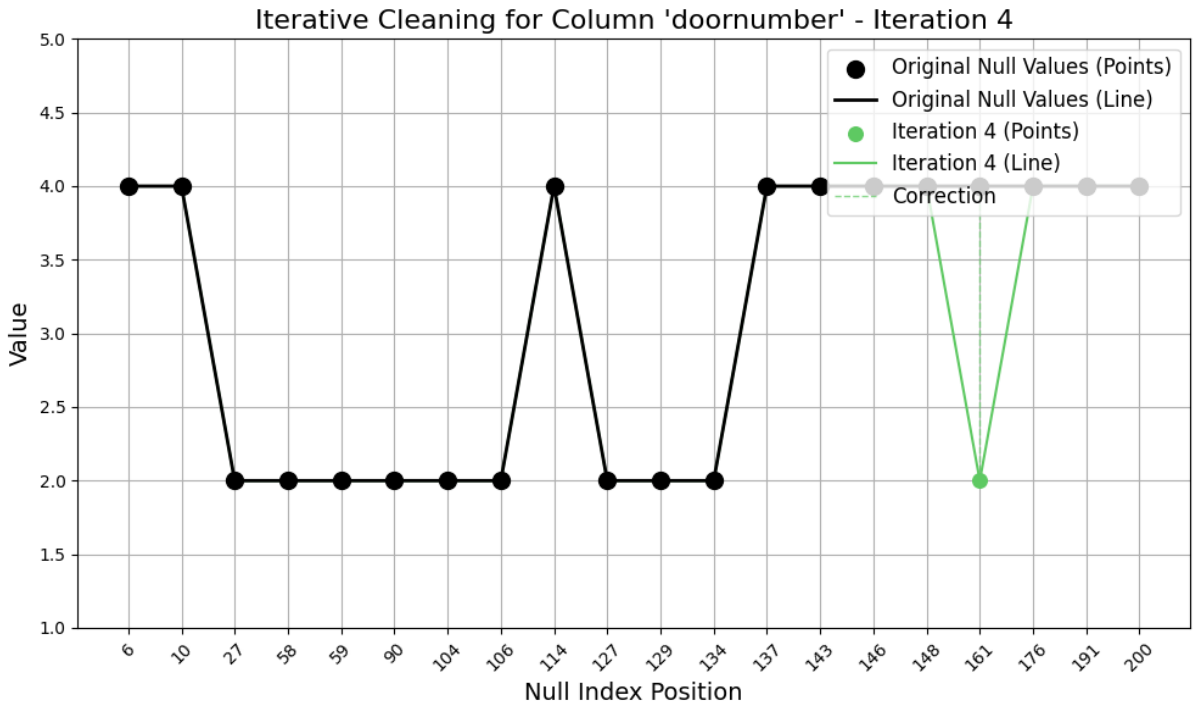

Fig 4 Iterative Cleaning Iteration-4

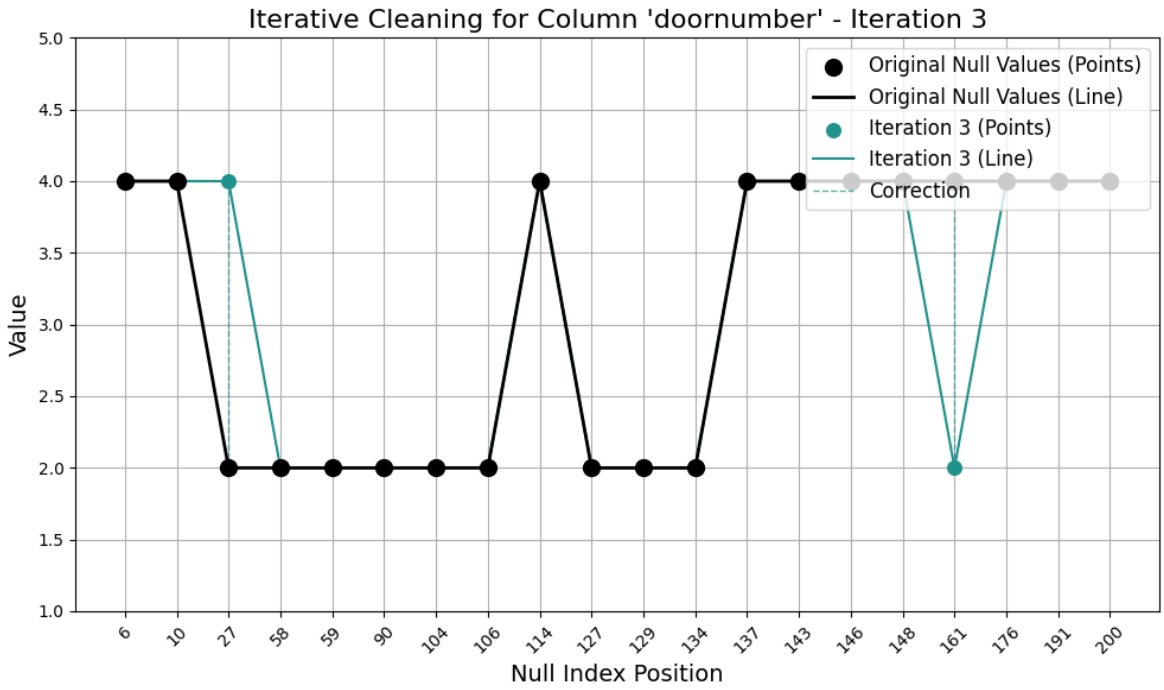

Fig 5 Iterative Cleaning Iteration-4

The graph showcases the iterative cleaning process for the "doornumber" column at Iteration 3, where missing (null) values, represented by black points and lines, are corrected. The corrections in this iteration are shown by teal points and lines, indicating the adjustments made to align the null values with the expected data trends. This visualization emphasizes how the iterative process gradually improves data quality by addressing inconsistencies.

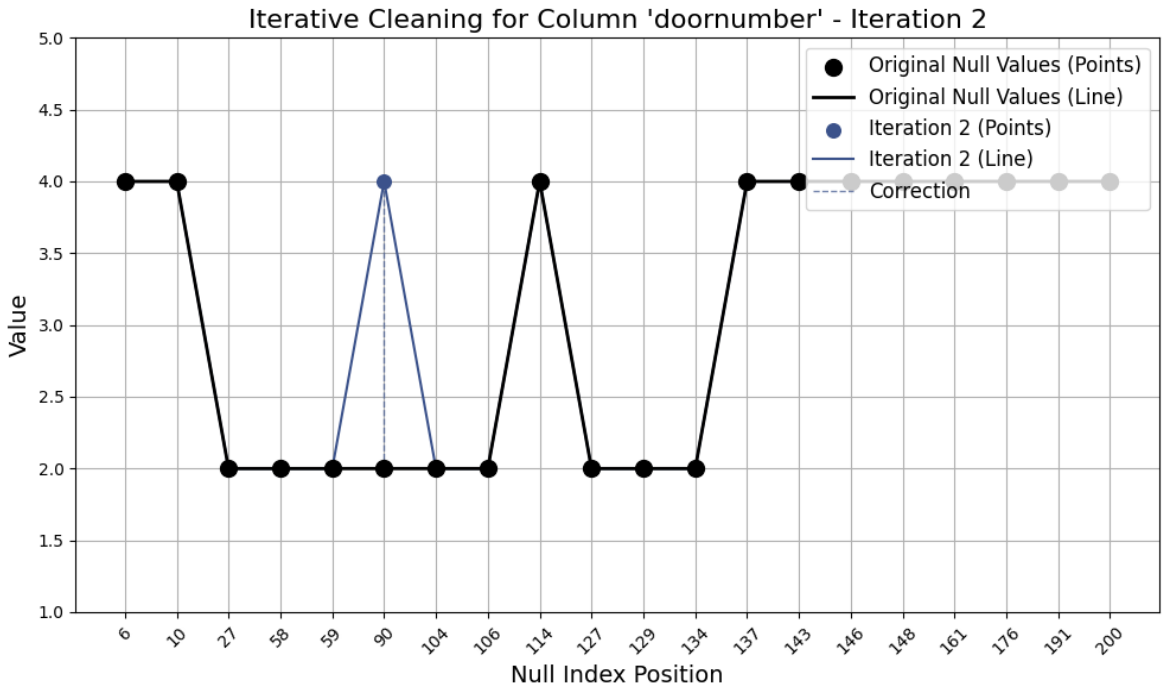

Fig 6 Iterative cleaning iteration-5

Analysis

The iterative cleaning process applied to the used car dataset highlights an effective method for handling missing data, ensuring the dataset's completeness and accuracy for further analysis. Missing data, if left unaddressed, can introduce biases or distortions in the dataset, significantly affecting the quality of insights or predictive models derived from it. The analysis of this iterative cleaning methodology demonstrates its ability to progressively refine imputed values and align them with the dataset's inherent structure and trends. By starting with an exploration phase to identify columns containing null values and their specific indices, the process established a clear understanding of the dataset's issues, allowing for tailored solutions based on the nature of the missing values.

Traditional imputation methods, such as mean, median, or mode substitution, served as an initial step to fill missing data quickly. However, these methods lacked the contextual depth required for datasets with complex interdependencies between variables. For instance, numerical columns like "wheelbase" exhibited improved results when handled through iterative imputation, as this method leveraged relationships between features to provide context-aware estimates. Similarly, categorical variables, such as "doornumber," were efficiently imputed with fewer iterations, underscoring the flexibility of the iterative approach across diverse data types. This progressive method ensured that imputed values were not only statistically plausible but also aligned with the overall data distribution, reducing the likelihood of introducing inconsistencies.

A key aspect of the iterative cleaning process was the visualization of imputation progress. Visual tools, including scatter and line plots, were instrumental in tracking changes to the missing values over successive iterations. These visualizations provided clarity on how imputed values evolved, offering transparency in the process. Correction lines drawn between original null values and their imputed counterparts illustrated the magnitude and direction of adjustments, further validating the method. For example, in the "wheelbase" column, imputed values gradually approached the dataset’s trend over five iterations, demonstrating the strength of iterative refinement for continuous data. Similarly, for the "doornumber" column, rapid convergence during the third iteration reflected the method's efficiency in dealing with discrete data.

The scalability and efficiency of this methodology became evident when applied to multiple columns within the dataset. By focusing on null indices and processing each column

individually, the approach minimized computational overhead, making it suitable for large datasets with numerous missing values. The number of iterations required for each column varied based on its characteristics and correlations with other variables, showcasing the adaptability of the method. This efficiency ensured that the iterative process could be seamlessly integrated into larger data cleaning workflows without significant performance trade-offs.

The final evaluation of the cleaned dataset highlighted the iterative approach's success in producing high-quality imputed values. Visual assessments confirmed that missing values were filled in a manner consistent with the dataset’s trends, while statistical analysis validated the imputation's accuracy. The iterative process allowed for refinement at each step, ensuring that the cleaned dataset was free of anomalies or distortions. This combination of progressive refinement and robust validation made the iterative cleaning methodology a reliable and transparent solution for addressing missing data. Overall, the analysis underscores the importance of iterative imputation in data preparation, particularly in scenarios where accuracy and consistency are critical.

Conclusion

To sum up, this research project took a bold step to conduct an extensive exploratory analysis of the variables associated with the occurrence of an aircraft incident as well as patterns and other interesting facts around accidents, failures, and hijack cases. The dataset was prepared for analysis by performing data cleaning and imputing missing data where necessary. Techniques such as Z-scores and interquartile range which are essentially statistical techniques were used to identify extreme values which otherwise would have been outliers and helped to enhance the trends in the data provided clarity. A univariate and bivariate analysis looked at some of the variables in detail such as deaths in which case graphs like histograms and boxplots were used to show where the variable expressed in regard to its frequency and distribution. The bivariate analysis assisted in identifying some relationships between the pairs of variables such as: incident types and the aircraft types involved, providing insight on the patterns of incident types as well as total deaths across various years. In the multi-variate analysis, stacked bar charts and advanced visualizations, including correlation heatmaps, further uncovered associations among multiple aspects allowing taking a better look at how distinctive characteristics worked together in high fatality incidents or in more specific types of incidents such as equipment failures or human errors. Pair plots and correlation analyses provided further understanding of the connections between the variables thus facilitating a better comprehension of the relations between the aircraft type, date of an incident, and number of fatalities. Analysis of the correlation among variables for example through the use of a heat map and a pair plot exposed some of the connections, and how types of incidents spread across categories and time, was presented in a more complex manner. Overall, this analysis forms a foundation for advancing the predictive modelling or risk assessment that can be used to improve safety and develop or modify the policies related to aviation activities. The analysis emphasizes the need for proper and robust data management and presentation in the conclusions drawn about the complicated data associated with the aviation incidents.

References

Dataset Source:

“https://www.kaggle.com/datasets/thedevastator/airplane-crashes-and-fatalities**”**

1. **Towards Data Science (EDA in Python)** - A blog site with various EDA tutorials in Python using popular libraries like Pandas, Matplotlib, and Seaborn.

   Website: EDA on Towards Data Science
2. **Kaggle** - Kaggle hosts a range of datasets, EDA projects, and notebooks shared by the data science community. You can find specific EDA projects on aviation incidents and related topics.

   Website: Kaggle EDA Projects
3. **Data Science Central** - This platform provides articles, tutorials, and guides on EDA and visualization. Search for aviation data projects or EDA tutorials.

   Website: Data Science Central
4. **GitHub** - You can find open-source EDA projects on GitHub by searching for "aircraft incident EDA" or "aviation data analysis."

   Website: GitHub
5. **DataCamp Blog: EDA Guide** - DataCamp provides articles and tutorials on EDA processes, data visualization techniques, and more.

   Website: DataCamp EDA Guide
6. **Statistical Data Visualization (Books)**

- "Python for Data Analysis" by Wes McKinney - A comprehensive guide on data manipulation and EDA with Python.
- "Practical Statistics for Data Scientists" by Peter Bruce and Andrew Bruce - Covers statistical concepts used in EDA.
- "Data Science from Scratch" by Joel Grus - Includes an EDA introduction and techniques for working with data.

7. **Matplotlib and Seaborn Official Documentation** - For in-depth visualization tutorials using Python libraries Matplotlib and Seaborn.
   - Matplotlib: Matplotlib Documentation
   - Seaborn: Seaborn Documentation

8. **Books on Data Visualization**

- "Storytelling with Data" by Cole Nussbaumer Knaflic - A guide to creating impactful data visualizations.
- "The Visual Display of Quantitative Information" by Edward R. Tufte - A classic on visualizing quantitative data.
- "Fundamentals of Data Visualization" by Claus O. Wilke - Covers practical techniques for effective visualization.

9. **D3.js Data Visualization Library** - For web-based interactive visualizations, D3.js is a powerful library with numerous aviation and EDA projects.

    Website: D3.js Documentation

Gitub Link: https://github.com/HARMESH095/no-code-analysis.git