Dhirubhai Ambani Institute of Information and Communication Technology

Gandhinagar, Gujarat

Group Number : Group_4

Group Members :

Ghori Zeel Jivrajbhai (202201287)

Aryankumar Panchasara (202201056)

Nischay Agrawal (202411031)

Course : IT 462 Exploratory Data Analysis

Professor : Gopinath Panda

Semester : Autumn 2024

Assignment 2 : MCAR_test

Date : 22/9/24

Missing Data and Its Challenges:

Missing data is a prevalent issue in datasets, often arising from data collection errors or unavailable information. Handling missing data is crucial because it can introduce biases, reduce statistical power, and lead to incorrect conclusions. Understanding the mechanisms of missing data—Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR)—is essential for selecting appropriate handling methods.

- **MCAR**: Missingness is random and unrelated to any data values. Little's MCAR Test is used to determine if data is MCAR, with a p-value greater than 0.05 suggesting randomness.
- **MAR**: Missingness depends on observed data but not on the missing values themselves.
- **MNAR**: Missingness depends on the unobserved missing values.

The `missingpy` Library

`missingpy` is a Python library designed for missing data imputation, particularly useful in machine learning. It offers methods like:

- **MissForest Imputation**: Uses Random Forests to impute missing values iteratively, handling both continuous and categorical data.
- **KNN Imputation**: Fills missing values using the similarity between observations.
- **Iterative Imputer**: A multivariate approach that models each variable with missing data using other observed variables iteratively.

While `missingpy` *offers robust imputation techniques, it is not always possible to use the library in every environment due to installation or dependency issues, as was the case in our analysis. In such cases, alternative imputation methods, such as those provided by* `fancyimpute`*, can be used.*

Missing Completely at Random (MCAR)

MCAR is the ideal scenario for missing data because the missingness does not introduce any bias. If data is MCAR, any missing values are truly random, and any analysis done on the available data will still be valid. To test whether missing data is MCAR, **Little's MCAR Test** is commonly used.

Little's MCAR Test

Little's MCAR Test is a statistical test specifically designed to assess whether the missing data mechanism is MCAR. The test compares the observed patterns of missing data to what would be expected if the missingness were truly random. Here's a brief overview of the process:

- **Hypothesis**:
    - Null Hypothesis (H0): The data is MCAR.
    - Alternative Hypothesis (H1): The data is not MCAR (suggesting it might be MAR or MNAR).
- **Test Statistic**: Little's MCAR Test calculates a test statistic based on the likelihood of the observed missing data patterns compared to expected patterns under the MCAR assumption.
- **P-Value Interpretation**:
    - If the p-value is greater than 0.05, there is insufficient evidence to reject the null hypothesis, suggesting that the missing data is MCAR.
    - If the p-value is less than or equal to 0.05, it indicates that the missing data is not MCAR, implying that the missingness might be related to observed or unobserved values (MAR or MNAR).
- **Importance**:
    - Identifying MCAR allows for simpler data handling techniques without the need for complex imputation models since the missingness is unbiased.

Example Application: If a dataset has missing values, applying Little's MCAR Test can help determine if simpler statistical analyses or unbiased imputation methods can be used without significantly altering the results.

KNN Imputation

K-Nearest Neighbors (KNN) imputation is a method used to handle missing data by leveraging the similarity between observations in a dataset. The core idea behind KNN imputation is that missing values can be predicted using the known values from other, similar data points (neighbors). This technique assumes that the missing values are related to the values of the nearest neighbors, which can be found by calculating the distance (or similarity) between observations.

```
knn_filled = KNN(k=2).fit_transform(df)
```

This line uses **K-Nearest Neighbors (KNN)** imputation to fill in missing values in the `numeric_df` dataframe. The KNN algorithm is based on the idea that the missing value of a sample can be estimated using the values from the most similar or "nearest" samples in the dataset.

- `KNN(k=2):` The `k=2` parameter specifies that for each missing value, the algorithm will look at the two nearest neighbors (based on similarity in the dataset) to predict the missing value. The algorithm computes the distances between observations using features that are not missing, and it fills in the missing value by averaging the values of its 2 closest neighbors.
- `.fit_transform(numeric_df)`: This method first fits the KNN model on the dataset and then transforms (fills) the missing values. The result is an imputed dataframe `knn_filled` where the missing values in `numeric_df` have been replaced using the KNN algorithm.

Iterative Imputation

Iterative Imputation is a multivariate imputation technique that fills missing values by modeling each feature with missing data as a function of the other features. It works by treating missing values as unknowns and predicting them iteratively based on the known values of other variables. The core idea behind iterative imputation is that each feature in the dataset is imputed one by one, using regression models that predict missing values based on the other available variables in the dataset. After each round of imputation, the dataset becomes more complete, and the model can refine its predictions in subsequent iterations.

```
iterative_filled = IterativeImputer().fit_transform(df)
```

The `IterativeImputer` uses a **multivariate iterative approach** to impute missing values. The method models each feature as a function of the other features and iteratively predicts missing values for each variable.

- `IterativeImputer():` This creates an instance of the iterative imputer, which works by iteratively predicting each feature with missing data using a regression model based on the other features in the dataset. It repeats the process multiple times, with each iteration refining the imputed values.
- `.fit_transform(numeric_df)`: This method first fits the iterative imputation model on the data and then transforms it by filling the missing values. The result, `iterative_filled`, is a fully imputed dataframe where the missing values in `numeric_df` have been replaced.

Missing Forest Imputation Method

Missing Forest Imputation is a method that uses Random Forests to iteratively impute missing values in a dataset. It's a robust, non-parametric approach that handles both continuous and categorical variables, making it ideal for complex datasets with mixed data types. This method works by training Random Forest models to predict missing values based on the observed data, refining the imputation iteratively until convergence.

Key Features:

- **Handles Mixed Data**: Suitable for datasets with numerical and categorical variables.
- **Iterative Refinement**: Continuously improves imputed values using Random Forest predictions.
- **No Assumptions**: Does not assume normal distribution, unlike some other imputation methods.

Advantages:

- Captures complex variable interactions.
- Reduces bias and improves data quality.
- More accurate than basic imputation techniques like mean or median.

```
missing_forest_filled = MissForest().fit  _transform(df)
```

Key Steps:

- **Initialize**: Create a `MissForest` object.
- **Fit and Transform**: Use `.fit_transform()` to apply the imputation.
- **Results**: The output is a DataFrame with missing values filled based on Random Forest predictions.

Google Drive link :

https://drive.google.com/drive/folders/1PlsaW3ZcXOKaJo9yC1PQsZ_JfLlwbxIF?usp=sharing'

Github link :

Exploratory_Data_Analysis/Group4_MCAR_test at main · zeelghori27012004/Exploratory_Data_Analysis (github.com)