Dhirubhai Ambani Institute of Information and Communication Technology

Gandhinagar, Gujarat

IT-462

Exploratory Data Analysis

(Prof. Gopinath Panda)

Assignment 2: MCAR Test Explanation

Group – 21

Dataset: Total Affected by Natural Disaster

Khelan Bhatt – 202411025

Dishant Patel – 202201260

Pritish Desai – 202201312

GitHub Link: https://github.com/pdishant5/IT-462-Exploratory-Data-Analysis-Group21

Assignment 2: MCAR Test Explanation

1. Introduction

1.1 What is "Missingpy" Library?

- **Missingpy** is a **Python** library designed to provide efficient, simple, and scalable tools to handle missing data.
- It contains methods that help with missing data imputation and analysis, particularly by utilizing machine learning approaches. Some of the main features of Missingpy include K-Nearest Neighbours (KNN) imputation and MissForest imputation, which use multivariate methods to estimate missing values.

1.2 Installation:

```
pip install missingpy
```

2. Core Functionalities

2.1 K - Nearest Neighbours (KNN) Imputation

➢ Example

```
# Let X be an array containing missing values

from missingpy import KNNImputer

imputer = KNNImputer()

X_imputed = imputer.fit_transform(X)
```

➢ Description

- The class provides a method for imputing missing values using the K-Nearest Neighbours (KNN) approach. For each sample with missing data, the missing values are estimated using the values from the n

neighbours' closest samples in the training set. If a sample has more than one missing feature, different sets of n neighbours may be used for each missing feature, depending on the feature being imputed.

- Each missing value is then replaced by the average (either weighted or unweighted) of the selected neighbours' values. If fewer than n neighbours are available for a feature, the training set average for that feature is used for imputation.
- The total number of samples available in the training set is always at least as large as the number of neighbours needed for imputation, though this may depend on the overall sample size and how many samples are excluded due to exceeding the limit for missing values per row (controlled by row_max_missing).
- The following code snippet demonstrates how to replace missing values (represented by np.nan) with the mean value of the two nearest neighbours of rows that contain missing data:

```
import numpy as np

from missingpy import KNNImputer

nan = np.nan

X = [[1, 2, nan], [3, 4, 3], [nan, 6, 5], [8, 8, 7]]

imputer = KNNImputer(n_neighbors=2, weights="uniform")

imputer.fit_transform(X)

array([[1. , 2. , 4. ],

       [3. , 4. , 3. ],

       [5.5, 6. , 5. ],

       [8. , 8. , 7. ]])
```

2.2 Random Forest Imputation (MissForest)

➢ Example

```
# Let X be an array containing missing values

from missingpy import MissForest

imputer = MissForest()

X_imputed = imputer.fit_transform(X)
```

➢ Description

- MissForest imputes missing values iteratively using Random Forests. The process begins by selecting the column with the fewest missing values—referred to as the "candidate column".
- In the first step, missing values in the other columns (non-candidate columns) are filled with an initial guess, which is typically the column mean for numerical variables and the column mode for categorical variables. Categorical variables must be explicitly identified during the imputer's fit() method call.
- Next, the imputer fits a Random Forest model, using the candidate column as the target variable and the remaining columns as predictors, based only on rows where the candidate column values are present. Once the model is trained, the imputer predicts and fills in the missing values of the candidate column. The fitted Random Forest uses the non-candidate columns' rows as input for the predictions.
- The imputation process then moves to the next column with the second smallest number of missing values, repeating the procedure for all columns with missing data. The entire process may involve multiple iterations or "epochs" for each column, continuing until the stopping criterion is satisfied. This criterion is based on the "difference" between the imputed arrays from one iteration to the

next. For numerical variables (num_vars_), this difference is defined as follows:

```
sum((X_new[:,num_vars_] - X_old[:,num_vars_])**2)/

sum((X_new[:,num_vars_])**2)
```

- For categorical variables(cat_vars), the difference is defined as follows:

```
sum(X_new[:, cat_vars_] != X_old[:, cat_vars_])) /
n_cat_missing
```

- The stopping criterion in MissForest is based on the difference between the newly imputed array (X_new) and the array imputed in the previous round (X_old).
- For numerical variables, this difference is calculated as the sum of squared differences between X_new and X_old, while for categorical variables, the difference is measured as the total number of categorical values that remain missing (n_cat_missing).
- The summation is performed across both rows and columns for all variables.
- The stopping criterion is met when the difference between X_new and X_old increases for the first time for both numerical and categorical variables (if available). When this happens, the imputer halts further iterations, assuming that the optimal imputation has been reached and further iterations would not improve the results.

```
from missingpy import MissForest

nan = float("NaN")

X = [[1, 2, nan],[3, 4, 3],[nan, 6, 5],[8, 8, 7]]
```

```
imputer = MissForest()

imputer.fit_transform(X)

array([[1.  , 2. , 4. ],

       [3.  , 4. , 3. ],

       [3.16, 6. , 5. ],

       [8.  , 8. , 7. ]])
```

3. Conclusion

- In the “Missingpy” library, the KNN-Imputer and MissForest functions offer two robust methods for imputing missing data.
- The KNN-Imputer fills in missing values by identifying the nearest neighbours in the dataset and averaging or selecting the most common value from them, making it effective for datasets with local similarities.
- MissForest uses Random Forest models to iteratively predict missing values, leveraging the power of decision trees to handle both categorical and continuous data, and is particularly useful for capturing complex, non-linear relationships in the data.
- Both methods are more advanced and accurate than simple imputation techniques, with MissForest being better suited for more complex datasets.