Detailed Notes on Data Imputation Techniques

September 12, 2024

1 Introduction to Data Imputation

Data imputation is the process of replacing missing values in datasets with substituted values. This is an important step in data preprocessing because missing values can lead to biased analyses or hinder machine learning models from functioning correctly. There are several methods of data imputation, each suited to different types of data and missing data mechanisms.

The main imputation methods are:

- Mean/Median/Mode Imputation
- Forward/Backward Fill Imputation
- K-Nearest Neighbors (KNN) Imputation
- Multivariate Imputation by Chained Equations (MICE)

In the following sections, we will cover each method in detail, including formulas and Python examples.

2 Mean/Median/Mode Imputation

2.1 Description

Mean/Median/Mode imputation replaces missing values with the mean, median, or mode of the available values in that column. This method is simple and quick but can introduce bias, especially if the missing data is not completely random.

- **Mean Imputation:** Used for numerical data. The missing value is replaced with the mean of the non-missing values.
- **Median Imputation:** Often used when the data has outliers, as it is more robust than mean imputation.
- **Mode Imputation:** Used for categorical variables where the most frequent value (mode) replaces the missing values.

2.2 Formulas

For mean imputation:

$$x_{\text{imputed}} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Where:

- x_{imputed} is the imputed value,
- x_i are the observed (non-missing) values,
- n is the number of observed values.

For median imputation, the missing value is replaced by the median of the observed values:

$$x_{\text{imputed}} = \text{Median}(x_1, x_2, \ldots, x_n)$$

For mode imputation:

$$x_{\text{imputed}} = \text{Mode}(x_1, x_2, \ldots, x_n)$$

2.3 Example in Python

```python
import pandas as pd
import numpy as np

# Sample data
data = {'Age': [25, 30, np.nan, 35, 40],
        'Salary': [50000, 60000, 55000, np.nan, 80000],
        'Experience': [2, 4, np.nan, 6, 8]}

df = pd.DataFrame(data)

# Mean imputation for numerical data
df['Age'].fillna(df['Age'].mean(), inplace=True)

# Median imputation for numerical data
df['Salary'].fillna(df['Salary'].median(), inplace=True)

# Mode imputation for categorical data (or a column with a few unique values)
df['Experience'].fillna(df['Experience'].mode()[0], inplace=True)

print(df)
```

2.4 Considerations

Mean, median, and mode imputation are simple but can distort the distribution of the data, particularly for non-random missingness. For example, if the data is skewed, mean imputation may not be the best choice.

3 Forward/Backward Fill Imputation

3.1 Description

Forward and backward fill are imputation methods that replace missing values by propagating the last or next observed value:

- **Forward Fill (ffill):** Replaces missing values with the previous non-null value.
- **Backward Fill (bfill):** Replaces missing values with the next non-null value.

3.2 Formulas

For forward fill, the missing value at time t is imputed using the value at time $t-1$:

$$x_t = x_{t-1}$$

For backward fill, the missing value at time t is imputed using the value at time $t+1$:

$$x_t = x_{t+1}$$

3.3 Example in Python

```
# Forward fill imputation
df_ffill = df.fillna(method='ffill')

# Backward fill imputation
df_bfill = df.fillna(method='bfill')

print(df_ffill)
print(df_bfill)
```

3.4 Considerations

Forward and backward fill are typically used for time series data where continuity is important. However, they can propagate errors if the missing values are numerous or if the time series fluctuates rapidly.

4 K-Nearest Neighbors (KNN) Imputation

4.1 Description

K-Nearest Neighbors (KNN) Imputation fills missing values by finding the k-nearest records (rows) in the dataset that are similar to the record with missing data. The missing values are then imputed by averaging the values of the neighbors.

4.2 Formulas

The Euclidean distance between two data points i and j is calculated as:

$$d(i,j) = \sqrt{\sum_{m=1}^{n} (x_{im} - x_{jm})^2}$$

Where:

- $d(i,j)$ is the distance between points i and j,
- x_{im} and x_{jm} are the observed values for feature m for points i and j,
- n is the number of features used for distance calculation.

Once the nearest neighbors are identified, the missing value is imputed as the mean (for continuous variables) or the mode (for categorical variables) of the values from the neighbors:

$$x_{\text{imputed}} = \frac{1}{k} \sum_{j=1}^{k} x_j$$

4.3 Example in Python

```
from sklearn.impute import KNNImputer

# Sample data with missing values
data_knn = {'Age': [25, 30, np.nan, 35, 40],
            'Salary': [50000, 60000, 55000, np.nan, 80000],
            'Experience': [2, 4, np.nan, 6, 8]}

df_knn = pd.DataFrame(data_knn)

# Apply KNN imputation
knn_imputer = KNNImputer(n_neighbors=2)
df_knn_imputed = pd.DataFrame(knn_imputer.fit_transform(df_knn), columns=df_knn.

print(df_knn_imputed)
```

4.4 Considerations

KNN imputation is computationally intensive, especially for large datasets, as it requires calculating distances between all points. It also assumes that similar points (neighbors) have similar values for the missing variable, which may not always be true.

5 Multivariate Imputation by Chained Equations (MICE)

5.1 Description

Multivariate Imputation by Chained Equations (MICE) is an iterative method where each variable with missing values is modeled using the other variables in the dataset as predictors. It performs multiple imputations by using chained equations for different variables.

5.2 Formulas

5.2.1 Linear Regression for Continuous Variables

$$\hat{X}_{\text{missing}} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

Where:

- $\hat{X}_{\text{missing}}$ is the predicted value for the missing entry,
- $X_1, X_2, \ldots, X_n$ are the observed variables,
- $\beta_0, \beta_1, \ldots, \beta_n$ are the regression coefficients.

5.2.2 Logistic Regression for Binary Categorical Variables

$$P(X = 1|X_1, X_2, \ldots, X_n) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n)}}$$

5.2.3 Multinomial Logistic Regression for Categorical Variables

For categorical variables with more than two classes:

$$P(X = k|X_1, X_2, \ldots, X_n) = \frac{e^{\beta_0^k + \beta_1^k X_1 + \beta_2^k X_2 + \cdots + \beta_n^k X_n}}{\sum_{j=1}^{K} e^{\beta_0^j + \beta_1^j X_1 + \beta_2^j X_2 + \cdots + \beta_n^j X_n}}$$

5.3 Example in Python

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

# Create a dataset with missing values
data_mice = {'Age': [25, 30, np.nan, 35, 40],
             'Salary': [50000, 60000, np.nan, 70000, 80000],
             'Experience': [2, 4, np.nan, 6, 8]}

df_mice = pd.DataFrame(data_mice)

# Apply MICE imputation
mice_imputer = IterativeImputer(max_iter=10, random_state=0)
df_mice_imputed = pd.DataFrame(mice_imputer.fit_transform(df_mice), columns=df_m

print(df_mice_imputed)
```

5.4 Considerations

MICE is flexible and works well with different types of variables. It is computationally expensive due to the iterative nature of the process. It assumes that the missing data is Missing At Random (MAR), meaning that the probability of missingness depends on other observed data, which may not always hold.

6 Conclusion

In this document, we have covered four main imputation methods: Mean/Median/Mode, Forward/Backward Fill, KNN, and MICE. Each method has its advantages and limitations, and the choice of method should depend on the type of data, the nature of the missing values, and the context of the analysis.