Research Documentations

Multivariate imputation by chained equations (MICE) has emerged as a principled method of dealing with missing data.

First situation is usually to call indistinctly missing data those **arising from a non-response in survey or an interview and also those arising from a break-down in measurement instruments** [1]. **At secondly situation is due to external (technical) reason, presumably due to change** [1]

However, there are a number of evident reasons, including **imperfect procedures of manual data entry, incorrect measurements, and equipment error [7]. In the context of missing data in surveys,** the problem has been studied extensively by Huisman, 2000; Schafer and Graham, 2002; Little and Rubin, 2002 and **can arise from non-response by the interviewee or poorly designed questionnaires [8]**

The standard missing data classification of missing data [11] as

**1) Missing Data Completely At Random (MCAR);**

**2) Missing Data At Random (MAR); and**

**3) Not Missing At Random (NMAR)**

**Not Missing Data At Random (NMAR)** if the probability of a record having a missing value for an attribute could depend on the value of the attribute. **Examples** include a sensor not detecting temperatures below a certain threshold, people not filling in yearly income in surveys if the income exceeds a certain value [12]. Thus, **NMAR**, (or informatively missing, as it is often known) **occurs when the missingness mechanism depends on the actual value of the missing data** [10, 11, 13]

**Data Imputation:**

* **Mean/ Mode/ Median Imputation**: Imputation is a method to fill in the missing values with estimated ones. The objective is to employ known relationships that can be identified in the valid values of the data set to assist in estimating the missing values. Mean / Mode / Median imputation is one of the most frequently used methods. It consists of replacing the missing data for a given attribute by the mean or median (quantitative attribute) or mode (qualitative attribute) of all known values of that variable. This can further be classified as generalized and similar case imputation.
* **Prediction Model:** Prediction model is one of the sophisticated method for handling missing data. Here, we create a predictive model to estimate values that will substitute the missing data. In this case, we divide our data set into two sets: One set with no missing values for the variable and another one with missing values. First data set become training data set of the model while second data set with missing values is test data set and variable with missing values is treated as target variable. Next, we create a model to predict target variable based on other attributes of the training data set and populate missing values of test data set.
* **KNN(k-nearest neighbor) Imputation:** In this method of imputation, the missing values of an attribute are imputed using the given number of attributes that are most similar to the attribute whose values are missing. The similarity of two attributes is determined using a distance function.

**IMPUTATION**

* Most machine learning algorithms require that their inputs have no [missing values](https://scikit-learn.org/stable/glossary.html#term-missing-values), and will not work if this requirement is violated. Algorithms that attempt to fill in (or impute) missing values are referred to as imputation algorithms.

so, what you should do is to create a indicator variable (binary with 0 if the value was there and 1 for missing) in addition to imputing the value with something (in fact 0 will do just fine). Why is that good? Imagine a linear model - when you include an indicator, you in fact ask the model to estimate the optimal constant that should have been the imputed values - the factor in front of the indicator variable is exactly that. So you do not need to guess whether to use mean or mode or whatever - let the model find out for you. And if you want to use a linear model you may consider including a few interaction effects for good measure: Now you are allowing the model to estimate different slopes when values are missing.

**MICE Imputation, short for ‘Multiple Imputation by Chained Equation’** is an advanced missing data imputation technique that uses multiple iterations of Machine Learning model training to predict the missing values using known values from other features in the data as predictors.

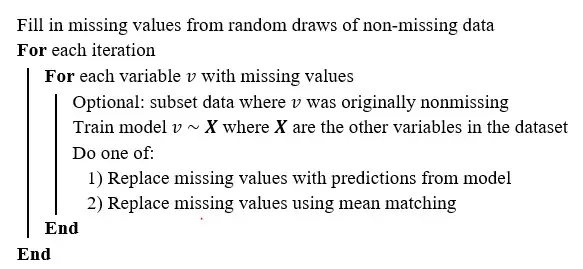
**How does MICE algorithm work?**

Here is a quick intuition (not the exact algorithm)

1. You basically take the variable that contains missing values as a response ‘Y’ and other variables as predictors ‘X’.

2. Build a model with rows where Y is not missing.

3. Then predict the missing observations.



**The MICE Algorithm (Step-by-step)**

For simplicity, let’s assume the dataset contains only 3 columns: **A, B, C** each of which contains missing values spread randomly.

The following steps are performed to perform MICE imputation:

1. **Decide on the number of iterations (k)** and create as many copies of the raw dataset.

2. In each column, **replace the missing values with an approximate value like the ‘mean’**, based on the non-missing values in that column. **This is a temporary replacement.** At the end of this step, there should be no missing values.

3. For the specific column you want to impute, eg: **column A alone, change the imputed value back to missing.**

4. Now, **build a regression model to predict A using (B and C) as predictors.** For this model, only the non-missing rows of A are included. So, A is the response, while, B and C are predictors. Use this model to predict the missing values in A.

5. Repeat steps 2-4 for columns B and C as well.

Completing 1 round of predictions for columns A, B and C forms 1 iteration.

Do this for the ‘k’ iterations you have pre-decided. With each iteration the predicted value of the temporary prediction for each column will keep improving. So, there is a continuity between the successive iteration, hence the name **‘chained’**.

By the end of the ‘kth iteration, the latest prediction (imputation) for each of the variables will be retained as the final imputation.

**Implement MICE with `IterativeImputer` from `sklearn`**

**Import the `IterativeImputer` and enable it.**

**Initialize the `IterativeImputer`.**

The default value for the number of iterations is specified using the `max\_iter` argument and is taken as 10. You might want to increase this if there are many missing values and takes more iterations to be accurate.

*Check the research note docs for the codes*

There are many different approaches to addressing missing data and the first question researchers might ask is **“why use multiple imputation?”** In certain circumstances (e.g. when there is less than 5% missingness and the missingness is totally random and does not depend on observed or unobserved values), complete case analysis may be an acceptable approach to addressing missing data (Graham, [2009](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/#mpr329-bib-0008); Schafer, [1999](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/#mpr329-bib-0028)).

 Single imputation procedures, such as **mean imputation, are an improvement but do not account for the uncertainty in the imputations; once the imputation is completed, analyses proceed as if the imputed values were the known, true values rather than imputed.**

Multiple imputation has a number of advantages over these other missing data approaches. **Multiple imputation involves filling in the missing values multiple times, creating multiple “complete” datasets.** Described in detail by Schafer and Graham ([2002](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/#mpr329-bib-0030)), the missing values are imputed based on the observed values for a given individual and the relations observed in the data for other participants, assuming the observed variables are included in the imputation model. **Multiple imputation procedures, particularly MICE, are very flexible and can be used in a broad range of settings. Because multiple imputation involves creating multiple predictions for each missing value, the analyses of multiply imputed data take into account the uncertainty in the imputations and yield accurate standard errors.** On a simple level, if there is not much information in the observed data (used in the imputation model) regarding the missing values, the imputations will be very variable, leading to high standard errors in the analyses. In contrast, if the observed data are highly predictive of the missing values the imputations will be more consistent across imputations, resulting in smaller, but still accurate, standard errors (Greenland and Finkle, [1995](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/#mpr329-bib-0010)).

Interpolation in Python – How to interpolate missing data, formula and approaches

But you need to be careful with this technique and try to really understand whether or not this is a valid choice for your data. Often, interpolation is applicable when the data is in a sequence or a series.

You should also know there are multiple interpolation methods available, the default is a *linear* method.

**When to use interpolation for imputing missing data?**

You can use interpolation when there is an order or a sequence and you want to estimate a missing value in the sequence. For example: Let’s say there are various classes of tickets in train travel, like, first class, second class, and so on. You would naturally expect the ticket price of the higher class to be more expensive than the lower class.

In that case, if the ticket price of an intermediate class is missing, you can use interpolation to estimate the missing value.

**When not to use interpolation?**

In case, there was no association between the order of the classes and the ticket fares, that is, if it was not necessary that the first class is more expensive than the second class, then, it might not be appropriate to use interpolation.

Let’s see this with an example.

import numpy as np

import pandas as pd

# class and ticket prices.

fare = {'first\_class':100,

'second\_class':np.nan,

'third\_class':60,

'open\_class':20}

Convert it to a pandas series object to make interpolation convenient.

# store as pandas series

ser = pd.Series(fare)

ser

first\_class 100.0

second\_class NaN

third\_class 60.0

open\_class 20.0

dtype: float64

Now you can use ser.interpolate() to predict the missing value. By default, ser.interpolate() will do a *linear interpolation*.

**Important caveat before you apply interpolation**

Linear interpolation will take the index (0,1,2..) as the X and the column you want to interpolate as Y and do the interpolation. So, **you need to make sure the X is sorted in your data** to make this work.

**MICE Imputation with LightGBM using miceforest**

MICE imputation can be made more efficient using the `miceforest` package. It is expected to perform significantly better because it implements `lightgbm` algorithm in the backend to do the imputation.

LightGBM is known for its high accuracy of predictions. Combining that power with the `mice` algorithm makes it a strong algorithm for imputations.

Here are some more compelling advantages:

1. It can handle categorical variables for imputations.  
2. You can customize how imputation happens.  
3. It's very fast. Can use GPU to go even faster.  
4. Data can be imputed in place to save memory.

Let's first install the miceforest dataset.

!pip install miceforest --no-cache-dir

To get the latest development version, it's available in the [github repository](https://github.com/AnotherSamWilson/miceforest" \t "_blank). Use the following command to install it.

!pip install git+https://github.com/AnotherSamWilson/miceforest.git

Import `miceforest`

import miceforest as mf

We have the original data with missing values in `df\_train`. Let's try to impute the missing values in the data with `miceforest`.

# Create kernel.

kds = mf.ImputationKernel(

df\_train,

save\_all\_iterations=True,

random\_state=100

)

# Run the MICE algorithm for 2 iterations

kds.mice(2)

# Return the completed dataset.

df\_imputed = kds.complete\_data()

**View original data with missing values.**

df\_train.head()

View Imputed dataset

df\_imputed.head()

It has predicted a value of '55' for the missing record.

Let's run for 5 more iterations and predict again.

kds.mice(iterations=5, n\_estimators=50)

**Now, let's predict.**

df\_imputed2 = kds.complete\_data()

df\_imputed2.head()

The prediction has now changed from 55 to 39.

The actual value of the data is 42, so the result gor more closer to the actual after running for 5 iterations.

**This is not a conclusive check. It will be a better idea to check the imputed value for all the missing observations in the data.**

**So, that's how you can use `miceforest`.**