Iterative Imputer – MiCE

17 July 2023 15:28

Iterative Imputer:

The Iterative Imputer is an imputation technique used to fill missing values in a dataset by **using a predictive model** to estimate the missing values based on the observed values. It works in an **iterative manner**, where missing values are imputed multiple times **until convergence** is achieved. Underlying algorithm for Iterative Imputer is MICE.

MICE (**Multivariate Imputation By Chained Equations** algorithm) is a technique by which we can impute missing values in a dataset on the basis of data in other features and trying to estimate the best prediction for each missing value.
It works by filling missing values first with **educated guesses (like Mean, Median, Mode etc.) for each missing value** based on the observed data. Then all the features with missing values are removed with educated guesses one by one and A predictive model (e.g., Linear regression, Decision trees, or k-Nearest Neighbors) is then trained using the observed and Imputed values as the input and the feature with missing values as the target variable. This process is repeated for a certain number of iterations or until the imputed values converge to best estimates of missing values.

When to Use:

1. When data is **MNAR**.
2. When the data is large with many features.

Advantages:

1. This technique is **more accurate** in terms of imputing missing values and giving better model accuracy.
2. It preserves the variability of the data.

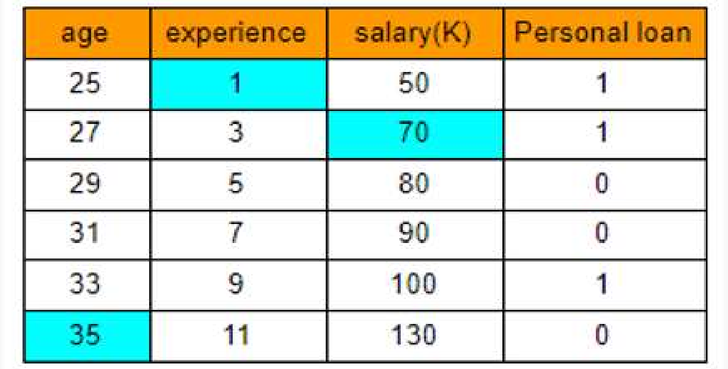
Disadvantages:

1. This involves lot of calculations**,** so it is **Computationally Intensive.**
2. In some cases, the iterative imputer may **struggle** to **converge to stable estimates**
3. This is **memory heavy** as we have to store the training data on server for imputing missing values while model is in production.

Demonstration of How Iterative Imputation happens using MICE Algorithms:

We have a dataset of Loan Applicants collected by a Banker. This is the **Actual Dataset** in which **first we will introduce some missing values** deliberately, then **impute them with mean** and finally **predict missing**

values in each column from **left to right** to demonstrate how Iterative Imputation happens.

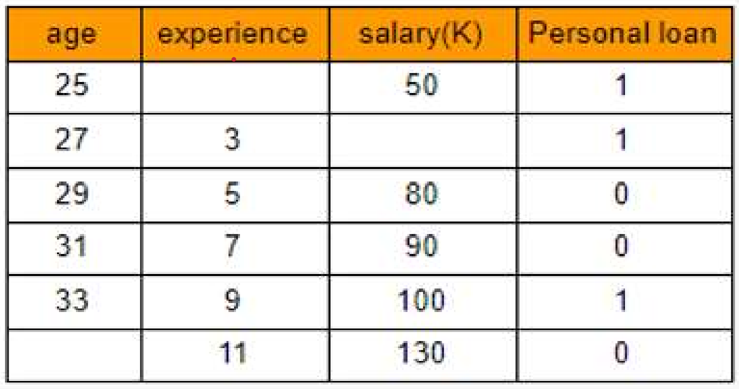


age	experience	salary(K)	Personal loan
25	1	50	1
27	3	70	1
29	5	80	0
31	7	90	0
33	9	100	1
35	11	130	0

Image by Author: Dataset with true values to verify the model output

All Steps involved in Iterative Imputation:

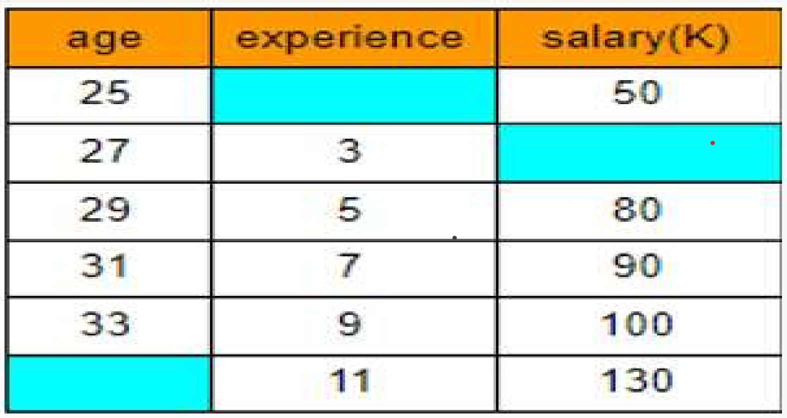
1. **Introducing Missing Values:**
   Here, some missing values have been introduced to understand how Iterative Imputation happens.



age	experience	salary(K)	Personal loan
25		50	1
27	3		1
29	5	80	0
31	7	90	0
33	9	100	1
	11	130	0

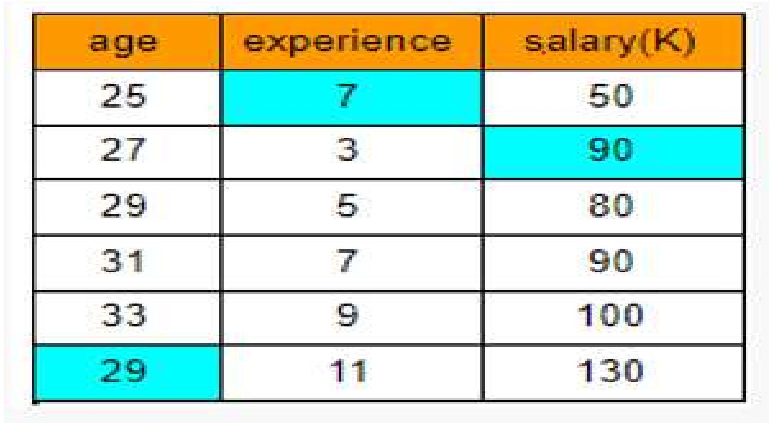
Image by Author: Dataset with missing values

2. **Removing target column as we just need to handle missing values in input features for final model.**



age	experience	salary(K)
25		50
27	3	
29	5	80
31	7	90
33	9	100
	11	130

3. **Missing Values Imputed with Mean:**

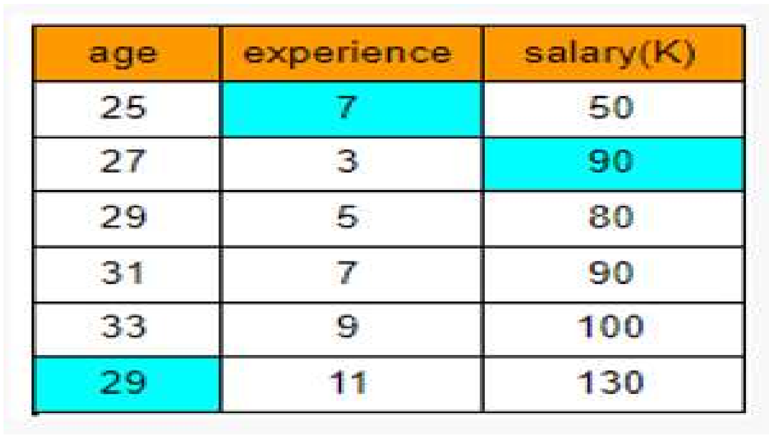


age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
29	11	130

Multivariate technique make better predictions of the missing values. So, to find the missing value in **Age column**, we will run a **regression model** on the **3** features with **Experience and Salary as features** and **Age as target**. Similarly we will try to get the missing values from experience and salary features as well.

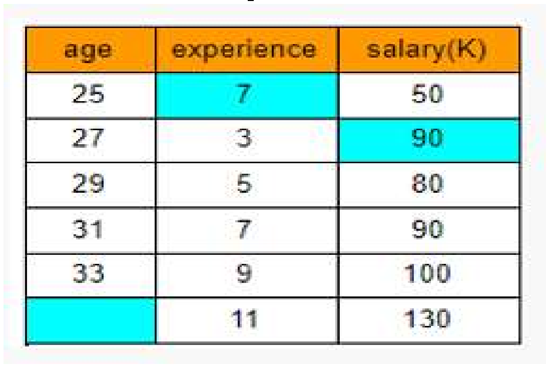
Iteration 1 Steps:

Step 1:
Impute all missing values using mean imputation with the mean of their respective columns.
We will call this as our "**Zeroth**" dataset.



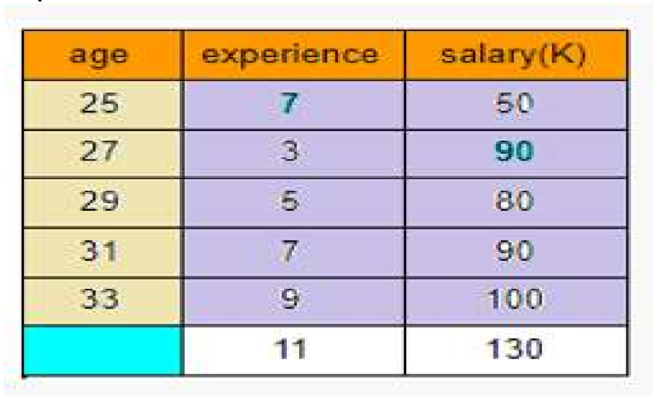
age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
29	11	130

Step 2:
Remove the "**Age**" imputed values and keep the imputed values in other columns as shown here.



age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
	11	130

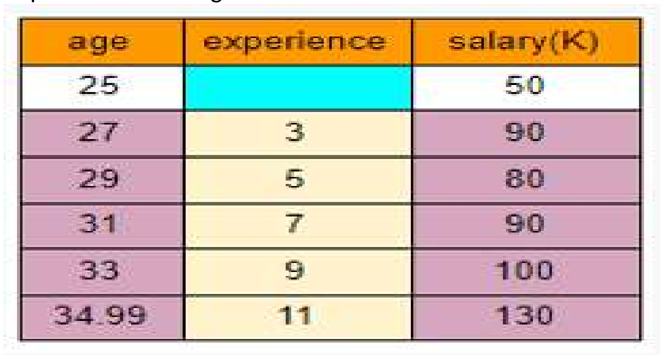
Step 3:



age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
	11	130

So, top 5 rows will be training data and the last row that has missing age will be test data. We will use (**experience = 11 and salary = 130**) to predict corresponding "age" value.
When we did this, we found that the model predicted the age as **34.99**.

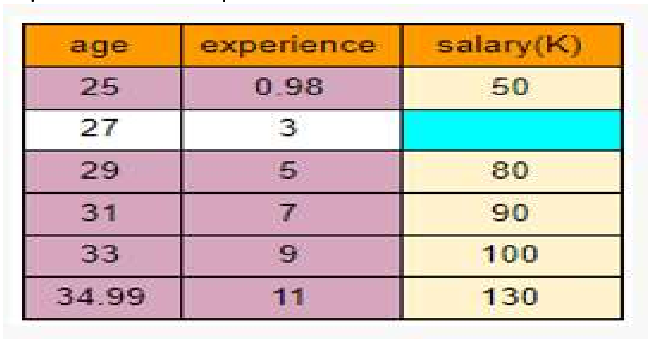
Step 4:
Imputed Predicted age in the data i.e. **34.99.**



age	experience	salary(K)
25		50
27	3	90
29	5	80
31	7	90
33	9	100
34.99	11	130

We updated the predicted age value in the missing cell in "age" column.
Now, remove "experience" imputed value. The remaining features and rows becomes the **feature matrix(purple cells)** and "**experience**" becomes the **target variable(yellow cells).**
We will run the **linear regression model** on the fully filled rows with **X= age and salary and Y=experience**.
To estimate the missing experience, we will use the missing value row (white cells) as the test data.
The predicted value for experience is **0.98**.

Step 5:
Imputed Predicted Experience **0.98** in the data.

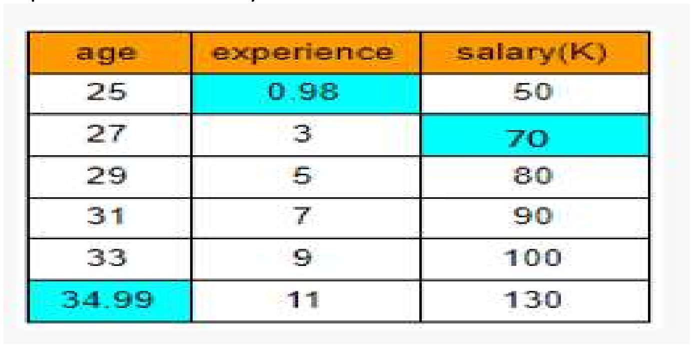


age	experience	salary(K)
25	0.98	50
27	3	
29	5	80
31	7	90
33	9	100
34.99	11	130

The predicted value for Salary is **70**.

Step 6: **First Dataset**

Imputed Predicted Salary **70** in the data.

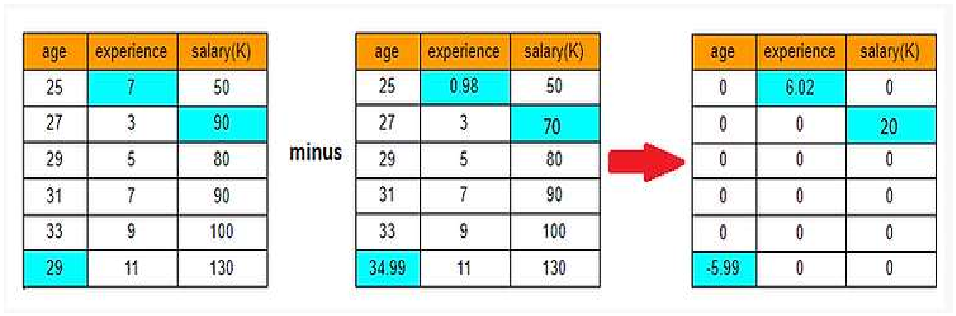


age	experience	salary(K)
25	0.98	50
27	3	70
29	5	80
31	7	90
33	9	100
34.99	11	130

We have now imputed the missing values with mean in the original dataset and the predicted values after 1st iteration is shown here.

Step 7:

Zeroth Dataset – First Dataset = Resultant Dataset showing absolute difference



age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
29	11	130

minus

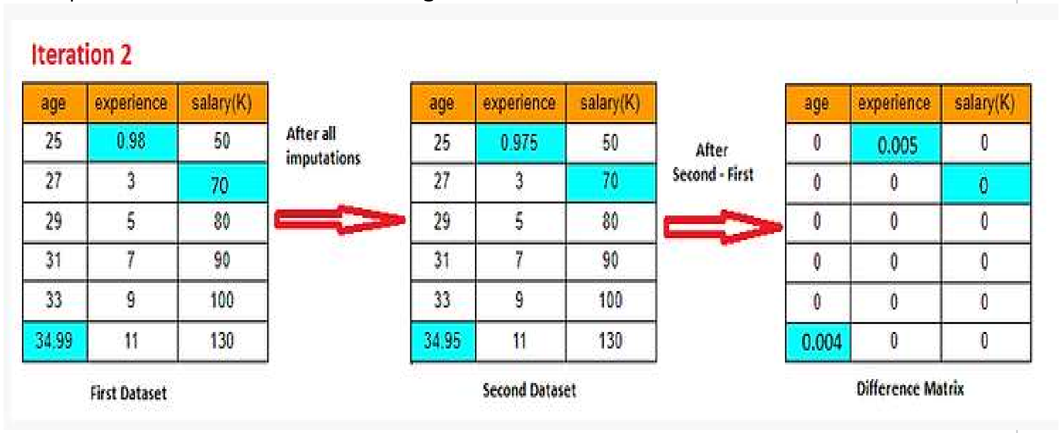
age	experience	salary(K)
25	0.98	50
27	3	70
29	5	80
31	7	90
33	9	100
34.99	11	130

age	experience	salary(K)
0	6.02	0
0	0	20
0	0	0
0	0	0
0	0	0
-5.99	0	0

If we observe, the absolute difference between 2 datasets are higher in few imputed values. Our aim is to reduce these differences close to 0. To achieve this we have to do many iterations.
So, now we repeat the steps 2-6 with the new dataset (first), until we get a stable model. i.e. until the difference between the 2 latest imputed datasets becomes very small, close to 0.

Iteration 2:

Now we will use the "**first**" dataset as our **base dataset** to do imputations, and **discard** the "**Zeroth**" dataset which had the mean imputations.
With "first" dataset as base, we performed all the steps and again predicted the imputed values for the initial 3 missing values.



Iteration 2

age	experience	salary(K)
25	0.98	50
27	3	70
29	5	80
31	7	90
33	9	100
34.99	11	130

First Dataset

After all imputations

age	experience	salary(K)
25	0.975	50
27	3	70
29	5	80
31	7	90
33	9	100
34.95	11	130

Second Dataset

After Second - First

age	experience	salary(K)
0	0.005	0
0	0	0
0	0	0
0	0	0
0	0	0
0.004	0	0

Difference Matrix

Now, after second iteration, we can see that the difference is very negligible.
The second dataset imputed values for **age is 34.95**, **experience is 0.95** and **Salary is 70**. When we compare these imputed values with the true values of the missing values which is **age =35, experience = 1 and Salary = 70K**, it is almost same with very small difference.
We can either stop here as we almost got the same numbers, or proceed with next iteration until we get 0 difference.

Important Parameters of Iterative Imputer:

estimator: ***estimator object, default=BayesianRidge()***

The "estimator" in iterative imputation is the machine learning model used to predict missing values, and its choice can impact the accuracy and efficiency of the imputation process. The default choice is BayesianRidge which is a probabilistic regression model, but other estimators can be used based on the specific characteristics of the data and the problem at hand.

tol*float, default=1e-3*

Tolerance of the stopping condition.

The "tol" parameter determines the convergence criterion for stopping the iterative process. When the absolute difference between consecutive imputations (estimations) falls below the tolerance level, the iterative imputation process stops, and the final imputed values are returned.

e.g. The "tol" parameter is set to **1e-3 (0.001)**, which means that the iterative process will stop when the absolute difference between consecutive imputations falls below 0.001.

max_iter: ***int, default=10***

Maximum number of imputation rounds to perform before returning the imputations computed during the final round.

initial_strategy: ***{'mean', 'median', 'most_frequent', 'constant'}, default='mean'***

Which strategy to use to initialize the missing values. Same as the `strategy` parameter in SimpleImputer.

imputation_order: ***{'ascending', 'descending', 'roman', 'arabic', 'random'}, default='ascending'***

The order in which the features will be imputed.
Possible values:

- `'ascending'`: From features with fewest missing values to most.
- `'descending'`: From features with most missing values to fewest.
- `'roman'`: Left to right.
- `'arabic'`: Right to left.

- `'random'`: A random order for each round.