Handling Null Values: Imputation Methods for Missing Data

Yael Dahari (325166510), Arbel Tepper (209222272)

Submitted as final project report for the Tabular Data Science course, BIU, Semester A, 2025

**Abstract**

**Problem Description**

Our goal is to improve the process of imputing missing values in datasets by providing users with multiple imputation options, along with an evaluation of their impact on model performance. This will enable users to make informed decisions about which method to use.

While working on previous assignments, we encountered the challenge of handling missing values—a process that required significant time and effort. We wished for an automated tool that could test different imputation methods and summarize their effects, allowing us to choose the most effective approach without trial and error. This project aims to create exactly that solution.

In class, we discussed various methods for imputing missing values and their potential impact on model performance. This project builds on those concepts by exploring how to quantitatively and qualitatively assess different imputation techniques.

Many users may not fully understand how different methods influence accuracy, fairness, and explainability. As a result, an imputation choice that seems beneficial may actually degrade model performance. Our project seeks to bridge this gap by automating the comparison of imputation methods, saving users time while ensuring they select the most suitable approach for their data.

**Solution Overview**

Our solution automates and improves the process of imputing missing values in datasets by systematically testing multiple imputation methods and evaluating their impact on model performance.

1. **Dataset Input & Missing Data Analysis**
   * The user uploads a dataset, and our tool automatically analyzes missing values.
   * A summary is presented, detailing the attributes with missing values, the number of affected samples, and their percentage relative to the dataset size.
2. **User Selection of Attributes for Imputation**
   * The user selects the attributes they wish to impute.
3. **Baseline Imputation for Benchmarking**
   * As a naive baseline, missing values are filled with random values within the attribute's observed range (min-max).
   * This serves as a benchmark to assess whether more sophisticated imputation methods improve model performance.
4. **Application of Multiple Imputation Techniques**
   * We apply several imputation strategies, including:
     + **Mean & Median Imputation** – Filling missing values with the attribute’s mean or median.
     + **K-Nearest Neighbors (KNN) Imputation** – Estimating missing values based on the nearest neighbors.
     + **Regression Imputation** – Predicting missing values using regression models trained on non-missing data.
     + **Row Deletion** – Removing samples with missing values as a last-resort option.
5. **Performance Evaluation & Comparison**
   * We evaluate each imputation method by training a model on the imputed data and computing multiple performance metrics, including:
     + **R² Score** – Measures how well the model explains variance.
     + **Mean Absolute Error (MAE)** – Average absolute difference between predicted and actual values.
     + **Mean Absolute Percentage Error (MAPE)** – Measures error as a percentage.
     + **Mean Squared Error (MSE) & Root Mean Squared Error (RMSE)** – Penalize larger errors more heavily.
   * The results provide a quantitative basis for comparing the imputation methods.
6. **User Guidance & Final Imputation**
   * The tool presents the evaluation results alongside explanations of each method’s strengths and weaknesses.
   * The user selects their preferred imputation method(s), which are then applied to the entire dataset.
   * The final imputed dataset is saved to a location specified by the user.

**Key Benefits**

* **Automation** – Reduces the manual effort required for testing imputation techniques.
* **Data-Driven Decision Making** – Enables users to make informed choices based on empirical results.
* **Flexibility** – Supports multiple imputation methods and user-defined preferences.
* **Improved Model Performance** – Ensures that missing data handling contributes positively to downstream tasks.

This structured approach provides users with a practical and efficient tool for handling missing data, ultimately improving the data science workflow.

**Experimental Evaluation**

Our experiment involved using four different datasets and the six missing data imputation algorithms. The selected datasets originally do not contain missing values. The missing data were introduced artificially, using the MCAR model, into each of the datasets. We chose arbitrarily the attributes that we added null values to, and the percentage of missing data. The missing data was artificially generated to enable verification of the quality of imputation, which was performed by comparing the imputed values with the original values and including this evaluation metric in the final scores.

The similarity score calculates how much the imputed data is similar to the original data. This score is ranged [0,1], where 1 = perfect recovery and 0 = total mismatch.  
Categorical similarity is the fraction of correctly imputed values (values that match the original). Numerical similarity uses Mean Absolute Error (MAE) but normalizes it by dividing by the attribute's range. Since lower MAE means better similarity, we compute `1 - normalized\_MAE`, ensuring the score is in [0,1].  
Something important about this score is that it's inapplicable in the drop method. We don't impute new values and therefore, can't compare them to the original.

An important thing to say is that the null values we introduce are being drawn at **random** meaning that results may differ from run to run and all implications we conclude about the data from the results are not final and need to be thoroughly checked.

**First Dataset: Movies Revenue**

The dataset contains 34 features about movies including budget, title, runtime, crew, cast etc. and aims to predict the revenue of the movie.  
Dimensions: (5368, 34).  
Target: Revenue.

We introduced nulls to the attributes `budget`, `original\_language`, `vote\_average`, `popularity` and in the percentages of 27%, 32%, 36%, 25% to match.  
Then, we filled each of them with the different methods and received the final scores. Let's take a look at the scores we got for the attribute `budget`:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **R^2** | **MSE** | **RMSE** | **MAPE** | **MAE** | **Similarity** |
| **Random** | 0.448 | 2.267702e+16 | 1.505889e+08 | 5019295.022% | 9.855290e+07 | 0.887 |
| **Mean** | 0.709 | 1.195051e+16 | 1.093184e+08 | 3720342.997% | 5.736363e+07 | 0.977 |
| **Median** | 0.705 | 1.212208e+16 | 1.101003e+08 | 2721735.336% | 5.731354e+07 | 0.979 |
| **Frequent** | 0.706 | 1.207491e+16 | 1.098859e+08 | 2930868.433% | 5.720527e+07 | 0.979 |
| **KNN** | 0.709 | 1.195051e+16 | 1.093184e+08 | 3720342.997% | 5.736363e+07 | 0.977 |
| **LR** | 0.694 | 1.256332e+16 | 1.120862e+08 | 2374022.588% | 5.933177e+07 | 1.000 |
| **Drop** | 0.737 | 1.114388e+16 | 1.055646e+08 | 8554331.544% | 5.408191e+07 | NA |

As we can see, when using the random method for filling the missing values, the R^2 score is pretty low – only 0.448. However, when we're using any of the other methods the score jumps to around 0.7, which is a big improvement.  
The error scores (MSE, MAPE) are extremely high no matter what method we're using but this is due to the fact that revenue is measured in millions of dollars, and therefore every mistake is amplified. If the data was normalized, we believe it would be significantly smaller. The similarity score strengthen that belief because it's essentially the normalized opposite MAE.   
The similarity score itself is pretty high in the random, which means that the budget of the data is probably distributed normally. That still, the other methods give higher similarity scores. All in all, the Mean and the KNN methods gave the best results, but we usually prefer a simpler method to a complex one so the Mean method would be the one to choose.

**Second Dataset: Laptop Price**

This dataset contains 11 attributes about laptops such as Memory, Ram, Cpu etc. and aims to predict the laptop's price.

Dimensions: (1303, 11).  
Target: Price.

We introduced nulls to the attributes `Cpu`, `Company\_Cpu`, `ScreenResolution`with the percentages of 26%, 17% ,20% accordingly. Here, let's take a look at the `Company\_Cpu` attribute imputations:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **R^2** | **MSE** | **RMSE** | **MAPE** | **MAE** | **Similarity** |
| **Random** | 0.345 | 400863.157 | 633.138 | 60.607% | 492.426 | 0.911 |
| **Mean** | 0.409 | 362041.728 | 601.699 | 53.669% | 456.028 | 0.972 |
| **Median** | 0.412 | 359996.010 | 599.997 | 53.63% | 454.635 | 0.974 |
| **Frequent** | 0.417 | 356953.755 | 597.456 | 53.97% | 452.950 | 0.959 |
| **KNN** | 0.409 | 362041.728 | 601.699 | 53.669% | 456.028 | 0.972 |
| **LR** | 0.417 | 356952.813 | 597.455 | 53.971% | 452.951 | 1.000 |
| **Drop** | 0.395 | 382960.707 | 618.838 | 52.246% | 462.653 | NA |

First of all, when looking at the R^2 score, we can see that the improvement from using imputation methods different than random, isn't that significant (only a 0.05 improvement). Therefore, the user can understand that using a complex method isn't the best course of action in this case.   
The random imputation does pretty well in restoring the values – we can see a similarity score of 0.911. All in all, from this table we can deduce that the preferable imputation method should be the Frequent. It is fast and simple, and gains a nice improvement while managing to predict the missing values correctly.

**Third Dataset: Cars Price**

This dataset contains 32 features about cars such as age, number of cylinders, brand, accident, clean title etc. and aims to predict their price.

Dimensions: (188533, 32).  
Target: Price.

We introduced null values to the attributes `milage`, `horse\_power`, `brand`, `age` with the percentages of 28%, 26%, 32%, 30% accordingly. Let's observe the results we got for the `brand` attribute:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **R^2** | **MSE** | **RMSE** | **MAPE** | **MAE** | **Similarity** |
| **Random** | 0.475 | 0.605 | 0.778 | 308.944% | 0.571 | 0.96 |
| **Mean** | 0.475 | 0.605 | 0.778 | 307.872% | 0.570 | 0.97 |
| **Median** | 0.475 | 0.605 | 0.778 | 307.875% | 0.570 | 0.97 |
| **Frequent** | 0.475 | 0.605 | 0.778 | 307.895% | 0.570 | 0.97 |
| **KNN** | 0.475 | 0.605 | 0.778 | 307.872% | 0.570 | 0.97 |
| **LR** | 0.472 | 0.608 | 0.780 | 309.581% | 0.572 | 1.000 |
| **Drop** | 0.484 | 0.590 | 0.768 | 299.941% | 0.565 | NA |

Here, we can see that all methods give pretty much the same results. There could be several explanations for that – the dataset might be one such that the mean and the median are the same and so imputing with one or the other generates the same results. Another important thing to consider is that the data is too complex for the Linear Regression model we're using. This is supported by the fact that the error metrics are all high, and even when using LR to impute the values, where we see a perfect restoration with a score of 1, the errors are really high (MSE of 0.6 and MAE of 0.5). This means that the imputation methods are good (the other methods have a similarity score of 0.96) and the thing that causes big errors is the complexity of data – it simply doesn't behave in a linear way. So, we actually gained insight of the way the data behaves without even having to study it – another bonus of our tool.

**Fourth Dataset:**  **Avocado Average Price**

This dataset contains 12 features about avocado such as type, year, region etc. and aims to predict their average price.

Dimensions: (18249, 12).  
Target: AveragePrice.

We introduced null values to the attributes `TotalBags`, `type`, `year` with the percentages 28%, 31%, 27% accordingly. Let’s observe the results for the attribute `type`:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **R^2** | **MSE** | **RMSE** | **MAPE** | **MAE** | **Similarity** |
| **Random** | 0.244 | 0.121 | 0.348 | 20.896% | 0.272 | 0.840 |
| **Mean** | 0.294 | 0.113 | 0.336 | 20.103% | 0.262 | 0.843 |
| **Median** | 0.229 | 0.123 | 0.351 | 21.184% | 0.273 | 0.846 |
| **Frequent** | 0.229 | 0.123 | 0.351 | 21.184% | 0.273 | 0.846 |
| **KNN** | 0.294 | 0.113 | 0.336 | 20.103% | 0.262 | 0.843 |
| **LR** | 0.124 | 0.140 | 0.374 | 23.142% | 0.297 | 1.000 |
| **Drop** | 0.403 | 0.096 | 0.310 | 17.995% | 0.238 | NA |

Just from a quick inspection of the table we can learn that the median and the frequent values of this attribute in the data are the same, since the scores across all metrics are the same. Also, we can gather that the Mean and the KNN imputations cause the same results, since again all scores are the same. Interestingly though, the R^2 scores are pretty low, and the lowest one is actually in the LR method, although it succeeds to perfectly restore the missing data. The highest value is at the Drop method and it's significantly higher than the rest (by at least 0.1). What we can infer from that, is that the data doesn't behave in a linear way. When the imputed data was more different than the original data we actually saw improvements, and we ignored the null samples and dropped them altogether it yielded the best results.

Now show graphs !!

**Related Work**

Several studies have explored different approaches to handling missing values in datasets. Our solution builds on existing methods while introducing key enhancements to improve usability, flexibility, and practical applicability for data scientists.

One relevant work is *“Experimental Analysis of Methods for Imputation of Missing Values in Databases”* [1]. This paper evaluates multiple imputation algorithms, including Hot-Deck Imputation, Naïve Bayes, and Mean Imputation. Hot-Deck Imputation fills all missing values in a sample simultaneously, rather than handling each attribute separately. Since our approach focuses on per-attribute imputation, this method was not directly applicable to our problem. However, their methodology for evaluating imputation performance was insightful.

Another key reference is *“A Survey on Missing Data in Machine Learning”* [2], which provides a broad overview of imputation techniques ranging from simple methods (mean, median) to more complex approaches like Expectation-Maximization (EM), Support Vector Machines (SVM), and Decision Trees. While these advanced methods can be powerful, they introduce significant computational complexity. Given that our solution is designed for practical, real-time decision-making in the data science workflow, we opted for a more interpretable and computationally efficient set of imputation techniques. For example, since we are using Linear Regression as a learning model, we deemed it excessive to use an imputation method that is more complex than the model itself.

**Key Differences & Contributions**

While prior research has focused on proposing and evaluating different imputation methods, our solution provides a **personalized and interactive** approach:

1. **User-Centric Decision-Making** – Unlike most existing tools that apply a single imputation strategy to the entire dataset, our solution allows users to choose which attributes to impute and provides real-time comparisons of different methods.
2. **Evaluation & Explanation** – We not only compute performance metrics for each imputation method but also provide explanations about their advantages and drawbacks, helping users make informed decisions.
3. **Automated Dataset Processing** – Our tool produces a finalized, imputed dataset that users can directly use for further analysis, eliminating the need for manual data preprocessing.

**Inspiration from Existing Work**

One particularly valuable insight we gained from *“Experimental Analysis of Methods for Imputation of Missing Values in Databases”* [1] was their experimental setup. Instead of searching for datasets with naturally occurring missing values, they artificially introduced missing values using the *Missing Completely at Random (MCAR)* model. This ensures a controlled evaluation process, allowing us to compare the imputed values with the original ones. Inspired by this approach, we adopted a similar methodology to assess the effectiveness of our imputation techniques.

By building upon existing research and integrating a user-friendly evaluation process, our solution aims to bridge the gap between theoretical advancements in missing data imputation and practical implementation in data science workflows.

**Conclusion**

**References**

[1] A. Farhangfar, L. Kurgan, W. Pedrycz. "Experimental analysis of methods for imputation of missing values in databases". Proceedings Volume 5421, Intelligent Computing: Theory and Applications II; (2004).

[2] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, O. Tabona. "A survey on missing data in machine learning". Journal of Big Data volume 8, Article number: 140 (2021).