Photo by Kevin Ku on Unsplash

7 Ways to Handle Missing Values in Machine Learning

Popular strategies to handle missing values in the dataset

Satyam Kumar Jul 24 · 5 min read ★

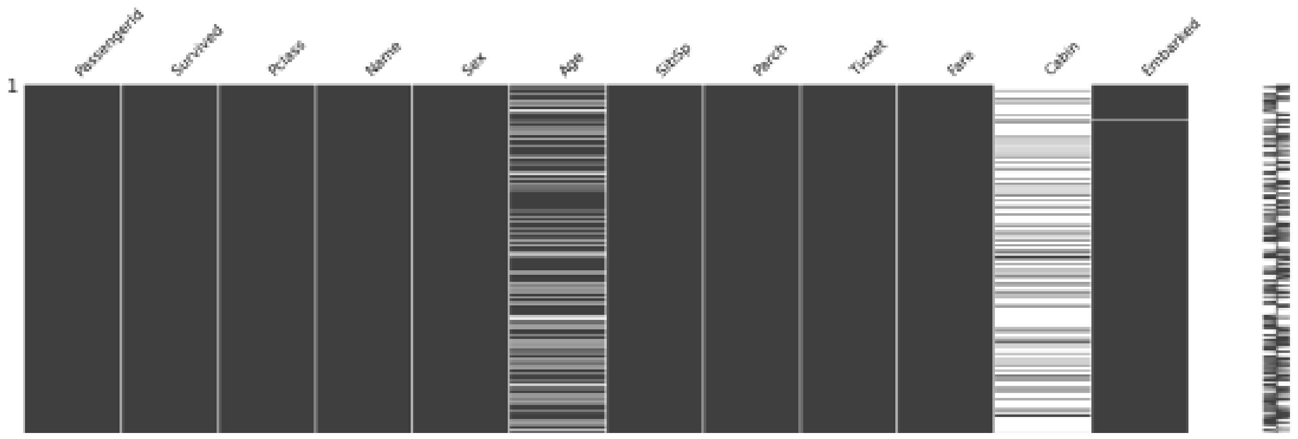
The real-world data often has a lot of missing values. The cause of missing values can be data corruption or failure to record data. The handling of missing data is very important during the preprocessing of the dataset as many machine learning algorithms do not support missing values.

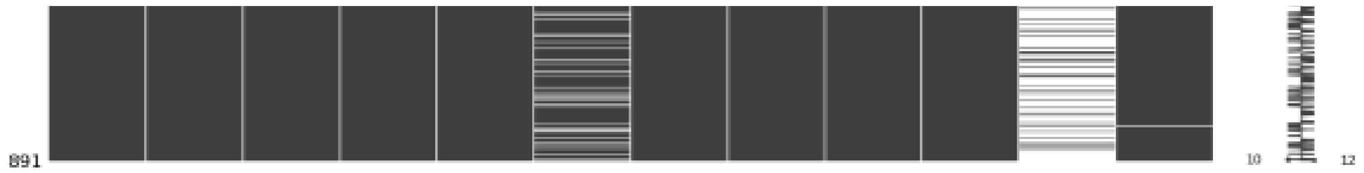
This article covers 7 ways to handle missing values in the dataset:

1. Deleting Rows with missing values
2. Impute missing values for continuous variable
3. Impute missing values for categorical variable
4. Other Imputation Methods
5. Using Algorithms that support missing values
6. Prediction of missing values
7. Imputation using Deep Learning Library — Datawig

> *Data used is Titanic Dataset from Kaggle*

```
data = pd.read_csv("train.csv")
msno.matrix(data)
```

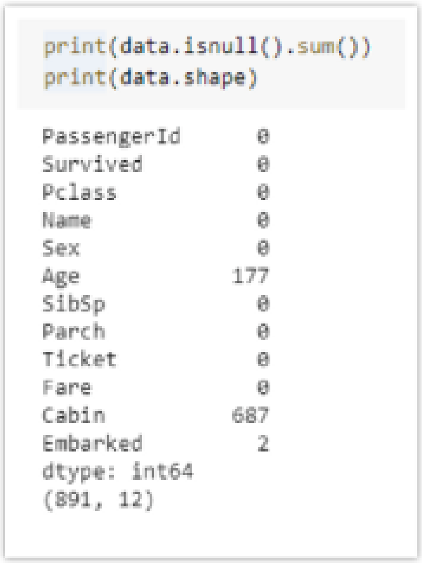




(Image by Author), **Visualization of Missing Values**: white lines denote the presence of missing value

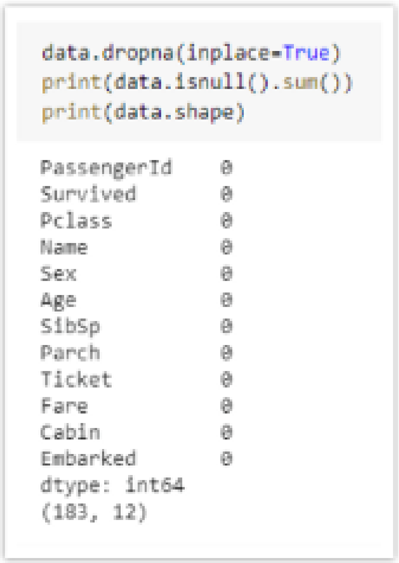
Delete Rows with Missing Values:

Missing values can be handled by deleting the rows or columns having null values. If columns have more than half of rows as null then the entire column can be dropped. The rows which are having one or more columns values as null can also be dropped.



```
print(data.isnull().sum())
print(data.shape)
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
(891, 12)
```



```
data.dropna(inplace=True)
print(data.isnull().sum())
print(data.shape)
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
(183, 12)
```

(Image by Author) **Left:** Data with Null values, **Right:** Data after removal of Null values

Pros:

- A model trained with the removal of all missing values creates a robust model.

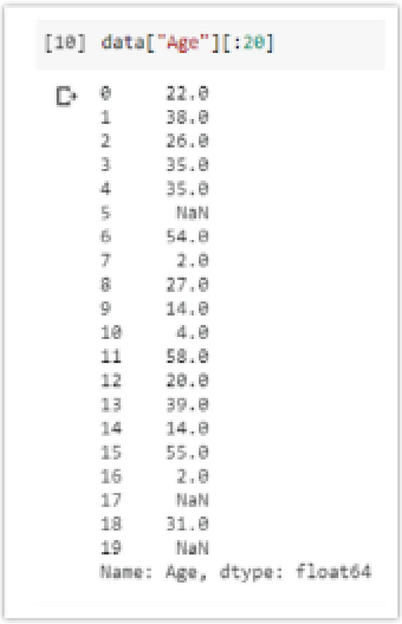
Cons:

- Loss of a lot of information.
- Works poorly if the percentage of missing values is excessive in comparison to the complete dataset.

. . .

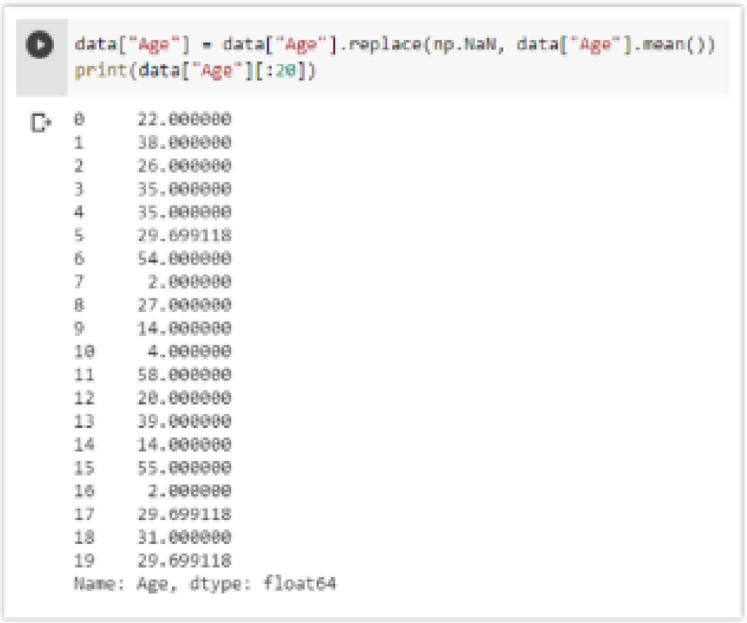
Impute missing values with Mean/Median:

Columns in the dataset which are having numeric continuous values can be replaced with the mean, median, or mode of remaining values in the column. This method can prevent the loss of data compared to the earlier method. Replacing the above two approximations (mean, median) is a statistical approach to handle the missing values.



```
[10] data["Age"][:20]

0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
5       NaN
6      54.0
7       2.0
8      27.0
9      14.0
10      4.0
11     58.0
12     20.0
13     39.0
14     14.0
15     55.0
16      2.0
17      NaN
18     31.0
19      NaN
Name: Age, dtype: float64
```



```
data["Age"] = data["Age"].replace(np.NaN, data["Age"].mean())
print(data["Age"][:20])

0     22.000000
1     38.000000
2     26.000000
3     35.000000
4     35.000000
5     29.699118
6     54.000000
7      2.000000
8     27.000000
9     14.000000
10     4.000000
11    58.000000
12    20.000000
13    39.000000
14    14.000000
15    55.000000
16     2.000000
17    29.699118
18    31.000000
19    29.699118
Name: Age, dtype: float64
```

(Image by Author) **Left:** Age column before Imputation, **Right:** Age column after imputation by the mean value

The missing values are replaced by the mean value in the above example, in the same way, it can be replaced by the median value.

```python
data["Age"] = data["Age"].replace(np.NaN, data["Age"].mean())
data["Age"] = data["Age"].replace(np.NaN, data["Age"].median())
```

impute_mean_median.py hosted with ❤ by **GitHub** view raw

Pros:

- Prevent data loss which results in deletion of rows or columns
- Works well with a small dataset and easy to implement.

Cons:

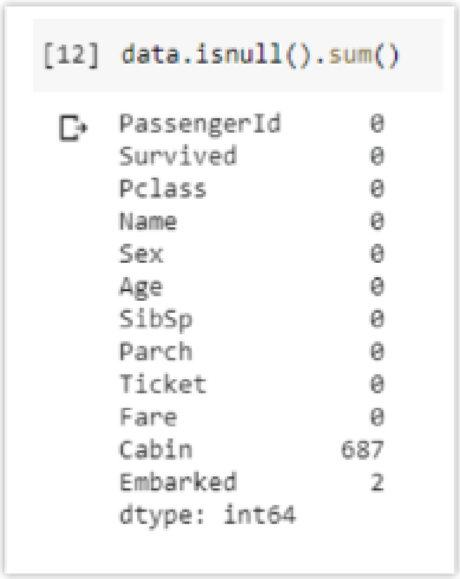
- Works only with numerical continuous variables.

- Can cause data leakage
- Does not factor the covariance between features.

. . .

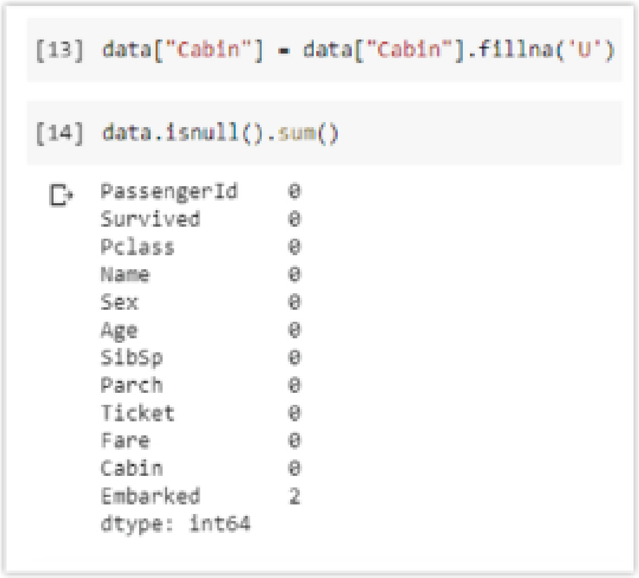
Imputation method for categorical columns:

When missing values is from categorical columns (string or numerical) then the missing values can be replaced with the most frequent category. If the number of missing values is very large then it can be replaced with a new category.



```
[12] data.isnull().sum()

PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```



```
[13] data["Cabin"] = data["Cabin"].fillna('U')

[14] data.isnull().sum()

PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       2
dtype: int64
```

(Image by Author) **Left:** Data before Imputation, **Right:** Cabin column after imputation by 'U'

Pros:

- Prevent data loss which results in deletion of rows or columns
- Works well with a small dataset and easy to implement.
- Negates the loss of data by adding a unique category

Cons:

- Works only with categorical variables.

- Addition of new features to the model while encoding, which may result in poor performance

. . .

Other Imputation Methods:

Depending on the nature of the data or data type, some other imputation methods may be more appropriate to impute missing values.

For example, for the data variable having longitudinal behavior, it might make sense to use the last valid observation to fill the missing value. This is known as the Last observation carried forward (LOCF) method.

For the time-series dataset variable, it makes sense to use the interpolation of the variable before and after a timestamp for a missing value.

. . .

Using Algorithms that support missing values:

All the machine learning algorithms don't support missing values but some ML algorithms are robust to missing values in the dataset. The k-NN algorithm can ignore a column from a distance measure when a value is missing. Naive Bayes can also support missing values when making a prediction. These algorithms can be used when the dataset contains null or missing values.

The sklearn implementations of naive Bayes and k-Nearest Neighbors in Python does not support the presence of the missing values.

Another algorithm that can be used here is RandomForest that works well on non-linear and the categorical data. It adapts to the data structure taking into consideration the high variance or the bias, producing better results on large datasets.

Pros:

- No need to handle missing values in each column as ML algorithms will handle it efficiently

Cons:

- No implementation of these ML algorithms in the scikit-learn library.

. . .

Prediction of missing values:

In the earlier methods to handle missing values, we do not use correlation advantage of the variable containing the missing value and other variables. Using the other features which don't have nulls can be used to predict missing values.

The regression or classification model can be used for the prediction of missing values depending on nature (categorical or continuous) of the feature having missing value.

```
Here 'Age' column contains missing values so for prediction of null
values the spliting of data will be,

y_train: rows from data["Age"] with non null values
y_test: rows from data["Age"] with null values
X_train: Dataset except data["Age"] features with non null values
X_test: Dataset except data["Age"] features with null values
```

(Code by Author)

Predict Missing Values in the Dataset

Understand how to predict missing values in the dataset using a Machine Learning model and its Implementation

towardsdatascience.com

Pros:

- Gives a better result than earlier methods
- Takes into account the covariance between missing value column and other columns.

Cons:

- Considered only as a proxy for the true values

. . .

Imputation using Deep Learning Library — Datawig

This method works very well with categorical, continuous, and non-numerical features. Datawig is a library that learns ML models using Deep Neural Networks to impute missing values in the datagram.

```
Install datawig library,
pip3 install datawig
```

Datawig can take a data frame and fit an imputation model for each column with missing values, with all other columns as inputs.

Below is the code to impute missing values in the *Age* column

(Code by Author)

Pros:

- Quite accurate compared to other methods.
- It supports CPUs and GPUs.

Cons:

- Can be quite slow with large datasets.

• • •

Conclusion:

Every dataset has missing values that need to be handled intelligently to create a robust model. In this article, I have discussed 7 ways to handle missing values that can handle missing values in every type of column. There is no thump rule to handle missing values in a particular manner, the method which gets a robust model with the best performance. One can use various methods on different features depending on how and what the data is about. Having a domain knowledge about the dataset is important, which can give an insight into how to preprocess the data and handle missing values.

References:

[1] Datawig: https://github.com/awslabs/datawig

Thank You for Reading

Machine Learning Data Science Artificial Intelligence Handling Missing Values Towards Data Science