1. Data Preprocessing

 Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

## Criteria for Data Quality

* Accuracy: A simple question we can ask to verify accuracy is - is the information contained in the data provisioned correctly in every detail? With data, we are trying to approximate patterns in the real world; accuracy helps define how true these approximations are.
* Relevance: Is this information impactful to what our desired end goal is?
* Consistency: Is the source of this data verified and can the information contained there be trusted? For example, having a customer who is age 15 but marital status is indicated to be married.
* Completeness: Is the data provisioned and comprehensive enough for the desired end goal? This is a common problem associated with missing data and one way to resolve this is by investigating the data source.
* Uniformity: Let's assume we have a sales column but not all transactions are made in the same currency. It becomes imperative to convert all the sales values into a single currency and that is what uniformity is about; ensuring that data measures are specified using the same units of measure in all systems.

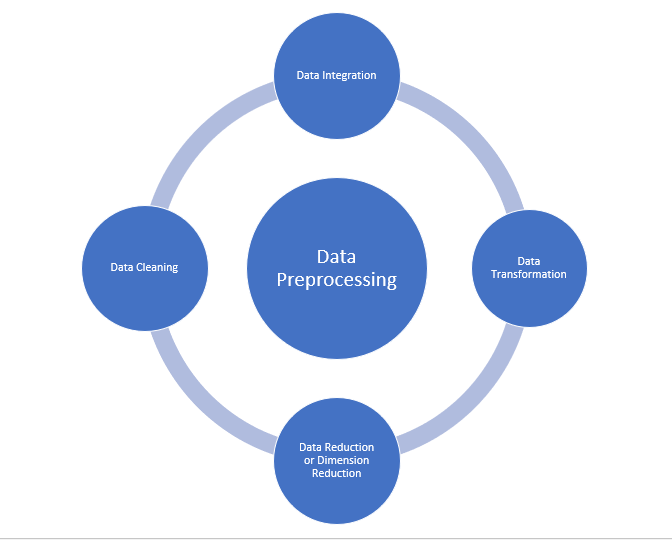
## Why is Data preprocessing important?

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following

* **Accuracy**: To check whether the data entered is correct or not.
* **Completeness**: To check whether the data is available or not recorded.
* **Consistency:** To check whether the same data is kept in all the places that do or do not match.
* **Timeliness**: The data should be updated correctly.
* **Believability**: The data should be trustable.
* **Interpretability**: The understandability of the data.

## Major Tasks in Data Preprocessing:

1. Data cleaning
2. Data integration
3. Data reduction
4. Data transformation



### Data cleaning:

Data cleaning is the process to remove incorrect data, incomplete data and inaccurate data from the datasets, and it also replaces the missing values. There are some techniques in data cleaning.

### **Handling missing values:**

* Standard values like “Not Available” or “NA” can be used to replace the missing values.
* Missing values can also be filled manually but it is not recommended when that dataset is big.
* The attribute’s mean value can be used to replace the missing value when the data is normally distributed  
  wherein in the case of non-normal distribution median value of the attribute can be used.
* While using regression or decision tree algorithms the missing value can be replaced by the most probable  
  value.

### **Noisy:**

          Noisy generally means random error or containing unnecessary data points. Here are some of the methods to handle noisy data.

* **Binning**: This method is to smooth or handle noisy data. First, the data is sorted then and then the sorted values are separated and stored in the form of bins. There are three methods for smoothing data in the bin. **Smoothing by bin mean method**: In this method, the values in the bin are replaced by the mean value of the bin; **Smoothing by bin median**: In this method, the values in the bin are replaced by the median value; **Smoothing by bin boundary**: In this method, the using minimum and maximum values of the bin values are taken and the values are replaced by the closest boundary value.
* **Regression**: This is used to smooth the data and will help to handle data when unnecessary data is present. For the analysis, purpose regression helps to decide the variable which is suitable for our analysis.
* **Clustering**: This is used for finding the outliers and also in grouping the data. Clustering is generally used in unsupervised learning.

### **Data integration:**

          The process of combining multiple sources into a single dataset. The Data integration process is one of the main components in data management. There are some problems to be considered during data integration.

* **Schema integration**: Integrates metadata(a set of data that describes other data) from different sources.
* **Entity identification problem:** Identifying entities from multiple databases. For example, the system or the use should know student \_id of one database and student\_name of another database belongs to the same entity.
* **Detecting and resolving data value concepts**: The data taken from different databases while merging  may differ. Like the attribute values from one database may differ from another database. For example, the date format may differ like “MM/DD/YYYY” or “DD/MM/YYYY”.

### Data reduction:

         This process helps in the reduction of the volume of the data which makes the analysis easier yet produces the same or almost the same result. This reduction also helps to reduce storage space. There are some of the techniques in data reduction are Dimensionality reduction, Numerosity reduction, Data compression.

* **Dimensionality reduction:**This process is necessary for real-world applications as the data size is big. In this process, the reduction of random variables or attributes is done so that the dimensionality of the data set can be reduced. Combining and merging the attributes of the data without losing its original characteristics. This also helps in the reduction of storage space and computation time is reduced. When the data is highly dimensional the problem called “Curse of Dimensionality” occurs.
* **Numerosity Reduction:**In this method, the representation of the data is made smaller by reducing the volume. There will not be any loss of data in this reduction.
* **Data compression:**The compressed form of data is called data compression. This compression can be lossless or lossy. When there is no loss of information during compression it is called lossless compression. Whereas lossy compression reduces information but it removes only the unnecessary information.

### Data Transformation**:**

       The change made in the format or the structure of the data is called data transformation. This step can be simple or complex based on the requirements. There are some methods in data transformation.

* **Smoothing**: With the help of algorithms, we can remove noise from the dataset and helps in knowing the important features of the dataset. By smoothing we can find even a simple change that helps in prediction.
* **Aggregation**: In this method, the data is stored and presented in the form of a summary. The data set which is from multiple sources is integrated into with data analysis description. This is an important step since the accuracy of the data depends on the quantity and quality of the data. When the quality and the quantity of the data are good the results are more relevant.
* **Discretization**: The continuous data here is split into intervals. Discretization reduces the data size. For example, rather than specifying the class time, we can set an interval like (3 pm-5 pm, 6 pm-8 pm).
* **Normalization:** It is the method of scaling the data so that it can be represented in a smaller range. Example ranging from -1.0 to 1.0.

Data Cleaning

* Data cleaning is a very basic building block of data science. Data science and analytics is garbage in, garbage out. This means that no matter how sophisticated our analytics or predictive algorithms are, the quality of output is dependent on the data input. Since data underpins all of these processes, it is important to spend an ample amount of time ensuring data is properly refined. When neglected, the result of it is costly, erroneous analytical results, both in terms of time and money, as well as other committed resources. Root causes:
* Human error during data entry, recording, or encoding
* A faulty sensory device as is the case in the internet of things (IoT)
* hardware or software damage
* Data is collected from various structured and unstructured sources and then combined, leading to duplicated and mislabelled values.
* Different data dictionary definitions for data stored at various locations.
* Manual entry error/Typos.
* Incorrect capitalization.
* Mislabelled categories/classes.

Types of data cleaning:

Inconsistent Records

A typical example of inconsistent records is a form where individuals are required to enter their gender. We can have different formats representing the same thing e.g 'M', 'm', 'Male', or 'MALE', and did we sell 'aples', 'apples', or 'APPLES' this month. The best way to spot them can be via a frequency chart or making a distinct display of all values in the column. Some operations on inconsistent records will include but are not limited to:

* Converting strings to lower or proper case
* Removing white spaces
* Renaming column names

Data Types and Type Conversion

The data type is simply how data is represented, which tells the compiler how the data is to be used. How data is represented determines the kind of operations possible with such data

**Removing Null/Duplicate Records**

If in a particular row a significant amount of data is missing, then it would be better to drop that row as it would not be adding any value to our model. you can impute the value; provide an appropriate substitute for the missing data. Also always remember to delete duplicate/ redundant values from your dataset as they might result in a bias in your model.

For example, let us consider the student dataset with the following records.

| **name** | **score** | **address** | **height** | **weight** |
| --- | --- | --- | --- | --- |
| A | 56 | Goa | 165 | 56 |
| B | 45 | Mumbai | 3 | 65 |
| C | 87 | Delhi | 170 | 58 |
| D |  |  |  |  |
| E | 99 | Mysore | 167 | 60 |

As we see that corresponding to student name “D”, most of the data is missing hence we drop that particular row.

*student\_df.dropna()* # drops rows with 1 or more Nan value

#output

| **name** | **score** | **address** | **height** | **weight** |
| --- | --- | --- | --- | --- |
| A | 56 | Goa | 165 | 56 |
| B | 45 | Mumbai | 3 | 65 |
| C | 87 | Delhi | 170 | 58 |
| E | 99 | Mysore | 167 | 60 |

**Dropping unnecessary Columns**

When we receive the data from stakeholders, generally it is huge. There can be a log of data that might not add any value to our model. Such data is better removed as it would valuable resources like memory and processing time.

For example, while looking at students’ performance over a test, students’ weight or their height does not have anything to contribute to the model.

*student\_df.drop(['height','weight'], axis = 1,inplace=True)* #Drops Height column form the dataframe

#Output

| **name** | **score** | **address** |
| --- | --- | --- |
| A | 56 | Goa |
| B | 45 | Mumbai |
| C | 87 | Delhi |
| E | 99 | mysore |

**Renaming columns**

It’s always better to rename the columns and format them to the most readable format which can be understood by both the data scientist and the business. For example in the student data set, renaming the column “name” as “Sudent\_Name” makes it meaningful.

*student\_df.rename(columns={'name': 'Student\_Name'}, inplace=True)* #renames name column to Student\_Name

#Output

| **Student\_Name** | **score** | **address** |
| --- | --- | --- |
| A | 56 | Goa |
| B | 45 | Mumbai |
| C | 87 | Delhi |
| E | 99 | Mysore |
|  |  |  |

## Treating missing values

The problem of missing value is quite common in many real-life datasets. Missing value can bias the results of the machine learning models and/or reduce the accuracy of the model. Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset. Below is a sample of the missing data from the Titanic dataset. You can see the columns ‘Age’ and ‘Cabin’ have some missing values.

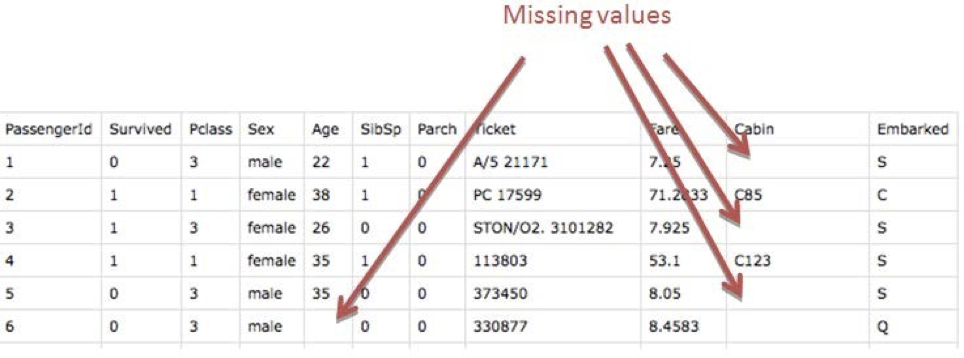
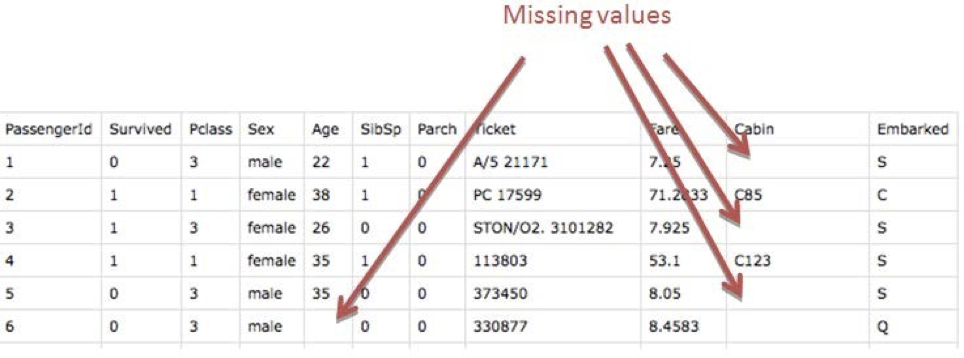


Image 1

How is Missing Value Represented In The Dataset?

In the dataset, blank shows the missing values.In Pandas, usually, missing values are represented by NaN.It stands for Not a Number.The above image shows the first few records of the Titanic dataset extracted and displayed using Pandas.



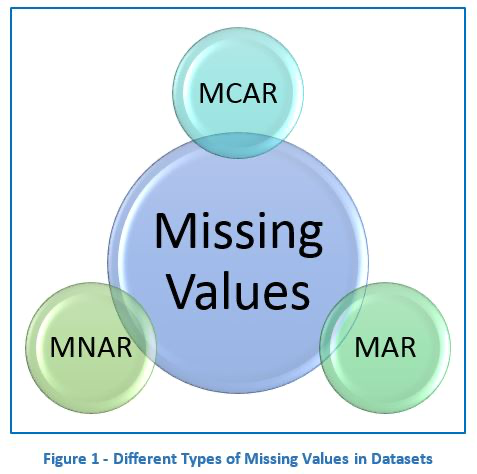
Why Is Data Missing From The Dataset

There can be multiple reasons why certain values are missing from the data.Reasons for the missing data from the dataset affect the approach of handling missing data. So it’s necessary to understand why the data could be missing.Some of the reasons are listed below:

* Past data might get corrupted due to improper maintenance.
* Observations are not recorded for certain fields due to some reasons. There might be a failure in recording the values due to human error.
* The user has not provided the values intentionally.

Types Of Missing Value

Formally the missing values are categorized as follows:



Missing Completely At Random (MCAR)

In MCAR, the probability of data being missing is the same for all the observations.In this case, there is no relationship between the missing data and any other values observed or unobserved (the data which is not recorded) within the given dataset.That is, missing values are completely independent of other data. There is no pattern.In the case of MCAR, the data could be missing due to human error, some system/equipment failure, loss of sample, or some unsatisfactory technicalities while recording the values.For Example, suppose in a library there are some overdue books. Some values of overdue books in the computer system are missing. The reason might be a human error like the librarian forgot to type in the values. So, the missing values of overdue books are not related to any other variable/data in the system.It should not be assumed as it’s a rare case. The advantage of such data is that the statistical analysis remains unbiased.

Missing At Random (MAR)

Missing at random (MAR) means that the reason for missing values can be explained by variables on which you have complete information as there is some relationship between the missing data and other values/data.In this case, the data is not missing for all the observations. It is missing only within sub-samples of the data and there is some pattern in the missing values.For example, if you check the survey data, you may find that all the people have answered their ‘Gender’ but ‘Age’ values are mostly missing for people who have answered their ‘Gender’ as ‘female’. (The reason being most of the females don’t want to reveal their age.)So, the probability of data being missing depends only on the observed data.In this case, the variables ‘Gender’ and ‘Age’ are related and the reason for missing values of the ‘Age’ variable can be explained by the ‘Gender’ variable but you cannot predict the missing value itself.Suppose a poll is taken for overdue books of a library. Gender and the number of overdue books are asked in the poll. Assume that most of the females answer the poll and men are less likely to answer. So why the data is missing can be explained by another factor that is gender.In this case, the statistical analysis might result in bias.Getting an unbiased estimate of the parameters can be done only by modeling the missing data.

Missing Not At Random (MNAR)

Missing values depend on the unobserved data.If there is some structure/pattern in missing data and other observed data can not explain it, then it is Missing Not At Random (MNAR).If the missing data does not fall under the MCAR or MAR then it can be categorized as MNAR.It can happen due to the reluctance of people in providing the required information. A specific group of people may not answer some questions in a survey.For example, suppose the name and the number of overdue books are asked in the poll for a library. So most of the people having no overdue books are likely to answer the poll. People having more overdue books are less likely to answer the poll.So in this case, the missing value of the number of overdue books depends on the people who have more books overdue.Another example, people having less income may refuse to share that information in a survey.In the case of MNAR as well the statistical analysis might result in bias.

Why Do We Need To Care About Handling Missing Value?

* It is important to handle the missing values appropriately.
* Many machine learning algorithms fail if the dataset contains missing values. However, algorithms like K-nearest and Naive Bayes support data with missing values.
* You may end up building a biased machine learning model which will lead to incorrect results if the missing values are not handled properly.
* Missing data can lead to a lack of precision in the statistical analysis.

How To Handle Missing Value?

Checking for missing values

The first step in handling missing values is to look at the data carefully and find out all the missing values.The following code shows the total number of missing values in each column.It also shows the total number of missing values in entire data set.

**IN:**

import pandas as pd

train\_df = pd.read\_csv("train.csv")

#Find the missing values from each column

train\_df.isnull().sum()

**OUT:**

Loan\_ID 0

Gender 13

Married 3

Dependents 15

Education 0

Self\_Employed 32

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 22

Loan\_Amount\_Term 14

Credit\_History 50

Property\_Area 0

Loan\_Status 0

dtype: int64

From the above output, we can see that there are 6 columns – Gender, Married, Dependents, Self\_Employed, LoanAmount, Loan\_Amount\_Term and Credit\_History having missing values.

**IN:**

#Find the total number of missing values from the entire dataset

train\_df.isnull().sum().sum()

**OUT:**

149

There are 149 missing values in total.

How To Handle The Missing Data

Analyze each column with missing values carefully to understand the reasons behind the missing values as it is crucial to find out the strategy for handling the missing values.

There are 2 primary ways of handling missing values:

* Deleting the Missing values
* Imputing the Missing Values

Deleting the missing values:

Generally, this approach is not recommended. It is one of the quick and dirty techniques one can use to deal with missing values. If the missing value is of the type Missing Not At Random (MNAR), then it should not be deleted.

If the missing value is of type Missing At Random (MAR) or Missing Completely At Random (MCAR) then it can be deleted.The disadvantage of this method is one might end up deleting some useful data from the dataset.There are 2 ways one can delete the missing values:

Deleting the entire row:If a row has many missing values then you can choose to drop the entire row.If every row has some (column) value missing then you might end up deleting the whole data.

Code to drop the entire row is as follows:

IN:

df = train\_df.dropna(axis=0)

df.isnull().sum()

**OUT:**

Loan\_ID 0

Gender 0

Married 0

Dependents 0

Education 0

Self\_Employed 0

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 0

Loan\_Amount\_Term 0

Credit\_History 0

Property\_Area 0

Loan\_Status 0

dtype: int64

Deleting the entire column:If a certain column has many missing values then you can choose to drop the entire column.Code to drop the entire column is as follows:

**IN:**

df = train\_df.drop(['Dependents'],axis=1)

df.isnull().sum()

**OUT:**

Loan\_ID 0

Gender 13

Married 3

Education 0

Self\_Employed 32

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 22

Loan\_Amount\_Term 14

Credit\_History 50

Property\_Area 0

Loan\_Status 0

dtype: int64

Imputing the Missing Value

Replacing With Arbitrary Value:If you can make an educated guess about the missing value then you can replace it with some arbitrary value using the following code.Ex. In the following code, we are replacing the missing values of the ‘Dependents’ column with ‘0’.

IN:

#Replace the missing value with '0' using 'fiilna' method

train\_df['Dependents'] = train\_df['Dependents'].fillna(0)

train\_df[‘Dependents'].isnull().sum()

OUT:

0

Replacing With Mean: This is the most common method of imputing missing values of numeric columns. If there are outliers then the mean will not be appropriate. In such cases, outliers need to be treated first. You can use the ‘fillna’ method for imputing the columns ‘LoanAmount’ and ‘Credit\_History’ with the mean of the respective column values.

**IN:**

#Replace the missing values for numerical columns with mean

train\_df['LoanAmount'] = train\_df['LoanAmount'].fillna(train\_df['LoanAmount'].mean())

train\_df['Credit\_History'] = train\_df[‘Credit\_History'].fillna(train\_df['Credit\_History'].mean())

**OUT:**

Loan\_ID 0

Gender 13

Married 3

Dependents 15

Education 0

Self\_Employed 32

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 0

Loan\_Amount\_Term 0

Credit\_History 0

Property\_Area 0

Loan\_Status 0

dtype: int64

**Replacing With Mode**

Mode is the most frequently occurring value. It is used in the case of categorical features.You can use the ‘fillna’ method for imputing the categorical columns ‘Gender’, ‘Married’, and ‘Self\_Employed’.

Replacing With Median

Median is the middlemost value. It’s better to use the median value for imputation in the case of outliers. You can use ‘fillna’ method for imputing the column ‘Loan\_Amount\_Term’ with the median value.

train\_df['Loan\_Amount\_Term']= train\_df['Loan\_Amount\_Term'].fillna(train\_df['Loan\_Amount\_Term'].median())

Replacing with previous value – Forward fill

IN:

import pandas as pd

import numpy as np

test = pd.Series(range(6))

test.loc[2:4] = np.nan

test

OUT:

0 0.0

1 1.0

2 Nan

3 Nan

4 Nan

5 5.0

dtype: float64

IN:

# Forward-Fill

test.fillna(method=‘ffill')

OUT:

0 0.0

1 1.0

2 1.0

3 1.0

4 1.0

5 5.0

dtype: float64

In some cases, imputing the values with the previous value instead of mean, mode or median is more appropriate. This is called forward fill. It is mostly used in time series data.

Replacing with next value – Backward fillIn backward fill, the missing value is imputed using the next value.

**IN:**

# Backward-Fill

test.fillna(method=‘bfill')

**OUT:**

0 0.0

1 1.0

2 5.0

3 5.0

4 5.0

5 5.0

dtype: float64

IN:

Interpolation

Missing values can also be imputed using interpolation. Pandas interpolate method can be used to replace the missing values with different interpolation methods like ‘polynomial’, ‘linear’, ‘quadratic’. Default method is ‘linear’.

**IN:**

test.interpolate()

**OUT:**

0 0.0

1 1.0

2 2.0

3 3.0

4 4.0

5 5.0

dtype: float64

Imputing Missing Values For Categorical Features

There are two ways to impute missing values for categorical features as follows:

Impute the Most Frequent Value

We will make use of ‘SimpleImputer’ in this case and as this is a non-numeric column we can’t use mean or median but we can use most frequent value and constant.

IN:

import pandas as pd

import numpy as np

X = pd.DataFrame({'Shape':['square', 'square', 'oval', 'circle', np.nan]})

X

Shape

OUT:

0 square

1 square

2 oval

3 circle

4 NaN

IN:

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='most\_frequent')

imputer.fit\_transform(X)

OUT:

array([['square'],

['square'],

['oval'],

['circle'],

['square']], dtype=object)

As you can see, the missing value is imputed with the most frequent value ’square’.

In any of the above approaches, you will still need to OneHotEncode the data (or you can also use some other encoder of your choice). After One Hot Encoding, in case 1, instead of the values ‘square’, ‘oval’,’ circle’, you will get three feature columns. And in case 2, you will get four feature columns (4th one for the ‘missing’ category). So it’s like adding the missing indicator column in the data. There is another way to add a missing indicator column, which we will discuss further.

Imputation of Missing Value Using sci-kit learn Library

Univariate Approach

In a Univariate approach, only a single feature is taken into consideration. You can use the class SimpleImputer and replace the missing values with mean, mode, median or some constant value.

Let’s see an example:

IN:

import numpy as np

from sklearn.impute import SimpleImputer

imp = SimpleImputer(missing\_values=np.nan, strategy='mean')

imp.fit([[1, 2], [np.nan, 3], [7, 6]])

OUT: SimpleImputer()

IN:

X = [[np.nan, 2], [6, np.nan], [7, 6]]

print(imp.transform(X))

OUT:

[[4. 2. ]

[6. 3.666...]

[7. 6. ]]

Multivariate Approach

In a multivariate approach, more than one feature is taken into consideration. There are two ways to impute missing values considering the multivariate approach. Using KNNImputer or IterativeImputer classes.

Let’s take an example of a titanic dataset.Suppose the feature ‘age’ is well correlated with the feature ‘Fare’ such that people with lower fares are also younger and people with higher fares are also older.In that case, it would make sense to impute low age for low fare values and high age for high fares values. So here we are taking multiple features into account by following a multivariate approach.

Let’s see how IterativeImputer works. For all rows, in which ‘Age’ is not missing sci-kit learn runs a regression model. It uses ‘Sib sp’ and ‘Fare’ as the features and ‘Age’ as the target. And then for all rows for which ‘Age’ is missing, it makes predictions for ‘Age’ by passing ‘Sib sp’ and ‘Fare’ to the training model. So it actually builds a regression model with two features and one target and then makes predictions on any places where there are missing values. And those predictions are the imputed values.

IN:

import pandas as pd

df = pd.read\_csv('http://bit.ly/kaggletrain', nrows=6)

cols = ['SibSp', 'Fare', 'Age']

X = df[cols]

X

|  | **SibSp** | **Fare** | **Age** |
| --- | --- | --- | --- |
| **0** | 1 | 7.2500 | 22.0 |
| **1** | 1 | 71.2833 | 38.0 |
| **2** | 0 | 7.9250 | 26.0 |
| **3** | 1 | 53.1000 | 35.0 |
| **4** | 0 | 8.0500 | 35.0 |
| **5** | 0 | 8.4583 | NaN |

IN:

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

impute\_it = IterativeImputer()

impute\_it.fit\_transform(X)

OUT:

array([[ 1. , 7.25 , 22. ],

[ 1. , 71.2833 , 38. ],

[ 0. , 7.925 , 26. ],

[ 1. , 53.1 , 35. ],

[ 0. , 8.05 , 35. ],

[ 0. , 8.4583 , 28.50639495]])

Nearest Neighbors Imputations (KNNImputer):Missing values are imputed using the k-Nearest Neighbors approach where a Euclidean distance is used to find the nearest neighbors.Let’s take the above example of the titanic dataset to see how it works.

IN:

from sklearn.impute import KNNImputer

impute\_knn = KNNImputer(n\_neighbors=2)

impute\_knn.fit\_transform(X)

OUT:

array([[ 1. , 7.25 , 22. ],

[ 1. , 71.2833, 38. ],

[ 0. , 7.925 , 26. ],

[ 1. , 53.1 , 35. ],

[ 0. , 8.05 , 35. ],

[ 0. , 8.4583, 30.5 ]])

In the above example, the n\_neighbors=2. So sci-kit learn finds the two most similar rows measured by how close the ‘Sib sp’ and ‘Fare’ values are to the row which has missing values. In this case, the last row has a missing value. And the third row and the fifth row have the closest values for the other two features. So the average of the ‘Age’ feature from these two rows is taken as the imputed value.

**Adding missing indicator to encode “missingness” as a feature**

In some cases, while imputing missing values, you can preserve information about which values were missing and use that as a feature.Because sometimes there may be a relationship between the reason for missing values (also called the “missingness”) and the target variable you are trying to predict.

Why do we need to do this?

Suppose you are predicting the presence of a disease and you can imagine a scenario in which a missing age is a good predictor of a disease because assume that we don’t have records for people in poverty. The age values are not missing at random. They are missing for people in poverty and poverty is a good predictor of disease. Thus, missing age or “missingness” is a good predictor of disease.

IN:

import pandas as pd

import numpy as np

X = pd.DataFrame({'Age':[20, 30, 10, np.nan, 10]})

X

|  | **Age** |
| --- | --- |
| **0** | 20.0 |
| **1** | 30.0 |
| **2** | 10.0 |
| **3** | NaN |
| **4** | 10.0 |

IN:

from sklearn.impute

import SimpleImputer

# impute the mean

imputer = SimpleImputer()

imputer.fit\_transform(X)

OUT:

array([[20. ],

[30. ],

[10. ],

[17.5],

[10. ]])

IN:

imputer = SimpleImputer(add\_indicator=True)

imputer.fit\_transform(X)

OUT:

array([[20. , 0. ],

[30. , 0. ],

[10. , 0. ],

[17.5, 1. ],

[10. , 0. ]])

In the above example, the second column indicates whether the corresponding value in the first column was missing or not. ‘1’ indicates that the corresponding value was missing and ‘0’ indicates that the corresponding value was not missing.If you don’t want to impute missing values but only want to have the indicator matrix then you can use the ‘MissingIndicator’ class from scikit learn.

### Handling Duplicates:

Duplicate rows occur usually when the data is combined from multiple sources. It gets replicated sometimes. A common problem is when users have the same identity number or the form has been submitted twice.

The solution to these duplicate tuples is to simply remove them. You can use the unique() function to find out the unique values present in the column and then decide which values need to be scraped.