**Data Cleaning With Pandas**

* Step 1: Import Dataset. To import the dataset, we use the read\_csv() function of pandas and store it in the pandas DataFrame named as data. ...
* Step 2: Merge Dataset. ...
* Step 3: Rebuild Missing Data. ...
* Step 4: Standardization and Normalization. ...
* Step 5: De-Duplicate Data.

Data cleaning is a critical task in data science that helps ensure the accuracy and reliability of analysis and decision-making. Through data cleaning, errors can be removed, data quality can be improved, and the data can be made more accurate and complete. By utilizing the various techniques and tools available for data cleaning in the Python Pandas library, data scientists can gain insights from the raw data and make better informed decisions.

**Q1. What do you mean by data type casting in the context of data analysis and data cleaning?**

In the context of data analysis, casting data types means converting data from one type to another. This is often done to ensure consistency and accuracy in data analysis, as well as to enable specific operations or functions that are available for certain data types. For example, casting a string to a numerical data type can enable mathematical operations, while casting a numerical data type to a string can enable string-based operations.

**Q2.When is it appropriate to drop missing values in data rather than imputing them in the context of data cleaning with Pandas?**

It is appropriate to drop missing values in data when the amount of missing data is small compared to the overall size of the dataset, and the missing data is randomly distributed or when they would skew the analysis. if the amount of missing data is substantial or the missing data is non-random, it may be more appropriate to impute the missing values rather than drop them, as dropping them may result in a biased or incomplete analysis.

**Q3.What are the different data-cleaning techniques in Python?**

In Python, various data cleaning techniques include removing duplicates, handling missing values, and converting data types. Additionally, you can use functions from libraries like pandas to filter out irrelevant information and correct errors in your datasets.

**Q4.Which library is used for data cleaning in Python?**

The primary library used for data cleaning in Python is Pandas. Pandas provide powerful tools and functions to manipulate and clean datasets efficiently, making it a go-to choice for data cleaning tasks in Python.

Let’s get started with data cleaning step by step.

To start working with Pandas, we need to first import it. We are using Google Colab as IDE, so we will import Pandas in Google Colab.

**#importing module**

**import pandas as pd**

**Step 1: Import Dataset**

**We are using a simple dataset for data cleaning, i.e., the iris species dataset. You can download this dataset from**[**kaggle.com**](http://kaggle.com/)[**.**](https://www.kaggle.com/uciml/iris)

To import the dataset, we use the read\_csv() function of pandas and store it in the pandas DataFrame named as data. As the dataset is in tabular format, when working with tabular data in Pandas, it will be automatically converted into a DataFrame. DataFrame is a two-dimensional, mutable data structure in Python. It is a combination of rows and columns like an excel sheet.

**Python Code:**

The head() function is a built-in function in pandas for the dataframe used to display the rows of the dataset. We can specify the number of rows by giving the number within the parenthesis. By default, it displays the first five rows of the dataset. If we want to see the last five rows of the dataset, we use the tail()function of the dataframe like this:

**#importing the dataset by reading the csv file**

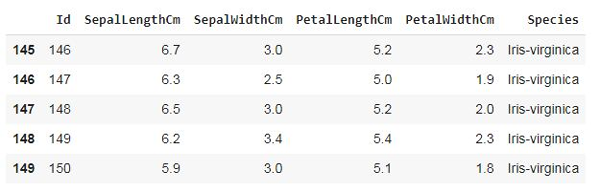
**data = pd.read\_csv('Iris.csv')**

**#displaying the first five rows of dataset**

**print(data.head())**

**#displayinf last five rows of dataset**

**data.tail()**



**Step 2: Merge Dataset**

Merging the dataset is the process of combining two datasets in one and lining up rows based on some particular or common property for data analysis. We can do this by using the merge() function of the dataframe. Following is the syntax of the merge function:

**DataFrame\_name.merge(right, how='inner', on=None, left\_on=None, right\_on=None, left\_index=False, right\_index=False, sort=False, suffixes=('\_x', '\_y'), copy=True, indicator=False, validate=None)**

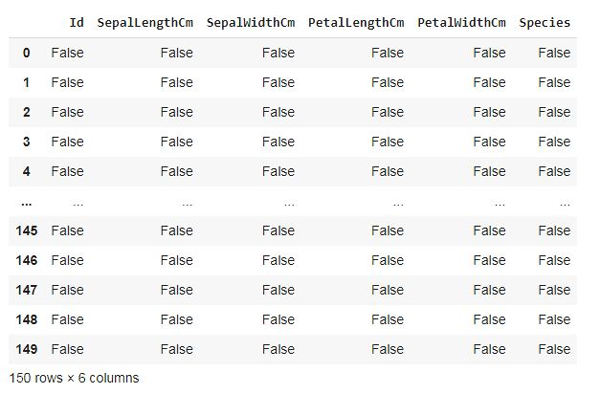
But in this case, we don’t need to merge two datasets. So, we will skip this step.

**Step 3: Rebuild Missing Data**

To find and fill in the missing data in the dataset, we will use another function. There are 4 ways to find the null values if present in the dataset. Let’s see them one by one:

**Using isnull() function:**

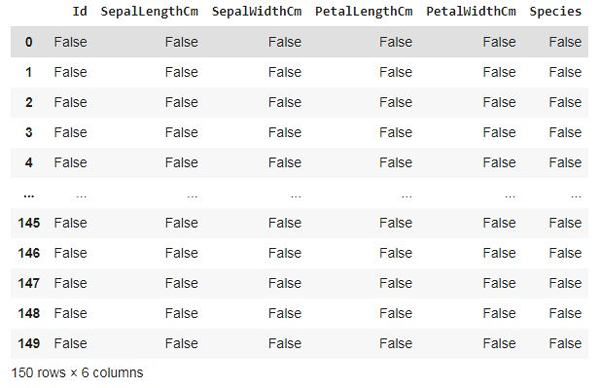
data.isnull()



This function provides the boolean value for the complete dataset to know if any null value is present or not.

**Using isna() function:**

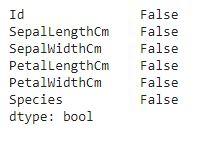
data.isna()



This is the same as the isnull() function. Ans provides the same output.

**Using isna().any()**

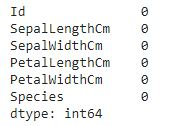
**data.isna().any()**



This function also gives a boolean value if any null value is present or not, but it gives results column-wise, not in tabular format.

**Using isna(). sum()**

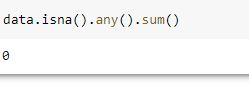
**data.isna().sum()**



This function gives the sum of the null values preset in the dataset column-wise.

**Using isna().any().sum()**

**data.isna().any().sum()**



This function gives output in a single value if any null is present or not.

There are no null values present in our dataset. But if there are any null values preset, we can fill those places with any other value using the fillna() function of DataFrame.Following is the syntax of fillna() function:

**DataFrame\_name.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)**

This function will fill NA/NaN or 0 values in place of null spaces. You may also drop null values using the dropna method when the amount of missing data is relatively small and unlikely to affect the overall.

**Step 4: Standardization and Normalization**

Data Standardization and Normalization is a common practices in machine learning.

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero, and the resultant distribution has a unit standard deviation.

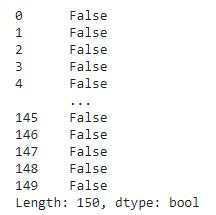
Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

This step is not needed for the dataset we are using. So, we will skip this step.

**Step 5: De-Duplicate Data**

De-Duplicate means removing all duplicate values. There is no need for duplicate values in data analysis. These values only affect the accuracy and efficiency of the analysis result. To find duplicate values in the dataset, we will use a simple dataframe function, i.e., duplicated(). Let’s see the example:

data.duplicated()



This function also provides bool values for duplicate values in the dataset. As we can see, the dataset doesn’t contain any duplicate values. If a dataset contains duplicate values, it can be removed using the drop\_duplicates() function. Following is the syntax of this function:

**DataFrame\_name.drop\_duplicates(subset=None, keep='first', inplace=False, ignore\_index=False)**

Step 6: Verify and Enrich the Data

After removing null, duplicate, and incorrect values, we should verify the dataset and validate its accuracy. In this step, we have to check that the data cleaned so far is making any sense. If the data is incomplete, we have to enrich the data again by data gathering activities like approaching the clients again, re-interviewing people, etc. Completeness is a little more challenging to achieve accuracy or quality in the dataset.

Step 7: Export Dataset

This is the last step of the data-cleaning process. After performing all the above operations, the data is transformed into a clean dataset, and it is ready to export for the next process in Data Science or Data Analysis.