Data Management in Python –

Handling Missing Values

Detecting, Excluding and Imputing NA's

Contents

Introduction

- Missing values in data is a common phenomenon in real world problems and can create problems for simple and complicated analysis.
- You need to know the mechanism of missingness and how to treat them is a requirement to reduce the bias and to produce powerful models.
- Let's get familiar with the mechanisms of missingness and explore various options of how to deal with them.

Missing Data Mechanism

Three types of missing data:

- Missing Completely at Random (MCAR):
  MCAR happens when missingness is totally unrelated to the variables in the dataset. For instance, if your equipment just flips out sometimes for no reason and doesn't record stuff, that will result in missing data that is MCAR.
- Missing at Random (MAR):
  MAR happens when the missingness is related to the information in your study. Other variables (but not the variable that is missing itself) in the dataset can be used to predict missingness. For instance, if men are more likely to tell you their weight than women, weight is MAR.
- Missing not at Random (MNAR):
  MNAR happens when missingness is related to missing data in your dataset. For instance, single people are less likely to report martial status than married people.

Missing Data Mechanism

- MNAR is ‘non ignorable’ because we have to include some model for why the data are missing and what the likely values are as we deal with the missing data.
- MCAR and MAR are both considered ‘ignorable’ because we don’t have to include any information about the missing data itself when we deal with the missing data.

Let’s go ahead with testing and dealing missing data

Data Snapshot

basicsalary data consist salary of each employee with it's Location & Grade. The data has 12 rows and 6 columns with 2 missing values.

Variables

First_Name	Last_Name	Grade	Location	ba	ms
Alan	Brown	GR1	DELHI	17990	16070
Agatha	Williams	GR2	MUMBAI	12390	6630

Observations

Columns	Description	Type	Measurement	Possible values
First_Name	First Name	character	-	-
Last_Name	Last Name	character	-	-
Grade	Grade	character	GR1, GR2	2
Location	Location	character	DELHI, MUMBAI	2
ba	Basic Allowance	numeric	Rs.	positive values
ms	Management Supplements	numeric	Rs.	positive values

Replacing Missing Values with NA while Importing the Data

A missing value is one whose value is unknown. Missing values in Python appears as NaN. NaN is not a string or a numeric value, but an indicator of missingness. Our data has two missing values, let's see what happens when we import this data in Python.

```
# Import Data and check how Python treats missing data while importing
import pandas as pd
salary_data = pd.read_csv("basic_salary.csv")
```

```
# Output
```

```
   First_Name Last_Name Grade Location     ba       ms
0        Alan     Brown   GR1    DELHI  17990  16070.0
1      Agatha  Williams   GR2   MUMBAI  12390   6630.0
2      Rajesh     Kolte   GR1   MUMBAI  19250  14960.0
3       Ameet    Mishra   GR2    DELHI  14780   9300.0
4        Neha       Rao   NaN   MUMBAI  19235  15200.0
5       Sagar    Chavan   GR2   MUMBAI  13390   6700.0
6       Aaron     Jones   GR1   MUMBAI  23280  13490.0
7        John     Patil   GR2   MUMBAI  13500  10760.0
8      Sneha      Joshi   GR1    DELHI  20660      NaN
9      Gaurav     Singh   GR2    DELHI  13760  13220.0
10      Adela    Thomas   GR2    DELHI  13660   6840.0
11       Anup      Save   GR2   MUMBAI  11960   7880.0
```

Note **read_csv()** replaces blank fields with NaN

Detecting NA's

```
# Check whether our data has missing values or not
salary_data.isnull()
```

```
   First_Name  Last_Name  Grade  Location     ba     ms
0       False      False  False     False  False  False
1       False      False  False     False  False  False
2       False      False  False     False  False  False
3       False      False  False     False  False  False
4       False      False   True     False  False  False
5       False      False  False     False  False  False
6       False      False  False     False  False  False
7       False      False  False     False  False  False
8       False      False  False     False  False   True
9       False      False  False     False  False  False
10      False      False  False     False  False  False
11      False      False  False     False  False  False
```

isnull() returns logical matrix with the same dimensions as the data frame.

```
# Check total missing values
salary_data.isnull().sum()
```

```
First_Name    0
Last_Name     0
Grade         0
Location      0
ba            0
ms            1
```

- **isnull()** for dataframe returns a logical matrix with the same dimensions as the data frame, and with dimnames taken from the row and column names of the data frame.
- **sum()** returns the total no. of missing values in the data by column.

Detecting NA's

```
# Check Number of missing data per column
salary_data.info()

# Output
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 6 columns):
First_Name     12 non-null object
Last_Name      12 non-null object
Grade          11 non-null object
Location       12 non-null object
ba             12 non-null int64
ms             11 non-null float64
dtypes: float64(1), int64(1), object(4)
memory usage: 656.0+ bytes
```

Using **info()** we can check how many NaN's our data contains.

Excluding Missing Values from Analysis

- Missing data is problematic because most statistical procedures require a value for each variable. When the data is incomplete, you have to decide how to deal with it.
- When Python encounters missing value, it attempts to perform the requested procedure and returns a missing (**NaN**) value as a result. One way of dealing with

```
import numpy as np
from statistics import *
x = [10,30,12,np.nan, 9]
mean(x)
```

```
Nan
```

This output can be interpreted as: our vector contains missing value, so the requested statistic - the mean - is undefined for this data.

```
# We can calculate mean by dropping missing value like in the next example.
# remove missing value
```

```
np.nanmean(x)
```

```
15.25
```

nanmean will remove all **NaN**'s while performing the requested procedure.

Excluding Missing Values from Analysis

Case wise deletion (complete case analysis) is the easiest way to deal with missing data. It simply removes all the cases with missing data anywhere in the data i.e. analysing only the cases with complete data.

```
# Case wise deletion
salary_data.dropna()
# Output
```

	First_Name	Last_Name	Grade	Location	ba	ms
0	Alan	Brown	GR1	DELHI	17990	16070.0
1	Agatha	Williams	GR2	MUMBAI	12390	6630.0
2	Rajesh	Kolte	GR1	MUMBAI	19250	14960.0
3	Ameet	Mishra	GR2	DELHI	14780	9300.0
5	Sagar	Chavan	GR2	MUMBAI	13390	6700.0
6	Aaron	Jones	GR1	MUMBAI	23280	13490.0
7	John	Patil	GR2	MUMBAI	13500	10760.0
9	Gaurav	Singh	GR2	DELHI	13760	13220.0
10	Adela	Thomas	GR2	DELHI	13660	6840.0
11	Anup	Save	GR2	MUMBAI	11960	7880.0

- **.dropna()** is used for case deletion.
- Here, **.dropna()** removes 2 rows that contains mising values

* Complete case analysis is widely used method for handling missing data, and is a default method in many statistical packages. But it has limitations like it may introduce bias and some useful information will be omitted from analysis.

Data Snapshot

Consumerpreference data consist information about 73 respondents & their preferences about 7 attributes on scale of 1-Least Important to 5-Most Important.

Variables

Observations

Sn	Gender	Color	Weight	Shape	Camera	Ram	Processor	Internet
1	M	3	3	4	3	3	3	3

Columns	Description	Type	Measurement	Possible values
Sn	Serial No. (Index Variable)	Character		-
Gender	Gender	Categorical	M;F	2
Color	Color	Categorical	1 to 5	5
Weight	Weight	Categorical	2 to 5	5
Shape	Shape	Categorical	3 to 5	5
Camera	Camera	Categorical	4 to 5	5
Ram	Ram	Categorical	5 to 5	5
Processor	Processor	Categorical	6 to 5	5
Internet	Internet	Categorical	7 to 5	5

Imputing Missing Values

- If the amount of missing data is very small relative to the size of the data then case wise deletion may be the best strategy in order not to bias the analysis, however deleting available data points deprives the data of some amount of information.
- Basically, we need to decide how we're going to use our missing data, if at all, then either remove cases from our data or impute missing values before wiping out potentially useful data points from our data and proceed with our analysis.
- Single imputation can be done by replacing missing values with the mean / median / mode (or any other procedure) of the other values in the variable.

In this tutorial we will primarily focus on performing single imputation .

> * Mean/ Median / mode imputations are simple but, like complete case analysis, can introduce bias on mean and deviation. Furthermore, they ignore relationship with other variables

Imputing Missing Values

Treating missing values in the variable ‘Processor’

```
# Import the data
# Check number of missing values for each variable
```

```
consumer_pref = pd.read_csv("consumerpreference.csv")
consumer_pref.isnull().sum()
```

```
# Output
Sn           0
Gender       0
Color        0
Weight       1
Shape        0
Camera       1
Ram          0
Processor    1
Internet     0
dtype: int64
```

Our data has 3 missing values.

```
# Treating missing values in variable ‘Proces
# Median imputation
```

Here, we are calculating median of the values in variable ‘Processor’ using **median()** and replacing the **NaN** value with the median value. Now there are no missing values in variable ‘**Processor**’.

```
consumer_pref['Processor'].fillna((consumer_pref['Processor'].median()
), inplace=True)
consumer_pref['Processor'].isnull().sum()
```

```
0
```

Imputing Missing Values

```
# Imputing missing value with forward fill
```

```
forward_fill = consumer_pref.fillna(method='ffill')
forward_fill.head(10)
```

```
# Output
```

	Sn	Gender	Color	Weight	Shape	Camera	Ram	Processor	Internet
0	1	M	3	3.0	4	3.0	3	3.0	3
1	2	M	3	4.0	3	3.0	4	4.0	4
2	3	M	1	2.0	2	5.0	4	4.0	4
3	4	M	1	1.0	2	5.0	5	5.0	5
4	5	F	4	3.0	4	5.0	5	5.0	5
5	6	F	4	3.0	4	4.0	3	3.0	3
6	7	F	4	4.0	4	1.0	2	2.0	2
7	8	F	5	4.0	5	1.0	1	1.0	1
8	9	M	1	1.0	1	$\boxed{1.0}$	4	3.0	4
9	10	M	5	$\boxed{1.0}$	4	3.0	2	3.0	2

- **fillna()** allows to fill the NaN values with the value specified or by predefined methods.
- **method = 'ffilll'** allows to forward fill the NaN values i.e. it fill's the missing value with the previous value.

Imputing Missing Values

```
# Imputing missing value with backward fill
```

```
backward_fill = consumer_pref.fillna(method='bfill')
backward_fill.head(10)
```

```
# Output
```

	Sn	Gender	Color	Weight	Shape	Camera	Ram	Processor	Internet
0	1	M	3	3.0	4	3.0	3	3.0	3
1	2	M	3	4.0	3	3.0	4	4.0	4
2	3	M	1	2.0	2	5.0	4	4.0	4
3	4	M	1	1.0	2	5.0	5	5.0	5
4	5	F	4	3.0	4	5.0	5	5.0	5
5	6	F	4	3.0	4	4.0	3	3.0	3
6	7	F	4	4.0	4	1.0	2	2.0	2
7	8	F	5	4.0	5	1.0	1	1.0	1
8	9	M	1	1.0	1	3.0	4	3.0	4
9	10	M	5	3.0	4	3.0	2	3.0	2

- **fillna()** allows to fill the NaN values with the value specified or by predefined methods.
- **method = ‘bfilll’** allows to backward fill the NaN values i.e. it fill’s the missing value with the next value.

Quick Recap

In this session, we learnt how to deal with different types of missing values. Here is the quick recap:

Replacing missing values	• Blank fields in numeric & character column are replaced with NaN while importing data. • `fillna()` is used to replace missing values with various methods.
Recoding values to missing	• **`isnull():`** returns the logical matrix which indicates which elements are missing. • **`sum():`** using this function we can calculate the count of NA's per column • **`info()`**: check number of NA's per variable
Excluding missing values from analysis	• **`dropna():`** performs case wise deletion
Imputing missing values	• Single imputation can be done by replacing missing values with the mean / median / mode (or any other procedure) of the other values in the variable