

Dealing with Missing Data

In this lesson, how to deal with missing or "NaN" values in pandas is explained.

We'll cover the following ^

- Missing data

Missing data

Until now, we have encountered various scenarios where `NaN` values were obtained and identified. This lesson focuses on removing or replacing these values based on the requirements. The `ffill()` method is already discussed [here](#); it deals with `NaN` values resulting from reindexing. Some more additional functions on how to handle `NaN` values are discussed below.

- `isnull()` : This function returns an object with all instances of `NaN` values marked as `True` and the rest as `False`. This function is also used [here](#) as an additional function for `Series`.

```
import numpy as np
import pandas as pd

null_val = np.nan

srs = pd.Series(['A','B','C',null_val]) # Declaring series with null values

print("Series with NaN values:")
print(srs.isnull()) # Checking for null values

# Declaring DataFrame with null values
df = pd.DataFrame(['A','B','C',null_val],[null_val,'B',null_val,'D'],[null_val,'B','C',null_val])

print("\nDataFrame with NaN values:")
print(df.isnull()) # Checking for null values
```



It can be seen from the output that in both cases of `Series` and `DataFrame` the *indexes* that had `NaN` values are marked as `True` and the rest as `False`. This can be a great function if we are asked to calculate or show the valid data points in a

given dataset.

- `dropna()` : This function drops the `NaN` values based on several parameters that we can define in it. It provides a vast range of parameters so accurate dead points in the data can be eliminated. If no parameter is defined, the rows containing at least one `NaN` value are deleted. The following are its parameterized types:
 - `dropna(how='all')` : Deletes the rows where all values are `NaN`
 - `dropna(axis=1)` : Deletes the entire column that contains at least one `NaN` value
 - `dropna(thresh=n)` : Deletes the rows that contain less than `n` number of non `NaN` values

```
import numpy as np
import pandas as pd

null_val = np.nan

df = pd.DataFrame([[ 'A', 'B', 'C', 'D'],[null_val, 'B', null_val, 'D'],[null_val, 'B', 'C', null_val],[null_val, 'B', 'C', null_val]])

print("Originl DataFrame:")
print(df)
# Dropping Null values
print("\nThe dropna() method:")
print(df.dropna())

print("\nThe dropna(how = all) method:")
print(df.dropna(how = 'all'))

print("\nThe dropna(axis=1) method:")
print(df.dropna(axis=1))

print("\nThe dropna(thresh = n) method:")
print(df.dropna(thresh = 1))
```

On **line 12**, the `dropna()` method returns one row as all the other rows contain `NaN` values.

On **line 15**, the `dropna(how = all)` method returns three rows as only the last row has all `NaN` values.

On **line 18**, the `dropna(axis=1)` method returns nothing as all the columns contain `NaN` values.

NaN values.

On **line 21**, the `dropna(thresh = 1)` method returns three rows as those rows have at least one element that is not NaN.

- `fillna(n)` : This function replaces the NaN data points with our defined value `n`.

```
import numpy as np
import pandas as pd

null_val = np.nan

df = pd.DataFrame([[ 'A', 'B', 'C', 'D'],[null_val, 'B', null_val, 'D'],[null_val, 'B', 'C', null_val],[null_val, 'B', 'C', null_val]])

print("Originl DataFrame:")
print(df)
# Fill the Null values
print("\nThe filled DataFrame:")
print(df.fillna('M'))
```

It can be seen from the output that the `fillna(n)` method replaces all the NaN values with `M`, which is passed as a parameter to this function. Any alphanumeric value can be assigned to it, and the NaN data points are replaced by it.

In the next lesson, some more important functions of pandas are explored.