Filling missing values

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_excel('./fill_na.xlsx')
data.head(5)
```

Out[2]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	NaN	NaN	NaN
2	2020-05-08	30.9343	NaN	rainy
3	2020-05-09	NaN	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy

Filling a common value to all missing data

In [3]:
```
data.fillna(0)
```

Out[3]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	0.0000	0.000000	0
2	2020-05-08	30.9343	0.000000	rainy
3	2020-05-09	0.0000	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy
5	2020-05-11	23.9382	0.000000	sunny

Adding missing data to individual columns

In [4]:
```
data.fillna({
    'temperature':0,
    'windSpeed':10,
    'status':'sunny'
})
```

Out[4]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	0.0000	10.000000	sunny
2	2020-05-08	30.9343	10.000000	rainy
3	2020-05-09	0.0000	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy
5	2020-05-11	23.9382	10.000000	sunny

We can access individual columns by an alternate way also.

In [5]:
```
data.status.fillna('windy')
```

Out[5]:
```
0     sunny
1     windy
2     rainy
3    cloudy
4     rainy
5     sunny
Name: status, dtype: object
```

Forward fill (row)

Forward fill is a method to forward the data from the row above the missing value. Thus all the missing value will get filled with the value above. If there are multiple missing values consecutively , they will also get filled with the same value of the above available data.

In [6]:
```
data.fillna(method="ffill")
```

Out[6]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	35.6582	10.788378	sunny
2	2020-05-08	30.9343	10.788378	rainy
3	2020-05-09	30.9343	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy
5	2020-05-11	23.9382	19.012990	sunny

Backward fill (row)

In [7]:
```
data.fillna(method="bfill")
```

Out[7]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	30.9343	6.889682	rainy
2	2020-05-08	30.9343	6.889682	rainy
3	2020-05-09	13.9082	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy
5	2020-05-11	23.9382	NaN	sunny

Forward fill (column)

In [8]:
```
data.fillna(method="ffill", axis="columns")
```

Out[8]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	2020-05-07 00:00:00	2020-05-07 00:00:00	2020-05-07 00:00:00
2	2020-05-08	30.9343	30.9343	rainy
3	2020-05-09	2020-05-09 00:00:00	6.889682	cloudy
4	2020-05-10	13.9082	19.01299	rainy
5	2020-05-11	23.9382	23.9382	sunny

Backward fill (column)

In [9]:
```
data.fillna(method="bfill", axis="columns")
```

Out[9]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	NaT	NaT	NaT
2	2020-05-08	30.9343	rainy	rainy
3	2020-05-09	6.889682	6.889682	cloudy
4	2020-05-10	13.9082	19.01299	rainy
5	2020-05-11	23.9382	sunny	sunny

Limiting the forward/backward fill

We can limit the number of rows or columns getting filled.

In [10]:
```
data.fillna(method="ffill", limit=1)
```

Out[10]:

	date	temperature	windSpeed	status
0	2020-05-06	35.6582	10.788378	sunny
1	2020-05-07	35.6582	10.788378	sunny
2	2020-05-08	30.9343	NaN	rainy
3	2020-05-09	30.9343	6.889682	cloudy
4	2020-05-10	13.9082	19.012990	rainy
5	2020-05-11	23.9382	19.012990	sunny

Filling with Pandas objects

There are many Pandas objects like df.sum(), df.max(), etc. we can fill the missing values with these too.

In [11]:
```
data.fillna(data.mean())
```

Out[11]:

	date	temperature	windSpeed	status
0	2020-05-06	35.658200	10.788378	sunny
1	2020-05-07	26.109725	12.230350	NaN
2	2020-05-08	30.934300	12.230350	rainy
3	2020-05-09	26.109725	6.889682	cloudy
4	2020-05-10	13.908200	19.012990	rainy
5	2020-05-11	23.938200	12.230350	sunny

Filling for specific range of columns

We can do filling for a specific range of column too as:

In [12]:
```
data.fillna(data.mean()['temperature':'windSpeed'])
```

Out[12]:

	date	temperature	windSpeed	status
0	2020-05-06	35.658200	10.788378	sunny
1	2020-05-07	26.109725	12.230350	NaN
2	2020-05-08	30.934300	12.230350	rainy
3	2020-05-09	26.109725	6.889682	cloudy
4	2020-05-10	13.908200	19.012990	rainy
5	2020-05-11	23.938200	12.230350	sunny

Interpolate missing value

In short, interpolation is a process of determining the unknown values that lie in between the known data points. We can interpolate missing values based on different methods. This is done by an object in DataFrame as interpolate() . By default, interpolate() does linear interpolation.

Linear interpolate

Linear interpolation involves estimating a new value by connecting two adjacent known values with a straight line.

In [13]:
```
data.temperature.interpolate()
```

Out[13]:
```
0    35.65820
1    33.29625
2    30.93430
3    22.42125
4    13.90820
5    23.93820
Name: temperature, dtype: float64
```

Time interpolate

time-weighted interpolation only works on Series or DataFrames with a DatetimeIndex

data.interpolate(method='time')

Other methods

In [14]:
```
data.temperature.interpolate(method='barycentric')
```

Out[14]:
```
0    35.65820
1    39.46530
2    30.93430
3    19.32775
4    13.90820
5    23.93820
Name: temperature, dtype: float64
```

In [15]:
```
data.temperature.interpolate(method='pchip')
```

Out[15]:
```
0    35.658200
1    34.220728
2    30.934300
3    21.496772
4    13.908200
5    23.938200
Name: temperature, dtype: float64
```

In [16]:
```
data.temperature.interpolate(method='akima')
```

Out[16]:
```
0    35.658200
1    34.448184
2    30.934300
3    22.421250
4    13.908200
5    23.938200
Name: temperature, dtype: float64
```

In [17]:
```
data.temperature.interpolate(method='spline', order = 2)
```

Out[17]:
```
0    35.658200
1    35.076089
2    30.934300
3    20.526966
4    13.908200
5    23.938200
Name: temperature, dtype: float64
```

In [18]:
```
data.temperature.interpolate(method='polynomial', order = 2)
```

Out[18]:
```
0    35.658200
1    36.196165
2    30.934300
3    19.872606
4    13.908200
5    23.938200
Name: temperature, dtype: float64
```