Как работать с датафреймом, содержащим даты

- `pd.to_datetime(data['date'], format='%d.%m.%Y')`
- `pd.read_csv(file, sep='\t', parse_dates=['date'], date_parser=lambda col: pd.to_datetime(col,format='%d.%m.%Y'))`
- `pd.Timestamp('1970-01-01')`
- `pd.Timedelta('1s')`
- `data.index.name`

In [16]:
```
import numpy as np
import pandas as pd
import polars as pl
import bottleneck as bn

import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt

# настройка визуализации
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
# импорт необходимых классов и функций
from catboost import CatBoostRegressor, Pool
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error
```

In [5]:
```
file = 'Data/example.csv'
data = pd.read_csv(file, sep='\t')
data.head()
```

Out[5]:

	date	sales
0	09.01.2018	2400
1	10.01.2018	2800
2	11.01.2018	2500
3	12.01.2018	2890
4	13.01.2018	2610

- `pd.to_datetime` - преобразовываем столбец с датой в тип **datetime**
- `format` - указываем нужный формат дат

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   date    12 non-null     object
 1   sales   12 non-null     int64 
```

```
dtypes: int64(1), object(1)
memory usage: 324.0+ bytes
```

In [7]:
```
data['date'] = pd.to_datetime(data['date'],
                              format='%d.%m.%Y')
data.head()
```

Out[7]:

	date	sales
0	2018-01-09	2400
1	2018-01-10	2800
2	2018-01-11	2500
3	2018-01-12	2890
4	2018-01-13	2610

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    12 non-null     datetime64[ns]
 1   sales   12 non-null     int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 324.0 bytes
```

Парсинг дат можно выполнить сразу при создании датафрейма: с помощью параметров:

- `parse_dates`
- `date_parser`

In [17]:
```
data = pd.read_csv(file, sep='\t',
                   parse_dates=['date'],
                   date_parser=lambda col: pd.to_datetime(col,
                                                          format='%d.%m.%Y'))
display(data.head())
print(data.info())
```

	date	sales
0	2018-01-09	2400
1	2018-01-10	2800
2	2018-01-11	2500
3	2018-01-12	2890
4	2018-01-13	2610

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   date    12 non-null     datetime64[ns]
 1   sales   12 non-null     int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 324.0 bytes
None
```

Задать столбец с датами в качестве индекса:

- `index_col`

Если хочется работать непосредственно с датами - их можно перевести в UNIX-время (POSIX- время):

In [20]:
```
# переводим столбец с датами в UNIX-время
start = pd.Timestamp('1970-01-01')
print(start)

d = pd.Timedelta('1s')
print(d)

data['unix'] = (data['date'] - start) // d
data.head()
```

```
1970-01-01 00:00:00
0 days 00:00:01
```

Out[20]:

	date	sales	unix
0	2018-01-09	2400	1515456000
1	2018-01-10	2800	1515542400
2	2018-01-11	2500	1515628800
3	2018-01-12	2890	1515715200
4	2018-01-13	2610	1515801600

Создадим датафрейм и при создании:

- прочитаем столбец с датами как индекс
- выполним парсинг дат

In [21]:
```
data = pd.read_csv(file,
                   sep='\t',
                   index_col=['date'],
                   parse_dates=['date'],
                   date_parser=lambda col: pd.to_datetime(col,
                                                          format='%d.%m.%Y'))
data.head()
```

Out[21]:

	sales
date	
2018-01-09	2400
2018-01-10	2800
2018-01-11	2500
2018-01-12	2890
2018-01-13	2610

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 12 entries, 2018-01-09 to 2018-01-20
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
```

```
---  ------  --------------  -----
 0   sales   12 non-null     int64
dtypes: int64(1)
memory usage: 192.0 bytes
```

Можно удалить имя индекса

In [23]:
```
data.index.name
```

Out[23]:
```
'date'
```

In [2]:
```
data.index.name = None
data.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 data.index.name = None
      2 data.head()

NameError: name 'data' is not defined
```