create_datasets

September 29, 2021

1 Download and manipulate data

This notebook contains information on downloading the Quandl Wiki stock prices and a few other sources that we use throughout the book.

1.1 Imports & Settings

[2]:
```
from pathlib import Path
import numpy as np
import pandas as pd
import pandas_datareader.data as web

pd.set_option('display.expand_frame_repr', False)
```

```
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-
packages/pandas_datareader/compat/__init__.py:7: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at
pandas.testing instead.
  from pandas.util.testing import assert_frame_equal
```

1.2 Set Data Store path

Modify path if you would like to store the data elsewhere and change the notebooks accordingly

[4]:
```
DATA_STORE = Path('assets.h5')
```

1.3 Quandl Wiki Prices

Quandl makes available a dataset with stock prices, dividends and splits for 3000 US publicly-traded companies. Quandl decided to discontinue support in favor of its commercial offerings but the historical data are still useful to demonstrate the application of the machine learning solutions in the book, just ensure you implement your own algorithms on current data.

> As of April 11, 2018 this data feed is no longer actively supported by the Quandl community. We will continue to host this data feed on Quandl, but we do not recommend using it for investment or analysis.

1. Follow the instructions to create a free Quandl account
2. Download the entire WIKI/PRICES data

3. Extract the .zip file,
4. Move to this directory and rename to wiki_prices.csv
5. Run the below code to store in fast HDF format (see Chapter 02 on Market & Fundamental Data for details).

```
[ ]: df = (pd.read_csv('wiki_prices.csv',
                      parse_dates=['date'],
                      index_col=['date', 'ticker'],
                      infer_datetime_format=True)
          .sort_index())

     print(df.info(null_counts=True))
     with pd.HDFStore(DATA_STORE) as store:
         store.put('quandl/wiki/prices', df)
```

1.3.1 Wiki Prices Metadata

Quandl no longer makes the metadata available. I've added the `wiki_stocks.csv` file to this directory so you can proceed directly with the next code cell and load the file.

1. Follow the instructions to create a free Quandl account if you haven't done so yet
2. Find link to download wiki metadata under Companies](https://www.quandl.com/databases/WIKIP/documentation) or use the download link with your API_KEY: https://www.quandl.com/api/v3/databases/WIKI/metadata?api_key=
3. Extract the .zip file,
4. Move to this directory and rename to wiki_stocks.csv
5. Run the following code to store in fast HDF format

```
[7]: meta_data = pd.read_csv('wiki_stocks.csv')
     meta_data = pd.concat([meta_data.loc[:, 'code'].str.strip(),
                            meta_data.loc[:, 'name'].str.split('(', expand=True)[0].str.
      ↪strip().to_frame('name')], axis=1)
     meta_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3199 entries, 0 to 3198
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   code    3199 non-null   object
 1   name    3199 non-null   object
dtypes: object(2)
memory usage: 50.1+ KB
```

```
[8]: meta_data.head()
```

```
[8]:   code                           name
     0     A     Agilent Technologies Inc.
```

```
1    AA                    Alcoa Inc.
2   AAL  American Airlines Group Inc.
3  AAMC   Altisource Asset Management
4   AAN                  Aaron's Inc.
```

[ ]:
```python
with pd.HDFStore(DATA_STORE) as store:
    store.put('quandl/wiki/stocks', meta_data)
```

1.4 S&P 500 Prices

The following code downloads historical S&P 500 prices from FRED (only last 10 years of daily data is freely available)

[ ]:
```python
df = web.DataReader(name='SP500', data_source='fred', start=2008)
print(df.info())
with pd.HDFStore(DATA_STORE) as store:
    store.put('sp500/prices', df)
```

1.4.1 S&P 500 Constituents

The following code downloads the current S&P 500 constituents from Wikipedia.

[5]:
```python
url = 'https://en.wikipedia.org/wiki/List_of_S%26P_500_companies'
df = pd.read_html(url, header=0)[0]
df.columns = ['name', 'ticker', 'sec_filings', 'gics_sector',␣
 ↪'gics_sub_industry',
              'location', 'first_added', 'cik', 'founded']
df = df.drop('sec_filings', axis=1).set_index('ticker')
print(df.info())
with pd.HDFStore(DATA_STORE) as store:
    store.put('sp500/stocks', df)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 505 entries, MMM to ZTS
Data columns (total 7 columns):
name                 505 non-null object
gics_sector          505 non-null object
gics_sub_industry    505 non-null object
location             505 non-null object
first_added          402 non-null object
cik                  505 non-null int64
founded              172 non-null object
dtypes: int64(1), object(6)
memory usage: 31.6+ KB
None
```

```
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3214: PerformanceWarning:
```

```
your performance may suffer as PyTables will pickle object types that it cannot
map directly to c-types [inferred_type->mixed,key->block0_values]
[items->['name', 'gics_sector', 'gics_sub_industry', 'location', 'first_added',
'founded']]

  if (yield from self.run_code(code, result)):
```

1.5 Metadata on US-traded companies

The following downloads several attributes for companies traded on NASDAQ, AMEX and NYSE

[44]:
```
url = 'https://www.nasdaq.com/screening/companies-by-name.aspx?
↪letter=0&exchange={}&render=download'
exchanges = ['NASDAQ', 'AMEX', 'NYSE']
df = pd.concat([pd.read_csv(url.format(ex)) for ex in exchanges]).
↪dropna(how='all', axis=1)
df = df.rename(columns=str.lower).set_index('symbol').drop('summary quote',␣
↪axis=1)
df = df[~df.index.duplicated()]
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6852 entries, YI to ZYME
Data columns (total 6 columns):
name         6852 non-null object
lastsale     6742 non-null float64
marketcap    5835 non-null object
ipoyear      3113 non-null float64
sector       5292 non-null object
industry     5292 non-null object
dtypes: float64(2), object(4)
memory usage: 374.7+ KB
None
```

[45]:
```
df.head()
```

[45]:
```
                                          name  lastsale marketcap  ipoyear  \
sector                    industry
symbol
YI                                  111, Inc.    7.7000   $627.89M   2018.0
Health Care     Medical/Nursing Services
PIH     1347 Property Insurance Holdings, Inc.    5.1700    $31.09M   2014.0
Finance   Property-Casualty Insurers
PIHPP   1347 Property Insurance Holdings, Inc.   24.7628    $17.33M      NaN
Finance   Property-Casualty Insurers
TURN                   180 Degree Capital Corp.    1.8922    $58.89M      NaN
Finance   Finance/Investors Services
FLWS                    1-800 FLOWERS.COM, Inc.   18.8200     $1.21B   1999.0
```

```
Consumer Services      Other Specialty Stores
```

1.5.1 Convert market cap information to numerical format

Market cap is provided as strings so we need to convert it to numerical format.

[46]:
```
mcap = df[['marketcap']].dropna()
mcap['suffix'] = mcap.marketcap.str[-1]
mcap.suffix.value_counts()
```

[46]:
```
M    3388
B    2441
0       6
Name: suffix, dtype: int64
```

Keep only values with value units:

[47]:
```
mcap = mcap[mcap.suffix.str.endswith(('B', 'M'))]
mcap.marketcap = pd.to_numeric(mcap.marketcap.str[1:-1])
mcaps = {'M': 1e6, 'B': 1e9}
for symbol, factor in mcaps.items():
    mcap.loc[mcap.suffix == symbol, 'marketcap'] *= factor
mcap.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5829 entries, YI to ZYME
Data columns (total 2 columns):
marketcap    5829 non-null float64
suffix       5829 non-null object
dtypes: float64(1), object(1)
memory usage: 296.6+ KB
```

[49]:
```
df['marketcap'] = mcap.marketcap
df.marketcap.describe(percentiles=np.arange(.1, 1, .1).round(1)).apply(lambda x:
 ↪ f'{int(x):,d}')
```

[49]:
```
count              5,829
mean       7,495,050,037
std       36,240,773,357
min            1,410,000
10%           35,188,000
20%           92,888,000
30%          196,769,999
40%          345,400,000
50%          619,210,000
60%        1,117,999,999
70%        2,200,000,000
80%        4,724,000,000
```

```
90%       13,713,999,999
max      961,260,000,000
Name: marketcap, dtype: object
```

1.5.2 Store result

[28]:
```
with pd.HDFStore(DATA_STORE) as store:
    store.put('us_equities/stocks', df)
```

```
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3214: PerformanceWarning:
your performance may suffer as PyTables will pickle object types that it cannot
map directly to c-types [inferred_type->mixed,key->block1_values]
[items->['name', 'sector', 'industry']]

  if (yield from self.run_code(code, result)):
```

1.6 Bond Price Indexes

The following code downloads several bond indexes from the Federal Reserve Economic Data service (FRED)

[ ]:
```
securities = {'BAMLCC0A0CMTRIV'   : 'US Corp Master TRI',
              'BAMLHYH0A0HYM2TRIV': 'US High Yield TRI',
              'BAMLEMCBPITRIV'    : 'Emerging Markets Corporate Plus TRI',
              'GOLDAMGBD228NLBM'  : 'Gold (London, USD)',
              'DGS10'             : '10-Year Treasury CMR',
              }

df = web.DataReader(name=list(securities.keys()), data_source='fred',␣
 ↪start=2000)
df = df.rename(columns=securities).dropna(how='all').resample('B').mean()

with pd.HDFStore(DATA_STORE) as store:
    store.put('fred/assets', df)
```