00_build_dataset

September 29, 2021

1 Create a dataset formatted for RNN examples

1.1 Imports & Settings

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
from pathlib import Path

import numpy as np
import pandas as pd
```

[3]:
```
np.random.seed(42)
```

[4]:
```
idx = pd.IndexSlice
```

1.2 Build daily dataset

[5]:
```
DATA_DIR = Path('..', 'data')
```

[6]:
```
prices = (pd.read_hdf(DATA_DIR / 'assets.h5', 'quandl/wiki/prices')
          .loc[idx['2010':'2017', :], ['adj_close', 'adj_volume']])
prices.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 5698754 entries, (Timestamp('2010-01-04 00:00:00'), 'A') to
(Timestamp('2017-12-29 00:00:00'), 'ZUMZ')
Data columns (total 2 columns):
 #   Column      Dtype
---  ------      -----
 0   adj_close   float64
 1   adj_volume  float64
dtypes: float64(2)
memory usage: 109.5+ MB
```

1.2.1 Select most traded stocks

[7]:
```
n_dates = len(prices.index.unique('date'))
dollar_vol = (prices.adj_close.mul(prices.adj_volume)
              .unstack('ticker')
              .dropna(thresh=int(.95 * n_dates), axis=1)
              .rank(ascending=False, axis=1)
              .stack('ticker'))
```

[8]:
```
most_traded = dollar_vol.groupby(level='ticker').mean().nsmallest(500).index
```

[9]:
```
returns = (prices.loc[idx[:, most_traded], 'adj_close']
           .unstack('ticker')
           .pct_change()
           .sort_index(ascending=False))
returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2013 entries, 2017-12-29 to 2010-01-04
Columns: 500 entries, AAPL to CNC
dtypes: float64(500)
memory usage: 7.7 MB
```

1.2.2 Stack 21-day time series

[10]:
```
n = len(returns)
T = 21 # days
tcols = list(range(T))
tickers = returns.columns
```

[11]:
```
data = pd.DataFrame()
for i in range(n-T-1):
    df = returns.iloc[i:i+T+1]
    date = df.index.max()
    data = pd.concat([data,
                      df.reset_index(drop=True).T
                      .assign(date=date, ticker=tickers)
                      .set_index(['ticker', 'date'])])
data = data.rename(columns={0: 'label'}).sort_index().dropna()
data.loc[:, tcols[1:]] = (data.loc[:, tcols[1:]].apply(lambda x: x.clip(lower=x.
 ↪quantile(.01),
                                                         upper=x.quantile(.99))))
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 995499 entries, ('A', Timestamp('2010-02-04 00:00:00')) to ('ZION',
Timestamp('2017-12-29 00:00:00'))
Data columns (total 22 columns):
```

```
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   label   995499 non-null  float64
 1   1       995499 non-null  float64
 2   2       995499 non-null  float64
 3   3       995499 non-null  float64
 4   4       995499 non-null  float64
 5   5       995499 non-null  float64
 6   6       995499 non-null  float64
 7   7       995499 non-null  float64
 8   8       995499 non-null  float64
 9   9       995499 non-null  float64
 10  10      995499 non-null  float64
 11  11      995499 non-null  float64
 12  12      995499 non-null  float64
 13  13      995499 non-null  float64
 14  14      995499 non-null  float64
 15  15      995499 non-null  float64
 16  16      995499 non-null  float64
 17  17      995499 non-null  float64
 18  18      995499 non-null  float64
 19  19      995499 non-null  float64
 20  20      995499 non-null  float64
 21  21      995499 non-null  float64
dtypes: float64(22)
memory usage: 171.0+ MB
```

[12]:
```
data.shape
```

[12]:
```
(995499, 22)
```

[13]:
```
data.to_hdf('data.h5', 'returns_daily')
```

1.3 Build weekly dataset

We load the Quandl adjusted stock price data:

[14]:
```
prices = (pd.read_hdf(DATA_DIR / 'assets.h5', 'quandl/wiki/prices')
          .adj_close
          .unstack().loc['2007':])
prices.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2896 entries, 2007-01-01 to 2018-03-27
Columns: 3199 entries, A to ZUMZ
dtypes: float64(3199)
memory usage: 70.7 MB
```

1.3.1 Resample to weekly frequency

We start by generating weekly returns for close to 2,500 stocks without missing data for the 2008-17 period, as follows:

[15]:
```
returns = (prices
           .resample('W')
           .last()
           .pct_change()
           .loc['2008': '2017']
           .dropna(axis=1)
           .sort_index(ascending=False))
returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 522 entries, 2017-12-31 to 2008-01-06
Freq: -1W-SUN
Columns: 2489 entries, A to ZUMZ
dtypes: float64(2489)
memory usage: 9.9 MB
```

[16]:
```
returns.head().append(returns.tail())
```

[16]:

ticker	A	AAL	AAN	AAON	AAP	AAPL	\
date							
2017-12-31	-0.005642	-0.010648	-0.010184	-0.001361	-0.008553	-0.033027	
2017-12-24	-0.003846	0.029965	0.090171	0.044034	-0.001490	0.006557	
2017-12-17	0.003413	0.000784	-0.052591	-0.014006	0.003888	0.026569	
2017-12-10	-0.019071	0.041012	-0.005359	-0.017882	0.010375	-0.009822	
2017-12-03	-0.009660	0.009267	0.105501	0.013947	0.112630	-0.022404	
2008-02-03	0.038265	0.252238	0.002941	0.095182	0.097833	0.028767	
2008-01-27	-0.013963	-0.048762	0.191310	0.071788	0.043997	-0.194286	
2008-01-20	-0.065000	0.086627	-0.080541	-0.054762	-0.007176	-0.065609	
2008-01-13	0.035375	-0.041902	-0.037818	-0.046538	-0.101486	-0.040878	
2008-01-06	-0.072553	-0.156356	-0.068707	-0.133301	-0.065496	-0.098984	

ticker	AAWW	ABAX	ABC	ABCB	…	ZEUS	ZIGO	\
date					…			
2017-12-31	-0.024938	-0.001814	-0.006922	-0.019329	…	-0.029797	0.000000	
2017-12-24	0.046087	0.032681	-0.007620	0.017598	…	0.032153	0.000000	
2017-12-17	0.004367	0.008396	0.074625	0.026567	…	0.036715	0.000000	
2017-12-10	-0.028014	-0.010386	0.020600	-0.054271	…	-0.002410	0.000000	
2017-12-03	0.073838	-0.028456	0.045796	0.024717	…	0.065742	0.000000	
2008-02-03	0.006245	-0.078058	0.036913	0.083217	…	0.137066	0.127561	
2008-01-27	-0.008984	-0.090807	-0.034771	0.054572	…	0.018349	-0.026292	
2008-01-20	0.015818	-0.019721	-0.015219	-0.044397	…	0.040573	0.010999	
2008-01-13	-0.052095	0.097385	0.080137	-0.017313	…	-0.054176	-0.047993	
2008-01-06	-0.029478	-0.098374	-0.037363	-0.132733	…	-0.027290	-0.075806	

```
ticker          ZINC      ZION      ZIOP      ZIXI       ZLC       ZMH  \
date
2017-12-31  0.000000 -0.009741  0.022222 -0.015730  0.000000  0.000000
2017-12-24  0.000000  0.026395 -0.068966 -0.024123  0.000000  0.000000
2017-12-17  0.000000 -0.018064 -0.018059  0.075472  0.000000  0.000000
2017-12-10  0.000000  0.016973 -0.015556 -0.055679  0.000000  0.000000
2017-12-03  0.000000  0.080475  0.014656 -0.006637  0.000000  0.000000
2008-02-03  0.286550  0.167722 -0.087879  0.069364  0.171949  0.193189
2008-01-27 -0.046975  0.136418 -0.003021  0.145695  0.042164 -0.014553
2008-01-20 -0.167109 -0.051614 -0.054286 -0.124638  0.037172 -0.037312
2008-01-13 -0.102381  0.037264 -0.022346 -0.172662  0.011799  0.051880
2008-01-06 -0.004739 -0.081058  0.101538 -0.143737 -0.134100  0.000752

ticker           ZQK       ZUMZ
date
2017-12-31  0.000000 -0.029138
2017-12-24  0.000000  0.067164
2017-12-17  0.000000 -0.051887
2017-12-10  0.000000  0.062657
2017-12-03  0.000000  0.047244
2008-02-03  0.127811  0.149083
2008-01-27  0.141892  0.118666
2008-01-20 -0.030144 -0.076969
2008-01-13  0.018692 -0.094249
2008-01-06 -0.133102 -0.269012

[10 rows x 2489 columns]
```

1.3.2 Create & stack 52-week sequences

We'll use 52-week sequences, which we'll create in a stacked format:

[17]:
```
n = len(returns)
T = 52 # weeks
tcols = list(range(T))
tickers = returns.columns
```

[18]:
```
data = pd.DataFrame()
for i in range(n-T-1):
    df = returns.iloc[i:i+T+1]
    date = df.index.max()
    data = pd.concat([data, (df.reset_index(drop=True).T
                             .assign(date=date, ticker=tickers)
                             .set_index(['ticker', 'date']))])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 1167341 entries, ('A', Timestamp('2017-12-31 00:00:00')) to ('ZUMZ',
Timestamp('2009-01-11 00:00:00'))
Data columns (total 53 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   0       1167341 non-null  float64
 1   1       1167341 non-null  float64
 2   2       1167341 non-null  float64
 3   3       1167341 non-null  float64
 4   4       1167341 non-null  float64
 5   5       1167341 non-null  float64
 6   6       1167341 non-null  float64
 7   7       1167341 non-null  float64
 8   8       1167341 non-null  float64
 9   9       1167341 non-null  float64
 10  10      1167341 non-null  float64
 11  11      1167341 non-null  float64
 12  12      1167341 non-null  float64
 13  13      1167341 non-null  float64
 14  14      1167341 non-null  float64
 15  15      1167341 non-null  float64
 16  16      1167341 non-null  float64
 17  17      1167341 non-null  float64
 18  18      1167341 non-null  float64
 19  19      1167341 non-null  float64
 20  20      1167341 non-null  float64
 21  21      1167341 non-null  float64
 22  22      1167341 non-null  float64
 23  23      1167341 non-null  float64
 24  24      1167341 non-null  float64
 25  25      1167341 non-null  float64
 26  26      1167341 non-null  float64
 27  27      1167341 non-null  float64
 28  28      1167341 non-null  float64
 29  29      1167341 non-null  float64
 30  30      1167341 non-null  float64
 31  31      1167341 non-null  float64
 32  32      1167341 non-null  float64
 33  33      1167341 non-null  float64
 34  34      1167341 non-null  float64
 35  35      1167341 non-null  float64
 36  36      1167341 non-null  float64
 37  37      1167341 non-null  float64
 38  38      1167341 non-null  float64
 39  39      1167341 non-null  float64
 40  40      1167341 non-null  float64
 41  41      1167341 non-null  float64
```

```
 42  42      1167341 non-null  float64
 43  43      1167341 non-null  float64
 44  44      1167341 non-null  float64
 45  45      1167341 non-null  float64
 46  46      1167341 non-null  float64
 47  47      1167341 non-null  float64
 48  48      1167341 non-null  float64
 49  49      1167341 non-null  float64
 50  50      1167341 non-null  float64
 51  51      1167341 non-null  float64
 52  52      1167341 non-null  float64
dtypes: float64(53)
memory usage: 476.6+ MB
```

[19]:
```
data[tcols] = (data[tcols].apply(lambda x: x.clip(lower=x.quantile(.01),
                                                  upper=x.quantile(.99))))
```

[20]:
```
data = data.rename(columns={0: 'fwd_returns'})
```

[21]:
```
data['label'] = (data['fwd_returns'] > 0).astype(int)
```

[22]:
```
data.shape
```

[22]:
```
(1167341, 54)
```

[23]:
```
data.sort_index().to_hdf('data.h5', 'returns_weekly')
```