assignment_1

June 21, 2025

1 Working with parquet files

1.1 Objective

- In this assignment, we will use the data downloaded with the module `data_manager` to create features.

(11 pts total)

1.2 Prerequisites

- This notebook assumes that price data is available to you in the environment variable `PRICE_DATA`. If you have not done so, then execute the notebook `01_materials/labs/2_data_engineering.ipynb` to create this data set.
- Load the environment variables using dotenv. (1 pt)

[1]:
```
# Write your code below.
%load_ext dotenv
%dotenv
```

[2]:
```
import dask.dataframe as dd
```

```
/tmp/ipykernel_9407/676544213.py:1: DeprecationWarning: The current Dask
DataFrame implementation is deprecated.
In a future release, Dask DataFrame will use a new implementation that
contains several improvements including a logical query planning.
The user-facing DataFrame API will remain unchanged.

The new implementation is already available and can be enabled by
installing the dask-expr library:

    $ pip install dask-expr

and turning the query planning option on:

    >>> import dask
    >>> dask.config.set({'dataframe.query-planning': True})
    >>> import dask.dataframe as dd
```

```
API documentation for the new implementation is available at
https://docs.dask.org/en/stable/dask-expr-api.html

Any feedback can be reported on the Dask issue tracker
https://github.com/dask/dask/issues

To disable this warning in the future, set dask config:

    # via Python
    >>> dask.config.set({'dataframe.query-planning-warning': False})

    # via CLI
    dask config set dataframe.query-planning-warning False
```

```
import dask.dataframe as dd
```

- Load the environment variable `PRICE_DATA`.
- Use glob to find the path of all parquet files in the directory `PRICE_DATA`.

(1pt)

[28]:
```python
import os
from glob import glob

# Write your code below.

print(os.getenv("PRICE_DATA"))

# Load the environment variable PRICE_DATA
price_data_dir = os.getenv("PRICE_DATA")

# Use glob to find all parquet files in the directory
parquet_files = glob(os.path.join(price_data_dir, "**/*.parquet"), recursive =␣
 ↪True)
```

```
../../05_src/data/prices/
```

For each ticker and using Dask, do the following:

- Add lags for variables Close and Adj_Close.
- Add returns based on Close:
  - `returns`: (Close / Close_lag_1) - 1
- Add the following range:
  - `hi_lo_range`: this is the day's High minus Low.
- Assign the result to `dd_feat`.

(4 pt)

[34]:
```python
# Write your code below.

# Read all parquet files into a Dask DataFrame
dd_px = dd.read_parquet(parquet_files, index = 'ticker')

# Display the first few rows of the Dask DataFrame
dd_px.head()

print(dd_px.columns)
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume', 'source',
       'Year'],
      dtype='object')
```

[38]:
```python
# Create a new feature in the Dask DataFrame, grouping by 'ticker' and adding␣
↪lags for Close and Adj Close,
# calculating returns, and the high-low range.
dd_feat = dd_px.groupby('ticker', group_keys=False).apply(
    lambda df: df.assign(
        Close_lag_1 = df['Close'].shift(1),
        Adj_Close_lag_1 = df['Adj Close'].shift(1),
        returns = df['Close'] / df['Close'].shift(1) - 1,
        hi_lo_range = df['High'] - df['Low']
    )
)
```

```
/tmp/ipykernel_9407/3832338523.py:3: UserWarning: `meta` is not specified,
inferred from partial data. Please provide `meta` if the result is unexpected.
  Before: .apply(func)
  After:  .apply(func, meta={'x': 'f8', 'y': 'f8'}) for dataframe result
  or:     .apply(func, meta=('x', 'f8'))            for series result
  dd_feat = dd_px.groupby('ticker', group_keys=False).apply(
```

- Convert the Dask data frame to a pandas data frame.
- Add a new feature containing the moving average of `returns` using a window of 10 days. There are several ways to solve this task, a simple one uses `.rolling(10).mean()`.

(3 pt)

[44]:
```python
# Write your code below.

# Compute the Dask DataFrame and convert it to a Pandas DataFrame with a moving␣
↪average of returns of 10 days
pd_feat = dd_feat.compute().groupby('ticker', group_keys=False).apply(
    lambda df: df.assign(moving_average = df['returns'].rolling(10).mean())
)

pd_feat
```

[44]:
```
               Date  Open  High   Low  Close  Adj Close    Volume    source  \
ticker
WRN     2015-01-02  0.58  0.62  0.58   0.60       0.60    1700.0   WRN.csv
WRN     2015-01-05  0.60  0.60  0.60   0.60       0.60       0.0   WRN.csv
WRN     2015-01-06  0.60  0.60  0.60   0.60       0.60    2200.0   WRN.csv
WRN     2015-01-07  0.54  0.60  0.54   0.59       0.59   40400.0   WRN.csv
WRN     2015-01-08  0.59  0.60  0.59   0.60       0.60    1000.0   WRN.csv
…              …     …     …     …      …          …         …         …
GXGX    2019-10-14  9.89  9.89  9.89   9.89       9.89    2300.0  GXGX.csv
GXGX    2019-10-15  9.93  9.93  9.93   9.93       9.93    1000.0  GXGX.csv
GXGX    2019-10-16  9.93  9.93  9.93   9.93       9.93       0.0  GXGX.csv
GXGX    2019-10-17  9.93  9.93  9.93   9.93       9.93       0.0  GXGX.csv
GXGX    2019-10-18  9.82  9.93  9.82   9.84       9.84  945300.0  GXGX.csv

        Year  Close_lag_1  Adj_Close_lag_1   returns  hi_lo_range  \
ticker
WRN     2015          NaN              NaN       NaN     0.040000
WRN     2015         0.60             0.60  0.000000     0.000000
WRN     2015         0.60             0.60  0.000000     0.000000
WRN     2015         0.60             0.60 -0.016667     0.060000
WRN     2015         0.59             0.59  0.016949     0.010000
…        …            …                …         …            …
GXGX    2019         9.94             9.94 -0.005030     0.000000
GXGX    2019         9.89             9.89  0.004044     0.000000
GXGX    2019         9.93             9.93  0.000000     0.000000
GXGX    2019         9.93             9.93  0.000000     0.000000
GXGX    2019         9.93             9.93 -0.009063     0.110001

        moving_average
ticker
WRN                NaN
WRN                NaN
WRN                NaN
WRN                NaN
WRN                NaN
…                    …
GXGX         -0.000593
GXGX          0.000816
GXGX         -0.000199
GXGX         -0.000199
GXGX         -0.001105

[330200 rows x 14 columns]
```

Please comment:

- Was it necessary to convert to pandas to calculate the moving average return? > No, it was not necessary to convert to pandas to perform the calculation, but for smaller sets of data, sometimes it is easier or quicker for analysis.

- Would it have been better to do it in Dask? Why? > It may have been better to perform this task in Dask to take advantage of parallel processing. For large datasets, Dask is superior to avoid memory issues.

(1 pt)

1.3 Criteria

The rubric contains the criteria for grading.

1.4 Submission Information

Please review our Assignment Submission Guide for detailed instructions on how to format, branch, and submit your work. Following these guidelines is crucial for your submissions to be evaluated correctly.

1.4.1 Submission Parameters:

- Submission Due Date: `HH:MM AM/PM - DD/MM/YYYY`
- The branch name for your repo should be: `assignment-1`
- What to submit for this assignment:
  - This Jupyter Notebook (assignment_1.ipynb) should be populated and should be the only change in your pull request.
- What the pull request link should look like for this assignment: `https://github.com/<your_github_username>/production/pull/<pr_id>`
  - Open a private window in your browser. Copy and paste the link to your pull request into the address bar. Make sure you can see your pull request properly. This helps the technical facilitator and learning support staff review your submission easily.

Checklist: - [ ] Created a branch with the correct naming convention. - [ ] Ensured that the repository is public. - [ ] Reviewed the PR description guidelines and adhered to them. - [ ] Verify that the link is accessible in a private browser window.

If you encounter any difficulties or have questions, please don't hesitate to reach out to our team via our Slack at `#cohort-3-help`. Our Technical Facilitators and Learning Support staff are here to help you navigate any challenges.