DWV Checkpoint

Ruslan Gatiatullin, Ilya Grigorev, Salavat Faizullin

March 2025

1 Data collection and manipulation

1.1 Scraping

Data are scraped from a set of websites stored on the webpage. The scraper is specified with the index name, start and end dates, sorting factor, and sorting order using queries mechanism.

A page is accessed using `playwright` python module to render the page and click the specific button. Then, the loaded page with the information is stored as an `html` file. Thus, selected pages are loaded and stored locally.

1.2 Manipulation

Next, the scraper can scrape information from `html` code. The data from a single page are extracted using `BeautifulSoup` and stored in the `pandas` data frame. Then, the raw data (represented as strings) is parsed according to the schema:

```
date: string (format yyyy-mm-dd)
price_at_opening: float
max_price: float
min_price: float
price_at_closure: float
volume_of_trade: float
capitalization: float
```

After the pre-processing step, the scraper allows one to store the information in the storage, which can be either a JSON file or `MongoDB`.

1.3 Example of Work

The following code snippet can be used to see how the presented architecture works:

```python
import scraper

scr = scraper.Scraper()
my_url = scraper.URL.construct_from_url(
    url=("https://www.moex.com/ru/index/IMOEX/archive"
         "?from=2025-01-26&till=2025-02-26&sort=TRADEDATE&order=desc")
)
scr.load_content([my_url]) # load page with driver
scr.scrape_pages() # scrape html files in the page/ folder

df = scr.load_page_data('page') # load data from example page
print(df.head())
```

1.4 Limitations & Insights

So far, the architecture of the parser is sequential. This configuration significantly reduces the effectiveness of scraping and parsing data, though the implementation is straightforward.

In future checkpoints, we plan to extend the functionality to an asynchronous architecture.

2 MOEX Investment Returns Heatmap Analysis

2.1 Reference

First chart(click!)

2.2 What is Displayed?

- **X-axis (horizontal):** Year/month/day of **investment start**
- **Y-axis (vertical):** Year/month/day of **investment end**
- **Cell color:**
  - Green → Positive returns (brighter = higher profit)
  - Red → Negative returns (losses)
  - Gray → Neutral/zero returns

2.3 How It Works

Each cell answers:

> *"If I invested in MOEX during [X]–[Y], what would my annualized return be?"*

Examples:

- Cell **2025-03** → **2025-06**: 3-month return (March–June 2025)
- Cell **2025-01** → **2025-12**: Annual return for 2025

2.4 Interactive Features

- **Granularity selection:**
  - Years (`year` → `year`)
  - Months (`2025-03` → `2025-06`)
  - Days (`2025-02-27` → `2025-03-05`)
- Date range selectors
- Hover tooltips showing exact returns

2.5 Example Calculation

Given data:

```json
{
  "date": "2025-02-27",
  "price_at_opening": 3249.63,
  "price_at_closure": 3232.65
}
```

Daily return calculation:

$$\text{Return} = \frac{P_{\text{close}} - P_{\text{open}}}{P_{\text{open}}} \times 100 = \frac{3232.65 - 3249.63}{3249.63} \times 100 \approx -0.52\% \qquad (1)$$

2.6 Limitations & Insights

- Sample data covers only **2025**
- Requires multi-year data for comprehensive analysis
- Visualizes:
  - Optimal investment periods
  - Loss risk patterns
  - Market volatility trends

3 Interactive Bar Plot Features

3.1 Reference

Second chart(click!)

3.2 Key Functionalities

- **Dynamic Parameter Control:**
  - Switch between Price, Daily Change (%), Yearly Change (%), and Market Cap
  - Auto-play mode cycles parameters every 3 seconds
- **Interactive Elements:**
  - Hover effects with brightness increase (180%)
  - Floating tooltips with company details
  - Animated sorting and transitions

3.3 Visual Design

- **Color Coding:**
  - Blue: Positive values
  - Red: Negative values
- **Annotations:**
  - Value labels with automatic formatting
  - Reference line at zero value
  - Rotated axis labels for readability

3.4 Technical Implementation

- **Core Stack:**
  - D3.js for data binding and animations
  - SVG for crisp rendering
  - CSS for styling and transitions
- **Data Handling:**
  - Real-time numeric conversion
  - Automatic sorting by selected metric