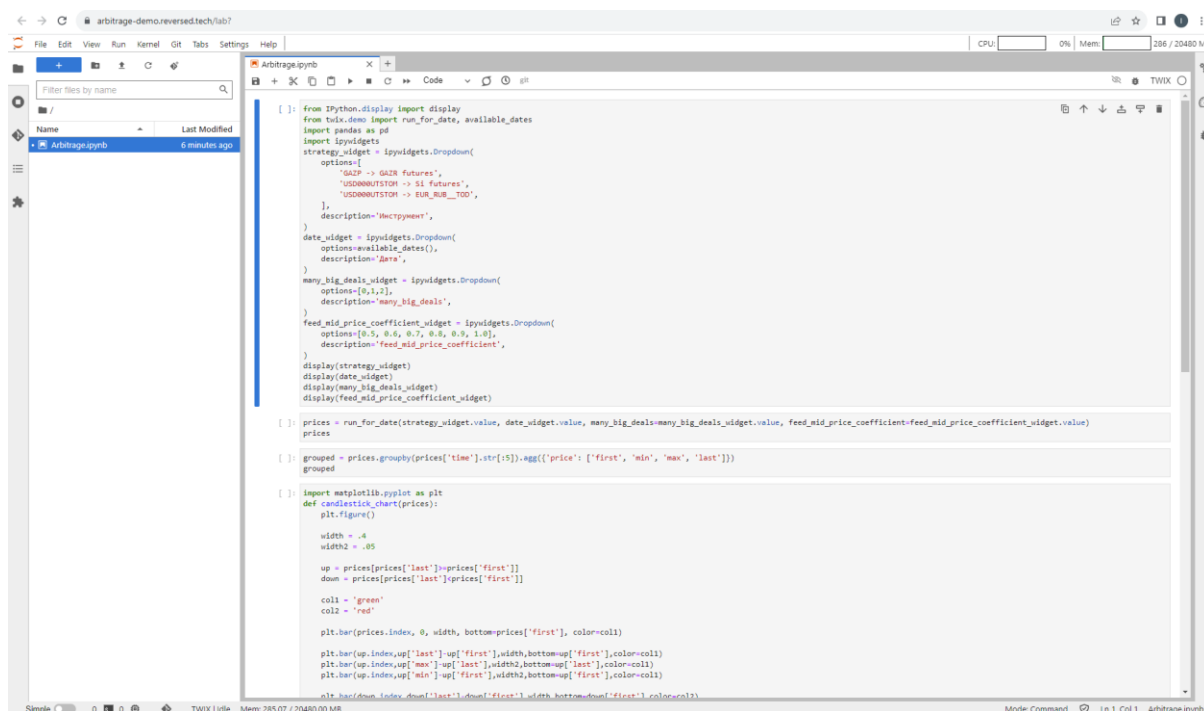


Инструкция по эксплуатации экземпляра ПО «Arbitrage MOEX / Арбитраж MOEX»

Программа «Arbitrage MOEX / Арбитраж MOEX» (далее Программа) представляет собой дистрибутив (файл tar.gz), внутри которого находится код, написанный на языке C++. Данная программа должна быть встроена в ПО клиента для анализа биржевых данных или торговли на Московской Бирже. Для эксплуатации ПО требуется иметь доступ до данных с биржевой информацией. Для проведения экспертной проверки предоставлен удалённый доступ до демонстрационной платформы, на которой развёрнута скомпилированная версия Программы и реализован доступ до биржевых данных за июль 2023 года. Таким образом, демонстрируется возможная работа Программы в комплексе с клиентским ПО для анализа биржевых данных.

Эксперт получает возможность запуска ПО на различных торговых инструментах, с различными наборами параметров на выбранных датах. Интерфейс демонстрационной платформы является стандартным Jupyter Notebook под названием Arbitrage.ipynb. Подробнее о функционале Jupyter Notebook можно прочесть, например, [здесь](#).

Демонстрационная платформа перед стартом работы выглядит следующим образом:



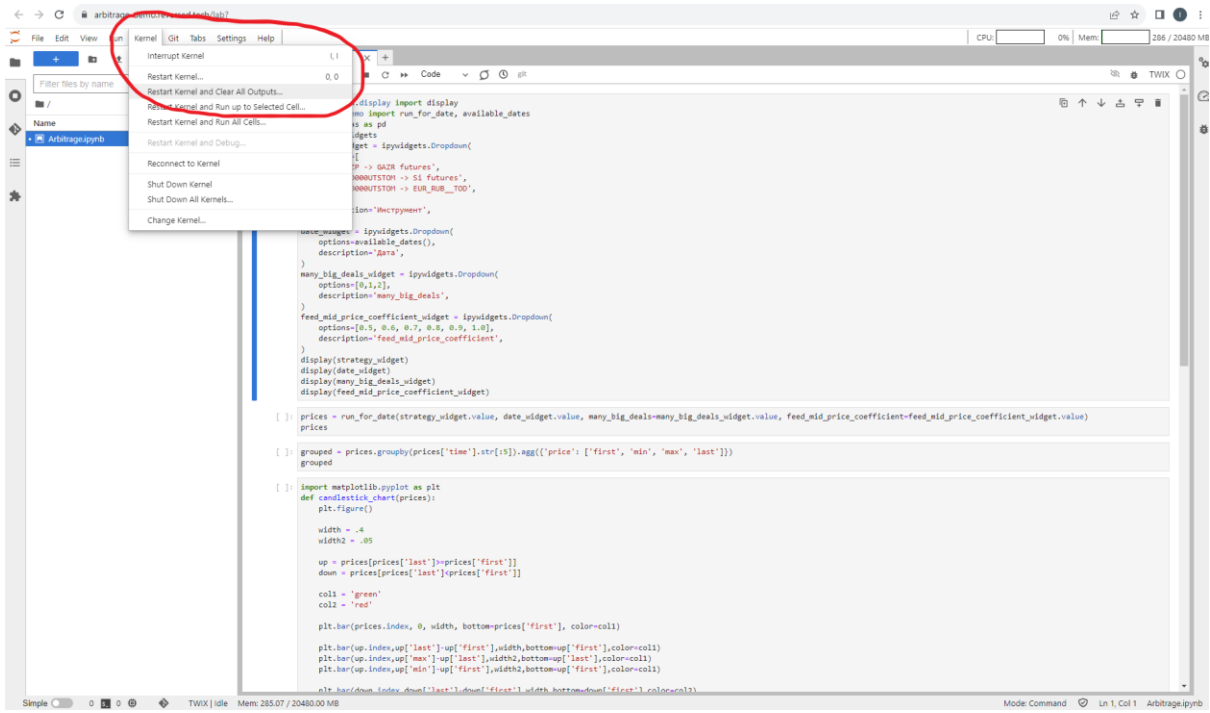
```
[ ]: from IPython.display import display
from twix_demo import run_for_date, available_dates
import pandas as pd
import ipynbwidgets
strategy_widget = ipynbwidgets.Dropdown(
    options=[
        'SAP -> GAZP futures',
        'USDNONUSTOM -> SI futures',
        'USDNONUSTOM -> EUR_RUB_TOD',
    ],
    description='Инструмент',
)
date_widget = ipynbwidgets.Dropdown(
    options=available_dates(),
    description='Дата',
)
many_big_deals_widget = ipynbwidgets.Dropdown(
    options=[0,1,2],
    description='many_big_deals',
)
feed_mid_price_coefficient_widget = ipynbwidgets.Dropdown(
    options=[0.5, 0.4, 0.7, 0.8, 0.9, 1.0],
    description='feed_mid_price_coefficient',
)
display(strategy_widget)
display(date_widget)
display(many_big_deals_widget)
display(feed_mid_price_coefficient_widget)

[ ]: prices = run_for_date(strategy_widget.value, date_widget.value, many_big_deals=many_big_deals_widget.value, feed_mid_price_coefficient=feed_mid_price_coefficient_widget.value)
prices

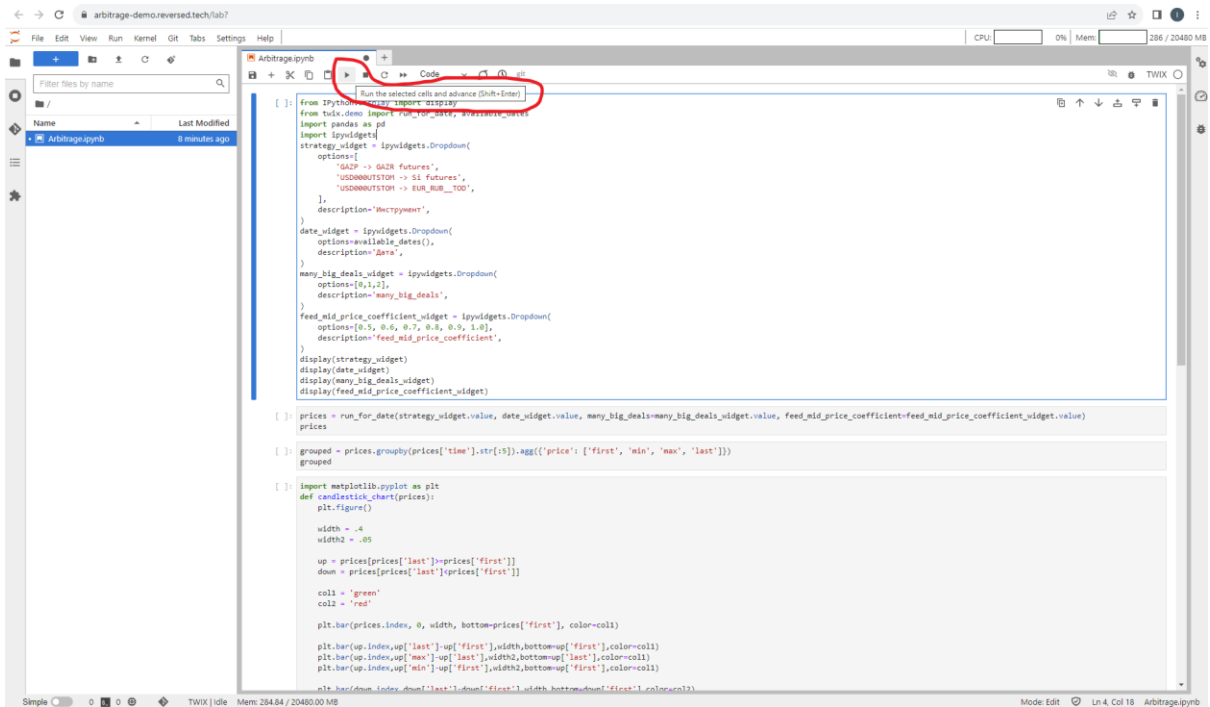
[ ]: grouped = prices.groupby(prices['time'].str[5]).agg({'price': ['first', 'min', 'max', 'last']})
grouped

[ ]: import matplotlib.pyplot as plt
def candlestick_chart(prices):
    plt.figure()
    width = .4
    width2 = .05
    up = prices[prices['last'] >= prices['first']]
    down = prices[prices['last'] < prices['first']]
    col1 = 'green'
    col2 = 'red'
    plt.bar(prices.index, 0, width, bottom=prices['first'], color=col1)
    plt.bar(up.index, up['last'] - up['first'], width, bottom=up['first'], color=col1)
    plt.bar(up.index, up['max'] - up['last'], width2, bottom=up['last'], color=col1)
    plt.bar(down.index, down['first'] - down['last'], width2, bottom=down['last'], color=col2)
    plt.bar(down.index, down['last'] - down['first'], width, bottom=down['first'], color=col2)
```

В случае, если доступ до демонстрационной платформы с использованием заданного пароля будет осуществлять несколько экспертов, при очередном запуске платформы могут быть видны результаты предыдущих запусков. В таком случае, требуется перед стартом работы очистить ноутбук командой Kernel → Restart Kernel and Clear All Outputs...:



Первая клетка Jupyter ноутбука Arbitrage.ipynb задаёт параметры для запуска ПО. Для запуска каждой клетки Jupyter ноутбука требуется нажать Shift+Enter или кнопку Play.



Перед запуском ПО требуется заполнить несколько форм, определяющих параметры, с которыми будет осуществляться запуск.

- **Инструмент.** Эксперту предлагается совершить выбор одной из трёх доступных на платформе арбитражных пар (ведущий инструмент – зависимый инструмент). ПО будет получать биржевую информацию по обоим из них и осуществлять предсказание цен ведомого инструмента. Доступен выбор одной из трёх опций: 1) предсказание цены фьючерса на акцию ПАО «Газпром» в зависимости от биржевой информации по акции ПАО «Газпром»; 2) предсказание цены фьючерса на курс рубля к доллару США в зависимости от биржевой информации по торговле инструментом USDRUB000TOM на валютном рынке ПАО «Московская биржа»; или 3) предсказание цены курса рубля к евро (инструмент EUR_RUB__TOD на валютном рынке ПАО «Московская биржа») в зависимости от биржевой информации по торговле инструментом USDRUB000TOM на валютном рынке ПАО «Московская биржа».
- **Дата.** Доступен список дней из июля 2023 года, когда осуществлялась биржевая торговля выбранными инструментами на площадках ПАО «Московская биржа». После запуска ПО будет сформирована таблица с предсказанием цен зависимого инструмента за указанную дату.
- **many_big_deals.** Параметр ПО, отвечающий за то, насколько сильно сделки на ведущем инструменте влияют на цену зависимого инструмента, для которого осуществляются предсказания
- **feed_mid_price_coefficient.** Параметр ПО, отвечающий за то, насколько сильно котировки ведущего инструмента влияют на цену зависимого инструмента, для которого осуществляются предсказания

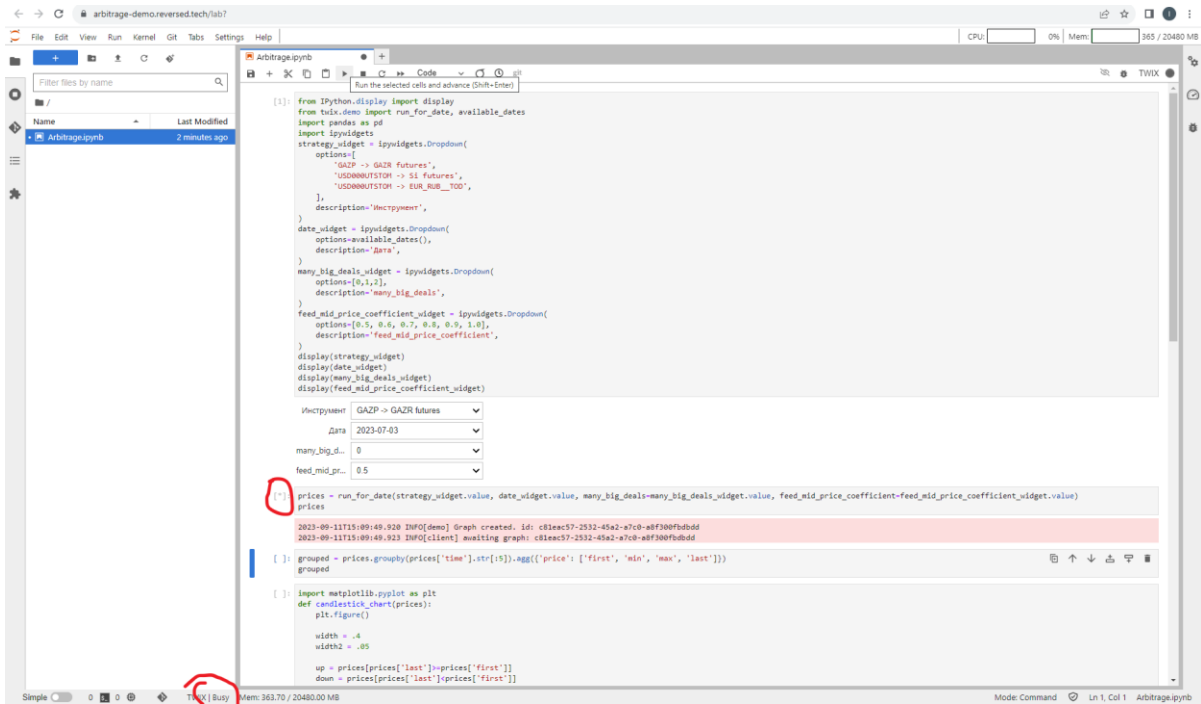
По умолчанию формы заполнены следующим образом:

The screenshot displays a Jupyter Notebook environment. The top part shows a file browser with a folder named 'Arbitrage.ipynb'. The main area contains Python code for data processing and visualization. Below the code, there is a form with four dropdown menus for parameter selection:

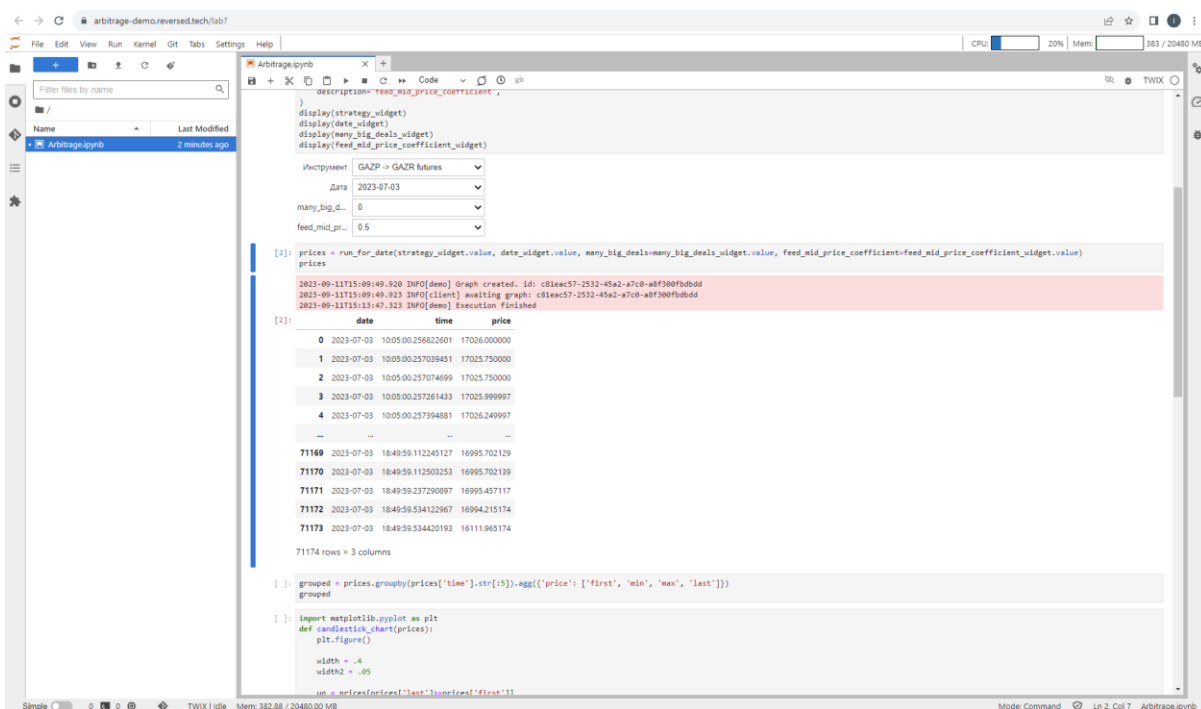
- Instrument: GAZP -> GAZR futures
- Date: 2023-07-03
- many_big_d.: 0
- feed_mid_pr.: 0.5

The code includes imports for IPython, pandas, and matplotlib, and performs data grouping and plotting. The plot area shows a candlestick chart with green and red bars, representing price movements over time.

Вторая клетка Jupyter ноутбука Arbitrage.ipynb запускает ПО, которое получает биржевую информацию по указанным инструментам за указанный день и выдаёт предсказание цены после каждого обновления биржевой информации. Расчёт предсказаний занимает несколько минут, т. к. в течение торгового дня происходят тысячи обновлений биржевой информации. При проведении расчётов демонстрационная платформа выглядит следующим образом:



Результатом работы ПО является таблица со значениями предсказаний цен зависимого инструмента за весь выбранный торговый день. Полученная таблица это pandas.DataFrame, подробнее о данном функционале можно прочесть [здесь](#).



В третьей и четвёртой клетках Jupyter ноутбука Arbitrage.ipynb добавлены примеры возможного дальнейшего использования полученного набора предсказаний. Предсказания сгруппированы по минутам и визуализированы в виде таблицы (клетка 3) и в виде свечного графика (клетка 4). При желании эксперты имеют возможность провести собственный анализ таблицы со значениями предсказаний, используя стандартный инструмент Jupyter notebook.

The screenshot shows a Jupyter Notebook cell with the following code and output:

```
[3]: grouped = prices.groupby(prices['time'].str[:5]).agg({'price': ['first', 'min', 'max', 'last']})
grouped
```

time	first	min	max	last
10:05	17026.000000	17020.667281	17028.697734	17022.947244
10:06	17022.741080	17020.226588	17024.358676	17024.104584
10:07	17023.732555	17023.443298	17031.499810	17031.499810
10:08	17030.794720	17025.966208	17032.260653	17025.966208
10:09	17025.927595	17017.966542	17025.927595	17018.341542
18:45	16992.286180	16992.243981	17838.171328	17081.105864
18:46	17077.910840	16990.974638	17078.143369	16990.974638
18:47	16990.937562	16990.552412	16994.605385	16994.443837
18:48	16994.201315	16993.108186	16994.389784	16993.108186
18:49	16993.043394	16111.965174	16995.702139	16111.965174

```
519 rows x 4 columns
```

The screenshot shows a Jupyter Notebook cell with the following code and output:

```
[4]: import matplotlib.pyplot as plt
def candlestick_chart(prices):
    plt.figure()

    width = .4
    width2 = .05

    up = prices[prices['last'] >= prices['first']]
    down = prices[prices['last'] < prices['first']]

    col1 = 'green'
    col2 = 'red'

    plt.bar(prices.index, 0, width, bottom=prices['first'], color=col1)

    plt.bar(up.index, up['last'] - up['first'], width, bottom=up['first'], color=col1)
    plt.bar(down.index, down['last'] - down['first'], width, bottom=down['first'], color=col2)
    plt.bar(up.index, up['min'] - up['first'], width2, bottom=up['first'], color=col1)
    plt.bar(down.index, down['max'] - down['first'], width2, bottom=down['first'], color=col2)
    plt.bar(down.index, down['min'] - down['last'], width2, bottom=down['last'], color=col2)

    plt.xticks(rotation=60, ha='right')
    plt.show()

candlestick_chart(grouped[grouped.index >= '12:00'] & [grouped.index <= '12:30']).price
```

