1. **ПЕРЕЧЕНЬ СЕРВИСОВ**

|  |  |
| --- | --- |
| **Скрипт** | **Наименование сервиса** |
| data\_markup.py | Оценка потенциальной доходности торговли и инвестиций. |
| data\_gen.py | Генерация датасетов |
| edu\_neural.py | Обучение нейронных сетей |
| calc\_profit.py | Оценка доходности нейронных сетей (бек тесты) |
| calc\_signals.py | Генерация торговых и инвестиционных сигналов |

1. **Оценка потенциальной доходности торговли и инвестиций.**

Входные данные:

|  |  |  |
| --- | --- | --- |
| Наименование параметра | Пример | Описание |
| --config\_file |  | Параметр указывается, если данные для сервиса передаются через конфиг |
| --config\_path | /app/cfg.json | Путь до файла конфигурации сервиса |
| --cmd\_config |  | Параметр указывается, если данные для сервиса передаются командную строку |
| --task\_id | 123 | Числовое значение идентификатора задачи |
| --ticker | SBER | Идентификатор финансового инструмента |
| --timeframe | 10m, 1D | Таймфрейм |
| --start\_date | 2007-01-01 | Дата начала получаемых для анализа данных |
| --end\_date | 2023-11-26 | Дата окончания получаемых для анализа данных |
| --count\_points | 5 | Параметр разметки тренда. Данное число показывает, что точка является экстремумом для последующих N точек |
| --extr\_bar\_count | 20 | Число размечаемых в тренде баров как сигнальных к покупке или продаже |

Выходные данные:

Выходные данные генерируются в JSON формате и передаются в API по итогам работы сервиса.

Пример результата:



Структура json:

* task\_id (id задачи)
* markup (разметка):
  + description (описания)
  + values (данные графика разметки)
    - Datetime
    - Open
    - Close
    - High
    - Low
    - Volume
    - Extrems (точки экстремума)
    - Trend (тренды: 0 – продажа, 1 - покупка)
    - Singals (сигналы: 0 – sell, 1 – hold, 2 - buy)
* profit\_without\_shift (теоретическая доходность по разметке):
  + description
  + value
* profit\_with\_shift (доходность при смещении точек открытия и закрытия позиции на 1 бар):
  + description
  + value

1. **Сервис генерации датасетов**

Входные данные:

|  |  |  |
| --- | --- | --- |
| Наименование параметра | Пример | Описание |
| --config\_file |  | Параметр указывается, если данные для сервиса передаются через конфиг |
| --config\_path | /app/cfg.json | Путь до файла конфигурации сервиса |
| --cmd\_config |  | Параметр указывается, если данные для сервиса передаются командную строку |
| --task\_id | 123 | Числовое значение идентификатора задачи |
| --timeframe | 10m, 1D | Таймфрейм |
| --start\_date | 2007-01-01 | Дата начала получаемых для анализа данных |
| --end\_date | 2023-11-26 | Дата окончания получаемых для анализа данных |
| --count\_points | 5 | Параметр разметки тренда. Данное число показывает, что точка является экстремумом для последующих N точек |
| --extr\_bar\_count | 20 | Число размечаемых в тренде баров как сигнальных к покупке или продаже |
| --size\_df | 2 | Размер генерируемого датасета в Гб |
| --max\_unmark | 0.33 | Какую долю конца тренда размечаем как hold |
| --data\_path | app/data | Где сохраняем датасет |

Выходные данные:

Выходные данные генерируются в JSON формате и передаются в API по итогам работы сервиса.

Пример json:



Структура json:

* task\_id (id задачи)
* status (done – датасет сгенерирован)

1. **Сервис обучения нейронных сетей**

Входные данные:

|  |  |  |
| --- | --- | --- |
| Наименование параметра | Пример | Описание |
| --config\_file |  | Параметр указывается, если данные для сервиса передаются через конфиг |
| --config\_path | /app/cfg.json | Путь до файла конфигурации сервиса |
| --cmd\_config |  | Параметр указывается, если данные для сервиса передаются командную строку |
| --task\_id | 123 | Числовое значение идентификатора задачи |
| --data\_path | app/data | Откуда загружаем датасет для обучения |
| --scaler\_path | app/scalers | Куда сохраняем scaler (необходим для масштабирования данных) |
| --neural\_path | app/neurals | Куда сохраняем нейронные сети |
| --new\_model\_flag | True | False | Обучаем ли новую модель (True), False - дообучение старой модели |
| --learning\_rate | 0.00001 | Скорость обучения |
| --epochs | 350 | Число эпох |
| --steps\_per\_epoch | 128 | Число шагов да эпоху обучения |
| --validation\_steps | 128 | Число шагов на валидацию |

Выходные данные:

Выходные данные генерируются в JSON формате и передаются в API по итогам работы сервиса.

Пример json:



Пример структуры:

result = {

'task\_id': task\_id,

'edu\_graph\_losses': {

'loss': {

'description': 'Loss тренировочной выборки',

'values': his.history['loss']

},

'val\_loss': {

'description': 'Loss валидационной выборки',

'values': his.history['val\_loss']

}

},

'train\_accuracy\_score': {

'description': 'Метрика точности accuracy тренировочной выборки',

'values': train\_accuracy\_score

},

'train\_roc\_auc\_score': {

'description': 'Метрика точности roc\_auc тренировочной выборки',

'values': train\_roc\_auc\_score

},

'train\_precision\_score': {

'description': 'Метрика точности precision тренировочной выборки',

'values': train\_precision\_score

},

'train\_recall\_score': {

'description': 'Метрика точности recall тренировочной выборки',

'values': train\_recall\_score

},

'train\_f1\_score': {

'description': 'Метрика точности f1 тренировочной выборки',

'values': train\_f1\_score

},

'train\_log\_loss': {

'description': 'Метрика точности log\_loss тренировочной выборки',

'values': train\_log\_loss

},

'test\_accuracy\_score': {

'description': 'Метрика точности accuracy тестовой выборки',

'values': test\_accuracy\_score

},

'test\_roc\_auc\_score': {

'description': 'Метрика точности roc\_auc тестовой выборки',

'values': test\_roc\_auc\_score

},

'test\_precision\_score': {

'description': 'Метрика точности precision тестовой выборки',

'values': test\_precision\_score

},

'test\_recall\_score': {

'description': 'Метрика точности recall тестовой выборки',

'values': test\_recall\_score

},

'test\_f1\_score': {

'description': 'Метрика точности f1 тестовой выборки',

'values': test\_f1\_score

},

'test\_log\_loss': {

'description': 'Метрика точности log тестовой выборки',

'values': test\_log\_loss

}

}

1. **Сервис оценки доходности нейронных сетей / бек тесты**

Входные данные:

|  |  |  |
| --- | --- | --- |
| Наименование параметра | Пример | Описание |
| --config\_file |  | Параметр указывается, если данные для сервиса передаются через конфиг |
| --config\_path | /app/cfg.json | Путь до файла конфигурации сервиса |
| --cmd\_config |  | Параметр указывается, если данные для сервиса передаются командную строку |
| --task\_id | 123 | Числовое значение идентификатора задачи |
| --ticker | SBER | Идентификатор финансового инструмента |
| --timeframe | 10m, 1D | Таймфрейм |
| --start\_date | 2007-01-01 | Дата начала получаемых для анализа данных |
| --end\_date | 2023-11-26 | Дата окончания получаемых для анализа данных |
| --count\_points | 5 | Параметр разметки тренда. Данное число показывает, что точка является экстремумом для последующих N точек |
| --extr\_bar\_count | 20 | Число размечаемых в тренде баров как сигнальных к покупке или продаже |
| --data\_path | app/data | Откуда загружаем датасет для обучения |
| --scaler\_path | app/scalers | Куда сохраняем scaler (необходим для масштабирования данных) |
| --neural\_path | app/neurals | Куда сохраняем нейронные сети |

Выходные значения:

Выходные данные генерируются в JSON формате и передаются в API по итогам работы сервиса.

Пример:



Структура:

result = {

'task\_id': task\_id,

'singals\_example': {

'markup\_signals': {

'description': 'Сигнлаы про разметке',

'values': str(list(ansamble\_signals\_temp[2][-len\_dataset:]))

},

'neural\_signals': {

'description': 'Сигнлаы нейронной сети',

'values': str(list(ansamble\_signals\_temp[0][-len\_dataset:]))

},

'neural\_trends': {

'description': 'Тренды нейронной сети',

'values': str(list(ansamble\_signals\_temp[1][-len\_dataset:]))

}

},

'test\_accuracy\_score': {

'description': 'Показатели точности датасета по метрике accuracy',

'value': test\_accuracy\_score

},

'test\_roc\_auc\_score': {

'description': 'Показатели точности датасета по метрике roc\_auc',

'value': test\_roc\_auc\_score

},

'test\_precision\_score': {

'description': 'Показатели точности датасета по метрике precision',

'value': test\_precision\_score

},

'test\_recall\_score': {

'description': 'Показатели точности датасета по метрике recall',

'value': test\_recall\_score

},

'test\_f1\_score': {

'description': 'Показатели точности датасета по метрике f1',

'value': test\_f1\_score

},

'test\_log\_loss': {

'description': 'Показатели точности датасета по метрике log\_loss',

'value': test\_log\_loss

},

'data\_std': {

'description': 'Стандартное отклонение по всей выборке',

'value': results\_ideal\_strategy['std']

},

'max\_risk': {

'description': 'Максимальная просадка',

'value': results\_ideal\_strategy['max\_risk']

},

'buy\_hold\_std': {

'description': 'Стандартное отклонение стратегии buy&hold',

'value': results\_ideal\_strategy['buy\_hold\_std']

},

'buy\_hold\_sharp': {

'description': 'Коэффициент шарпа стратегии buy&hold',

'value': results\_ideal\_strategy['buy\_hold\_sharp']

},

'ideal\_strategy\_profit\_without\_shift': {

'description': 'Доходность стратегии по разметке',

'value': results\_ideal\_strategy['strategy\_profit\_without\_shift']

},

'ideal\_strategy\_profit\_with\_shift': {

'description': 'Доходность стратегии по разметке со мещением на 1 бар',

'value': results\_ideal\_strategy['strategy\_profit\_with\_shift']

},

'ideal\_strategy\_std\_without\_shift': {

'description': 'Стандартное отклонение стратегии по разметке',

'value': results\_ideal\_strategy['strategy\_std\_without\_shift']

},

'ideal\_strategy\_std\_with\_shift': {

'description': 'Стандартное отклонение стратегии по разметке со мещением на 1 бар',

'value': results\_ideal\_strategy['strategy\_std\_with\_shift']

},

'ideal\_strategy\_sharp\_without\_shift': {

'description': 'Коэффициент Шарпа стратегии по разметке',

'value': results\_ideal\_strategy['strategy\_sharp\_without\_shift']

},

'ideal\_strategy\_sharp\_with\_shift': {

'description': 'Коэффициент Шарпа стратегии по разметке со мещением на 1 бар',

'value': results\_ideal\_strategy['strategy\_sharp\_with\_shift']

},

'ideal\_strategy\_trade\_count\_without\_shift': {

'description': 'Число сделок стратегии по разметке',

'value': results\_ideal\_strategy['strategy\_trade\_count\_without\_shift']

},

'ideal\_strategy\_trade\_count\_with\_shift': {

'description': 'Число сделок стратегии по разметке со мещением на 1 бар',

'value': results\_ideal\_strategy['strategy\_trade\_count\_with\_shift']

},

'neural\_strategy\_profit\_without\_shift': {

'description': 'Доходность стратегии нейронной сети',

'value': results\_calc\_strategy['strategy\_profit\_without\_shift']

},

'neural\_strategy\_profit\_with\_shift': {

'description': 'Доходность стратегии нейронной сети при смещении на 1 бар',

'value': results\_calc\_strategy['strategy\_profit\_with\_shift']

},

'neural\_strategy\_std\_without\_shift': {

'description': 'Стандартное отклонение стратегии нейронной сети',

'value': results\_calc\_strategy['strategy\_std\_without\_shift']

},

'neural\_strategy\_std\_with\_shift': {

'description': 'Стандартное отклонение стратегии нейронной сети при смещении на 1 бар',

'value': results\_calc\_strategy['strategy\_std\_with\_shift']

},

'neural\_strategy\_sharp\_without\_shift': {

'description': 'Коэффициент Шарпа стратегии нейронной сети',

'value': results\_calc\_strategy['strategy\_sharp\_without\_shift']

},

'neural\_strategy\_sharp\_with\_shift': {

'description': 'Коэффициент Шарпа стратегии нейронной сети при смещении на 1 бар',

'value': results\_calc\_strategy['strategy\_sharp\_with\_shift']

},

'neural\_strategy\_trade\_count\_without\_shift': {

'description': 'Число сделок стратегии нейронной сети',

'value': results\_calc\_strategy['strategy\_trade\_count\_without\_shift']

},

'neural\_strategy\_trade\_count\_with\_shift': {

'description': 'Число сделок стратегии нейронной сети при смещении на 1 бар',

'value': results\_calc\_strategy['strategy\_trade\_count\_with\_shift']

},

'dyn\_ideal\_trading': {

'description': 'Динамика доходности стратегии по разметке',

'value': result\_ideal\_df.to\_json()

},

'dyn\_neural\_trading': {

'description': 'Динамика доходности нейронной сети',

'value': result\_calc\_df.to\_json()

}

}

1. **Сервис генерации сигналов**

Входные данные:

|  |  |  |
| --- | --- | --- |
| Наименование параметра | Пример | Описание |
| --config\_file |  | Параметр указывается, если данные для сервиса передаются через конфиг |
| --config\_path | /app/cfg.json | Путь до файла конфигурации сервиса |
| --cmd\_config |  | Параметр указывается, если данные для сервиса передаются командную строку |
| --task\_id | 123 | Числовое значение идентификатора задачи |
| --tickers | ["SBER", "ABIO"] | Перечень финансовых инструментов для расчёта сигналов |
| --timeframe | 10m, 1D | Таймфрейм |
| --start\_date | 2007-01-01 | Дата начала получаемых для анализа данных |
| --end\_date | 2023-11-26 | Дата окончания получаемых для анализа данных |
| --count\_points | 5 | Параметр разметки тренда. Данное число показывает, что точка является экстремумом для последующих N точек |
| --extr\_bar\_count | 20 | Число размечаемых в тренде баров как сигнальных к покупке или продаже |
| --data\_path | app/data | Откуда загружаем датасет для обучения |
| --scaler\_path | app/scalers | Куда сохраняем scaler (необходим для масштабирования данных) |
| --neural\_path | app/neurals | Куда сохраняем нейронные сети |

Выходные данные:

Выходные сигналы записываются в БАЗУ ДАННЫХ

Структура:

SELECT

id,

task\_id,

ticker,

signal,

signal\_position,

created\_at

FROM public.cals\_signals\_results;

