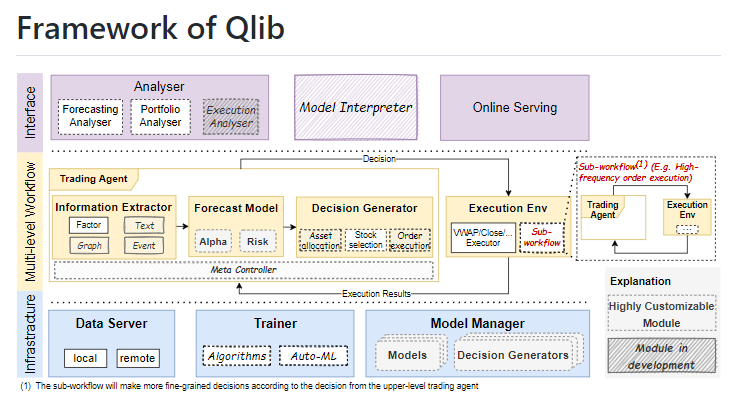
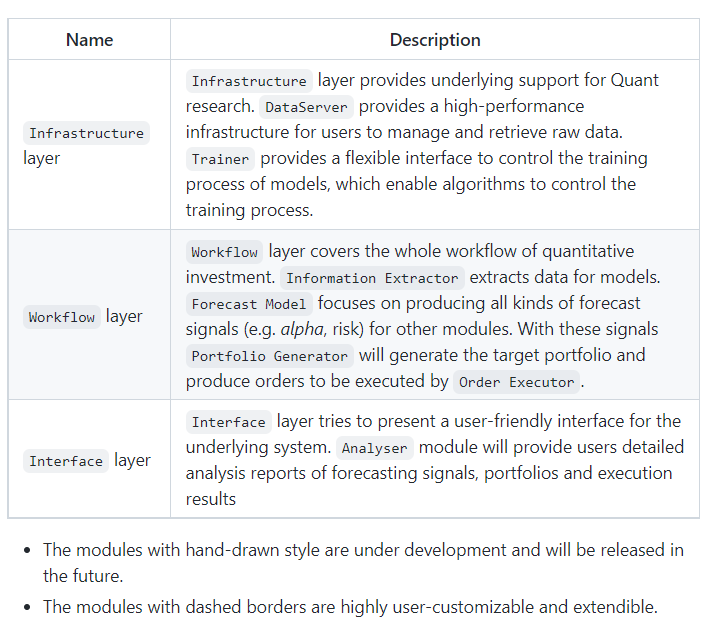
10.26, 10,27, 10,28，10.29，10.30，10.31, 11.1

# Qlib学习：

## Qlib架构：





## Qlib 初始化：

**import** **qlib***# region in [REG\_CN, REG\_US]*

**from** **qlib.config** **import** REG\_CN

provider\_uri = "~/.qlib/qlib\_data/cn\_data" *# target\_dir*

qlib.init(provider\_uri=provider\_uri, region=REG\_CN)

Parameters

Besides provider\_uri and region, qlib.init has other parameters. The following are several important parameters of qlib.init:

provider\_uri

Type: str. The URI of the Qlib data. For example, it could be the location where the data loaded by get\_data.py are stored.

region

Type: str, optional parameter(default: qlib.config.REG\_CN).

Currently: qlib.config.REG\_US (‘us’) and qlib.config.REG\_CN (‘cn’) is supported. Different value of region will result in different stock market mode. - qlib.config.REG\_US: US stock market. - qlib.config.REG\_CN: China stock market.

Different modes will result in different trading limitations and costs.

exp\_manager

Type: dict, optional parameter, the setting of experiment manager to be used in qlib. Users can specify an experiment manager class, as well as the tracking URI for all the experiments. However, please be aware that we only support input of a dictionary in the following style for exp\_manager. For more information about exp\_manager, users can refer to Recorder: Experiment Management.

*# For example, if you want to set your tracking\_uri to a <specific folder>, you can initialize qlib below*qlib.init(provider\_uri=provider\_uri, region=REG\_CN, exp\_manager= {

"class": "MLflowExpManager",

"module\_path": "qlib.workflow.expm",

"kwargs": {

"uri": "python\_execution\_path/mlruns",

"default\_exp\_name": "Experiment",

}

})

mongo

Type: dict, optional parameter, the setting of MongoDB which will be used in some features such as Task Management, with high performance and clustered processing. Users need to follow the steps in installation to install MongoDB firstly and then access it via a URI. Users can access mongodb with credential by setting “task\_url” to a string like “mongodb://%s:%s@%s” % (user, pwd, host + “:” + port).

*# For example, you can initialize qlib below*qlib.init(provider\_uri=provider\_uri, region=REG\_CN, mongo={

"task\_url": "mongodb://localhost:27017/", *# your mongo url*

"task\_db\_name": "rolling\_db", *# the database name of Task Management*})

为了获取数据，用户需要先Qlib用qlib.init进行初始化。请参考初始化。

如果用户按照初始化步骤下载数据，则应使用以下代码初始化qlib

>> import qlib

>> qlib.init(provider\_uri='~/.qlib/qlib\_data/cn\_data')

加载具有给定时间范围和频率的交易日历：

>> **from** **qlib.data** **import** D

>> D.calendar(start\_time='2010-01-01', end\_time='2017-12-31', freq='day')[:2][Timestamp('2010-01-04 00:00:00'), Timestamp('2010-01-05 00:00:00')]

将给定的市场名称解析为股票池配置：

>> **from** **qlib.data** **import** D

>> D.instruments(market='all')

{'market': 'all', 'filter\_pipe': []}

在给定的时间范围内加载特定股票池的工具：

>> **from** **qlib.data** **import** D

>> instruments = D.instruments(market='csi300')

>> D.list\_instruments(instruments=instruments, start\_time='2010-01-01', end\_time='2017-12-31', as\_list=**True**)[:6]['SH600036', 'SH600110', 'SH600087', 'SH600900', 'SH600089', 'SZ000912']

根据名称过滤器从基础市场加载动态工具

>> **from** **qlib.data** **import** D

>> **from** **qlib.data.filter** **import** NameDFilter

>> nameDFilter = NameDFilter(name\_rule\_re='SH[0-9]*{4}*55')

>> instruments = D.instruments(market='csi300', filter\_pipe=[nameDFilter])

>> D.list\_instruments(instruments=instruments, start\_time='2015-01-01', end\_time='2016-02-15', as\_list=**True**)['SH600655', 'SH601555']

在给定时间范围内加载某些工具的特征：

>> from qlib.data import D

>> instruments = ['SH600000']

>> fields = ['$close', '$volume', 'Ref($close, 1)', 'Mean($close, 3)', '$high-$low']

>> D.features(instruments, fields, start\_time='2010-01-01', end\_time='2017-12-31', freq='day').head()

$close $volume Ref($close, 1) Mean($close, 3) $high-$low

instrument datetime

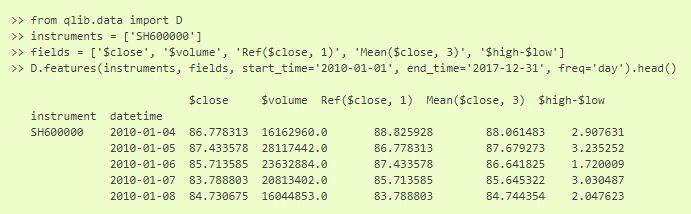
SH600000 2010-01-04 86.778313 16162960.0 88.825928 88.061483 2.907631

2010-01-05 87.433578 28117442.0 86.778313 87.679273 3.235252

2010-01-06 85.713585 23632884.0 87.433578 86.641825 1.720009

2010-01-07 83.788803 20813402.0 85.713585 85.645322 3.030487

2010-01-08 84.730675 16044853.0 83.788803 84.744354 2.047623



在给定的时间范围内加载某些股票池的特征：

启用缓存后，qlib 数据服务器会一直缓存请求的股票池和字段的数据，第一次处理请求可能比没有缓存的时间更长。但是在第一次之后，即使请求的时间段发生变化，具有相同库存池和字段的请求也会命中缓存并得到更快的处理。

从上面可以看到，head( )函数的原型中，默认的参数size大小是 5，所以会返回 5 个数据。

>> from qlib.data import D

>> from qlib.data.filter import NameDFilter, ExpressionDFilter

>> nameDFilter = NameDFilter(name\_rule\_re='SH[0-9]{4}55')

>> expressionDFilter = ExpressionDFilter(rule\_expression='$close>Ref($close,1)')

>> instruments = D.instruments(market='csi300', filter\_pipe=[nameDFilter, expressionDFilter])

>> fields = ['$close', '$volume', 'Ref($close, 1)', 'Mean($close, 3)', '$high-$low']

>> D.features(instruments, fields, start\_time='2010-01-01', end\_time='2017-12-31', freq='day').head()

$close $volume Ref($close, 1) Mean($close, 3) $high-$low

instrument datetime

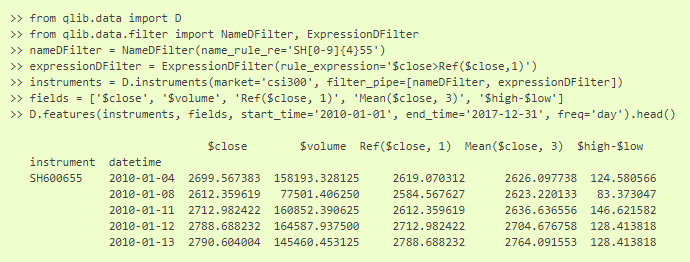
SH600655 2010-01-04 2699.567383 158193.328125 2619.070312 2626.097738 124.580566

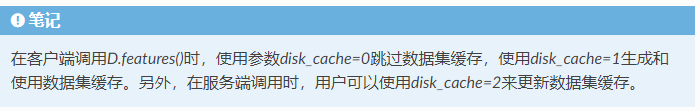
2010-01-08 2612.359619 77501.406250 2584.567627 2623.220133 83.373047

2010-01-11 2712.982422 160852.390625 2612.359619 2636.636556 146.621582

2010-01-12 2788.688232 164587.937500 2712.982422 2704.676758 128.413818

2010-01-13 2790.604004 145460.453125 2788.688232 2764.091553 128.413818





## 数据处理

接下来的就是本地数据处理的内容

介绍包括以下几个部分。Data Layer

* 数据准备
* 数据接口
* 数据加载器
* 数据处理程序
* 数据集
* 缓存
* 数据和缓存文件结构

### 下载中国市场的数据

Qlib has provided an off-the-shelf dataset in .bin format, users could use the script scripts/get\_data.py to download the China-Stock dataset as follows.

*# download 1d*

python scripts/get\_data.py qlib\_data --target\_dir ~/.qlib/qlib\_data/cn\_data --region cn

*# download 1min*

python scripts/get\_data.py qlib\_data --target\_dir ~/.qlib/qlib\_data/qlib\_cn\_1min --region cn --interval 1min

### 下载美国市场的数据;

python scripts/get\_data.py qlib\_data --target\_dir ~/.qlib/qlib\_data/us\_data --region us

After running the above command, users can find china-stock and us-stock data in Qlib format in the ~/.qlib/qlib\_data/cn\_data directory and ~/.qlib/qlib\_data/us\_data directory respectively.

### Qlib提供爬虫

Qlib also provides the scripts in scripts/data\_collector to help users crawl the latest data on the Internet and convert it to qlib format.

### 自动更新每日频率的数据

It is recommended that users update the data manually once (–trading\_date 2021-05-25) and then set it to update automatically.

For more information refer to: yahoo collector

* use crontab: crontab -e
* set up timed tasks:

\* \* \* \* 1-5 python <script path> update\_data\_to\_bin --qlib\_data\_1d\_dir <user data dir>

### 下载每日数据：

Besides downloading the prepared demo data, users could download demo data directly from the Collector as follows for reference to the CSV format. Here are some example:

Qlib has provided the script scripts/dump\_bin.py to convert ****any**** data in CSV format into .bin files (Qlib format) as long as they are in the correct format.

Besides downloading the prepared demo data, users could download demo data directly from the Collector as follows for reference to the CSV format. Here are some example:

for daily data:

### 每日数据：

python scripts/get\_data.py csv\_data\_cn --target\_dir ~/.qlib/csv\_data/cn\_data

### 分钟数据：

python scripts/data\_collector/yahoo/collector.py download\_data --source\_dir ~/.qlib/stock\_data/source/cn\_1min --region CN --start 2021-05-20 --end 2021-05-23 --delay 0.1 --interval 1min --limit\_nums 10

### 很重要的：Converting CSV Format into Qlib Format

Users can also provide their own data in CSV format. However, the CSV data must satisfies following criterions:

CSV file is named after a specific stock or the CSV file includes a column of the stock name

Name the CSV file after a stock: SH600000.csv, AAPL.csv (not case sensitive).

CSV file includes a column of the stock name. User must specify the column name when dumping the data. Here is an example:

python scripts/dump\_bin.py dump\_all ... --symbol\_field\_name symbol

where the data are in the following format:

CSV file must includes a column for the date, and when dumping the data, user must specify the date column name. Here is an example:

python scripts/dump\_bin.py dump\_all ... --date\_field\_name date

where the data are in the following format:

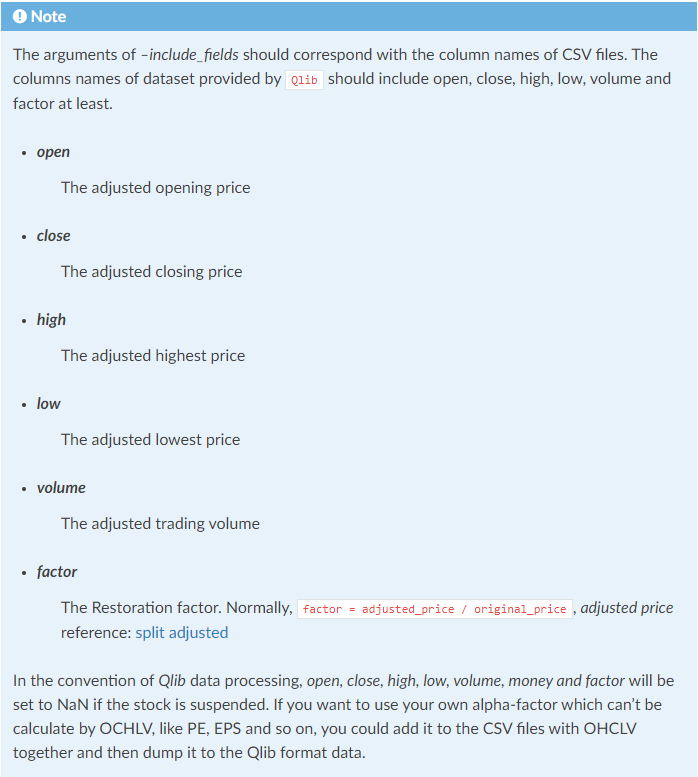
Supposed that users prepare their CSV format data in the directory ~/.qlib/csv\_data/my\_data, they can run the following command to start the conversion.

python scripts/dump\_bin.py dump\_all --csv\_path ~/.qlib/csv\_data/my\_data --qlib\_dir ~/.qlib/qlib\_data/my\_data --include\_fields open,close,high,low,volume,factor

For other supported parameters when dumping the data into .bin file, users can refer to the information by running the following commands:

python dump\_bin.py dump\_all --help

After conversion, users can find their Qlib format data in the directory ~/.qlib/qlib\_data/my\_data.



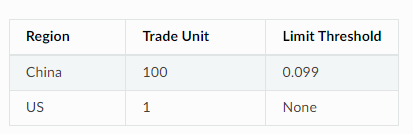
### 股票池设置Stock pool

Qlib defines stock pool as stock list and their date ranges. Predefined stock pools (e.g. csi300) may be imported as follows.

python collector.py --index\_name CSI300 --qlib\_dir <user qlib data dir> --method parse\_instruments

### 两种交易模式Multiple Stock Modes

Qlib now provides two different stock modes for users: China-Stock Mode & US-Stock Mode. Here are some different settings of these two modes:



The trade unit defines the unit number of stocks can be used in a trade, and the limit threshold defines the bound set to the percentage of ups and downs of a stock.

If users use Qlib in china-stock mode, china-stock data is required. Users can use Qlib in china-stock mode according to the following steps:

* Download china-stock in qlib format, please refer to section Qlib Format Dataset.
* Initialize Qlib in china-stock mode

Supposed that users download their Qlib format data in the directory ~/.qlib/qlib\_data/cn\_data. Users only need to initialize Qlib as follows：

**from** **qlib.config** **import** REG\_CN

qlib.init(provider\_uri='~/.qlib/qlib\_data/cn\_data', region=REG\_CN)

If users use Qlib in US-stock mode, US-stock data is required. Qlib also provides a script to download US-stock data. Users can use Qlib in US-stock mode according to the following steps:

* Download us-stock in qlib format, please refer to section Qlib Format Dataset.
* Initialize Qlib in US-stock mode

Supposed that users prepare their Qlib format data in the directory ~/.qlib/qlib\_data/us\_data. Users only need to initialize Qlib as follows.

**from** **qlib.config** **import** REG\_USqlib.init(provider\_uri='~/.qlib/qlib\_data/us\_data', region=REG\_US)

NOTE: PRs for new data source are highly welcome! Users could commit the code to crawl data as a PR like the examples here. And then we will use the code to create data cache on our server which other users could use directly.

## Data API

### Data Retrieval

Users can use APIs in qlib.data to retrieve data, please refer to Data Retrieval.

### Feature

Qlib provides Feature and ExpressionOps to fetch the features according to users’ needs.

Feature

* Load data from the data provider. User can get the features like $high, $low, $open, $close, .etc, which should correspond with the arguments of –include\_fields, please refer to section Converting CSV Format into Qlib Format.

ExpressionOps

* ExpressionOps will use operator for feature construction. To know more about Operator, please refer to Operator API. Also, Qlib supports users to define their own custom Operator, an example has been given in tests/test\_register\_ops.py.

### Filter

<https://qlib.readthedocs.io/en/latest/reference/api.html#module-qlib.data.filter>

Qlib provides NameDFilter and ExpressionDFilter to filter the instruments according to users’ needs.

* NameDFilter

Name dynamic instrument filter. Filter the instruments based on a regulated name format. A name rule regular expression is required.

主要用来筛选相关的股票的连续时间下的价格变动

* ExpressionDFilter

Expression dynamic instrument filter. Filter the instruments based on a certain expression. An expression rule indicating a certain feature field is required.

basic features filter: rule\_expression = ‘$close/$open>5’

cross-sectional features filter : rule\_expression = ‘$rank($close)<10’

time-sequence features filter: rule\_expression = ‘$Ref($close, 3)>100’

主要用来筛选不满足要求的股票

Here is a simple example showing how to use filter in a basic Qlib workflow configuration file:

filter: &filter

filter\_type: ExpressionDFilter

rule\_expression: "Ref($close, -2) / Ref($close, -1) > 1"

filter\_start\_time: 2010-01-01

filter\_end\_time: 2010-01-07

keep: False

data\_handler\_config: &data\_handler\_config

start\_time: 2010-01-01

end\_time: 2021-01-22

fit\_start\_time: 2010-01-01

fit\_end\_time: 2015-12-31

instruments: \*market

filter\_pipe: [\*filter]

### Reference

To know more about Data API, please refer to Data API.

<https://qlib.readthedocs.io/en/latest/reference/api.html#data>

### Data Loader

Data Loader in Qlib is designed to load raw data from the original data source. It will be loaded and used in the Data Handler module.

### QlibDataLoader

### Interface

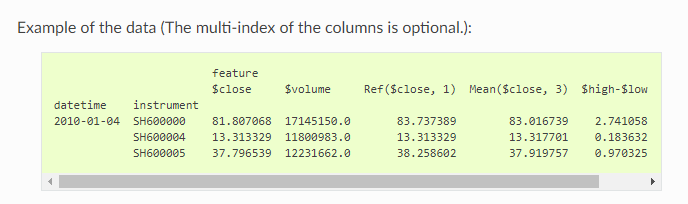
Here are some interfaces of the QlibDataLoader class:

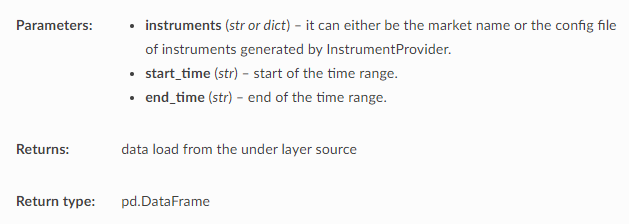
classqlib.data.dataset.loader.DataLoader

DataLoader is designed for loading raw data from original data source.

load(instruments, start\_time=None, end\_time=None) → pandas.core.frame.DataFrame

直接读取成pandas格式，更易于处理





### API

<https://qlib.readthedocs.io/en/latest/reference/api.html#module-qlib.data.dataset.loader>

如果是详细的如何按照特征加载数据看这个API，非常重要，直接可以把本机的原始数据直接读取出来

### Data Handler

The Data Handler module in Qlib is designed to handler those common data processing methods which will be used by most of the models.

Users can use Data Handler in an automatic workflow by qrun, refer to Workflow: Workflow Management for more details.

<https://qlib.readthedocs.io/en/latest/component/workflow.html>

### DataHandlerLP

In addition to use Data Handler in an automatic workflow with qrun, Data Handler can be used as an independent module, by which users can easily preprocess data (standardization, remove NaN, etc.) and build datasets.

用来处理训练的数据集

In order to achieve so, Qlib provides a base class qlib.data.dataset.DataHandlerLP. The core idea of this class is that: we will have some leanable Processors which can learn the parameters of data processing(e.g., parameters for zscore normalization). When new data comes in, these trained Processors can then process the new data and thus processing real-time data in an efficient way becomes possible. More information about Processors will be listed in the next subsection.

### Interface

Here are some important interfaces that DataHandlerLP provides:

<https://qlib.readthedocs.io/en/latest/component/recorder.html>

很重要，用来设计训练集，比如提取每一只股票的时间序列（按照股票的代号）

**classqlib.data.dataset.handler.DataHandlerLP(instruments=None, start\_time=None, end\_time=None, data\_loader: Union[dict, str, qlib.data.dataset.loader.DataLoader] = None, infer\_processors: List[T] = [], learn\_processors: List[T] = [], shared\_processors: List[T] = [], process\_type='append', drop\_raw=False, \*\*kwargs)**

DataHandler with ****(L)earnable (P)rocessor****

Tips to improving the performance of data handler - To reduce the memory cost

* drop\_raw=True: this will modify the data inplace on raw data;

### Processor

The Processor module in Qlib is designed to be learnable and it is responsible for handling data processing such as normalization and drop none/nan features/labels.

很重要，如何去除空白的特征，并且完成归一化

Qlib provides the following Processors:

* DropnaProcessor: processor that drops N/A features.
* DropnaLabel: processor that drops N/A labels.
* TanhProcess: processor that uses tanh to process noise data.
* ProcessInf: processor that handles infinity values, it will be replaces by the mean of the column.
* Fillna: processor that handles N/A values, which will fill the N/A value by 0 or other given number.
* MinMaxNorm: processor that applies min-max normalization.
* ZscoreNorm: processor that applies z-score normalization.
* RobustZScoreNorm: processor that applies robust z-score normalization.
* CSZScoreNorm: processor that applies cross sectional z-score normalization.
* CSRankNorm: processor that applies cross sectional rank normalization.
* CSZFillna: processor that fills N/A values in a cross sectional way by the mean of the column.

Users can also create their own processor by inheriting the base class of Processor. Please refer to the implementation of all the processors for more information (Processor Link).

源代码：

<https://github.com/microsoft/qlib/blob/main/qlib/data/dataset/processor.py>

Example：

我们看一个样例：

Data Handler can be run with qrun by modifying the configuration file, and can also be used as a single module.

Know more about how to run Data Handler with qrun, please refer to Workflow: Workflow Management

Qlib provides implemented data handler Alpha158. The following example shows how to run Alpha158 as a single module.

NOTE: Users need to initialize Qlib with qlib.init first, please refer to initialization.

**import** **qlibfrom** **qlib.contrib.data.handler** **import** Alpha158

data\_handler\_config = {

"start\_time": "2008-01-01",

"end\_time": "2020-08-01",

"fit\_start\_time": "2008-01-01",

"fit\_end\_time": "2014-12-31",

"instruments": "csi300",}

**if** \_\_name\_\_ == "\_\_main\_\_":

qlib.init()

h = Alpha158(\*\*data\_handler\_config)

*# get all the columns of the data*

print(h.get\_cols())

*# fetch all the labels*

print(h.fetch(col\_set="label"))

*# fetch all the features*

print(h.fetch(col\_set="feature"))

NOTE: In the Alpha158, Qlib uses the label Ref($close, -2)/Ref($close, -1) - 1 that means the change from T+1 to T+2, rather than Ref($close, -1)/$close - 1, of which the reason is that when getting the T day close price of a china stock, the stock can be bought on T+1 day and sold on T+2 day.

### Dataset

The Dataset module in Qlib aims to prepare data for model training and inferencing.

The motivation of this module is that we want to maximize the flexibility of of different models to handle data that are suitable for themselves. This module gives the model the flexibility to process their data in an unique way. For instance, models such as GBDT may work well on data that contains nan or None value, while neural networks such as MLP will break down on such data.

If user’s model need process its data in a different way, user could implement his own Dataset class. If the model’s data processing is not special, DatasetH can be used directly.

The DatasetH class is the dataset with Data Handler. Here is the most important interface of the class:

设置自己的数据库：

**classqlib.data.dataset.\_\_init\_\_.DatasetH(handler: Union[Dict[KT, VT], qlib.data.dataset.handler.DataHandler], segments: Dict[str, Tuple], \*\*kwargs)**

Dataset with Data(H)andler

User should try to put the data preprocessing functions into handler. Only following data processing functions should be placed in Dataset:

* The processing is related to specific model.
* The processing is related to data split

**\_\_init\_\_(handler: Union[Dict[KT, VT], qlib.data.dataset.handler.DataHandler], segments: Dict[str, Tuple], \*\*kwargs)**

**Setup the underlying data.**

**Parameters:**

**handler (Union[dict, DataHandler]) –**

**handler could be:**

**instance of DataHandler**

**config of DataHandler. Please refer to DataHandler**

**segments (dict) – Describe the options to segment the data. Here are some examples:**

**config(handler\_kwargs: dict = None, \*\*kwargs)**

Initialize the DatasetH用于初始化数据集

**Parameters:**

**handler\_kwargs (dict) –**

**Config of DataHandler, which could include the following arguments:**

**这里调用DataHandler类用来对数据集进行划分**

**arguments of DataHandler.conf\_data, such as ‘instruments’, ‘start\_time’ and ‘end\_time’.**

**kwargs (dict) –**

**Config of DatasetH, such as**

**segments :dict**

**Config of segments which is same as ‘segments’ in self.\_\_init\_\_**

**setup\_data(handler\_kwargs: dict = None, \*\*kwargs)**

**是否使用缓存**

Setup the Data

|  |  |
| --- | --- |
| **Parameters:** | ****handler\_kwargs**** (dict) –  init arguments of DataHandler, which could include the following arguments:   * init\_type : Init Type of Handler * enable\_cache : whether to enable cache |

**prepare(segments: Union[List[str], Tuple[str], str, slice], col\_set='\_\_all', data\_key='infer', \*\*kwargs) → Union[List[pandas.core.frame.DataFrame], pandas.core.frame.DataFrame]**

Prepare the data for learning and inference.

|  |  |
| --- | --- |
| **Parameters:** | * ****segments**** (Union[List[Text], Tuple[Text], Text, slice]) –   Describe the scope of the data to be prepared Here are some examples:   * + ’train’   + [‘train’, ‘valid’] * ****col\_set**** (str) – The col\_set will be passed to self.handler when fetching data. * ****data\_key**** (str) – The data to fetch: DK\_\* Default is DK\_I, which indicate fetching data for ****inference****. * ****kwargs**** –   **The parameters that kwargs may contain:**  **flt\_col :str**  It only exists in TSDatasetH, can be used to add a column of data(True or False) to filter data. This parameter is only supported when it is an instance of TSDatasetH. |
| **Returns:** |  |
| **Return type:** | Union[List[pd.DataFrame], pd.DataFrame] |
| **Raises:** | NotImplementedError: |

### Cache

**Cache is an optional module that helps accelerate providing data by saving some frequently-used data as cache file. Qlib provides a Memcache class to cache the most-frequently-used data in memory, an inheritable ExpressionCache class, and an inheritable DatasetCache class.**

### Global Memory Cache

Memcache is a global memory cache mechanism that composes of three MemCacheUnit instances to cache Calendar, Instruments, and Features. The MemCache is defined globally in cache.py as H. Users can use H[‘c’], H[‘i’], H[‘f’] to get/set memcache.

classqlib.data.cache.MemCacheUnit(\*args, \*\*kwargs)

Memory Cache Unit.

**\_\_init\_\_(\*args, \*\*kwargs)**

Initialize self. See help(type(self)) for accurate signature.

**limited**

whether memory cache is limited

**classqlib.data.cache.MemCache(mem\_cache\_size\_limit=None, limit\_type='length')**

Memory cache.

**\_\_init\_\_(mem\_cache\_size\_limit=None, limit\_type='length')**

|  |  |
| --- | --- |
| **Parameters:** | * ****mem\_cache\_size\_limit**** (cache max size.) – * ****limit\_type**** (length or sizeof; length(call fun: len), size(call fun: sys.getsizeof)) – |

### ExpressionCache

ExpressionCache is a cache mechanism that saves expressions such as Mean($close, 5). Users can inherit this base class to define their own cache mechanism that saves expressions according to the following steps.

Override self.\_uri method to define how the cache file path is generated

Override self.\_expression method to define what data will be cached and how to cache it.

The following shows the details about the interfaces:

classqlib.data.cache.ExpressionCache(provider)

Expression cache mechanism base class.

This class is used to wrap expression provider with self-defined expression cache mechanism.

可以自己设计Cache的使用

**expression(instrument, field, start\_time, end\_time, freq)**

**update(cache\_uri: Union[str, pathlib.Path], freq: str = 'day')**

### DatasetCache

**DatasetCache is a cache mechanism that saves datasets. A certain dataset is regulated by a stock pool configuration (or a series of instruments, though not recommended), a list of expressions or static feature fields, the start time, and end time for the collected features and the frequency. Users can inherit this base class to define their own cache mechanism that saves datasets according to the following steps.**

* **Override self.\_uri method to define ho their cache file path is generated**
* **Override self.\_expression method to define what data will be cached and how to cache it.**

**The following shows the details about the interfaces:**

**Class qlib.data.cache.DatasetCache(provider)**

**Dataset cache mechanism base class.**

**This class is used to wrap dataset provider with self-defined dataset cache mechanism.**

**dataset(instruments, fields, start\_time=None, end\_time=None, freq='day', disk\_cache=1, inst\_processors=[])**

**update(cache\_uri: Union[str, pathlib.Path], freq: str = 'day')**

Update dataset cache to latest calendar.

Overide this method to define how to update dataset cache corresponding to users’ own cache mechanism.

|  |  |
| --- | --- |
| **Parameters:** | * ****cache\_uri**** (str or Path) – the complete uri of dataset cache file (include dir path). * ****freq**** (str) – |
| **Returns:** | 0(successful update)/ 1(no need to update)/ 2(update failure) |
| **Return type:** | int |

**staticcache\_to\_origin\_data(data, fields)**

cache data to origin data

|  |  |
| --- | --- |
| **Parameters:** | * ****data**** – pd.DataFrame, cache data. * ****fields**** – feature fields. |
| **Returns:** | pd.DataFrame. |

**staticnormalize\_uri\_args(instruments, fields, freq)**

normalize uri args

Qlib has currently provided implemented disk cache DiskDatasetCache which inherits from DatasetCache . The datasets’ data will be stored in the disk.

### Data and Cache File Structure

We’ve specially designed a file structure to manage data and cache, please refer to the [File storage design section in Qlib paper](https://arxiv.org/abs/2009.11189) for detailed information. The file structure of data and cache is listed as follows.

## 现在开始写训练神经网络选股的代码

这篇文章有很强的指导意义

https://finance.sina.com.cn/stock/stockzmt/2020-12-23/doc-iiznezxs8458252.shtml

Alpha158因子库默认的标签定义方式为：股票t日的标签对应t+2日收盘价相对于t+1日收盘价的涨跌幅，相当于t日收盘后发信号，t+1日收盘时刻开仓，t+2日收盘时刻平仓。下面我们展示两种自定义标签方法。

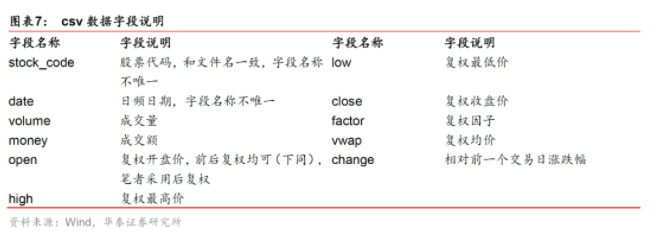
如果希望以vwap价交易，将标签定义为t+2日vwap价相对于t+1日vwap价的涨跌幅，那么可以在设置模型训练参数task时，将task[‘dataset’][‘handler’][’class’]的值从Alpha158改为Alpha158vwap，同时在设置回测参数port\_analysis\_config时，将交易价格deal\_price的值从close改为vwap即可，代码实现如下图。

### 1.数据处理中出现的问题：

股票csv数据需要至少包含下表所示字段。其中有两个“坑”需要注意：

1. 价格数据的要求在官方文档中未提及，笔者建议采用复权价格，原因在于后续数据标注（源码见qlib.contrib.data.handler）采用close或vwap计算股票未来收益率，且回测（源码见qlid.contrib.evaluate.backtest）使用close、open或vwap进行交易。

2. 数据需包含factor或change字段，否则运行Qlib官方提供的策略全流程范例代码examples/workflow\_by\_code.py时，策略收益和净值将出现异常。



## LSTM代码

我们看一下Qlib的代码通过代码了解一下架构，以及每一个class的意义：

### 首先提出的问题：

1. 这个神经网络是如何选股的，也就是如何设计的数据集？
2. 神经网络采用的什么回测方式
3. 是否有使用future infomation
4. 是否可以不断地更新神经网络
5. 评估这个策略的性能指标是什么？

### 神经网络初始化：

scripts\_dir = Path.cwd().parent.joinpath("scripts")

if not scripts\_dir.joinpath("get\_data.py").exists():

    # download get\_data.py script

    scripts\_dir = Path("~/tmp/qlib\_code/scripts").expanduser().resolve()

    scripts\_dir.mkdir(parents=True, exist\_ok=True)

    import requests

    with requests.get("https://raw.githubusercontent.com/microsoft/qlib/main/scripts/get\_data.py") as resp:

        with open(scripts\_dir.joinpath("get\_data.py"), "wb") as fp:

            fp.write(resp.content)

学习：

1. pathlib 获取当前文件路径

from pathlib import Path

v = Path.cwd()

# 输出 F:\moocInterafce

os即operating system（操作系统），Python 的 os 模块封装了常见的文件和目录操作。

1. os.path模块主要用于文件的属性获取,exists是“存在”的意思，所以顾名思义，os.path.exists()就是判断括号里的文件是否存在的意思，括号内的可以是文件路径。

举个栗子：

user.py为存在于当前目录的一个文件

1. 这一块可以在初始化数据集的时候使用

### 安装一些支持包：

! pip install catboost

! pip install --upgrade numpy

! pip install distributed==2021.9.0

! pip install LGBModel

### 并加入包依赖

import qlib

import pandas as pd

from qlib.config import REG\_CN

#配置数据集

from qlib.contrib.model.gbdt import LGBModel

#Forecast Model is designed to make the prediction score about stocks.

使用LGBModel

from qlib.contrib.data.handler import Alpha158

设置数据库

from qlib.contrib.evaluate import (

    backtest as normal\_backtest,

    risk\_analysis,

)

载入评估以及风险分析的包

问题：在回测过程中有哪几种：

* ****str\_type**** ('amount', 'weight' or 'dropout') – strategy type: TopkAmountStrategy ,TopkWeightStrategy or TopkDropoutStrategy.

**到时候看一下这一个回测程序**

**多空策略(Long/Short Strategy)，是对冲基金最常见的策略，就是买进一部分股票，卖出另一部分股票，以达到控制风险的目的。**

from qlib.utils import exists\_qlib\_data, init\_instance\_by\_config

Qlib provides a util named: **init\_instance\_by\_config** to initialize any class inside Qlib with the configuration includes the fields: class, module\_path and kwargs.

from qlib.workflow import R

QlibRecorder provides a high level API for users to use the experiment management system. The interfaces are wrapped in the variable R in Qlib, and users can directly use R to interact with the system. The following command shows how to import R in Python:

from qlib.workflow.record\_temp import SignalRecord, PortAnaRecord

The RecordTemp class is a class that enables generate experiment results such as IC and backtest in a certain format. We have provided three different Record Template class:

### 获取数据

from qlib.utils import flatten\_dict

# use default data

# NOTE: need to download data from remote: python scripts/get\_data.py qlib\_data\_cn --target\_dir ~/.qlib/qlib\_data/cn\_data

provider\_uri = "~/.qlib/qlib\_data/cn\_data"  # target\_dir

if not exists\_qlib\_data(provider\_uri):

    print(f"Qlib data is not found in {provider\_uri}")

    sys.path.append(str(scripts\_dir))

    from get\_data import GetData

    GetData().qlib\_data(target\_dir=provider\_uri, region=REG\_CN)

qlib.init(provider\_uri=provider\_uri, region=REG\_CN)

这里获取了中国的A股数据并且对qlib进行初始化：初始化函数

Besides provider\_uri and region, qlib.init has other parameters. The following are several important parameters of qlib.init:

provider\_uri

Type: str. The URI of the Qlib data. For example, it could be the location where the data loaded by get\_data.py are stored.

region

Type: str, optional parameter(default: qlib.config.REG\_CN).

Currently: qlib.config.REG\_US (‘us’) and qlib.config.REG\_CN (‘cn’) is supported. Different value of region will result in different stock market mode. - qlib.config.REG\_US: US stock market. - qlib.config.REG\_CN: China stock market.

Different modes will result in different trading limitations and costs.

redis\_host

Type: str, optional parameter(default: “127.0.0.1”), host of redis

The lock and cache mechanism relies on redis.

redis\_port

Type: int, optional parameter(default: 6379), port of redis

设置市场初始参数

market = "csi300"

benchmark = "SH000300"

**ExpressionDFilter**

Expression dynamic instrument filter. Filter the instruments based on a certain expression. An expression rule indicating a certain feature field is required.

* + basic features filter: rule\_expression = ‘$close/$open>5’
  + cross-sectional features filter : rule\_expression = ‘$rank($close)<10’
  + time-sequence features filter: rule\_expression = ‘$Ref($close, 3)>100’

可以使用filter筛选出来一些股票

###################################

### # train model

###################################

data\_handler\_config = {

    "start\_time": "2008-01-01",

    "end\_time": "2020-08-01",

    "fit\_start\_time": "2008-01-01", #这个fit\_start\_time是用来标注训练集的

    "fit\_end\_time": "2014-12-31", #这个fit\_end\_time是用来标注训练集的

    "instruments": market,

}

task = {

    "model": {

        "class": "LGBModel",

        "module\_path": "qlib.contrib.model.gbdt",

        "kwargs": {

            "loss": "mse",

            "colsample\_bytree": 0.8879,

            "learning\_rate": 0.0421,

            "subsample": 0.8789,

            "lambda\_l1": 205.6999,

            "lambda\_l2": 580.9768,

            "max\_depth": 8,

            "num\_leaves": 210,

            "num\_threads": 20,

        },

    },

    "dataset": {

        "class": "DatasetH",

        "module\_path": "qlib.data.dataset",

        "kwargs": {

            "handler": {

                "class": "Alpha158",

                "module\_path": "qlib.contrib.data.handler",

                "kwargs": data\_handler\_config,

            },

            "segments": {

                "train": ("2008-01-01", "2014-12-31"),

                "valid": ("2015-01-01", "2016-12-31"),

                "test": ("2017-01-01", "2020-08-01"),

            },

        },

    },

}

学习：

**dataset**:

**class**: DatasetH

**module\_path**: qlib.data.dataset

**kwargs**:

**handler**:

**class**: Alpha158

**module\_path**: qlib.contrib.data.handler

**kwargs**: \***data\_handler\_config**

**segments**:

**train**: [2008-01-01, 2014-12-31]

**valid**: [2015-01-01, 2016-12-31]

**test**: [2017-01-01, 2020-08-01]

Label是机器学习的标签，Ref($close,-2)是T+2的收盘价，这里使用的label是T+2天的收益率，也就是未来2天的收益率——这是考虑A股市场T+1的机制，在T+1天买入，T+2天卖出。

它们使用一个类来实现,都继承了DataHandlerLP，需要实现两个函数，一个是定义feature,另一个是label。

class Alpha360(DataHandlerLP):

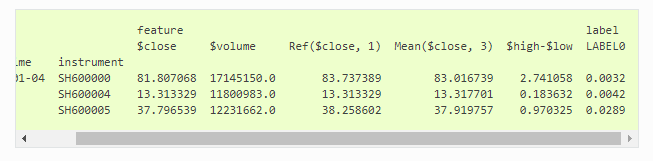
def get\_label\_config(self):

return (["Ref($close, -2)/Ref($close, -1) - 1"], ["LABEL0"])

def get\_feature\_config(self):

pass

这里实际上是预测每只股票的未来两天内的价格增长率，但是只通过当天的数据情况来得出，我们可以看出这样的准确率并不会很高



# model initiaiton

model = init\_instance\_by\_config(task["model"])

dataset = init\_instance\_by\_config(task["dataset"])

# start exp to train model

with R.start(experiment\_name="train\_model"):

    R.log\_params(\*\*flatten\_dict(task))

    model.fit(dataset)

    R.save\_objects(trained\_model=model)

    rid = R.get\_recorder().id

get\_recorder().id的用法

**get\_recorder(\*, recorder\_id=None, recorder\_name=None, experiment\_id=None, experiment\_name=None) → qlib.workflow.recorder.Recorder**

Method for retrieving a recorder.

If active recorder exists:

* + no id or name specified, return the active recorder.
  + if id or name is specified, return the specified recorder.

If active recorder not exists:

* + no id or name specified, raise Error.
  + if id or name is specified, and the corresponding experiment\_name must be given, return the specified recorder. Otherwise, raise Error.

The recorder can be used for further process such as save\_object, load\_object, log\_params, log\_metrics, etc

**储存目标的参数**

**save\_objects(local\_path=None, artifact\_path=None, \*\*kwargs)**

Method for saving objects as artifacts in the experiment to the uri. It supports either saving from a local file/directory, or directly saving objects. User can use valid python’s keywords arguments to specify the object to be saved as well as its name (name: value).

In summary, this API is designs for saving ****objects**** to ****the experiments management backend path****, 1. Qlib provide two methods to specify ****objects**** - Passing in the object directly by passing with \*\*kwargs (e.g. R.save\_objects(trained\_model=model)) - Passing in the local path to the object, i.e. local\_path parameter. 2. artifact\_path represents the ****the experiments management backend path****

* If active recorder exists: it will save the objects through the active recorder.
* If active recorder not exists: the system will create a default experiment, and a new recorder and save objects under it.

*# Case 1***with** R.start(experiment\_name='test'):

pred = model.predict(dataset)

R.save\_objects(\*\*{"pred.pkl": pred}, artifact\_path='prediction')

rid = R.get\_recorder().id...R.get\_recorder(recorder\_id=rid).load\_object("prediction/pred.pkl") *# after saving objects, you can load the previous object with this api*

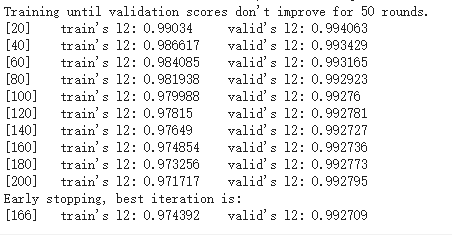
*# Case 2***with** R.start(experiment\_name='test'):

R.save\_objects(local\_path='results/pred.pkl', artifact\_path="prediction")

rid = R.get\_recorder().id

R.get\_recorder().id...R.get\_recorder(recorder\_id=rid).load\_object("prediction/pred.pkl") *# after saving objects, you can load the previous object with this api*

读取后的object预测模型读取



可以看到梯度下降并不是很明显说明这个模型并不是能够很好的收敛

### 下面是回测的部分：

###################################

# prediction, backtest & analysis

###################################

port\_analysis\_config = {

    "executor": {

        "class": "SimulatorExecutor",

        "module\_path": "qlib.backtest.executor",

        "kwargs": {

            "time\_per\_step": "day",

            "generate\_portfolio\_metrics": True,

        },

    },

    "strategy": {

        "class": "TopkDropoutStrategy",

        "module\_path": "qlib.contrib.strategy.model\_strategy",

        "kwargs": {

            "model": model,

            "dataset": dataset,

            "topk": 50,

            "n\_drop": 5,

        },

    },

    "backtest": {

        "start\_time": "2017-01-01",

        "end\_time": "2020-08-01",

        "account": 100000000,

        "benchmark": benchmark,

        "exchange\_kwargs": {

            "freq": "day",

            "limit\_threshold": 0.095,

            "deal\_price": "close",

            "open\_cost": 0.0005,

            "close\_cost": 0.0015,

            "min\_cost": 5,

        },

    },

}

# backtest and analysis

with R.start(experiment\_name="backtest\_analysis"):

    recorder = R.get\_recorder(recorder\_id=rid, experiment\_name="train\_model")

    model = recorder.load\_object("trained\_model")

获取模型参数

    # prediction

    recorder = R.get\_recorder()

    ba\_rid = recorder.id

    sr = SignalRecord(model, dataset, recorder)

    sr.generate()

## **Record Template**

这一个用来生成模型输出信号

The RecordTemp class is a class that enables generate experiment results such as IC and backtest in a certain format. We have provided three different Record Template class:

* **SignalRecord**: This class generates the prediction results of the model.
* SigAnaRecord: This class generates the IC, ICIR, Rank IC and Rank ICIR of the model.

    # backtest & analysis

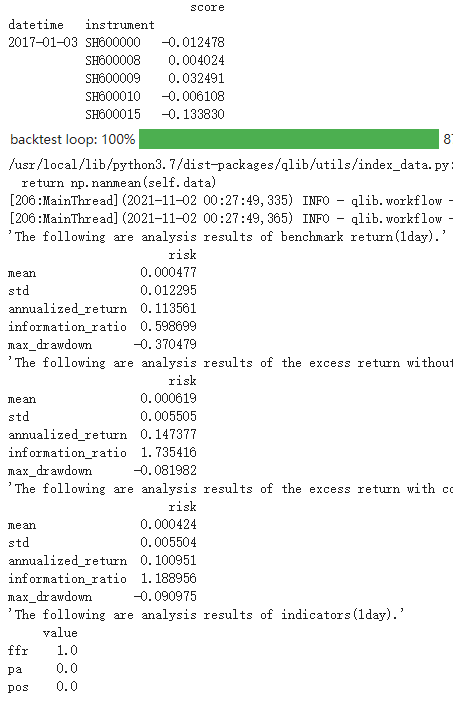
    par = PortAnaRecord(recorder, port\_analysis\_config, "day")

    par.generate()

这一个用来生成回测报告

* PortAnaRecord: This class generates the results of backtest. The detailed information about backtest as well as the available strategy, users can refer to [Strategy](https://qlib.readthedocs.io/en/latest/component/strategy.html) and [Backtest](https://qlib.readthedocs.io/en/latest/component/backtest.html).

Here is a simple exampke of what is done in PortAnaRecord, which users can refer to if they want to do backtest based on their own prediction and label.



### Analyze graphs:

import qlib

from qlib.contrib.report import analysis\_model, analysis\_position

import yaml

#from qlib.contrib.report

import qlib.contrib.report.analysis\_model, qlib.contrib.report.analysis\_position

from qlib.data import D

recorder = R.get\_recorder(recorder\_id=ba\_rid, experiment\_name="backtest\_analysis")

print(recorder)

pred\_df = recorder.load\_object("pred.pkl")

pred\_df\_dates = pred\_df.index.get\_level\_values(level='datetime')

report\_normal\_df = recorder.load\_object("portfolio\_analysis/report\_normal\_1day.pkl")

positions = recorder.load\_object("portfolio\_analysis/positions\_normal\_1day.pkl")

analysis\_df = recorder.load\_object("portfolio\_analysis/port\_analysis\_1day.pkl")

score是预测值

pred\_df.index.get\_level\_values这个是为了获取预测的日期

pred\_df存了预测值



可以看到report\_normal\_df的结构



可以看到analysis\_df的结构

## 搭建一个自己的训练模型

运行程序时遇到的坑：

需要安装Redis

### **2. qlib.data.cache.QlibCacheException：它看到redis锁的key(…)现在已经存在于你的redis db中。**

它看到 redis 锁的密钥现在已经存在于你的 redis 数据库中。您可以使用以下命令清除您的 redis 密钥并重新运行您的命令

**$** redis-cli> select 1> flushdb

如果问题未解决，请使用查找是否存在多个键。如果是这样，请尝试使用清除所有键。keys \*flushall

"""

        "record":[{

        "class": "SignalRecord",

        "module\_path": "qlib.workflow.record\_temp",

        "kwargs":{

            #"model": "<model>",

            #"dataset": "<dataset>",

        }},

        {"class": "SigAnaRecord",

        "module\_path": "qlib.workflow.record\_temp",

        "kwargs":{

            "ana\_long\_short": False,

            "ann\_scaler": 252,

        }},

        {"class": "PortAnaRecord",

        "module\_path": "qlib.workflow.record\_temp",

        "kwargs":{

            "config": port\_analysis\_config,

        }},

    ]

"""

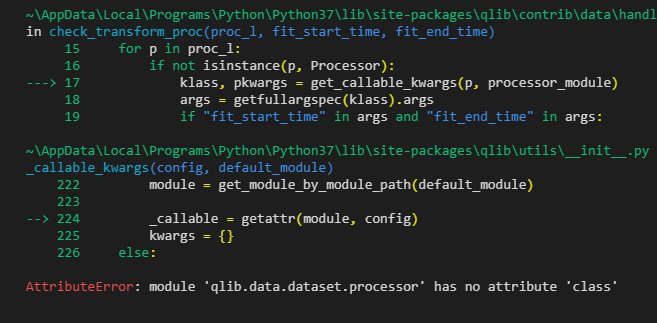
运行示例代码的方法：

<https://github.com/microsoft/qlib>

**classqlib.workflow.record\_temp.SignalRecord(model=None, dataset=None, recorder=None)**

**classqlib.workflow.record\_temp.SigAnaRecord(recorder, ana\_long\_short=False, ann\_scaler=252, label\_col=0)**

**yaml的模块表示什么意思?**

****

**这个需要调整ysml转义到python的代码格式：**

**接下来看一下在分析模型的时候遇到的bug**

## Qlib的业绩归因模块

### 什么是数据归因？

基金研究中经常出现业绩归因这个概念。

业绩归因的上层概念是业绩评价，****业绩评价****作为一个综合体系分为三个要素：

业绩度量

业绩度量是业绩评价体系的基础，研究理论成熟。基于资本资产定价模型（CAPM模型）的三大经典风险调整收益度量法依旧服役。

Treynor(1965)指数：引入投资组合的系统性风险因子β \betaβ来计算单位风险所获得的超额收益，其值越大，承担单位系统风险所获得的超额收益越高，说明投资组合的业绩越好。

Sharpe(1966)指数：将投资组合收益率的标准差作为系统性风险来计算单位风险所获得的超额收益，其值越大，说明投资组合的业绩越好

Jensen(1968)指数：投资组合的实际期望收益率于位于证券市场线上组合的期望收益率之差。此差额则表示投资组合绩效的高低。Jesnsen指数应用最广泛，大多数的业绩归因模型也是基于Jensen指数基础上构建的。

业绩归因

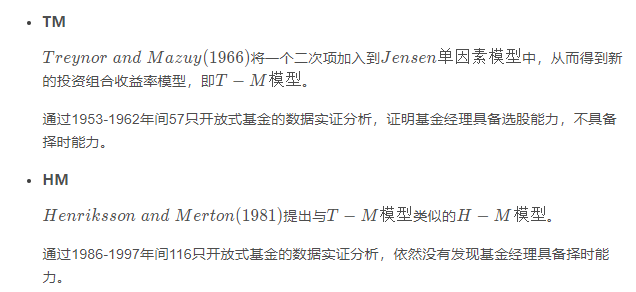
所谓业绩归因，是指将投资组合于基准组合的实际收益率进行比较，并将两个收益率的差额即超额收益进行分解，得到能够解释超额收益来源的各归因项，直观地反映出投资经理的决策行为，从而有针对性的做出改进。

业绩持续性分析。

业绩归因的理论很多，从研究对象角度可分为：

### 股票型基金

普遍将超额收益来源归结为择时能力和选股能力。



TM模型和HM模型，均以Jensen指数为基础，以市场无风险利率为基准，将基金的超额收益来源主要归因于基金经理的择时能力和选股能力。

Jensen指数是建立在CAMP模型基础上的风险调整收益度量，依赖严格的CAPM模型假设，实证分析结果不理想。

C-L

ChangandLewellen(1984)对H − M 模 型 H-M模型H−M模型进行优化，为了避免CAPM模型中理论假设与现实情况不一致所导致的偏差，建立了基于APT套利定价模型的C-L模型。

1. L模型基于APT理论

Brinson

Brinson(1985)等直接抛弃CAPM模型，建立了一个涉及单期及多期收益率的Brinson模型。

BHB

Brinson;Hood;Beebower(1986)Brinson;Hood;Beebower(1986)一起改进模型，形成了B H B 模 型 BHB模型BHB模型。由于脱离了理论假设，目前应用最广，适用于股票型基金和混合型基金。

B H B 模 型 BHB模型BHB模型通过严格的数学推导将基金超额收益的来源分为三个部分：

* 资产配置效应：反映了投资组合和基准组合在各类别资产上配置不同的比例所带来的超额收益
* 证券选择效应：反映了投资组合和基准组合由于选择不同的证券所带来的超额收益
* 交互效应：资产配置效应和政权选择效应的相互影响。

FamaandFrench(1993)改进传统的CAPM模型，提出基于市场收益率、规模、账面与市场价值比例这三个因素的三因素模型，2014年提升为五因素模型（增加公司的盈利能力、投资模式）

### 混合型基金

基本偏向于股票型基金分析方法

### 债券型基金

由于债券于股票本质属性的差异，模型独立于股票型基金。

* W-T

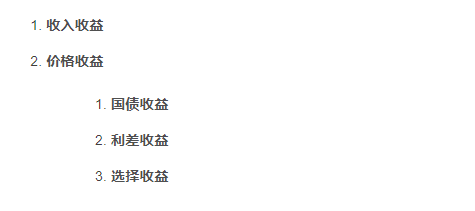
借鉴Fama业绩分解模型，将其中的系统性风险β \betaβ替换为债券的久期，以衡量债 券的系统性风险。

* 加权久期归因模型

结合W-T模型和BHB模型，针对‘自上而下’的投资决策过程提出了加权久期归因模型。

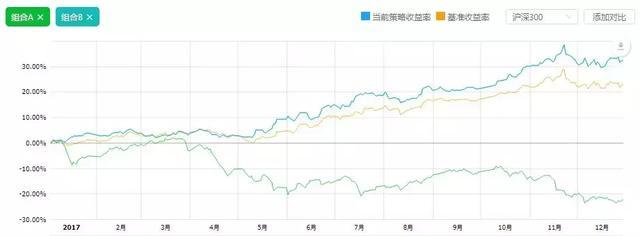
* Campisi

从债券投资组合的收益分解着手，提出了“Campisi框架”。将债券组合的总收益分解为：



现在看一下业绩归因的一些公式帮助理解

比如下面两个投资组合A和B，我们可以看到，在2017年全年，两个组合的表现迥异，相较于黄色的沪深300指数收益曲线，组合B的走势与指数收益更为接近，而组合A则看上去特立独行。对于投资者来说，我们需要知道为什么组合B可以取得相较于基准指数更高的收益，而组合A的巨幅回撤又来自哪里？



△组合A和组合B收益走势情况

数据来源：倍发科技投资研究系统(Betalpha BAR 1.0)

通常来说，常用的归因模型有三类：第一类是业绩拆分，这类拆分需要我们寻找一个基准收益作为对比，考察与基准相比，我们的组合在权重配置和股票选择上存在的差异，并认为这样的差异是我们获得与基准不同收益的原因，这类模型中常见的就是Brinson归因；第二类是因子分析，也就是基于多因子模型，查看组合收益在各个风险因子上的暴露程度，其实就是我们的多因子个股风险模型在组合中的应用；第三类是风格分析，我们对组合的收益率进行回归分析，判断组合偏重的投资风格，比如著名的FF三因子模型就是其中最有代表性的应用。

### 业绩拆分

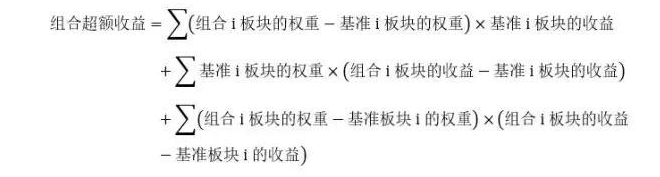
从业界应用来说，以Brinson归因为代表的业绩拆分方法使用最为广泛。这套方法由Brinson、Hood和Beebower在1986年提出，他们的思路是这样的，如果我们要进行投资，首先应当进行基准的设定，然后对比基准中的各个板块权重超配或者低配，最后再在这些板块中进行个股的选择，这一系列选择造成的投资组合与基准组合的差异，就是我们最终超额收益的来源。

首先我们先选择一个业绩基准，它的收益状况是这样的：

等号右侧最开始的那个符号表示连加，所以上面的公式就表示所有板块的加权收益率之和。类似地，我们自己投资组合的收益可以表示为

所以组合相对于基准的超额收益就表示为

很显然，我们通过移项可以把上面式子的右边改写一番，此时要求组合各个板块权重之和为1，各个板块的收益之和为组合的总收益：



这是一个很复杂的公式，不过我们可以按照加号，把等式右边分为三部分。

1. 也就是第一个加号前的部分是我们的板块配置业绩，即在同一个板块上，由于我们在组合中对板块相较基准进行了超配或者低配导致的股票收益，通常在应用中，我们用股票行业来指代板块。问题：这个基准是从哪里来的呢？



比如假设沪深300中银行股的权重是20%，而我们组合的银行股权重是30%，就意味着我们在银行业中超配了10%，这可能会导致我们与沪深300指数的收益率存在差异。（解答）

第二部分是我们的选股业绩，它表示的含义是，在同一板块中，我们对股票权重的设定与基准不同，导致了我们最终收益的差异。



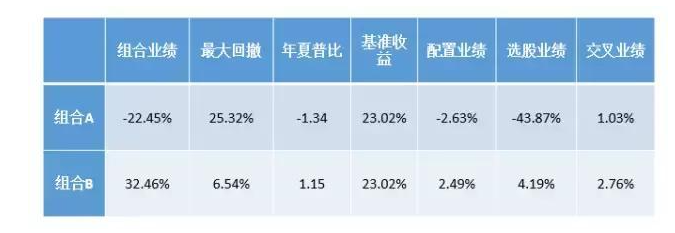
假设我们都按基准进行了板块权重的安排，但是我自己在食品饮料板块的收益是10%，而沪深300在这个行业的板块收益是20%，那么可能的原因就是我在这个行业中个股的配置与基准不一致，比方食品饮料行业中沪深300在贵州茅台上的比例是90%，但是我却在组合中大量配置了某跨省药酒，那就说明我对食品饮料行业的了解有限，跑输大盘在情理之中。

最后一个部分是交叉项，这一项解释起来有点乱，它实际上是选股业绩和配置业绩的一种叠加效应（也就是选择哪只很强的股票，以及配置板块资金的百分比的对抗）。比如你特别牛逼，有非常强的选股能力，在食品饮料行业买入了更多的贵州茅台，并且还在配置中超配了食品饮料业，这是交互项就能给你带来更高的超额收益。反过来如果你在某个热门行业中买错了股票，还低配了这个行业（高配吧？），那自然就会被大盘抛远。



解释了一大通，Brinson归因的意思就是，相较于基准组合，我们自己投资组合的超额收益是由我们在行业权重配置、选股能力以及两者叠加的效应共同决定的。利用Brinson归因就能看到，我们的收益究竟来源于哪里。

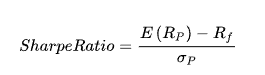
在BAR中，我们对文初的组合A和组合B，进行2017年全年业绩的Brinson归因，其中基准业绩我们选择沪深300指数。从概要上来看，组合A全年的收益为-22.45%，而组合B全年的收益为32.46%。

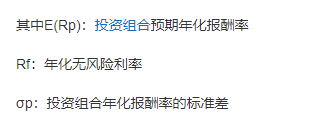


#### 补充：

#### 夏普比率（Sharpe Ratio），

又被称为夏普指数 --- 基金绩效评价标准化指标。夏普比率在现代投资理论的研究表明，风险的大小在决定组合的表现上具有基础性的作用。风险调整后的收益率就是一个可以同时对收益与风险加以考虑的综合指标，长期能够排除风险因素对绩效评估的不利影响。夏普比率就是一个可以同时对收益与风险加以综合考虑的三大经典指标之一。 投资中有一个常规的特点，即投资标的的预期报酬越高，投资人所能忍受的波动风险越高；反之，预期报酬越低，波动风险也越低。所以理性的投资人选择投资标的与投资组合的主要目的为：在固定所能承受的风险下，追求最大的报酬；或在固定的预期报酬下，追求最低的风险。





目的是计算投资组合每承受一单位总风险，会产生多少的超额报酬。比率依据资产配置线(Capital Allocation Line,CAL)的观念而来，是市场上最常见的衡量比率。当投资组合内的资产皆为风险性资产时，适用夏普比率。夏普指数代表投资人每多承担一分风险，可以拿到几分超额报酬；若大于1，代表基金报酬率高过波动风险；若为小于1，代表基金操作风险大过于报酬率。这样一来，每个投资组合都可以计算Sharpe Ratio,即投资回报与多冒风险的比例，这个比例越高，投资组合越佳。

举例而言，假如国债的回报是3%，而你的投资组合预期回报是15%，你的投资组合的标准偏差是6%，那么用15%－3%,可以得出12%（代表超出无风险投资的回报），再用12%/6%=2，代表投资者风险每增长1%，换来的是2%的多余收益。

夏普理论告诉我们，投资时也要比较风险，尽可能用科学的方法以冒小风险来换大回报。所以说，投资者应该成熟起来，尽量避免一些不值得冒的风险。同时在投资时如缺乏投资经验与研究时间，可以让真正的专业人士（不是只会卖金融产品给你的SALER）来帮到你建立起适合自己的，可承受风险最小化的投资组合。这些投资组合可以通过Sharpe Ratio来衡量出风险和回报比例。

#### 特雷诺指数

特雷诺指数（Treynor ratio）用TR表示，是每单位系统性风险获得的风险溢价，是投资者判断某一基金管理者在管理基金过程中所冒风险是否有利于投资者的判断指标。特雷诺指数越大，单位风险溢价越高，开放式基金的绩效越好，基金管理者在管理的过程中所冒风险有利于投资者获利。相反特雷诺指数越小，单位风险溢价越低，开放式基金的绩效越差，基金管理者在管理的过程中所冒风险不有利于投资者获利。

特雷诺指数是对单位风险的超额收益的一种衡量方法。在该指数中,超额收益被定义为基金的投资收益率与同期的无风险收益率之差,该指数计算公式为:

T=（Rp―Rf）/βp

其中：T表示特雷诺业绩指数，Rp表示某只基金的投资考察期内的平均收益率，Rf表示考察期内的平均无风险利率，βp表示某只基金的系统风险。

特雷诺业绩指数的含义就是每单位系统风险资产获得的超额报酬（超过无风险利率Rf）。特雷诺业绩指数越大，基金的表现就越好；反之，基金的表现越差。

#### ****净值类指标、****

基金净值代表基金的价格，象征着盈亏变化，能直观的反应基金的业绩变化，一般净值增加，就代表基金有了盈利。净值指标又分单位净值、累计净值、复权净值。

1. 基金单位净值

基金单位净值指每份基金单位的净资产价值，简单的说就是这个基金每一份的价值，即单价。基金刚成立时，单位净值一般是1。

用公式来表示是：基金单位净值=(总资产-总负债)/基金单位总数。

其中，总资产指基金拥有的所有资产，包括股票、债券、银行存款和其他有价证券等；总负债指基金运作及融资时所形成的负债，包括应付给他人的各项费用、应付资金利息等；

基金单位总数是指当时发行在外的基金单位的总量。

2. 累计净值

基金累计份额净值是指基金份额净值与基金历来分红的累计份额分红金额之和。所谓基金份额净值就是一份基金份额的价值。为了评价基金的投资效益和业绩，评价基金的业绩指标。常常采用基金份额净值和基金累计份额净值来表示相当于一股股票的价格。

累计份额净值是评价基金投资业绩的一个重要指标，一般说来，累计净值越高，表明基金的历史业绩越好。当然，累计份额净值的高低并不是选择基金的主要依据，基金净值未来的成长性才是判断基金投资价值的关键。

用公式来表示是： 基金累计份额净值=基金份额净值+基金累计份额分红金额

例如：一只基金，共经过二次分红，第一次分红为0.5元/份；第二次分红为0.3元/份；现在的基金份额净值是1.2元/份，则 ：

基金份额累计净值=1.2元/份+0.5元/份+0.3元/份=2.0元/份

3. 复权净值

基金复权净值，考虑了红利再投资的因素，对基金的单位净值进行了复权计算。也就是将“分红+单位净值”再投资进行复利计算。

而累计净值是把分红加回单位净值，但不作为再投资计算复利。

针对以上3种净值，我们举个简单的例子进行说明。私募基金A刚成立时(时间点T0)，单位净值为1，此时没有涨跌或分红，累计净值和复权净值也都为1。过了一段时间(时间点T1)，基金进行了分红，分红金额为0.2。不久之后(时间T2)，基金净值上涨了10%。三个时间点的净值可以用下表表示。

注：T2时，单位净值=0.8\*10%=0.88，累计净值=0.88+0.2=1.08，复权净值=0.88+[0.2\*(1+10%)]=1.1

#### 收益类指标

衡量基金收益类的指标有很多，这里我们只选取市场上各平台常用的收益率指标来说。

#### 累计收益

累计收益是指基金从成立开始到至今的总收益，包括分红收益，是用来衡量该基金成立以来的收益情况的指标。

用公式表示是：(最新净值-1)\*100%

#### 年化收益率

年化收益率是指投资期限为一年所获的收益率。该指标把当前收益率换算成年度收益率，是一种理论收益率，并不是真正已取得的收益率。年化收益率货币基金过去七天每万份基金份额净收益折合成的年收益率。货币市场基金存在两种收益结转方式：1. "日日分红，按月结转"，相当于日日单利，月月复利；2."日日分红，按日结转"，相当于日日复利。

用公式表示是：[(投资内收益/本金)/投资天数]\* 365\* 100%

#### 区间收益率

区间收益率是指在既定区间内，基金实现的净值绝对收益，即累计净值增长率。

区间通常可以包括：近一个月、近三个月、近半年、近一年、近两年、近三年、近五年、近十年、成立以来、今年以来等。

用公式表示是：区间收益率=(区间末期累计净值-区间初期累计净值)/区间初期累计净值

#### 平均回报

平均回报反应的是基金收益高低的一个指标，代表的是基金在一定时间内的平均回报率的多少。

不用说，平均回报当然是越高越好，数值越大，说明收益就越高。

#### 阿尔法系数

阿尔法系数是基金的实际收益和按照贝塔系数计算的期望收益之间的差额，即跑赢市场的部分。

公式：基金的收益=市场收益+超额收益。

贝塔系数代表市场收益，阿尔法则代表着超额收益。阿尔法系数越大，超额收益就越大。

阿尔法系数代表着基金有多大程度能跑赢预期收益率，这个数值，我们当然是倾向于越大越好。

#### 卡玛比率

Calmar比率(Calmar Ratio) 描述的是收益和最大回撤之间的关系。计算方式为年化收益率与历史最大回撤之间的比率。（收益与最大风险之间的比）

Calmar比率数值越大，基金的业绩表现越好。

反之，基金的业绩表现越差。

#### 胜率

胜率就是成功次数和失败次数的比值。但是这个指标不能有效地反映真实的盈利水平。举例来说，假设胜率为66.67%，就是说，平均每交易3次中，有2次是盈利的，1次是亏损的。但是如果每次盈利是3%，2次总计盈利2\*3%=6%，1次亏损亏损10%，那么，实际的收益率是6%-10%=-4%，也就是说虽然胜率是66.67%的较高水平，但是实际上投资者是亏损4%。

#### 最大回撤

最大回撤是指基金净值(一般用累计净值)从最大值回落到最小值的幅度。该指标用来描述任一投资者可能面临的的最大亏损。最大回撤是一个重要的风险指标，对于对冲基金和数量化策略交易，该指标比波动率还重要。 也能在一定程度上反映私募基金管理人的风控能力。

用公式表示是：D为某一天的净值，i为某一天，j为i后的某一天，Di为第i天的产品净值，Dj则是Di后面某一天的净值，drawdown就是最大回撤率。

drawdown=max（（Di-Dj）/Di），其实就是对每一个净值进行回撤率求值，然后找出最大的。可以使用程序实现。

举个例子：一只基金刚开始净值是1元，随着市场震荡上行，最高涨到了1.5元。然后，市场开始一路下跌，基金最低跌到了0.8元。随后，市场开始震荡，先是涨到了1.2元，又跌至0.9元，又涨到了1.1元。可以看到，期间亏损幅度最大的时候是从1.5元跌到0.8元，即最大回撤率=(1.5-0.8)/1.5=46.7%!

#### 索提诺比率 (Sortino Ratio)

索提诺比率是一种衡量投资组合相对表现的方法。与夏普比率(Sharpe Ratio)有相似之处，但索提诺比率运用下偏标准差而不是总标准差，以区别不利和有利的波动。

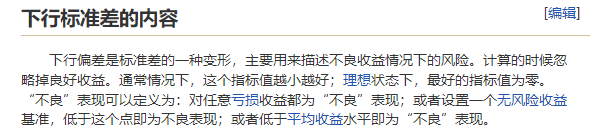
和夏普比率类似，这一比率越高，表明基金承担相同单位下行风险能获得更高的超额回报率。索提诺比率可以看做是夏普比率在衡量对冲基金/私募基金时的一种修正方式。

索提诺比率（Sortino Ratio）与夏普比率类似，所不同的是它区分了波动的好坏，因此在计算波动率时它所采用的不是标准差，而是下行标准差。这其中的隐含条件是投资组合的上涨（正回报率）符合投资人的需求，不应计入风险调整。

索提诺比率区分了波动的好坏，衡量计算期内相同单位下行风险能获得的超额回报率。

用公式表示是： Sortino Ratio=(Rp-Rf)/DR，其中，Rp为平均值，Rf为无风险利率，DR为下行标准差

该比率越高，基金承担单位下行风险得到的超额回报率越高。



#### 贝塔系数

贝塔系数体现的是基金相对于整个市场的波动情况，即基金相对于大盘的偏离程度。贝塔系数是一个相对指标，β越高，波动越大，风险也就越大。

假设某只基金的贝塔系数是1.1，说明上证指数涨10%，基金涨11%；上证指数10％，基金跌11%。

如果贝塔系数为1，则上证指数涨10％，基金涨10％；上证指数跌10％，基金相应跌10％。

如果贝塔系数为0.8，则上证指数涨10％，基金涨8％；上证指数跌 10％，基金相应跌8％。

大家可以看出，贝塔系数大于1，说明基金波动比大盘要大，代表着这只基金的风险比大盘大。如果贝塔系数小于1，说明基金波动比大盘小，风险比大盘小。

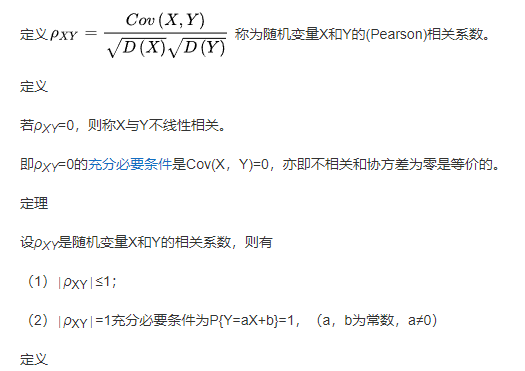
当处在牛市，市场处于上升阶段，可以选高贝塔系数的基金；当处在熊市，市场在下跌阶段，选低贝塔系数的基金。

计算方式

单项资产系统风险用β系数来计量，通过以整个市场作为参照物，用单项资产的风险收益率与整个市场的平均风险收益率作比较，即：

IMG_256 COV（X，Y）=E［（X-E（X））（Y-E（Y））

Β系数的数学意义：就是基金收益与市场收益的线性相关性（倍率关系）



其中Cov(ra,rm)是证券 a 的收益与市场收益的协方差；

是市场收益的方差。其中ρam为证券a与市场的相关系数；σa为证券a的标准差；σm为市场的标准差。

#### 詹森指数

詹森指数是测定证券组合经营绩效的一种指标，是证券组合的实际期望收益率与位于证券市场线上的证券组合的期望收益率之差。1968年，美国经济学家迈克尔·詹森（Michael C. Jensen）发表了《1945-1964年间共同基金的业绩》一文，提出了这个以资本资产定价模型（CAPM）为基础的业绩衡量指数，它能评估基金的业绩优于基准的程度，通过比较考察期基金收益率与由定价模型CAPM得出的预期收益率之差，即基金的实际收益超过它所承受风险对应的预期收益的部分来评价基金，此差额部分就是与基金经理业绩直接相关的收益。

詹森指数又称为阿尔法系数，是衡量基金超额收益大小的一种指标。这一指标综合考虑了基金收益与风险因素，比单纯的考虑基金收益大小要更科学。

詹森指数=Ri,t—[Rf,t+βi(Rm,t-Rft)]

Rm,t为市场投资组合在t时期的收益率；Ri,t为i基金在t时期的收益率；Rf,t为t时期的无风险收益率，βi为基金投资组合所承担的系统风险。詹森指数所代表的就是基金业绩中超过市场基准组合所获得的超额收益。即詹森指数>0，表明基金的业绩表现优于市场基准组合，大得越多，业绩越好；反之，如果詹森指数〈 0，则表明其绩效不好。

#### R平方

R平方反映的是业绩基准的变动对基金表现的影响，数值按0-100进行衡量。

如果R平方为100，说明基金回报变动完全取决于业绩基准变动；如果R平方为50，说明有50%的基金回报来自于业绩基准变动。（这个系数是由谁提供？）

此外，R平方还可以用来确定贝塔系数与阿尔法系数的准确性。一般来说，基金的R平方越大，β和α这两个系数的准确性越高。

#### 年化波动率

年化波动率用来衡量投资标的的波动风险。一般在投资者购买某种理财产品时会碰到。而且在比较不同投资理财产品年化波动率时，最好保持计算方法的一致性。这样可以让用户作出正确的判断。

计算公式: 年化波动率=收益率标准差\*(n^0.5)

其中：计算周期为日，对应n为250；计算周期为周，对应n为52；计算周期为月，对应n为12；计算周期为年，对应n为1。

#### 下行风险

下行风险是指，由于市场环境变化，未来价格走势有可能低于分析师或投资者所预期的目标价位。下行风险是我们投资可能出现的最坏的情况，也是投资者可能需要承担的损失。

不同投资的下行风险各异，有的很小，有的则可能无限大。例如，购买股票就需要承担有限的下行风险，最坏的结果是投资者损失全部的投资成本，即本金。但是，卖空操作的下行风险却是无限的，因为被卖空股票的价格的上升空间是无限的

### Qlib业绩归因函数

Analysis is designed to show the graphical reports of Intraday Trading , which helps users to evaluate and analyse investment portfolios visually. The following are some graphics to view:

analysis\_position

* report\_graph
* score\_ic\_graph
* cumulative\_return\_graph
* risk\_analysis\_graph
* rank\_label\_graph

analysis\_model

* model\_performance\_graph

Turnover

周转率、成交量：. 在证券交易上，指一个投资组合或交易所在一段期间交易股份的数量占总股票数量的百分比

Benchmark

是大盘的一个基准

# backtest and analysis

with R.start(experiment\_name="backtest\_analysis"):

    recorder = R.get\_recorder(recorder\_id=rid, experiment\_name="train\_model")

    model = recorder.load\_object("trained\_model")

    # prediction

    recorder = R.get\_recorder()

    ba\_rid = recorder.id

    sr = SignalRecord(model, dataset, recorder)

    sr.generate()

    # backtest & analysis

    par = PortAnaRecord(recorder, port\_analysis\_config, "day")

    par.generate()

PortAnaRecord主要是负责记录回测的函数

This is the Portfolio Analysis Record class that generates the analysis results such as those of backtest. This class inherits the RecordTemp class.

The following files will be stored in recorder - report\_normal.pkl & positions\_normal.pkl:

The return report and detailed positions of the backtest, returned by qlib/contrib/evaluate.py:backtest

port\_analysis.pkl : The risk analysis of your portfolio, returned by qlib/contrib/evaluate.py:risk\_analysis

simulatorExecutor

表示当前执行器在最内层，这里使用的最内层策略是TWAP策略，框架目前也支持VWAP策略

Qlib provides a implemented strategy classes named TopkDropoutStrategy.

#### 问：什么是VWAP交易策略：（日内交易）

阿齐兹博士的书里讲解了基于VWAP (Volume Weighted Average Price) 成交量加权平均价的日内交易策略。VWAP的一般用法是当机构客户下大额订单时，交易经理用其来评估手下交易员的交易水平，看他当天的成交价是高于或低于当天的VWAP。例如，一个机构当天想买入一百万股的苹果股票，他可不能直接下一个市价单，交易员会在一天之中尝试在股价低于VWAP线的时候分批买入，当一天的交易结束，如果成交均价低于VWAP，这就是一笔成功的交易。

他概括了三种交易策略：

1. VWAP开盘反转：

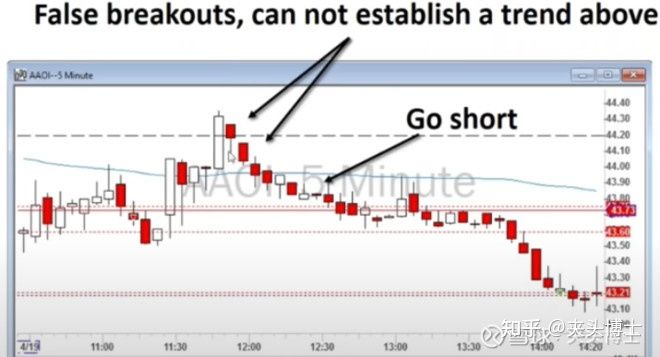
一只弱势股开盘大跌后股价持续低于VWAP线，如果股价没有持续创5分钟K线新低，那么有可能股价会开始向VWAP反转甚至突破VWAP向上。

反之，如果一只强势股开盘大涨后股价没能继续创5分K新高，那也许意味着做空的机会，股价可能靠近VWAP甚至跌破VWAP往下。

上图是京东JD的一个VWAP开盘反转例子，JD开盘大跌，但25分钟后没有继续新低，Aziz在30分钟左右入场做多JD，止损设为5分钟K线的最低点，之后JD股价开始反转，突破VWAP线向上，直到碰到日K线阻力。

2. VWAP假突破：

一只股票交易了半天左右的时候，应该已经呈现出了一天的趋势(持续向上高于VWAP或向下低于VWAP)，然而，如果形势出现逆转，就可能形成一个假突破，有反向操作的机会。



上图中，应用光电AAOI的股价在美东时间中午之前突破了VWAP向上，但很快趋势反转跌破VWAP，此时就形成了一个做空的机会。



上图中，BBBY开盘大跌，之后股价形成假突破，Aziz开始做空，同时把VWAP价设为止损，当股价跌到当日前期低点后逐步止盈。

3. VWAP移动平均线趋势交易：

当一只股票经历了早盘的博弈以后，通常会维持趋势在VWAP以上或以下。散户交易者可以根据移动平均线EMA进行做空和做多操作。



上图中，太阳城SCTY在早盘大涨后股价一直保持在VWAP以上，VWAP线形成了当天很好的支撑位，Aziz在支撑位做多，直到遇到下一级别的日K线阻力位离场。



上图中，脸书FB在早盘出现下跌，但一直维持在VWAP附近，此后股价一路沿着9天移动平均线往上。



上图的JWN是一个很好的例子，同时呈现了VWAP开盘反转、假突破和沿趋势线交易这三种交易策略。

标准VWAP策略原理

标准的VWAP策略是一种静态策略，即在交易开始之前，利用已有信息确定提交策略，交易开始之后按照此策略进行交易，而不考虑交易期间的信息。

需要买入的股票数量记为V，区间的划分与预测交易量分布时一样，并假设已经通过预测技术获得了当天的交易量分布预测值

IMG_256

以1分钟为单位，按照预测的成交比例分配每个区间内的交易量，在区间内再平均分配。

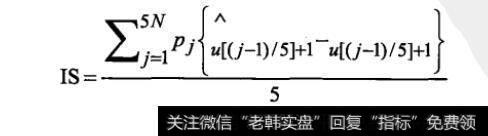
设认

IMG_256

为各分钟的成交价格，市场最终的成交量分布为

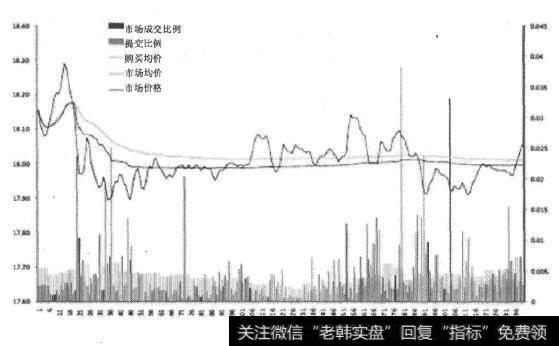


则执行差额定义为决策者的交易均价与市场成交均价的差，可得：



由上式可以看出，VWAP策略的好坏直接受交易量分布预测质量的影响。预测越准确，比较误差越小。但正如已经看到的那样，交易量分布预测很难做到十分精确，从而普通的VWAP策略的执行效果将很难保证。

如图8-2所示是上海机场2010年1月12日的标准VWAP策略执行效果图。从图上可以看出，标准VWAP策略基本上跟随了市场交易量加权均价，但明显比市场均价要差。



标准VWAP策略是一种非常简单的静态策略。它涉及的变量较少，执行比较容易，在正常情况下能够较好地跟随市场成交价格。

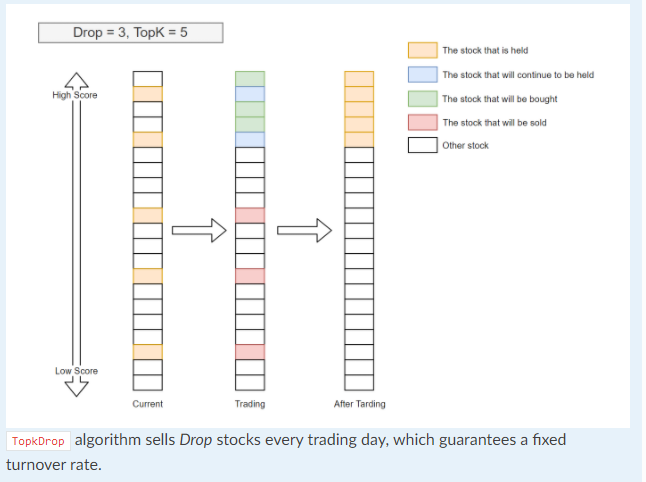
标准的VWAP策略虽然简单易行，但是有两个很明显的缺点：第一是它完全依赖于日内交易量分布预测，如果预测不准确，VWAP策略的执行效果将非常不确定；第二是它是一种完全静态的策略，也就是说在交易开始之前就完成了决策，在交易时间内执行策略即可，没有将市场的最新信息如价格变化、交易量变化等考虑进去，使得它不能更好地适应市场的变化，从而无法获得更好的交易价格。

TopkDropoutStrategy也就是买入卖出的策略，量化要做的是买入的交易点，以及交易的持仓，我个人感觉这两个都是要配合起来的

TopkDropoutStrategy is a subclass of BaseStrategy and implement the interface generate\_order\_list whose process is as follows.

Adopt the Topk-Drop algorithm to calculate the target amount of each stock

* Topk: The number of stocks held
* Drop: The number of stocks sold on each trading day



benchmark = "SH000300"

基准就是沪深300指数

## Qlib的analyse模块

analysis\_model

analysis\_position

* report\_graph
* score\_ic\_graph
* cumulative\_return\_graph
* risk\_analysis\_graph
* rank\_label\_graph

analysis\_model

* model\_performance\_graph

#### Graphical Result

* Axis X: Trading day
* **Axis Y:**

**cum bench（****大盘回报累计）**

Cumulative returns series of benchmark

**cum return wo cost（股票回报累计无交易费）**

Cumulative returns series of portfolio without cost

**cum return w cost（股票回报累计有交易费）**

Cumulative returns series of portfolio with cost

**return wo mdd****（最大回撤无交易费）**

Maximum drawdown series of cumulative return without cost

**return w cost mdd:（最大回撤有交易费）**

Maximum drawdown series of cumulative return with cost

**cum ex return wo cost****（与大盘相比异常的回报无交易费）**

The CAR (cumulative abnormal return) series of the portfolio compared to the benchmark without cost.

**cum ex return w cost（与大盘相比异常的回报有交易费）**

The CAR (cumulative abnormal return) series of the portfolio compared to the benchmark with cost.

**turnover（交易量）**指一个投资组合或交易所在一段期间交易股份的数量占总股票数量的百分比

Turnover rate series

**cum ex return wo cost mdd****（从回报中的借款无花费）**

Drawdown series of CAR (cumulative abnormal return) without cost

**cum ex return w cost mdd（从回报中的借款有花费）**

Drawdown series of CAR (cumulative abnormal return) with cost

The shaded part above: Maximum drawdown corresponding to cum return wo cost（最大的借款对应无费用回报）

The shaded part below: Maximum drawdown corresponding to cum ex return wo cost（最大的借款对应无费用的异常回报）

#### rank\_label\_graph

也就是每日的股票收盘价预测排名

Ranking percentage of stocks buy, sell, and holding on the trading day. Average rank-ratio(similar to sell\_df[‘label’].rank(ascending=False) / len(sell\_df)) of daily trading

对交易日中买入、卖出和持有的股票百分比进行排序。 每日交易的平均排名比率(类似于sell\_df[' label '].rank(ascending=False) / len(sell\_df))

标签：预测价格与今日价格（收盘）的增长值

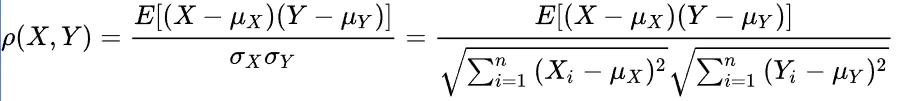
The label T is the change from T to T+1, it is recommended to use close, example: D.features(D.instruments(‘csi500’), [‘Ref($close, -1)/$close-1’])

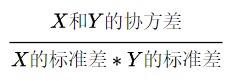
#### ic\_figure.

IC figure

ic

* The Pearson correlation coefficient series between label and prediction score. In the above example, the label is formulated as Ref($close, -1)/$close - 1. Please refer to Data Feature for more details.（也就是皮尔森相关系数，当天实际增长的价格与预测的增长的价格（明天与后天）的线性相关性，说明如果这样的线性相关性越好，那么预测值也就会越准确）
* ：皮尔逊相关系数用于度量两个变量(X和Y)之间的线性相关程度，其值介于-1与1之间。这种线性相关直观表述就是，随着X增大，Y是否同时增大或者减小；当二者分布在一条直线上时，皮尔逊相关系数等于1或-1; 两个变量之间没有线性关系，皮尔逊相关系数为0.



这个是对单样本来说的

协方差是一个反映两个随机变量相关程度的指标，协方差大于0的时候表示两者正相关，小于0的时候表示两者负相关），但是协方差值的大小并不能很好地度量两个随机变量的关联程度，值的大小受到两个变量量纲的影响，不适合用于比较。为了更好的度量两个随机变量的相关程度，引入了Pearson相关系数，其在协方差的基础上除以了两个随机变量的标准差，消除了量纲的影响。

rank\_ic(就相当于在看每一天的预测值排序与实际值排序的相关性有多大)

* The Spearman’s rank correlation coefficient series between label and prediction score.

spearman相关系数

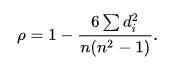
它是衡量两个变量的依赖性的 非参数 指标。 它利用单调方程评价两个统计变量的相关性。 如果数据中没有重复值， 并且当两个变量完全单调相关时，斯皮尔曼相关系数则为+1或−1。

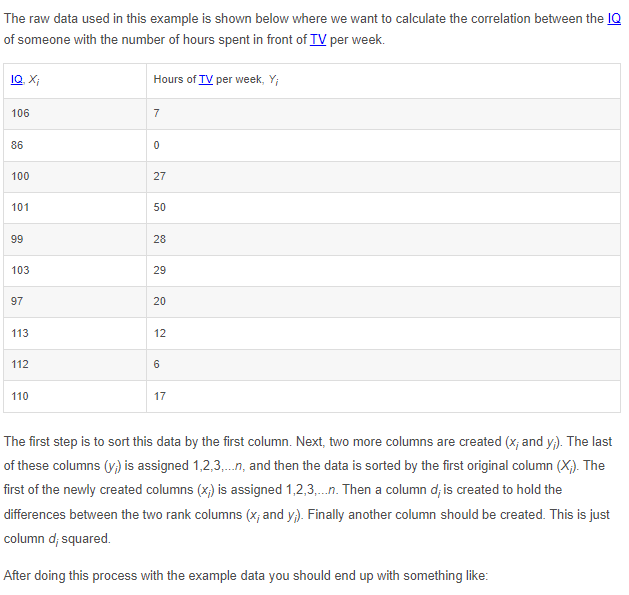
斯皮尔曼相关系数被定义成等级变量之间的皮尔逊相关系数。对于样本容量为n的样本，n个原始数据被转换成等级数据，相关系数ρ为

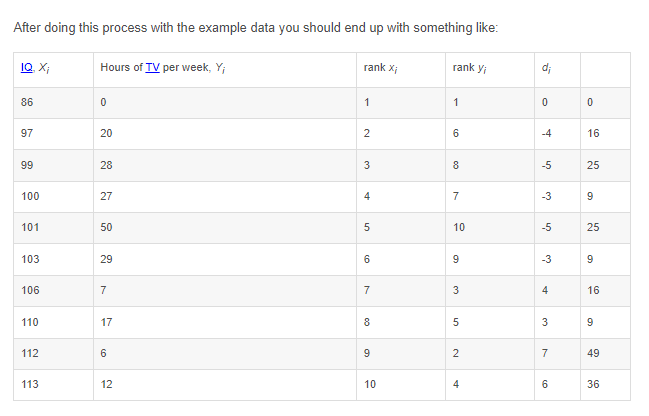


度量一对观测数据的统计依赖性还有其他的几种度量指标:在相关性和依赖性中有谈及。其中最常用的是皮尔逊积矩相关系数。

斯皮尔曼相关也可称为"级别相关";也就是说，被观测数据的"等级"被替换成"级别"。在连续的分布中，被观测数据的级别，通常总是小于等级的一半。然而，在这个案例中，级别和等级相关系数是一致的。更一般的，被观测数据的"级别"与估计的总体样本的比值小于给定的值，即被观测值的一半。也就是说，它是相应的等级系数的一种可能的解决方案。虽然不常用，"级别相关"还是仍然有被使用

举个例子：





#### long\_short\_backtest

A backtest for long-short strategy

|  |  |
| --- | --- |
| **Parameters:** | * ****pred**** – The trading signal produced on day T. * ****topk**** – The short topk securities and long topk securities. * ****deal\_price**** – The price to deal the trading. * ****shift**** – Whether to shift prediction by one day. The trading day will be T+1 if shift==1. * ****open\_cost**** – open transaction cost. * ****close\_cost**** – close transaction cost. * ****trade\_unit**** – 100 for China A. * ****limit\_threshold**** – limit move 0.1 (10%) for example, long and short with same limit.（每一次调用的资金量） * ****min\_cost**** – min transaction cost. * ****subscribe\_fields**** – subscribe fields.（选定购买哪一个领域的股票） * ****extract\_codes**** – bool. will we pass the codes extracted from the pred to the exchange. NOTE: This will be faster with offline qlib. * ****executor related arguments**** (-) – * ****executor**** (BaseExecutor()) – executor used in backtest. * ****verbose**** (bool) – whether to print log. |
| **Returns:** | The result of backtest, it is represented by a dict. { “long”: long\_returns(excess),  ”short”: short\_returns(excess), “long\_short”: long\_short\_returns} |

#### risk\_analysis\_graph

|  |  |
| --- | --- |
| **Parameters:** | * ****analysis\_df**** –   analysis data, index is ****pd.MultiIndex****; columns names is ****[risk]****     * ****report\_normal\_df**** –   ****df.index.name**** must be ****date****, df.columns must contain ****return****, ****turnover****, ****cost****, ****bench****.     * ****report\_long\_short\_df**** –   ****df.index.name**** must be ****date****, df.columns contain ****long****, ****short****, ****long\_short****. |

学习：infomation ratio信息比率

衡量某一投资组合优于一个特定指数的风险调整超额报酬。

信息比率（Information Ratio)：以马克维茨的均异模型为基础，用来衡量超额风险所带来的超额收益。它表示单位主动风险所带来的超额收益。

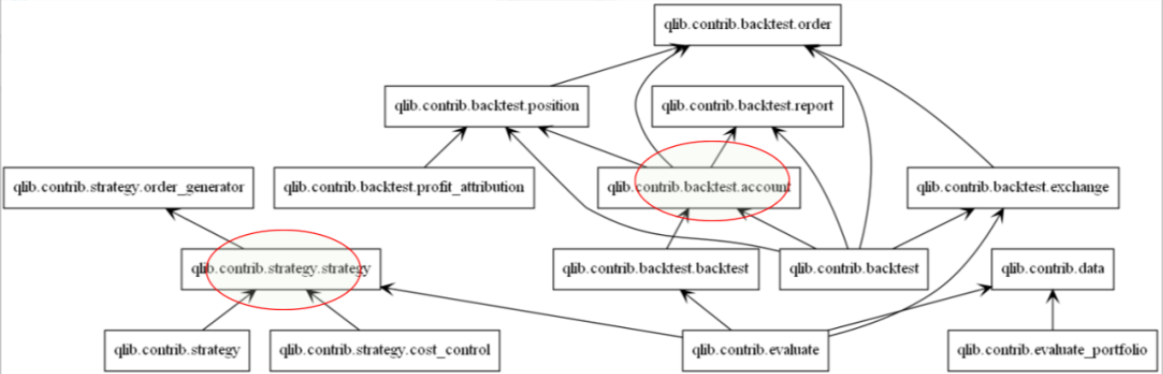
IR=α ∕ ω (α为组合的超额收益，ω为主动风险）

分子α为真实预期收益率与定价模型所计算出的收益率的差，α可以由风险溢价暗示。分母为残差风险即残差项的标准差。（就是α因子）

#### model\_performance\_graph

|  |  |
| --- | --- |
| **Parameters:** | ****pred\_label**** – index is ****pd.MultiIndex****, index name is ****[instrument, datetime]****; columns names is [\*\*](https://qlib.readthedocs.io/en/latest/reference/api.html?highlight=analysis_model" \l "id5)[score,label]\*\*. It is usually same as the label of model training(e.g. “Ref($close, -2)/Ref($close, -1) - 1”).label   * ****lag**** – pred.groupby(level=’instrument’)[‘score’].shift(lag). It will be only used in the auto-correlation computing. * ****N**** – group number, default 5. * ****reverse**** – if True, pred[‘score’] \*= -1. * ****rank**** – if ****True****, calculate rank ic. * ****graph\_names**** – graph names; default [‘cumulative\_return’, ‘pred\_ic’, ‘pred\_autocorr’, ‘pred\_turnover’]. * ****show\_notebook**** – whether to display graphics in notebook, the default is True. * ****show\_nature\_day**** – whether to display the abscissa of non-trading day. |

## Qlib的backtest模块：



### \_init\_.py

#### **换手函数**：

def get\_exchange(exchange=None,这是一个字典，每次更新换手率，这个换手率频率可调，为周频

freq="day",

start\_time=None,

end\_time=None,

codes="all",

subscribe\_fields=[],

open\_cost=0.0015,

close\_cost=0.0025,

min\_cost=5.0,

limit\_threshold=None,#这个是每天的涨跌幅

deal\_price: Union[str, Tuple[str], List[str]] = None,

\*\*kwargs,

):

#### **创建账户**：

def create\_account\_instance(

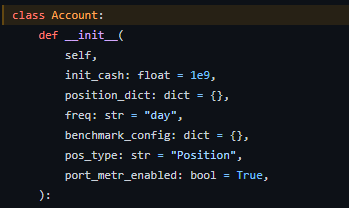
start\_time, end\_time, benchmark: str, account: Union[float, int, dict], pos\_type: str = "Position"

) -> Account:

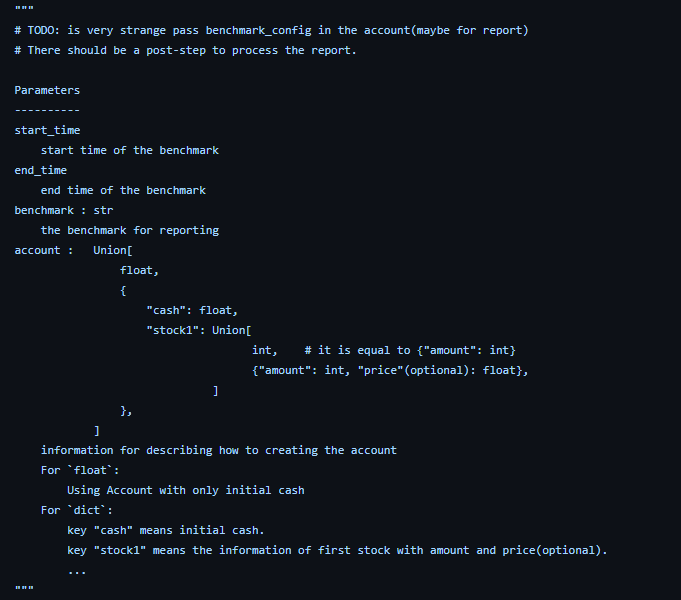
->符号的含义：这个符号通常放在我们定义的函数的函数名后面。

这里面，元数据表明了函数的返回值为int类型。

至于楼主问题中的，-> \_Attr则表明函数返回的是一个外部可访问的类的私有变量。



参数意义



### Account.py

### Backtest.py

### Decision.py

### Exchange.py

### Executor.py

### High\_performance\_ds.py

### Position.py

### Profit\_attribution.py

### Report.py

### Signal.py

### Utils.py