Using the API

January 29, 2022

1 Introduction

The REST API is the main tool used to contribute features and algorithms to the ecosystem. This document walks through how to do so, with specific attention to the alpha test that starts in Q1 2022.

During the alpha test, everything, including this API, is brand new. The good news is that the functionality is super simple and easy to use. The bad news is if you do anything wrong, it will just get rejected, and not handled by more robust error-handling scripts. If you encounter any issue, or get stuck on anything, give us an email![1] We are here to help and make it easy for you.

2 Summary

The basic workflow for contributing features to the ecosystem:

1. Get a private key from this form.
2. Format your data, and save it as a JSON object.
3. Submit the data that covers the appropriate in-sample time frame.
4. Schedule a cron job to send in (usually) a single row to update your feature in real time.

The following pages walk you through everything you need to know to submit features and begin claiming 'land' in the feature space.

[1] gina [at] judgeresearch [dot] co.

3 Submitting Your Features and Algorithms

3.1 Submitting Algorithms

For the alpha test, we do not have means to test algorithms you submit without looking at your code. We assume revealing your IP is not attractive to most quant devs. While you are encouraged to submit an, R, Python or C++ script at the below endpoint, we are organizing the alpha test primarily around the act of submitting features, and therefore will focus on that topic for the rest of this document.

If you do want to submit an algorithm, it must take in any matrix, X, and return $\hat{y}$, a vector of forecasts. Please see "How It Works and How to Participate" for more details.

3.2 Submitting Algorithms

The endpoint to submit features and algorithms is,

```
https://n1mqxz43aj.execute-api.us-east-2.amazonaws.com/dev
```

A feature should be delivered in the body through a `PUT` request as a JSON file. The request must have the following items in the header:

1. `x-api-key`: the API key we have sent you
2. `featureName`: the name of the feature in this file
3. `DV`: the dependent variable (target feature); for the alpha stage we are only using `BTC-USD` and `ETH-USD`.

You'll find the details for each step below.

3.3 The API Key

Use this form to get your private key. You'll receive an email with a key soon after you submit the form. You can reply to that email if you have any questions and an actual human being will respond.

3.4 Naming Your Feature

If you are submitting before the alpha test officially begins in late February, congratulations, you are likely claiming subspaces of the feature space that are of extreme value. We assume that you will not want to reveal your features' substance by labeling it accurately. So just make sure you give it a unique identifier that you can remember.

Remember, the system by design does nothing to track your features. If you mix up your feature names or otherwise mis-organize your features, an error may not be thrown but the AI will eventually throw your features out as not helpful.

3.5 Organizing your Features

In the future, you'll be able to send in matrices with one PUT request. For the alpha test, please only submit vectors - that is, one feature - per PUT request. You can submit up to 100 such requests in each time bloc.

3.6 Formatting Your Data

For the alpha test, we only accept JSON objects.

3.7 Specifying Your Dependent Variable (Target Feature)

When you submit a feature, you choose your target feature. For the alpha test, the header should include either `BTC-USD` or `ETH-USD`.

Your feature will be given a small chance to be run in models with other target features. Thus, if it does exceptionally well, it will grow in the population and become prominent for target features other than those you choose. However, more than 95% of the initial runs (chances) your feature gets will be related to the target feature you specify in the header.

3.8 JSON structure

The attribute - value pair must be a standard LINUX-format date, e.g: {"2000-11-22 08:00:00" : [your features value at this time period]}.

Missing data *must* be represented by an NA. This includes any time periods from markets that are not 24-7.

4 Updating Your Data

After the initial submission of the training sample, each observation should be submitted *before* the turn of the time period. For example, the cut-off submission time at the turn of the day is 23:59, not midnight. Like OHLC data and most time series data, the timestamp refers to the beginning of the UTC time period. For example, for an hourly time series, 20:00 refers to the time bloc 20:00-20:59.

You must submit a JSON identical in every way as your original submission, except that it contains only the rows to append to your original object. Typically, this will be just one row, so your JSON would contain only one attribute - value pair; but if you missed an observation, simply submit the rows not yet submitted.

5 The Alpha Test

The alpha test begins in late February, 2022, but you or your fund can start now. The below vignette gives you step-by-step instructions for participating.

It is a data experiment - no live positions will be taken, as the fund itself is not yet set up.

While the alpha test is an experiment, users still 'claim their land' (see 'How It Works') by moving first. The massive, overwhelming first-mover advantages we've built into the incentive structure could, therefore, end up affecting your rewards for years to come.

6 Submitting Features for the Alpha Test

The alpha test will run over two pairs, `ETH-USD` and `BTC-USD`, at the four hour interval.

The date window for you initial submission should begin at:

```
2019-01-01 00:00:00
```

and run until the most recent four hour block.

After your first submission, you should schedule a cron job to send in new data every period. The contest will run for 45 days.

The judging criteria will be the one-step-ahead forecasts during the 45 days of live data you send in in real time. Users that submit features before others, with an identity defined as as $\rho \geq .975$, will retain their 'land.'

The long-run emphasis of the project is for experts to send in high-value, hard-to-engineer features. However, in the beginning, easy-to-engineer features may prove invaluable land to hold in the feature space, while involving very little overhead for participants. For example, the simple logged differenced OHLCV data for various pairs will likely be included in innumerable models in the future, and whoever submits those first, if they do so reliably in the future, will retain that exceptional land.