FDML coding assignment for Raft

Thanks for applying and congratulations for reaching this point of the interview process!

In this stage we will have you write some code. We will evaluate your ability to develop something that's fairly common at Raft. We strive to make this test as close to our daily business as possible and guarantee that every backend engineer at Raft can complete this test. The informal wording of this test is close to our daily business talk and reading comprehension is part of the test

About this test

The code in the package

We are providing you with a base project that is very similar to the local setup we use for microservices. This includes all the tools you may need: docker infrastructure, database connectors, and, of course, main.py.

Extract the contents, install Docker if you don't already have it installed, then run `docker-compose up` in the directory. You can find more information in the README.md file.

Our goal is to save you the set-up time, but also verify that you'd know your way within our infrastructure.

How long it should take

The test includes 2 requirements. Once you manage to start the base microservice, we expect the tasks to take two to three hours(if you're familiar with the environment) - and above that feel free to spend any time you feel comfortable spending.

Sending back the code

Please, upload the code on GitHub as a **private repository** and share it with `raft-ml`.

Please, don't put this project on a public repository - we routinely change the content of the test, but we don't want to run out of ideas too soon.

Setting expectations

- We don't expect mature code. Even the simplest code gets refined over the years and we understand that you won't reach that level of stability in a couple hours of work.
- We care about **code quality and structure**. Spin up your favourite code linters, make sure it's easy to find where the functions are, that no stray files are included and that the spacing in the code is not random.
- We care about **simplicity**. Since this is a proof of concept, it wouldn't grow further than the end of this exercise. There's no need to code this as if you had to extend it for the next 5 years.
- We care about **modernity**. Not using `async` on a web server in year MMXXIII is not acceptable. Likewise, try to use the latest libraries and modern coding paradigms.
- We care about **communication**. If any part of the code that's not easily explained by patterns and naming, or you took some shortcut "hack", a comment would be welcome. If you changed any of the infrastructure, a mention in the README.md would be appreciated.

Making the tests pass is not a hard requirement. If you can explain your approach and the results that the approach produces, it's fine. But it would be great if all the tests pass!

Problem

We want to understand user clicking patterns. Users often click parts of the webpage and we want to find clusters within them. We have a stream of user clicks coming in. We want to cluster the clicks on-the-fly. The clustering algorithm should be smart enough to understand whether the coordinates belong to an existing cluster or to a new cluster. The clusters will be distinct, will never overlap, and will never have any ambiguity.

We do have multiple pages (each represented by a unique id). The clusters across pages are independent.

Input format:

The API calls will contain - (x, y) coordinates between 0 and 1. (float) - the page_uuid. We might have multiple pages and each page will have different clusters. (string) The data models for the input are stored in `app/data_models.py`

Requirements:

You need to implement the following functions/API endpoints in `app/main.py` :

1. API endpoint `/save_click_and_predict_cluster` - it takes click coordinates and a unique page id as input. It must assign the coordinates to an existing "click" cluster for that page id *or* creates a new cluster. The click input should also be stored in the database. It should return the assigned cluster index(starting index from 0) and whether the cluster is a new cluster.
2. API endpoint `/predict_cluster` - it takes click coordinates and a unique page id as input. It must predict what cluster the coordinates belong to. It shouldn't store the coordinates in the db and it should just return what cluster index the click belongs to. Return `null` if it doesn't belong to any cluster.

Testing

We've provided test cases inside `tests/cases` .

At submission, we'll be testing the solution with our secret test set.