Machine Learning Engineer Hands-on

Problem

Suppose you are a Machine Learning Engineer of a restaurant recommendation project in LINE MAN Wongnai. Your data scientist colleagues are developing machine learning models of two use cases, batch and real-time inference. The models are implemented using the nearest neighbor algorithm from the Scikit-Learn library.

Part 1: Model Serving

Please implement an HTTP API server for the model inference with at least one type of **database server**. The server must be able to serve **30 requests per second** of the given request parameters (`request.parquet`) with **the 90th percentile of response time within 100 milliseconds**. For this exercise, you can consider up to the top 2,000 restaurants in the nearest neighbor search as relevant results and disregard restaurants ranked lower than 2,000th place

HTTP API Specification

- Endpoint: `/recommend/<user_id>` (e.g. `/recommend/u00000`)
- Method: `GET` and `POST`
- Query parameters
    - `latitude`: User's latitude
    - `longitude`: User's longitude
    - `size`: Number of recommended restaurants (default: 20)
    - `max_dis`: Max **geodesic or great circle displacement** in meters between the user and restaurants. Restaurants further than this are considered irrelevant. (default: 5000) (Hint: H3 index)
    - `sort_dis`: Flag to sort restaurants by the displacement.
        - `sort_dist=1`: Sort restaurants by geodesic or great circle displacement
        - `sort_dist=0` or not provided: Sort restaurants by Euclidean distance returned from the model
- Response: recommended restaurants in JSON format
    - `restaurants`: Recommended restaurants that are composed of
        - `id`: Restaurant ID
        - `difference`: Euclidean distance returned from the model
        - `displacement`: Geodesic or great circle displacement

Example

```
{
    "restaurants": [
        {
            "id": "r00002",
            "difference": 25.2,
            "displacement": 800

        },
        {
            "id": "r00001",
            "difference": 28.2,
            "displacement": 1000
        }
}
```

Attachment

These files are provided in the attachment.

- **`model.pkl`**: binary of Scikit-Learn NearestNeighbors object
- **`inference.py`**: example script of model inference
- **`requirements.txt`**: Python dependencies for executing `inference.py`
- **`Dockerfile`**: for building a docker image to execute `inference.py`
- **`user.parquet`**: user data of type Parquet with following columns
  - **`user_id`**: ID of the user
  - **`feature_0`** to **`feature_999`**: numerical features of the user for model inference
- **`user.small.parquet`**: a small sample of `user.parquet` for **debugging purpose**. **Do not use this file in the submission and performance testing.**
- **`restaurant.parquet`**: restaurant data of type Parquet with the following columns
  - **`restaurant_id`**: ID of the restaurant
  - **`index`**: integer index to map the model result to restaurant ID
  - **`latitude`**: Restaurant's latitude
  - **`longitude`**: Restaurant's longitude
- **`request.parquet`**: Request parameters for performance testing

Submission

The submission must contain these files as minimum requirements

- Server code
- **`requirements.txt`**: Python dependencies for executing the server
- **`Dockerfile`**: for building a docker image to execute the server
- **`docker-compose.yml`**: for starting API and database server
- **`perf_test`**: a folder containing performance testing code
- **`README.md`**: an instruction to execute the server
- **`answer.pdf`**: performance report
- Do **not** include `user.parquet` and `restaurant.parquet` in the submission