CGG Incubator case study (ML Engineer)

Thanks for applying to this ML Engineer position @ CGG's Incubator team! This is a technical case study to help us understand better your technical stacks. It is an open assessment, the goal is not to find a perfect machine-learning model but to see how you will build a real-world industrial ML system. You can add anything that you think is useful to this pipeline.

General instructions:

The main goal of this assessment is to **build an end-to-end ML pipeline**.

- Please do the exercise using `Python` and your favorite IDE (PyCharm, VS code....).
- Upload your work to **GitHub** so that we can see your progress (your commits). Please make sure you give viewing permission to hanyuan.peng@cgg.com.
- **IMPORTANT**: Please make regular commits with your ongoing work. Do not only upload the final result when you get it done. Instead, do your work in public so we can follow your regular updates. If you use other online programming tools, please make sure that we can see you regular updates by making necessary notes.

The subjects are listed below. Please have a look, and do not hesitate to ask any questions about the format of the data files, or the requirements of the questions.

Subject

The goal of this ML workflow is to detect the existence of a `landfill` in one area using satellite images. It's a simple binary classification problem with a `satellite image` as input and predicts if it contains (label= `1`) or not (label= `0`) a landfill site.

Raw Data

You will have 2 files attached in this mail, `train.geojson` and `test.geojson` .

Those raw data files are in `geojson` format, which contains a list of polygons (squares in this case) and their coordinates, and also their `labels` in the `properties` . You will need to download the corresponding satellite images by following this example using Microsoft Planetary Computer STAC API.

- For the parameter `time_of_interest` shown in the example notebook, please use this value: `2022-01-01/2022-12-30`
- You can keep the `"eo:cloud_cover": {"lt": 10}` to select all assets that have `cloud_cover` less than 10%.

- You can use the `visual` channel (like shown in the example) to have normalized satellite images.

After you have downloaded all the images (20 for the training & validation and 6 for testing). Please answer the following questions:

1. What are the sizes of these images? Do you need to do some processing for the model building?
2. With the help of this description, can you tell the real-world resolution of downloaded images? (optional)
3. Any other insight that you can draw from the data? (optional)

Modeling

You will need to build a simple Machine Learning model to detect the existence of landfills in given satellite images. Evaluate the performance with the test data.

> **Important**: The goal here is **NOT** to produce a state-of-the-art machine learning model, just pick any model that you want (or the one that you can run with your hardware). It doesn't matter if you don't have a model who has a very good generalization.

This test should be okay to run on your laptops as you could use a very simple model, but if you don't have any device that allows you to run the training, you could use the free Google Colab or any other free resources that you prefer.

Here are some questions:

- How do you evaluate your model?
- What do you think about the performance that you got with your model? Any reason?
- Any suggestions to improve the performance?
- What could impact the performance of a machine learning model?

ML pipeline

The goal here is to build a reproducible pipeline.

Based on the previous steps of data downloading and modeling, to deploy your model in production, develop a machine learning pipeline that:

- Reads the raw data and downloads the satellite images
- Trains the model (If you have limited computing resources, you don't need to run the training part or you can replace your model with the simplest one).
- Evaluate the model with metrics that you defined
- Make predictions on new locations and show the result.
- Add any bricks that you think are useful to this pipeline.

- Propose an interface so that this pipeline can be run easily. (optional)
- Propose a solution for monitoring in production (optional)