Rappi Machine Learning Engineer

Challenge

The Challenge

Based on the well known Kaggle problem "Titanic: Machine Learning from Disaster" and the jupyter notebook provided as a zip, , we ask that you complete the following two tasks, adhering to your coding best practices:

● Create a pipeline for model training, evaluation, and deployment process.

● Create an API that will receive a list of passengers and provide accurate valuation predictions.

Documenting any assumptions made during model development and identifying potential areas for improvement is essential. To support this, you should create a detailed README file that explains the project's structure, dependencies, and provides instructions for running the pipeline and API.

In the README, add a concise conclusion that covers:

- **Evaluation metrics**: Is the model good or bad? Why?
- **Key features**: Which features are most important and why?
- **Production deployment**: Outline the technologies, MLOps best practices, system architecture, automation strategies, and cloud services you'd use and why.

While external resources like libraries and frameworks are encouraged, avoid relying on pre-built templates. These can obscure your individual design choices, making it harder for us to assess your unique skills. We want to evaluate your personal approach and best practices in coding and system design.

Pipeline for training a classifier

Create a pipeline for training a binary classifier. The input for this training can be found in the mentioned jupyter notebook. The output should be a binary classifier model and exported somewhere so it can be used from the API.

Your solution should cover the following points:

● Binary classifier as described above (you can use the code provided in the notebook).

● Automatized pipeline execution.

● Pipeline profiling (CPU and RAM usage).

● Dockerized solution.

It would be great if your solution covers the following points (optional):

● Improved implementation of the classifier provided in the notebook.

● Logs configuration.

● Virtual environment configuration.

● CI/CD Workflows.

● Data version control

Create an API

After exporting the classifier, it is required to use it for real time predictions. With the mentioned objective in mind, you would need to create an API that receives a list of passengers and predicts for each of them if they're likely to survive or not.

Your solution should cover the following points:

● API as described above.

● Proper error handling (http status code, message, unhandled exceptions, etc).

● API should be documented, it's recommended to use a framework such as FastAPI or similar.

● API logging, monitoring and alerts.

● API profiling (CPU, RAM usage, maximum requests handled) and how it would scale.

● Unit testing of model's predict function.

● Dockerized solution.

It would be great if your solution covers the following points (optional):

● Download model and load it on API deployment.

● Basic security system for the API (API-Keys or similar).

● Logs configuration.

● An endpoint to print feature importances.

● Support for A/B Testing so two models can be deployed and compete between them.

● CI/CD Workflow.

Results delivery

1. Implementation code folder/repository and instructions to run it.

2. You may use any language, tools, and cloud services you want.

4. We expect to receive your work within <> days since we sent you the challenge.

Documentation

We encourage you to write down every step down the road. Design decisions, references you used, difficulties and insights, and partial results. As mentioned, not all the points are mandatory but as mentioned in the respective sections it would be great if your solution covers them all.

If you have any doubts, feel free to contact us at rodrigo.pizarro@rappi.com