REMLA24 – Team 8

Tip ten Brink, Felipe Bononi Bello, Dielof van Loon
4927192, 5069238, 5346894

1 RELEASE PIPELINE DOCUMENTATION

In this section two different release pipelines will be documented. The first of which releases the software package containing the preprocessing logic that is used for both training and new queries. The second one will highlight the release pipeline of the model-service container image. This container image represents a wrapper service for the released ML model. For both release pipelines the different steps will be elaborated

Software Package: lib-ml The source code of the released software package can be found on Github [1]. This repository holds the logic regarding managing of the tokenizer, publishing of new releases and the continuous integration and deployment (CI/CD) of the package.

Where the source file holds the preprocessing logic, including ways to create, store and retrieve a tokenizer. A tokenizer holds the tokens in which the train, validation and test datasets are encoded. To account for other possible tokens an out of variable token is included. The tokenizer will be stored making use of joblib [2]. We might still need to implement a way to upload the tokenizer to gdrive, as currently we do this manually. The tokenizer can be retrieved either locally when the force download parameter is set to false or from google drive making use of the gdown package [3] when the force download parameter is set to true and the url of a google drive folder is passed on.

When publishing a new release, create a tag on Github matching the version specified in ‘pyproject.toml‘ file. Github Actions automates the build and publishing processes. If a release fails, the tag can be deleted and re-published. Using Github Actions enables every push to be automatically tested and build ensuring that only stable and functional versions are released. This CI/CD pipeline ensures the set software standards.

Container Image: model-service The source code of this container image can be found on Github [4]. This repository contains all necessary configurations for deploying and managing the service efficiently, as well as the underlying logic that facilitates interaction with the machine learning model through a RESTful API. Furthermore it takes care of publishing new releases and the continuous integration and deployment (CI/CD) of the container image.

The model-service container image is a wrapper for the created ML model. It provides a scalable and flexible solution for model deployment, allowing external components to interact with the model via a REST API. This setup ensures that the model can be easily integrated into larger systems or workflows, supporting both single and bulk prediction requests.

[1]https://github.com/remla24-team8/lib-ml
[2]https://joblib.readthedocs.io/en/stable/
[3]https://pypi.org/project/gdown/
[4]https://github.com/remla24-team8/model-service

When publishing a new container image, create a tag on Github matching the version specified in ‘pyproject.toml‘ file the same as done for the software package. Github Actions takes care of creating a new release and publishing it on the Github Container Registry (ghcr). After which it becomes accessible for orchestration tools such as Kuberenetes and Docker to dynamically scale the service.

2 DEPLOYMENT DOCUMENTATION

As we have not yet finished exactly our final deployment, this section is still unfinished.

Below shows the basic setup of our cluster.

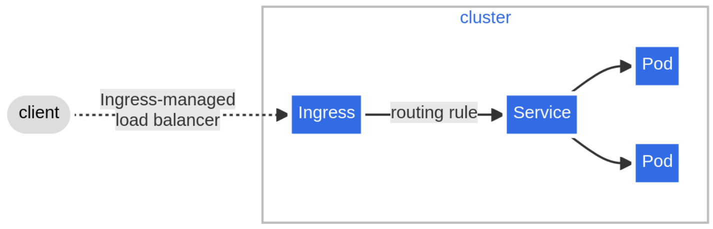

Figure 1: Deployment Diagram

3 EXTENSION PROPOSAL

As we have not yet finished the primary portion of our code, we cannot yet evaluate what exactly needs an extension.

4 ADDITIONAL USE CASE

5 EXPERIMENTAL SETUP

6 ML PIPELINE

7 ML TESTING DESIGN

Most of the testing is done in the model-training branch. This is because the model is the main testing subject and thus needs testing every time the model receives an update, which happens in the model-training branch.

7.1 Current tests

The current tests are listed in table 1. The international URLs test is currently failing, as the model supplied by our instructors scores the international URL lower in phishing then its legitimate counterpart. We also devised a test on an important data slice, namely HTTP vs HTTPS. This is because HTTP is much less secure than HTTPS, which could be a sign of a phishing attack. Thus, it is important that HTTPS URLs get a lower phishing score than its HTTP counterpart. We modified the top100.txt dataset to 2500 URLs and generated an HTTP counterpart of these URLs [1]. Our criteria says that 90% of our predictions should favour the HTTPS counterpart.

Name	Description
Predict types	This test enforces the model to return an numpy array for every query it gets, regardless if the input is a list or a string.
Correct URLs	This test checks if legitimate URLs return low scores
Incorrect URLs	This test checks if illegitimate URLs return high scores
International URLs	This test checks if URLs that use international characters made to look like English characters get a high score.
HTTP vs HTTPS	This tests if the same legitimate URL scores higher when using HTTP instead of HTTPS

Table 1: Different PyTests in model-training

7.2 Limitations and Future Work

REFERENCES

[1] Y Nizipli. 2023. url-dataset. https://github.com/ada-url/url-dataset.