Project: Data science micro-service

This is a short project to introduce some typical problems faced by machine learning engineers in the data science team. It should not take more than a few hours (although significant prior knowledge may be required) and can be completed over the weekend.

Summary:

Develop and deploy a simple model to predict whether a sentence is relevant.

Objectives:

The objective of this project is to demonstrate your proficiency as a machine learning engineer.

A machine learning engineer is responsible for:

- Engineering best practices
- Application design, service architecture and reliability
- CI/CD pipelines with appropriates tests and fallbacks
- Data science service deployments in production
- Data science cloud resources and IAMs

A machine learning engineer should be familiar with some of the followings:

- Engineering best practices (code smell, code reviews practices, etc)
- Web stack (e.g. rest, grpc, ngnix, redis, rabbitmq, etc)
- Big data stack (e.g. spark, kafka, elastics, etc)
- Cloud stack and basic devops (e.g. kubernetes, docker, terraform, IAMs, VPC, shell scripting, etc)
- Data science and machine learning concepts

You are NOT expected to do a full-scale production ready project. However, you should try to demonstrate your proficiency in **engineering designs and best practices for data science products**, as well as some basic knowledge in data science and machine learning.

You are expected to **prioritize** and implement a MVP to best demonstrate your engineering design skill. You are expected to explain your design rationales and decisions during the interview. You are also expected to expand on how you would improve the product (engineering pov) if more time is provided. There will also be general architecture and infrastructure design questions during the interview.

You can contact william.teo@grab.com if you need any further clarification. However, you can make any assumptions you need as long as you can justify your rationales.

Dataset:

A SQLite database will be provided with this assignment. The database has a single table “data”. The provided file is a tarball with a gzip compression. There is only a single file inside the archive. This file is the base64 representation of the SQLite database - i.e. you will need to encode it back to binary before you can actually use the database.

The original source of the data can be found in the following link: NLP Starter Test. This dataset contains sentences with the following labels: “Relevant”, “Not Relevant”, or “Can’t Decide”.

You should use this sqlite database if possible (with the corresponding scripts to unpack and encode the database). You should also commit this file in the repo for this assignment.

Preparation:

You will need to create one or more project repos for this assignment and share it with us. You can use any of the source control providers (e.g. Github, Gitlab, BitBucket, etc). The repo can be public as nothing is confidential in this assignment.

Task1: Project Repo

Create one or more project repos with the provider of your choice for the assignment. You should design the repo(s) as if the assignment is an end-to-end data science project - i.e. from getting the data, to training/validating the model, to deploying a web service for the model.

You will need to explain your project structure design during the interview.

NOTE:
When designing your repo(s), take into account that you will need to setup CI/CD for your model training, and model serving application.

Task2: Model training

Train a simple model to predict whether a sentence is “Relevant” or otherwise. There are no restrictions in the language, library, or techniques to use, but note that you will need to package this model into a web service.

The performance of the model is **NOT** important. You should **NOT** spend too much time in feature engineering, hyper-parameter tuning or model selection.

However, your codes should minimally include the following functionalities:

- Retrieve the dataset from some arbitrary source (i.e. sqlite db)
- Creating a test and train dataset
- Fitting a model
- Validating the model (a simple threshold on precision/recall is sufficient)
- Serializing the model (so that it can be loaded in the web service)
- Persist/Update the serialized model to some arbitrary store (can be a local fs)

The performance of the model(s) is NOT important, but you should demonstrate basic knowledge in interacting with databases, data preparation and model training.

However, you should focus more on demonstrating good engineering designs - e.g. readability, extensibility, separation of concerns, reproducibility, …

NOTE:
You will be setting up a pipeline to demonstrate how to train and update the model to simulate an automated CI process where the model is updated periodically.

Task3: Model serving

Develop a simple web service which accepts a sentence and response whether the sentence is “Relevant” or otherwise.

The web service can be REST or GRPC. REST service must come with the corresponding Swagger or OpenAPI specifications. GRPC must come with the corresponding protobuf specifications. There are no other restrictions in the language, or framework to use.

You should also include a Dockerfile for the web service.

Task4: CI/CD

Select your preferred code style and unit test packages for your preferred language and framework.

Setup CI pipelines to do the following:

- Code style (and complexity) checks
- Basic unit tests (or similar)
- Integration test for model training

You will NOT need to setup any CD for this assignment.

NOTE:
You do NOT need to write a 100% coverage unit test, but you should write the MOST important unit tests to demonstrate your proficiency.

The integration test is to simulate and test an actual CI run to train and update a model - i.e. you can provide mock parameters and data, and do the appropriate checks.

Task5: Deployment

Google Cloud Platform provides a service “Google Cloud Run” which allows easy deployment of stateless containers.

Demonstrate your initiative by setting up your repo(s) with a “Google Cloud Run Button” to serve your web app - i.e. anyone can just click on the “Google Cloud Run Button” to provision your prediction web app.