**Project 4: Model Development and Prediction Service**

**CS4981 ML Production Systems**

**Instructions**

**Part I: Offline Model Development**

You will write a script to optimize a model using the currently available data. Adapt your solution from project 1 to use the labeled emails stored in Minio. Divide the data the into training (80%), validation (10%), and testing sets (20%) by time. Make sure that you handle class imbalance in the training set.

Train models with different parameters for the CountVectorizer, different model types, and hyper-parameters for your model type. Evaluate the models on the validation set. Choose the settings that give you the best result on the validation set.

Finally, using those parameters, train a model on the training set and evaluate on the testing set. Output the parameters and metrics (as a JSON file).

**Part II: Offline Model Development**

Write a script that will train a model on all of the available. Data and save the resulting model to a file. The script should take the parameters JSON file along with the Minio logs as input. The output should be a [pickle](https://docs.python.org/3/library/pickle.html) file containing the serialized transformer and model objects and metadata such as the parameters used, date ranges for the input data and class balance metrics. Make sure that you include any transformers (e.g., CountVectorizer, StandardScaler) when you pickle the state of the trained model. Put the current date and time in the filename to allow multiple models to coexist. Upload the model file to Minio.

**Part III: Implement Classifier Service**

You will implement a service that classifies emails received via a REST interface. You can do everything in a single Python file named prediction\_service.py . The program should take the Minio path, username, and password as environmental variables. The program should download the trained model from Minio, unpickle it, and load it. The program should then start a Flask REST service.

The server should listen on port 8888 and implement a single REST endpoint:

GET /classify

Takes a JSON object as payload. The object contains the key “email” with the value being another object. The email object should have the key “body” with the value being a String. The request should return another JSON object with the key “predicted\_class” and a value of “spam” or “ham.”

Set up structured logging. For every request, log the predicted class (ham or spam) but not the email content.

**Part IV: Test It**

1. In one terminal instance, run the prediction service

2. In a second terminal, start the log collection script

3. In a separate terminal instance, run the simulator:

(venv) $ python spam\_simulator.py evaluate-model --email-dir path/to/email\_data --classifier-url http://127.0.0.1:8888/ --number-emails 10000

4. When done, write a script to count the fraction of ham and spam predictions. Compare to the fraction in the Minio logs – do they agree?

**Submission Instructions**

Submit a URL got a GitHub repository to Canvas.