Flask Load Balancing task

Python Practice Problem - LOAD BLANCER

Assume you have a flask API, that deals different ML models, but due to GPU/CPU/RAM restrictions, it can load only 2 ML model at once. This flask API has two following routes.
1. '/get_loaded_models' This will return the two loaded models info at that time (code is already provided below)
2. '/process_request' This will process the request, using the required model, specified in request against key 'model'. If that model is already loaded, the API will process that request using loaded model. Otherwise the API will load this model at the place of the one model which is previously loaded and has less frequently requested.
You are required to write its load balancer. Your load balancer should be able to record the number of requests processed by all models, so that it can intelligently load new model on the place of less frequent model. Sample code with dummy processing is provided below. You have to write its load balancer function. You may also also re structure the code, or change in code according to your need. But those changes should not change any previous functionality.

```
import traceback
from flask import Flask
from flask import request

class ML:
    def init(self):
        self.avaliable_models = {
            "face_detection": "/additional_drive/ML/face_detection",
            "car_detection": "/additional_drive/ML/car_detection",
            "shoe_detection": "/additional_drive/ML/shoe_detection",
            "cloth_detection": "/additional_drive/ML/cloth_detection",
            "signal_detection": "/additional_drive/ML/signal_detection",
            "water_level_detection": "/additional_drive/ML/water_level_detection",
            "missile_detection": "/additional_drive/ML/missile_detection"
        }
        self.loaded_models_limit = 2
        self.loaded_models = {
            model: self.load_weights(model)
            for model in list(self.avaliable_models)[:self.loaded_models_limit]
        }

    def load_weights(self, model):
        return self.avaliable_models.get(model,None)

    def load_balancer(self, new_model):
        print(self.loaded_models)
        # your code here
```

```
app = Flask(name)
ml = ML()

@app.route('/get_loaded_models', methods=['GET', 'POST'])
def get_loaded_models():
    return ml.loaded_models

@app.route('/process_request', methods=['GET', 'POST'])
def process_request():
    try:
        model = request.form["model"]
        if model not in ml.loaded_models:
            ml.load_balancer(model)
        return "processed by "+ ml.loaded_models[model]
    except:
        return str(tracback.format_exc())

app.run(host='0.0.0.0', port=5000)
```