Chaklam Silpasuwanchai, Akraradet Sinsamersuk
AT82.03: Machine Learning

A2: Predicting Car Price

In this assignment, you will continue using the `Car Price` dataset but with the code from scratch we learned in the class. We shall implement more code on top of what we have, perform some experiments using ML flow, and lastly deploy your docker to our prepared virtual machine.

Note: You are ENCOURAGED to work with your friends, but DISCOURAGED to blindly copy other's work. Both parties will be given 0.

Note: Comments should be provided sufficiently so we know you understand. Failure to do so can raise suspicion of possible copying/plagiarism.

Note: You will be graded upon (1) documentation, (2) experiment, (3) implementation.

Note: This is a two-weeks assignment, but start early.

Deliverables: The GitHub link containing the jupyter notebook, a README.md of the github, and the folder of your web application called 'app'.

Task 1. Implementation - Based on `03 - Regularization.ipynb`, modify `LinearRegression()` class as follows:

- Add a function `r2` that compute the r^2 score. Be reminded that you have learned this equation in the L1 lecture.
- As we have discussed in class, there are much better way to initialize the weight aside from zeros.
  - We shall start with the defacto **Xavier** method - please read roughly the paper http://proceedings.mlr.press/v9/glorot10a.html.
  - To summarize, the xavier initialization method is calculated as a random number with a uniform probability distribution (U) between the range

$$U[-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}]$$

    where m is the number of inputs.
  - We can easily implement this in python as follows:

```
# pseudocode for xavier weight initialization
# number of samples, let's say 10
m = 10
# calculate the range for the weights
lower, upper = -(1.0 / sqrt(m)), (1.0 / sqrt(m))
# summarize the range
print(lower, upper)
# you need to basically randomly pick weights within this range
# generate random numbers
numbers = rand(1000)
scaled = lower + numbers * (upper - lower)
print(scaled)

```

  - Modify the class such that it allows the user to choose between zeros initialization or xavier.
- Learn about **momentum**. In short, momentum is a improved gradient descent technique that help alleviate the local minima stucking problem. A little big longer version, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way. The same thing happens to our parameter updates: The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation. Implement it so that users can choose whether to use momentum or now, and also the momentum

as well which is in range of (0, 1) accordingly. For those who are struggling, here is some pseudocode hint:

```
# pseudocode for momentum
def update():
    step  = alpha * grad
    theta = theta - step + momentum * prev_step
    prev_step = step

```

- Implement a function inside the class that can plot the **feature importance** based on coefficients. Note that although large (positive or negative) magnitude of coefficients can usually indicate strong feature importance, it is important to assume that the input variables have the same scale or have been scaled prior to fitting a model - for further reading - https://inria.github.io/scikit-learn-mooc/python_scripts/dev_features_importance.html

Task 2. Experiment - Using `A1: Predicting Car Price` jupyter notebook that you have submitted as the starter, replace the modeling part with the class we have built above.

- Use cross-validation to compare (1) polynomial, lasso, ridge, normal; (2) with, without momentum; (3) stochastic, mini-batch, batch; (4) zero, xavier; (5) learning rate of 0.01, 0.001, and 0.0001. The comparison should be done in terms of r^2 and mse. The entire experiment must be performed using ML flow.
- Perform the prediction on the test set using the best model and report the mse and r^2. Plot the feature importance graph using the function we have built above.
- Write a short report inside the jupyter notebook discussing your findings, together with some captured screenshots of ML flow, and a final table depicting the comparisons.

Task 3. Deployment - We will continue from the `A1: Predicting Car Price` Docker website. I assume that you now have a working Docker container (because you submitted the first assignment that you did by yourself... right?). Now that you have a new amazing model, why not share this with the world? And this time I meant the real world. There are two objectives. (1) To improve your website that now makes use of your two models, and (2) deploy your site on an actual server that will be accessible from the internet.

Objective 1: Improve your website The scenario is now this.

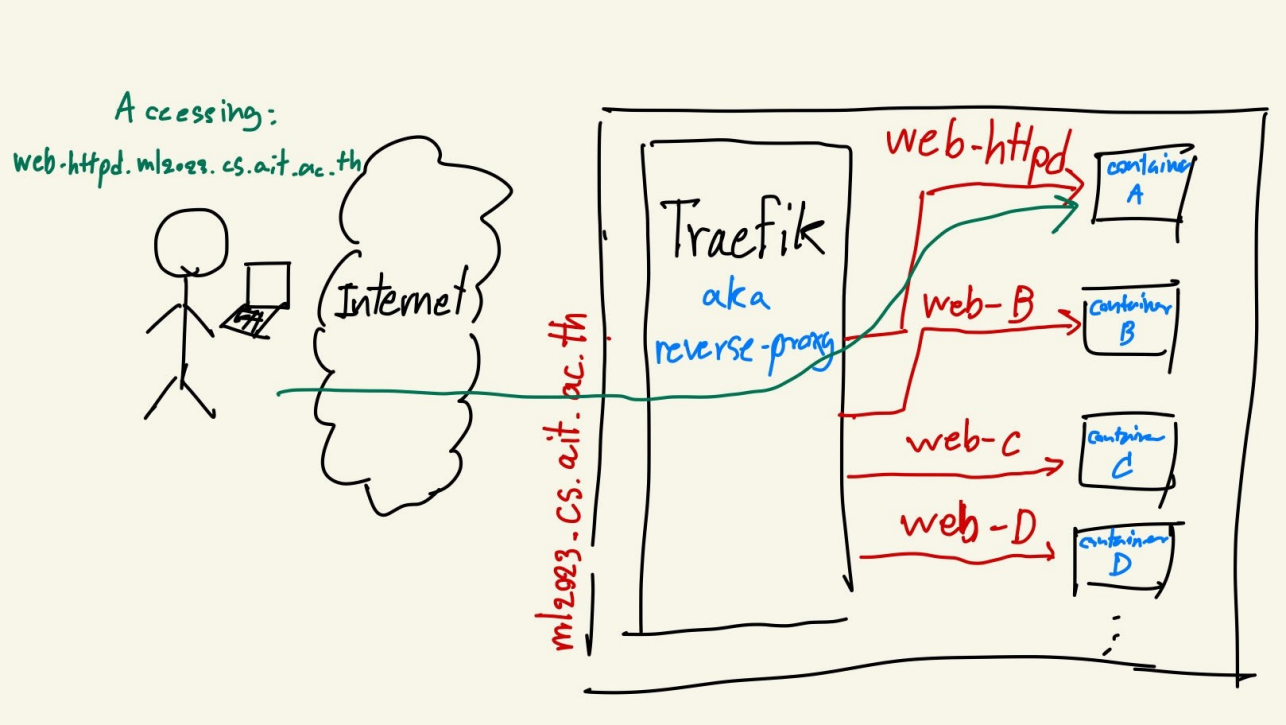
1) The user type in your URL into the browser and access the site.
2) (Optional) The user will land on the homepage. In the navigation bar is two menus indicate an old page and a newly developed page.
3) (Optional) The user clicks on the newly developed page to access the new model.
4) The site prompt message explaining how to use the new model and how the new model is better than the old one.
5) User inputs the appropriate data and clicks submit.
6) (Optional) The user waits for the output.
7) The output shows on the designated box.

This should not be a difficult task since the new page is mostly a clone of the existing one. The next objective is more fun.

Objective 2: Deploy to the server We have prepared a server inside CSIM for you to deploy your work. This is the information of that web server.

- **IP:** 192.41.170.25
- **URL:** ml.brain.cs.ait.ac.th
- **OS:** Ubuntu server 24.04

How is this done?



What we have currently is one server that serves multiple websites/services. Normally, each service would occupy a port, i.e. 80 or 443. However, based on the number of students in our class, we would have 40+ sites and I can not remember which port is whose. Thus, we use a unique subdomain that pairs with each service instead. So, the users can not access the service directly and can only access it through what is called the 'Reverse Proxy'.

For this assignment, I used 'Traefik' as a reverse proxy. One common benefit of having a reverse proxy in between the chain is off-loading SSL/TLS functionality to the reverse proxy. Thus, you, the developer, do not need to handle this yourself (which is bad for learning). The beauty of 'Traefik' is that it can talk to the Docker container without those containers exposing their service to the host. Thus, the developers, you, only need to prepare a container that, once spawned, serves the service. Thus, your website is not bounded/paired with the port.

Then, how do we access the services? Well, I already gave the answer. We use **subdomain**. In our case, `ml.brain.cs.ait.ac.th` and `*.ml.brain.cs.ait.ac.th` all means `192.41.170.25`. The '*' in front of the domain we call the sub-domain. With this, we can pair a unique sub-domain to each of the services. From the example, 'web-httpd' is paired with one site, and 'web-st121413' is paired with another site.

TL;DR All you need is access to the `ml-brain` server and Docker that is ready for deployment. You can come back and read this information again later. You will eventually need to understand this...

TO DO

(1) Have an access to `ml-brain` server. To keep this as a best practice, you will access the server using `SSH private key`.

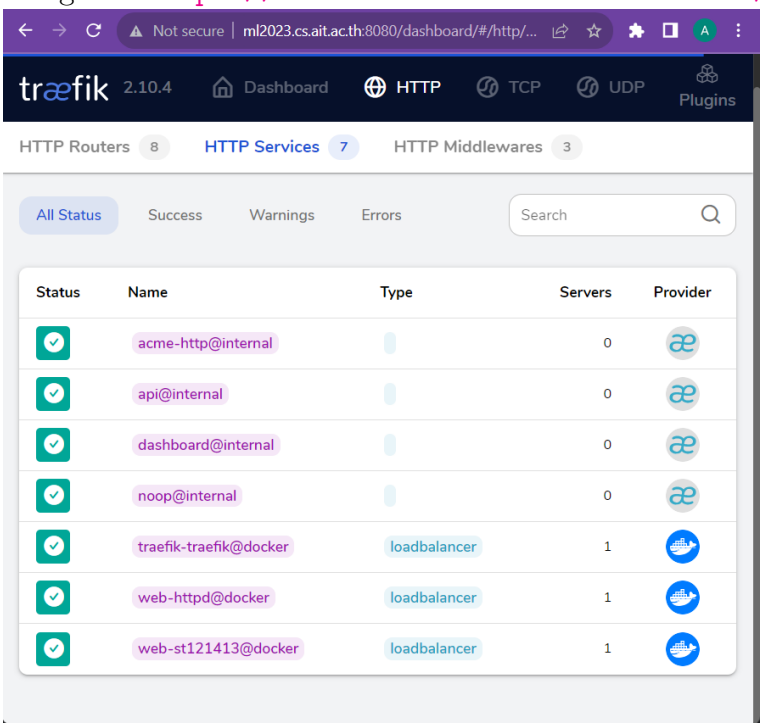
1) Create public/private key pair following this instruction 1. Create a public/private key.
2) Send an email to TA (st121413@ait.asia) using subject name **ML: private key of <your-stid>** and attach the **Public Key** in the email.
3) For some SLA, your TA will respond to indicate that your public key is installed in your account on the server. Thus, you can access the server using your private key. Try to access following this instruction 3. SSH to the server with the private key authentication option. (Change `bazooka` to `ml2023`). Note that you need to use CSIM wifi in order to access the server directly, otherwise, you will need to do `ProxyJump` (wow another fancy term) which is already discussed in the instruction link.

(2) Deploy the website on the server.

1) SSH to `ml-brain` server.
2) Verify that you have permission to run `docker` command with `docker ps`
   It should list down all the running containers. If you got either no containers or some error message, please contact TA.

```
st121413@ml2023:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED      STATUS      PORTS
                                                                                NAMES
ad179e7bde58   httpd          "httpd-foreground"       2 days ago   Up 2 days   80/tcp
                                                                                st121413-web-1
4888425f952d   httpd          "httpd-foreground"       2 days ago   Up 2 days   80/tcp
                                                                                httpd-web-1
b36eb4a02095   traefik:v2.10  "/entrypoint.sh --co…"   2 days ago   Up 2 days   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.
0.0:443->443/tcp, :::443->443/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   traefik-traefik-1
st121413@ml2023:~$
```

3) Prepare your Docker image and upload it to your Docker Hub.
4) In the server, use this docker-compose.yaml as a template to deploy your site.
5) If you are in CSIM and using CSIM wifi, you can verify that your container is recognized by 'Traefik' using this https://traefik.ml.brain.cs.ait.ac.th/



6) Give it a moment for 'Traefik' to finish issuing SSL/TLS and then you should be able to access your site using the domain specified in your `docker-compose.yaml`.

Good luck :-)