

# Report of technical test

---

POSIOLOGY CLASSIFICATION

Marwan AL OMARI

PHD APPLICANT | MARWAN.AL.OMARI@ETU.UNIV-POITIERS.FR

The training and the data preprocessing of the model are called by *make train*, which will initialize the model and create the docker image as well as the container. As we could see in the following figure (1), the docker image and container are created successfully.

**Figure 1-** make train command: create docker image and container with the trained model

```
marwan@Gamingologist: /mnt/c/Users/Marwan/rd-technical-test-master
marwan@Gamingologist:/mnt/c/Users/Marwan/rd-technical-test-master$ make train
docker build -t docker-ml-model -f train.Dockerfile .
[+] Building 1.4s (11/11) FINISHED
=> [internal] load build definition from train.Dockerfile                                0.0s
=> => transferring dockerfile: 44B                                                    0.0s
=> [internal] load .dockerignore                                                       0.0s
=> => transferring context: 2B                                                         0.0s
=> [internal] load metadata for docker.io/library/python:3.7                         1.2s
=> [internal] load build context                                                       0.1s
=> => transferring context: 80.12kB                                                    0.1s
=> [1/6] FROM docker.io/library/python:3.7@sha256:7f9798664c8b1cdaccc19f1b90105a27db736c861c8184198dc7a8c01f82cc 0.0s
=> CACHED [2/6] ADD negative_examples_posology.csv .                                0.0s
=> CACHED [3/6] ADD positive_examples_posology.csv .                                0.0s
=> CACHED [4/6] ADD build_model.py .                                                  0.0s
=> CACHED [5/6] RUN pip install scikit-learn                                         0.0s
=> CACHED [6/6] RUN pip install pandas                                              0.0s
=> exporting to image                                                                0.0s
=> => exporting layers                                                                0.0s
=> => writing image sha256:50bf15893ddd2b7c678bcf98db99f95327ac99323fc0faf634a5c381132480ef 0.0s
=> => naming to docker.io/library/docker-ml-model                                   0.0s

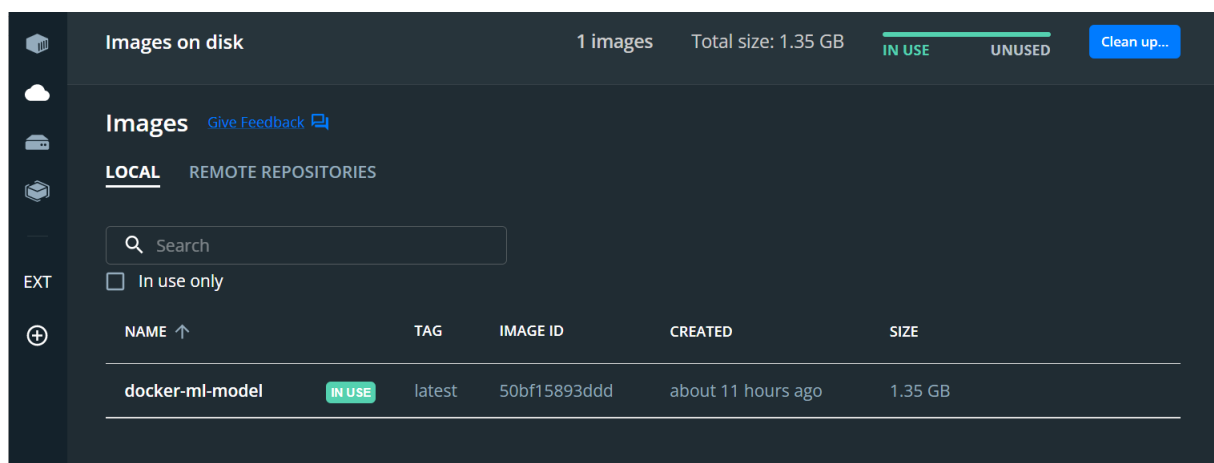
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
docker run docker-ml-model
precision    recall  f1-score   support

   0         0.93    0.97    0.95     299
   1         0.90    0.77    0.83     92

 accuracy          0.93    391
 macro avg         0.92    0.87    0.89    391
weighted avg         0.92    0.93    0.92    391
```

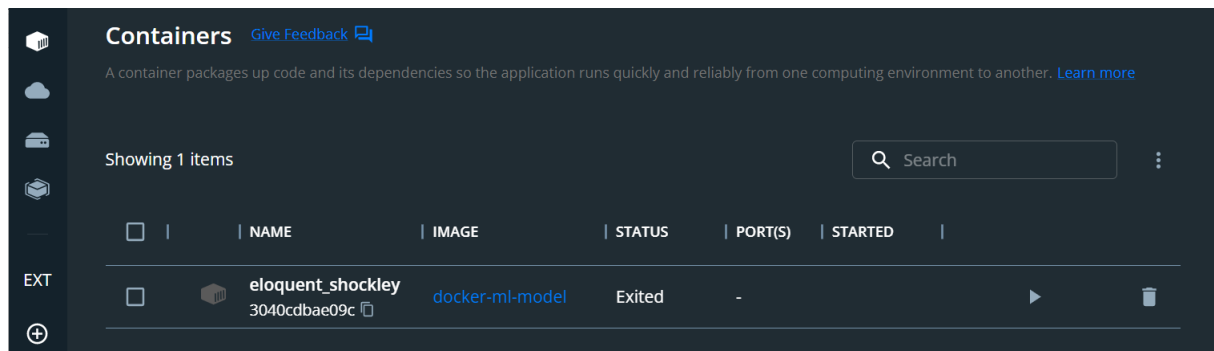
The following figure (2) shows the created image of machine learning model for production.

**Figure 2-** Docker image of the machine learning model



With the successful creation of docker image, the container is well placed for running the model and logging the model's progress. The container is well represented in the following figure (3).

**Figure 3-** Docker container of the machine learning model



Therefore, in the following figure (4), we could observe the logging of the machine learning image, which is the results of logistic regression model.

**Figure 4-** The logs of the machine learning model

The screenshot shows the 'Logs' tab for the 'eloquent\_shockley' container. It displays a table of log entries. The first part of the log shows precision, recall, f1-score, and support for two classes (0 and 1). The second part shows accuracy, macro avg, and weighted avg metrics.

	precision	recall	f1-score	support
0	0.93	0.97	0.95	299
1	0.90	0.77	0.83	92
accuracy			0.93	391
macro avg	0.92	0.87	0.89	391
weighted avg	0.92	0.93	0.92	391

On the other hand, for putting the model in real production as a microservice, the command *make api* would take the trained model in the first image into production using flask api on 4002 ports. The command and the process are well presented in the following figure (5):

**Figure 5-** The deployment of the machine learning model

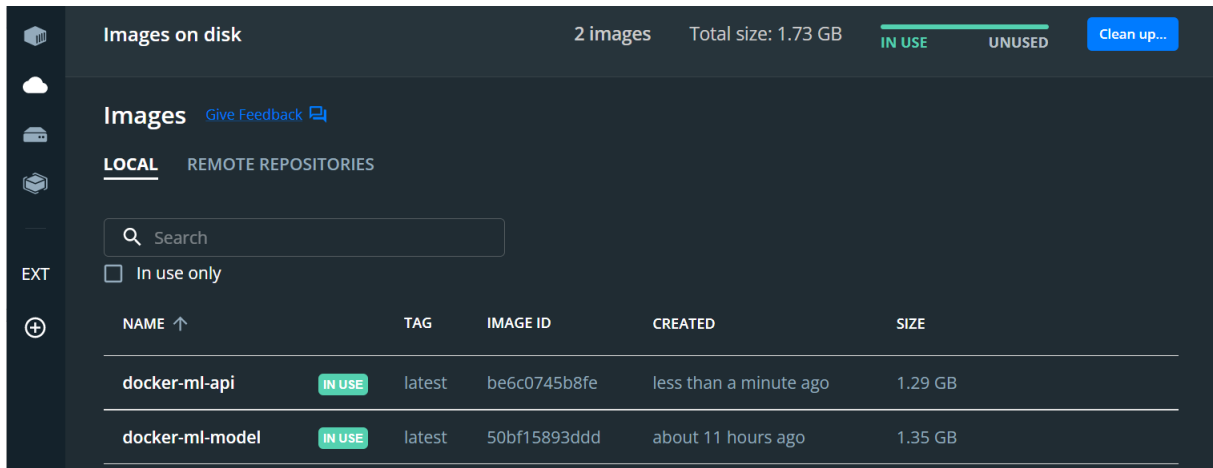
The screenshot shows a terminal window with the following commands and output:

```
marwan@Gamingologist:/mnt/c/Users/Marwan/rd-technical-test-master$ make api
docker build -t docker-ml-api -f api.Dockerfile .
[+] Building 142.5s (10/10) FINISHED
=> [internal] load build definition from api.Dockerfile 0.0s
=> => transferring dockerfile: 188B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/python:3.7 6.2s
=> [internal] load build context 0.3s
=> => transferring context: 390.26kB 0.3s
=> [1/5] FROM docker.io/library/python:3.7@sha256:7f9798664c8b1cdaccc19f1b90105a27db736c861c8184198dc7a8c01f82cc 0.0s
=> CACHED [2/5] WORKDIR /app 0.0s
=> [3/5] COPY . /app 0.0s
=> [4/5] RUN pip install flask 5.9s
=> [5/5] RUN pip install scikit-learn 128.3s
=> exporting to image 1.7s
=> => exporting layers 1.6s
=> => writing image sha256:be6c0745b8fe66d20a12d6680af95b054ebde9067a113c7bdf155d9f8a135f3 0.0s
=> => naming to docker.io/library/docker-ml-api 0.0s

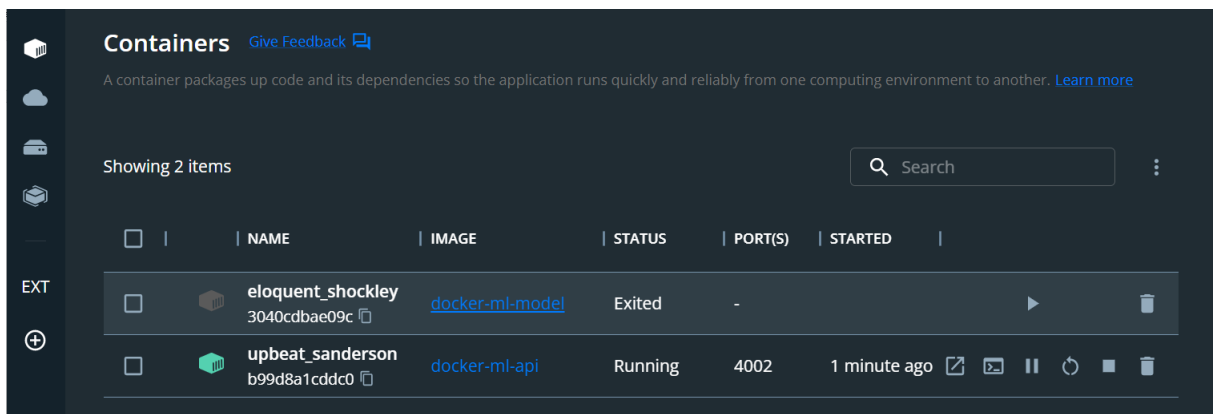
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
docker run -it -p 4002:4002 docker-ml-api
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses (0.0.0.0)
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:4002
* Running on http://172.17.0.2:4002 (Press CTRL+C to quit)
```

In the following two figure (6) and (7), the docker image and container of the deployed machine learning model are successfully created in docker.

**Figure 6-** Docker image of the deployed model

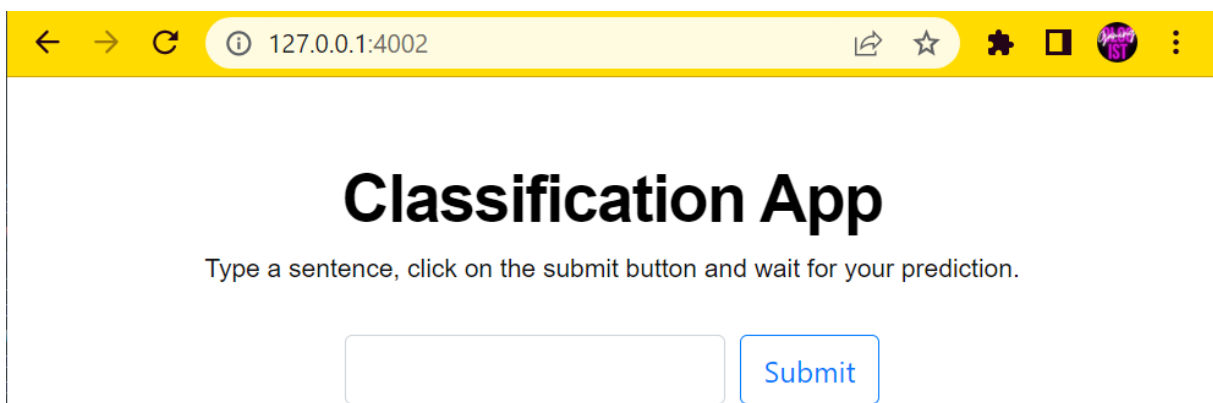


**Figure 7-** Docker container of the deployed model



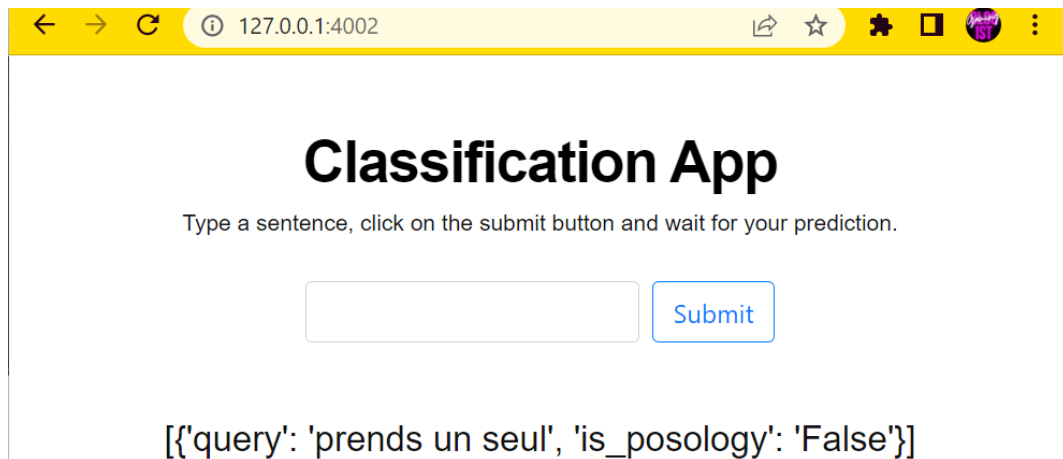
Hence, with the deployment of the model, now, it is ready in production, as accessing the local host in port 4002, as it is well represented in the following figure (8):

**Figure 8-** the model in production



Moreover, the following two figures (9) and (10) represent two examples of questioning the model. An example outputs false posology and the other output true positive.

**Figure 9-** A negative example of the deployed model

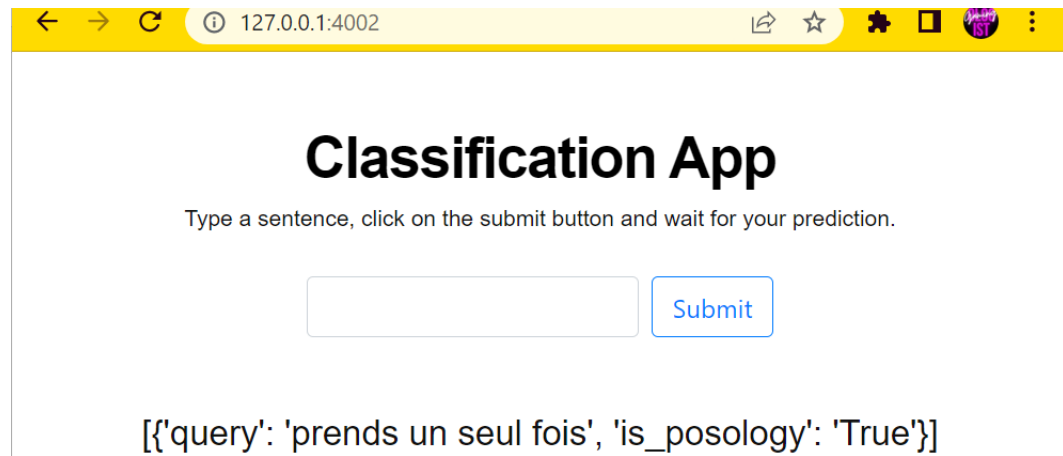


← → ↻ ⓘ 127.0.0.1:4002

# Classification App

Type a sentence, click on the submit button and wait for your prediction.

**Figure 10-** A positive example of the deployed model



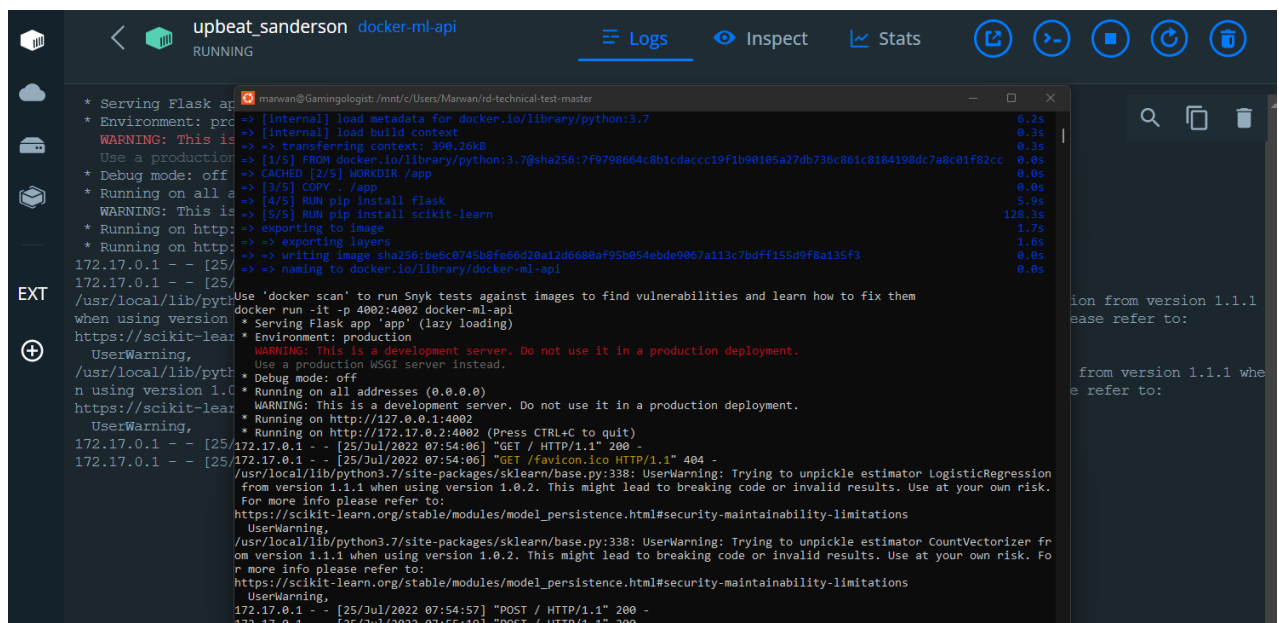
← → ↻ ⓘ 127.0.0.1:4002

# Classification App

Type a sentence, click on the submit button and wait for your prediction.

In conclusion, as we could see in the following figure (11), the model is well on production and the docker shows the logs in terminal as we have sent the two examples presented before.

**Figure 11-** The logs of the deployed model



```
upbeat_sanderson docker-ml-api
RUNNING

* Serving Flask app
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses (0.0.0.0)
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:4002 (Press CTRL+C to quit)
172.17.0.1 - - [25/Jul/2022 07:54:06] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [25/Jul/2022 07:54:06] "GET /favicon.ico HTTP/1.1" 404 -
/usr/local/lib/python3.7/site-packages/sklearn/base.py:338: UserWarning: Trying to unpickle estimator LogisticRegression from version 1.1.1 when using version 1.0.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
UserWarning,
/usr/local/lib/python3.7/site-packages/sklearn/base.py:338: UserWarning: Trying to unpickle estimator CountVectorizer from version 1.1.1 when using version 1.0.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
UserWarning,
172.17.0.1 - - [25/Jul/2022 07:54:57] "POST / HTTP/1.1" 200 -
172.17.0.1 - - [25/Jul/2022 07:55:19] "POST / HTTP/1.1" 200 -
```