AI Model In Production

Trung Nguyen
HAN University of Applied Sciences
Arnhem, The Netherlands

October 28, 2022

Contents

1 Introduction

Creating a model is the first step in making an AI (Artificial Intelligence) application. The next step is deploying or turning that model into a usable service. There are plenty of tutorials on how to deploy the machine learning model.

This report will only summarize the main points and give instructions for a quick demo on a simple web service-based prediction model.

The reader should have basic python knowledge.

2 AI model in production

Deploying a predictive model into production is quite complex. Choosing how to put models into production can depend on the specific use case. A good summary of different deployment approaches has been described in [1]. This section only shows a simple example of how to save and load a pre-trained model for production.

2.1 From model to production

Data scientists/engineer often do their prototype application in Jupyter Notebooks [2]. An ad-hoc trained model in Jupyter can be pushed to production with several types of libraries and other notebook providers.

The model can be save in different format:

- **Pickle** converts a python object to to a bitstream and allows it to be stored to disk and reloaded at a later time [3].
- **ONNX** the Open Neural Network Exchange format, is an open format that supports the storing and porting of predictive model across libraries and languages. Most deep learning libraries support it and sklearn [4] also has a library extension to convert their model to ONNX's format [3].
- **PMML** or predictive model markup language, is another interchange format for predictive models. Like for ONNX sklearn also has another library extension for converting the models to PMML format. It has the drawback however of only supporting certain type of prediction models.PMML has been around since 1997 and so has a large footprint of applications leveraging the format. Applications such as SAP for instance is able to leverage certain versions of the PMML standard, likewise for CRM applications such as PEGA [3].
- **POJO and MOJO** are H2O.ai's export format, that intendeds to offers an easily embeddable model into java application. They are however very specific to using the H2O's platform [3].

2.2 Save and Load the model

Python pickle module is one of the most popular object serialization packages. It serializes objects so they can be saved to a file and loaded into a program again later.

Datacamp website give a good explanation on Pickle in Python, when (not) to use it, how to compress pickled objects, multiprocessing [5]

In figure 1 the scikitlearn model can be saved and loaded with a few lines of code.

```
import pickle

# save the model to disk
filename = 'Skit_learn_model.sav'
pickle.dump(regressor, open(filename, 'wb'))
```

```
# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test_best, y_test)
print(result)
```

```
0.9317195953761216
```

Figure 1: Save and Load scikit learn model with pickle

The model also can be trained with "GPU" to accelerate the training process and load with "CPU" later to save the memory.

In figure 2 and 3 the model had been trained with GPU and later load in CPU using pytorch libraries.

```
PATH = "Solar_prediction_model.pt"
torch.save(lstm, PATH)
```

Figure 2: Save Pytorch model

```
input_size = 8
hidden_size = 128
num_layers = 1
seq_length =4
num_classes = 1
bidirectional = True

model = LSTM(num_classes, input_size, hidden_size, num_layers,bidirectional)

PATH = "../input/pv-model/Solar_prediction_model.pt"
# Load
model = torch.load(PATH,map_location='cpu')
model.eval()
```

```
.STM(
  (lstm): LSTM(8, 128, batch_first=True)
  (fc): Linear(in_features=128, out_features=1, bias=True)
)
```

Figure 3: Load Pytorch model

3 Model deployment

3.1 Deploy the model locally using Voila

Voilà allows you to convert a Jupyter Notebook into an interactive dashboard that allows you to share your work with others. It is secure and customizable, giving you control over what your readers experience [6].

Install Voila with conda (figure 4).

Figure 4: Voila package install

Move to the Jupiter notebook directory using cd "name of directory" and Run the Voila command : voila notebooknames.ipynb. An example is shown in figure 5.

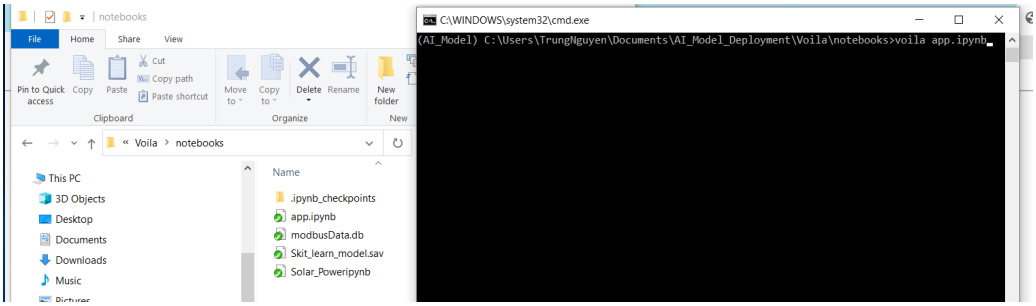

Figure 5: Voila Run Command

A web browser with localhost address (figure 6) will be opened

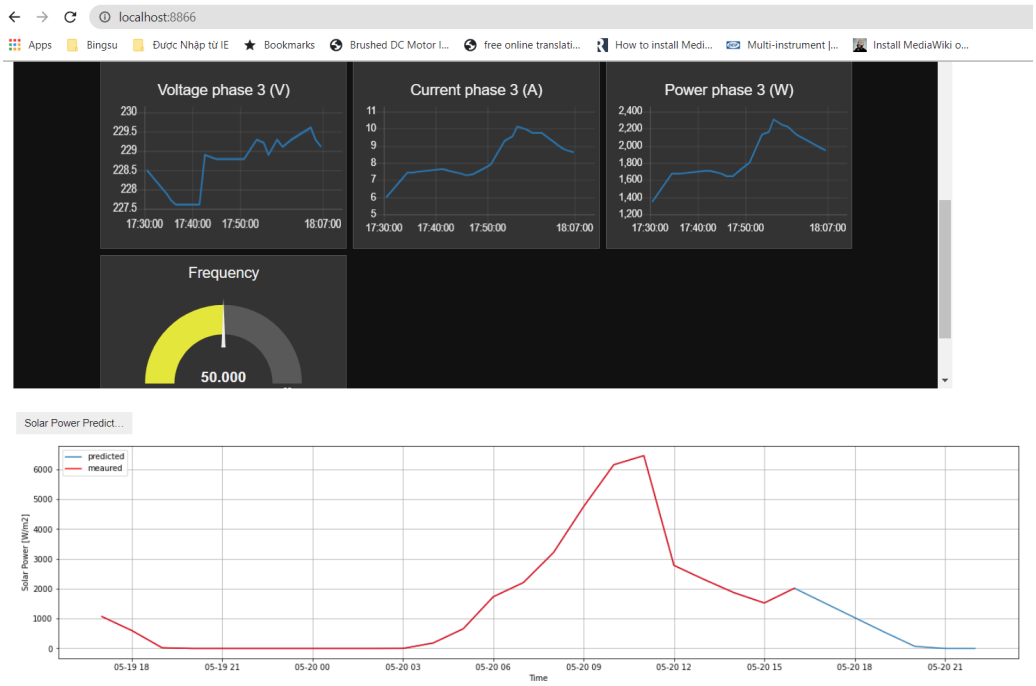

Figure 6: Voila Web

3.2 Deploy the model with Heroku

Heroku is a cloud Platform that helps developers quickly deploy, manage, and scale moderns applications. Heroku provides a free package for noncommercial apps such as proof of concept and personal project7.

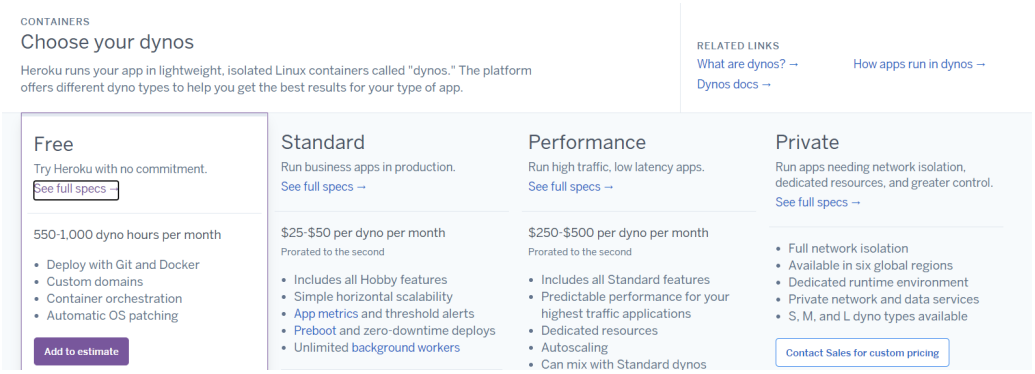

Figure 7: Heroku Pricing

The first step is to create three files that Heroku requires.

- requirements.txt : tells Heroku which Python packages to install when it run the web app.
- runtime.txt : specifies the version of Python we want Heroku to use.
- procfile: This file includes the instructions for Heroku to deploy our Voila app.

An example github prepare for Heroku deployment is shown on figure 8. The notebooks folder content the jupyternotebook.

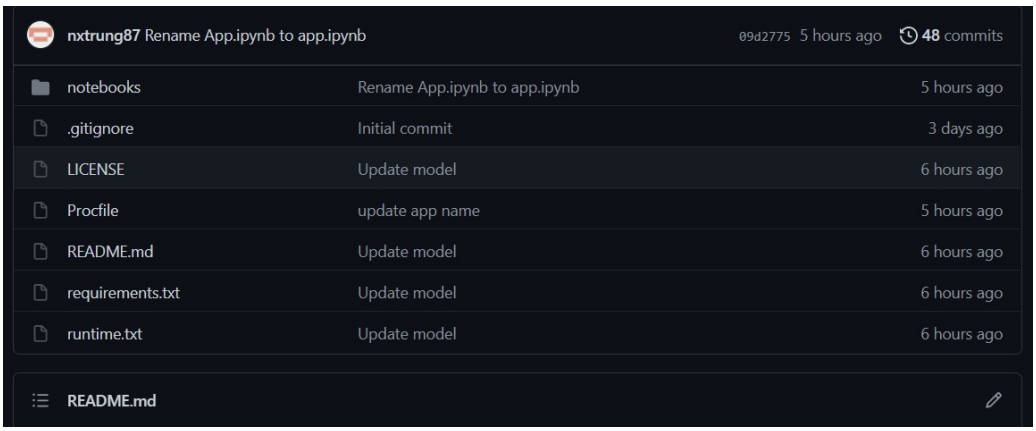

Figure 8: Heroku github example

The contents example of the three Heroku require files is shown on figure 9. On the top left of the figure is the requirements.txt, top right is the runtuime.txt and bottom is the Procfile.

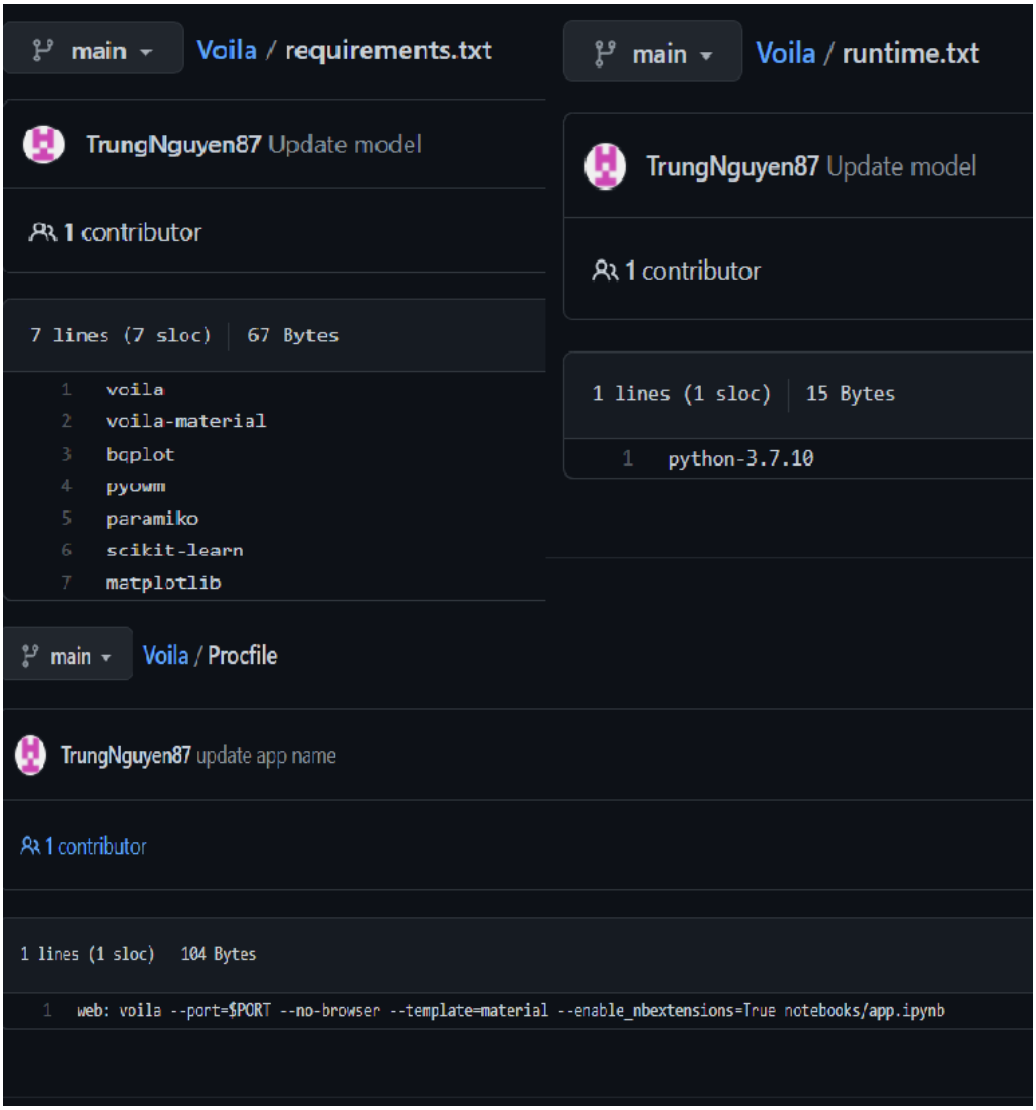

Figure 9: Heroku files content example

Next step is create an account with Heroku and follow the instruction in figure 10, 11, 12, 13, 14

In figure 10 select new on the right corner.

Figure 10: Create new app

Give a name for the app using lower letter case, select the deployment region and then select the create app button on the bottom.

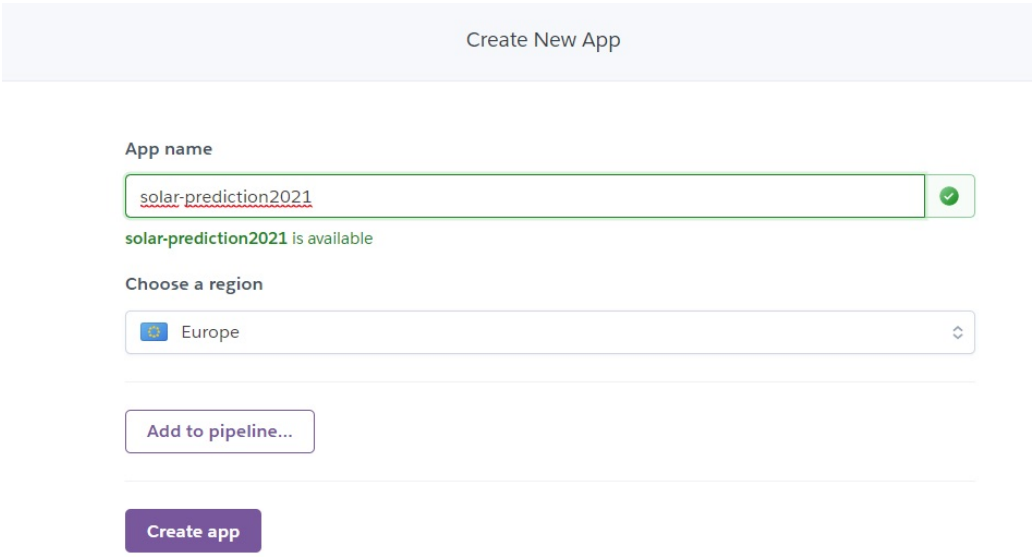

Figure 11: Create new app ctn

In figure 12 select the Deploy tab and connect your app github repository. Scroll down and choose to enable the automatic deploys in figure 13.

Automatic deploys means every time when something is changed (for example update the requirement file or code file), the web app will also be updated.

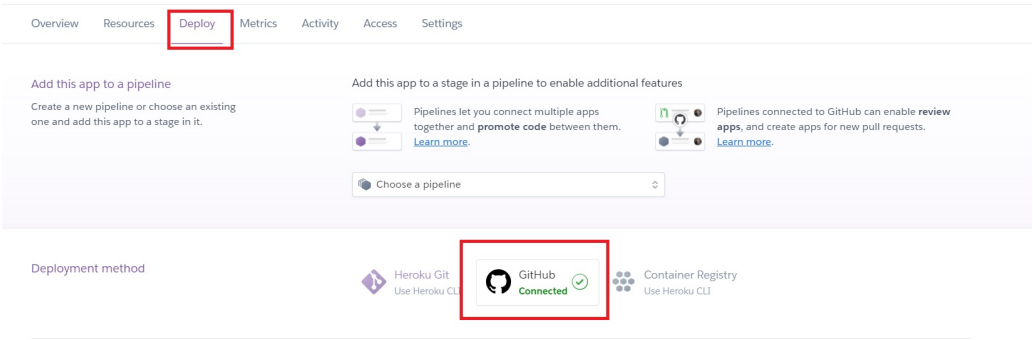

Figure 12: Set up deployment

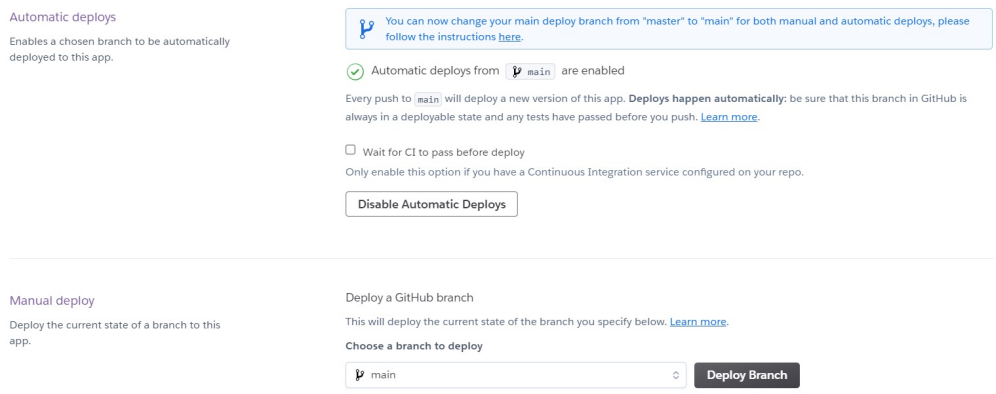

Figure 13: Set up deployment method

Link to the web app can be found on setting tab, scroll down to domains section (figure 14). User can also add their own domain name.

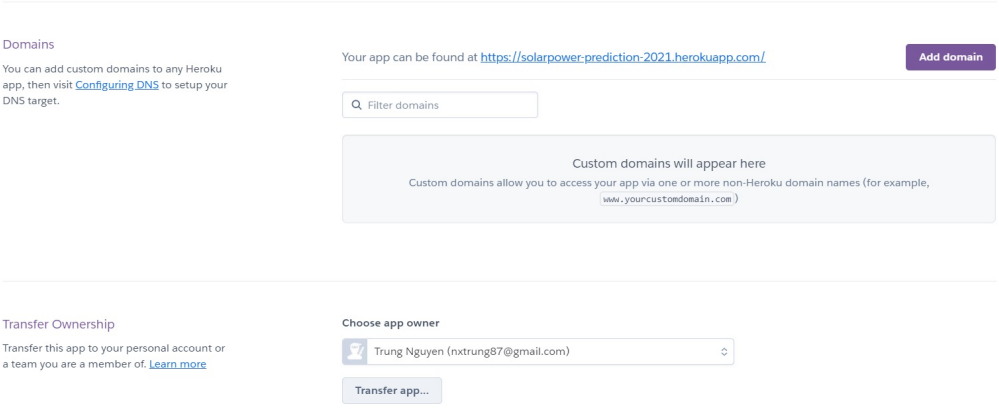

Figure 14: Link of the web app

Figure 15 is a demo web app.

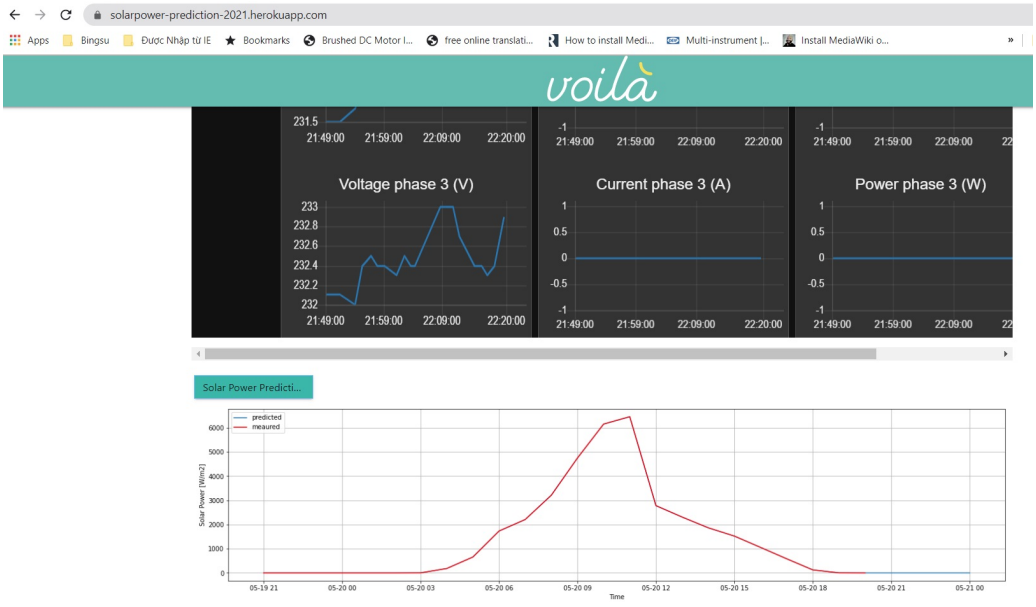

Figure 15: The demo webapp

3.3 Forked the demo Github

For the users who would like to try something quick. Forked the demo GitHub repository (figure 16) using the forked button on the top right of the demo GitHub page and follow the instruction to set up the deployment App with Heroku in the previous section.

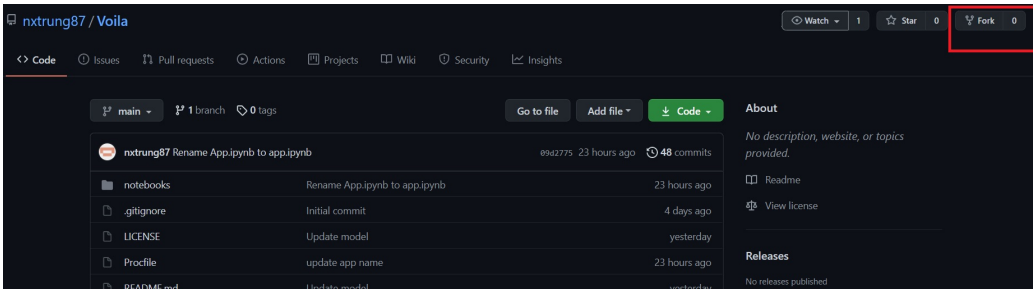

Figure 16: Forked a repository

A few more demo github page to try out:

- voila-heroku.
- voila-demo.
- jupyterlab-heroku.

4 Link to relevant information

More information about online - offline deployment platform can be found on the links below.

- 10 Ways to Deploy and Serve AI Models to Make Predictions
- 7 ML Model Deployment Cloud Platforms
- Deploying Voilà

References

[1] *Deploying machine learning in production.* URL: https://www.kdnuggets.com/2019/06/approaches-deploying-machine-learning-production.html.

[2] *Jupiter Notebooks.* URL: https://jupyter.org/.

[3] *Overview of Different Approaches to Deploying Machine Learning Models in Production.* URL: https://www.kdnuggets.com/2019/06/approaches-deploying-machine-learning-production.html.

[4] *Scikit-learn, Simple and efficient tools for predictive data analysis.* URL: https://scikit-learn.org/stable/.

[5] *Pickle in Python: Object Serialization.* URL: https://www.datacamp.com/community/tutorials/pickle-python-tutorial.

[6] *Voila.* URL: https://voila.readthedocs.io/en/stable/.