NAME: BERIL CHEPKORIR

BATCH CODE: LISP01

SUBMISSION DATE:29/03/2021

SUBMITTED TO: https://github.com/dangerous2/Cloud-and-API-Deployment

MODEL DEPLOYMENT USING HEROKU

STEPS:

- Train ML model
- Create a web app using Flask
- Commit the code to GitHub
- Connect GitHub to Heroku
- Deploy the model

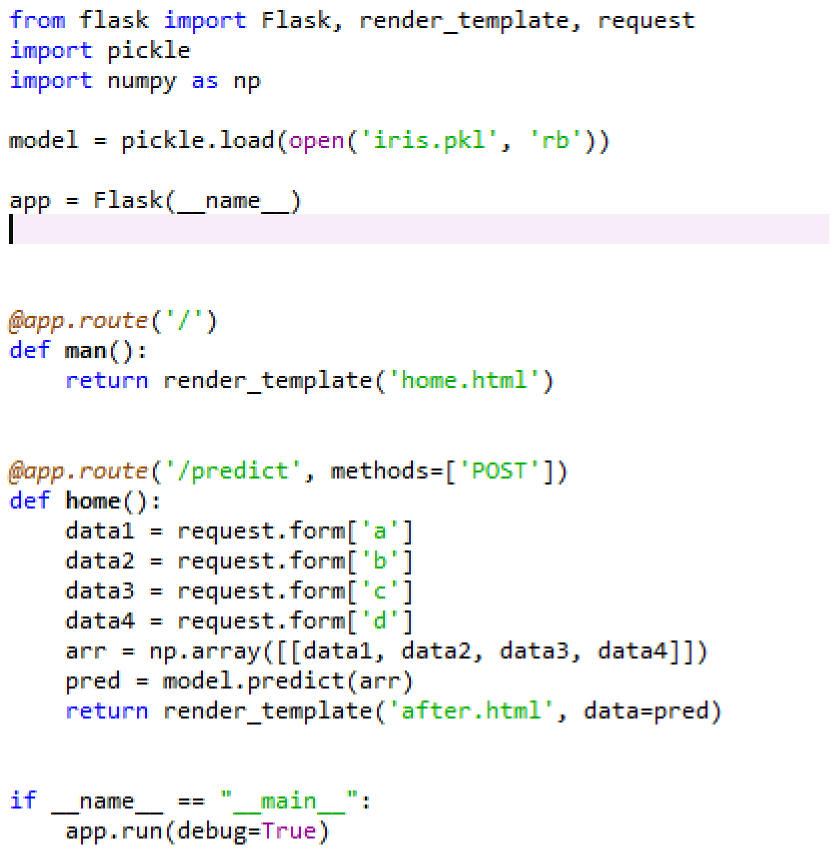
Training Machine Learning Model

I am using Support Vector Classifier as the classification algorithm. I have used iris dataset to predict the type of flower

```
import pandas as pd
import numpy as np
import pickle
import sklearn

df = pd.read_csv('Iris.csv')

#X = df.drop('Species', axis=1)
X = np.array(df.iloc[:, 0:4])
y = df.Species

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

from sklearn.svm import SVC
sv = SVC(kernel='linear').fit(X_train,y_train)


pickle.dump(sv, open('iris.pkl', 'wb'))
```

I have used pickle library, which basically helps you dump your model in the form character stream

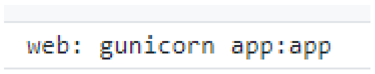
Creating a web app using flask



```python
from flask import Flask, render_template, request
import pickle
import numpy as np

model = pickle.load(open('iris.pkl', 'rb'))

app = Flask(__name__)


@app.route('/')
def man():
    return render_template('home.html')


@app.route('/predict', methods=['POST'])
def home():
    data1 = request.form['a']
    data2 = request.form['b']
    data3 = request.form['c']
    data4 = request.form['d']
    arr = np.array([[data1, data2, data3, data4]])
    pred = model.predict(arr)
    return render_template('after.html', data=pred)


if __name__ == "__main__":
    app.run(debug=True)
```

I have also created templates which will serve as the home page, where we will have all the field required to run the model.

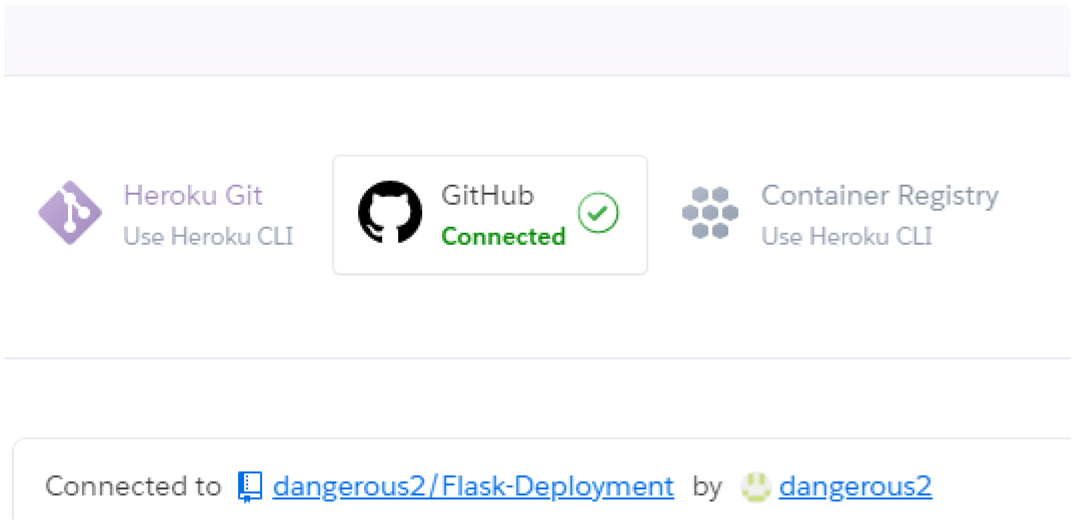
Most important thing is to create a Procfile and requirement.txt, which handles the configuration part in order to deploy the model into Heroku server



```
web: gunicorn app:app
```

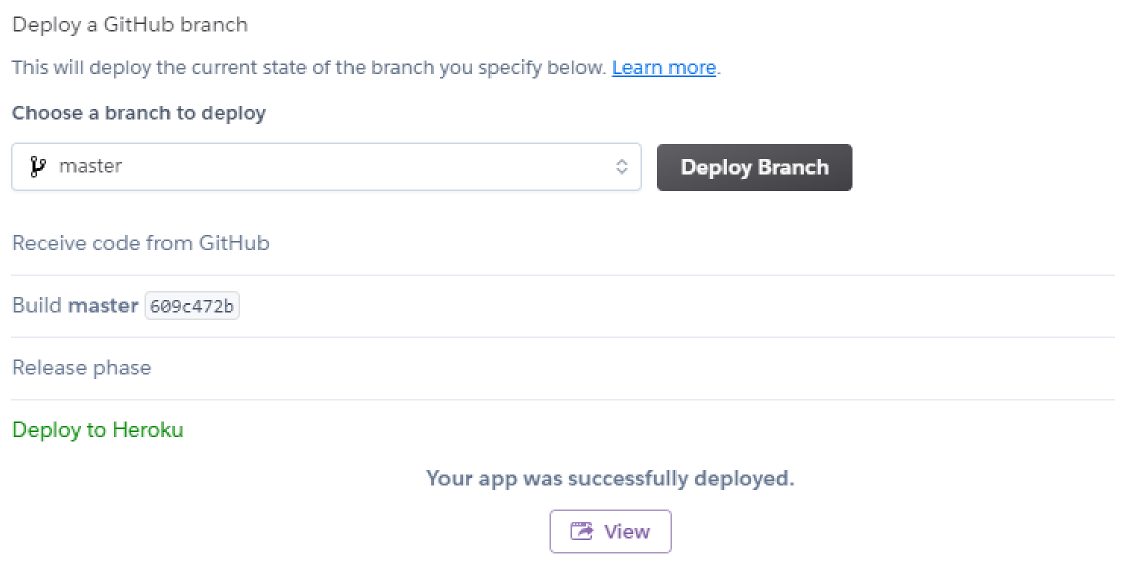
Commit your code to GitHub and connect Heroku to GitHub

Create an app name in Heroku and connect your GitHub repository as shown below



Connected to dangerous2/Flask-Deployment by dangerous2

After you connect, there are 2 ways to deploy your app. You could either choose automatic deploy or manual deploy. Automatic deployment will take place whenever you commit anything into your github repository. Automatically the build will start. I have deployed it using manual deploy.
Just by selecting the branch and clicking on deploy, build will start. After successful deployment, deployment tab should look as shown below:



Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

master

Deploy Branch

Receive code from GitHub

Build **master** `609c472b`

Release phase

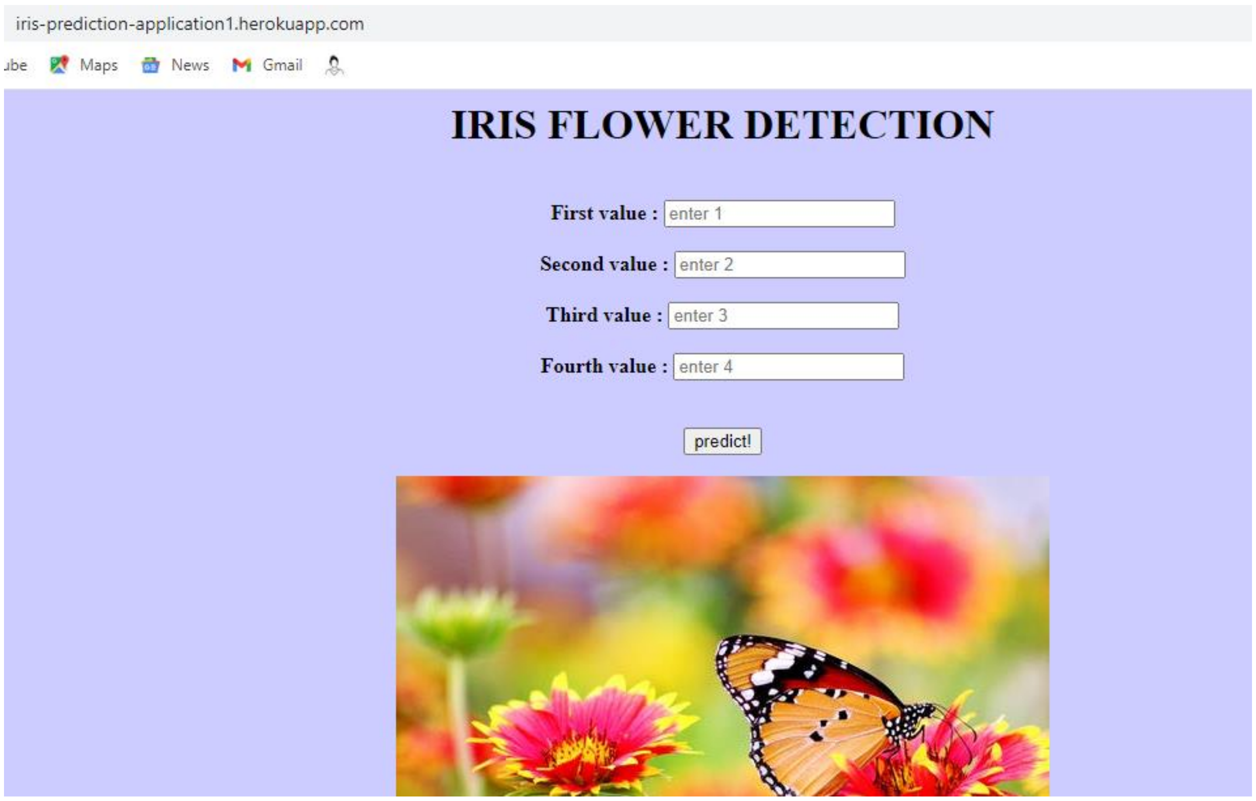
Deploy to Heroku

Your app was successfully deployed.

View

The final Application:

After successful deployment, app will be created



IRIS FLOWER DETECTION

First value : enter 1

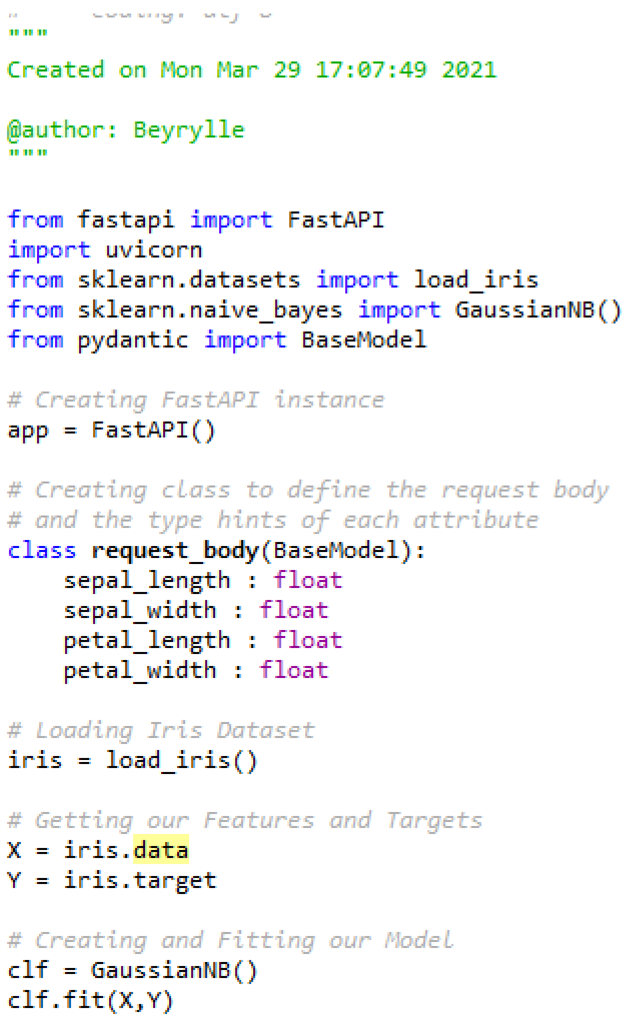
Second value : enter 2

Third value : enter 3

Fourth value : enter 4

DEPLOYING MACHINE LEARNIG MODEL WITH FASTAPI

1. Imports – you'll need pandas, NumPy GaussianNB () from scikit-learn, BaseModel from pydantic
2. Declare a class request body which inherits from BaseModel. This class contains only fields which are used to predict a single flower species
3. Make an instance of FASTAPI
4. Load the dataset and fit the model
5. Creating an Endpoint to receive the data to make prediction on.



```python
"""
Created on Mon Mar 29 17:07:49 2021

@author: Beyrylle
"""

from fastapi import FastAPI
import uvicorn
from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB()
from pydantic import BaseModel

# Creating FastAPI instance
app = FastAPI()

# Creating class to define the request body
# and the type hints of each attribute
class request_body(BaseModel):
    sepal_length : float
    sepal_width : float
    petal_length : float
    petal_width : float

# Loading Iris Dataset
iris = load_iris()

# Getting our Features and Targets
X = iris.data
Y = iris.target

# Creating and Fitting our Model
clf = GaussianNB()
clf.fit(X,Y)
```

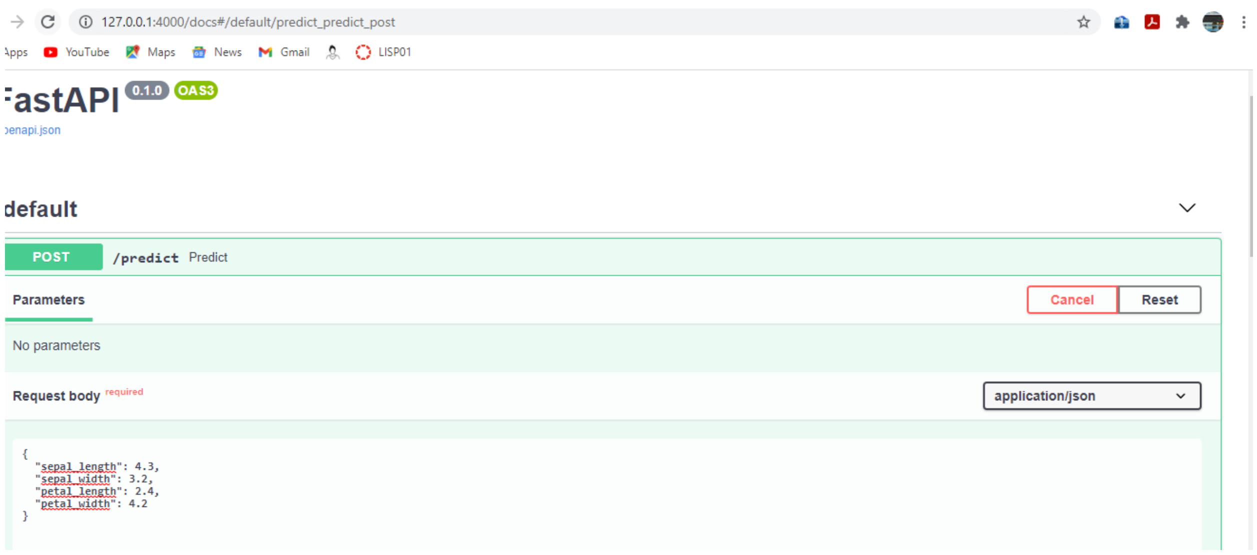
```python
# Creating an Endpoint to recieve the data
# to make prediction on.
@app.post('/predict')
def predict(data : request_body):
    # Making the data in a form suitable for prediction
    test_data = [[
            data.sepal_length,
            data.sepal_width,
            data.petal_length,
            data.petal_width
    ]]

    # Predicting the Class
    class_idx = clf.predict(test_data)[0]

    # Return the Result
    return { 'class' : iris.target_names[class_idx]}



if __name__ == "__main__":
    uvicorn.run(app,host ='127.0.0.1',port =4000)
```

Run the file and below are the results

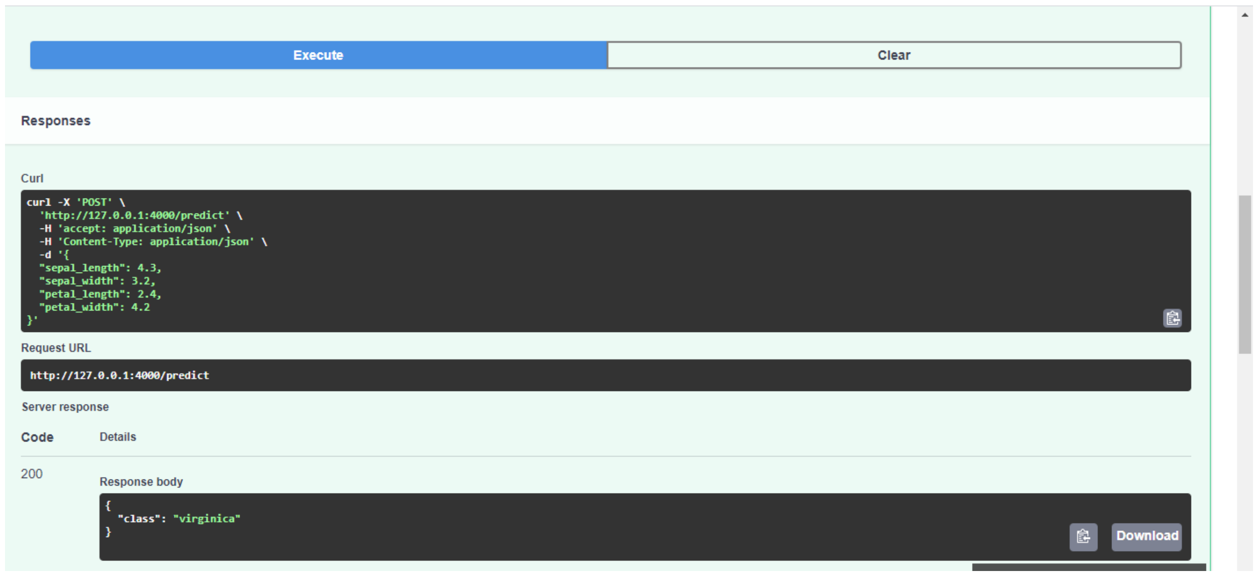
```
INFO:     Started server process [11064]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:4000 (Press CTRL+C to quit)

In [3]: INFO:     127.0.0.1:9867 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:9868 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:9874 - "POST /predict HTTP/1.1" 200 OK
```

Copy the address to the browser and add docs at the end eg: http://127.0.0.1:4000/docs and run it

Fill in the details



```
{
  "sepal_length": 4.3,
  "sepal_width": 3.2,
  "petal_length": 2.4,
  "petal_width": 4.2
}
```



```
curl -X 'POST' \
  'http://127.0.0.1:4000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "sepal_length": 4.3,
  "sepal_width": 3.2,
  "petal_length": 2.4,
  "petal_width": 4.2
}'
```

```
http://127.0.0.1:4000/predict
```

200

```
{
  "class": "virginica"
}
```