

Cloud and API Deployment

Name: Huu Thien Nguyen

Submission date: 4th June 2022

Submitted to: Data Glacier canvas platform

Internship Batch: LISUM09

Methods

The description below contains the detailed approach, including a snapshot of each step of the deployment process.

1. Select data

The Iris data was chosen for this implementation due to this particular dataset's simplicity and low resource cost. The Iris flower data set, often known as Fisher's Iris data set, is a multivariate data set first published in 1936 by British statistician, eugenicist, and biologist Ronald Fisher as an example of linear discriminant analysis in his paper The use of numerous measurements in taxonomic issues. Sepal length, sepal width, petal length, and petal width are the four independent aspects of sizes. The output is the flower's class.

iris				
Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
5.1	3.5	1.4	0.2	Setosa
4.9	3	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Setosa
5	3.6	1.4	0.2	Setosa
5.4	3.9	1.7	0.4	Setosa
4.6	3.4	1.4	0.3	Setosa
5	3.4	1.5	0.2	Setosa
4.4	2.9	1.4	0.2	Setosa
4.9	3.1	1.5	0.1	Setosa
5.4	3.7	1.5	0.2	Setosa

Figure 1: Iris dataset features

2. Save the model

The model was coded using IntelliJ IDE. It read the input file and the Iris dataset, performed the feature engineering with the Standard Scaler library, and then applied the Random Forest model with the split train and test data. After running the python code, a package file was created and ready to be deployed.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle

# Load the csv file
df = pd.read_csv("iris.csv")

print(df.head())

# Select independent and dependent variable
X = df[["Sepal_Length", "Sepal_Width", "Petal_Length", "Petal_Width"]]
y = df["Class"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Instantiate the model
classifier = RandomForestClassifier()

# Fit the model
classifier.fit(X_train, y_train)

# Make pickle file of our model
pickle.dump(classifier, open("model.pkl", "wb"))
```

Figure 2: Model creation using IntelliJ IDE

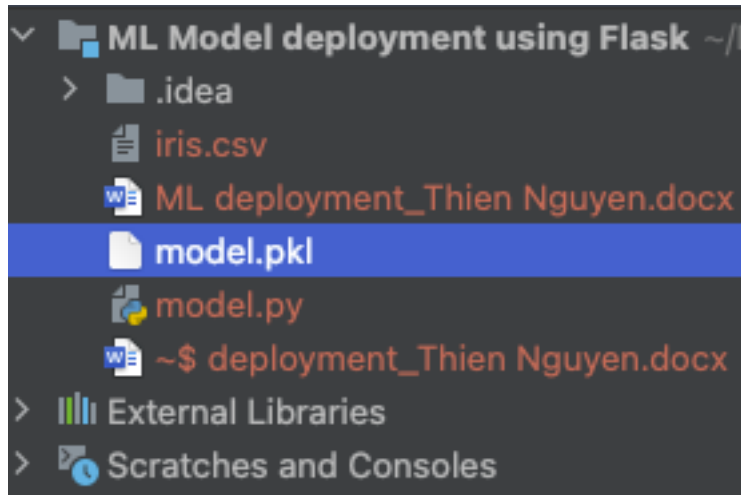


Figure 3: A package file was created

3. Prepare Flask application

The app.py contains the Flask code to build the application. Flask framework and code were included in the flask library. The homepage, route settings, and prediction are specified in each function.

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

# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text = "The flower species is {}".format(prediction))

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

Figure 4: Flask application deployment

4. Webpage setup

The front end of the homepage website was designed in the index.html file.

```
<!DOCTYPE html>
<html >
  <!--From https://codepen.io/frytyler/pen/EGdtg-->
  <head>
    <meta charset="UTF-8">
    <title>ML API</title>
    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
  </head>
  <body>
    <div class="login">
      <h1>Flower Class Prediction</h1>

      <!-- Main Input For Receiving Query to our ML -->
      <form action="{{ url_for('predict')}}" method="post">
        <input type="text" name="Sepal_Length" placeholder="Sepal_Length" required="required" />
        <input type="text" name="Sepal_Width" placeholder="Sepal_Width" required="required" />
        <input type="text" name="Petal_Length" placeholder="Petal_Length" required="required" />
        <input type="text" name="Petal_Width" placeholder="Petal_Width" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
      </form>

      <br>
      <br>
      {{ prediction_text }}
    </div>
```

Figure 5: Homepage in HTML code

5. Create an account on Heroku

Create a new account, log in on Heroku, and create a new app.

Create New App

App name

predict-app-thiennguyen

predict-app-thiennguyen is available

Choose a region

Europe


Add to pipeline...


Create app


Figure 6: Create a new app on Heroku

6. Connect to the GitHub repository

Deployment method

 Heroku Git
Use Heroku CLI

 GitHub
Connect to GitHub

 Container Registry
Use Heroku CLI

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

View your code diffs on GitHub

Connect your app to a GitHub repository to see commit diffs in the activity log.

Deploy changes with GitHub

Connecting to a repository will allow you to deploy a branch to your app.

Automatic deploys from GitHub

Select a branch to deploy automatically whenever it is pushed to.

Create review apps in pipelines

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. [Learn more.](#)

Connect to GitHub

Figure 7: Connect to the GitHub repository

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to [AndrewNguyen27296/VC](#) by [AndrewNguyen27296](#) [Disconnect...](#)

Releases in the [activity feed](#) link to GitHub to view commit diffs

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please [follow the instructions here.](#)

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more.](#)

Choose a branch to deploy

[nguyenuuthien27296@gmail.com](#)

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

Choose a branch to deploy

[nguyenuuthien27296@gmail.com](#) [Deploy Branch](#)

Figure 8: Connect success

7. Deploy project to Heroku

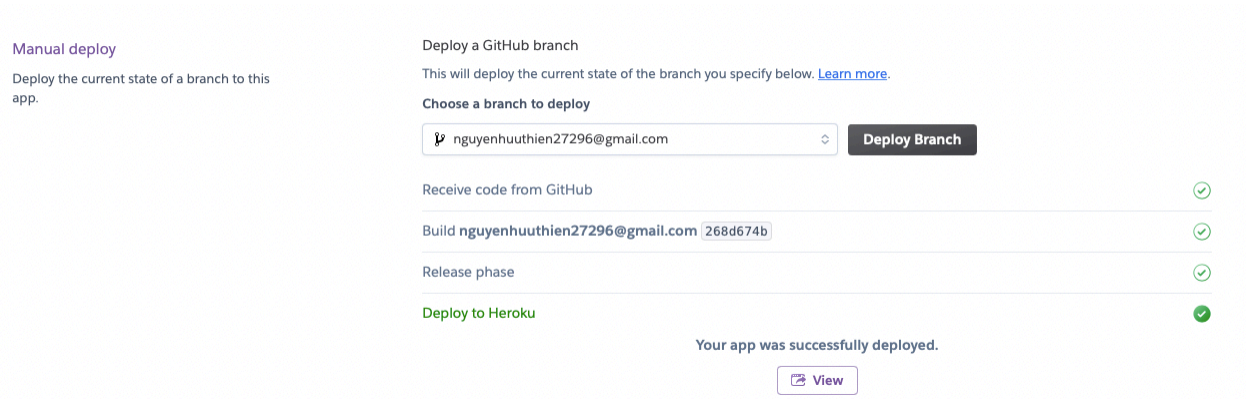


Figure 9: Deploy success



Figure 10: Website front-end