Kristina Kaliagina
Submission date: 26.07.2022
Submitted to: Data Glacier

Deployment on Flask

Batch code:

```
$python app.py
```

Steps of deployment:

1. Dataset

The data was taken from the library of *sklearn.datasets*, which are called *'iris'* and contain information about these flowers:

- petal length;
- petal width;
- sepal length;
- sepal width;
- class of irises.

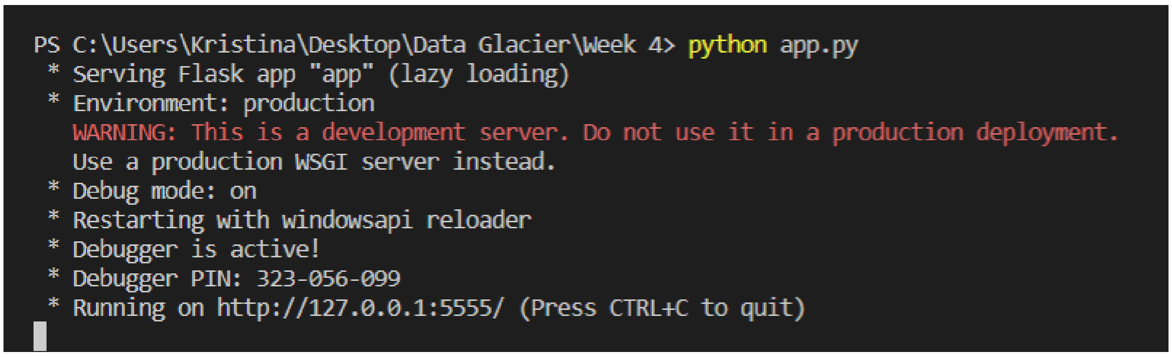
2. Build the model

A *LogisticRegression* model from the *sklearn* package was used to predict the class of irises, saved in pickle format (filename - *build.py*).

3. Flask App

The next step in deploying the model is to create a *Flask* application with input and output data for the model (filename - *app.py*).

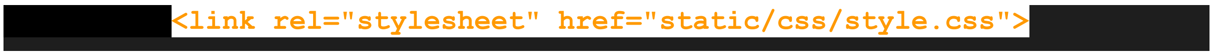
Below is a screenshot with *a command* to run the application, as well as *a link* where you can use the web application:



```
PS C:\Users\Kristina\Desktop\Data Glacier\Week 4> python app.py
 * 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: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 323-056-099
 * Running on http://127.0.0.1:5555/ (Press CTRL+C to quit)
```

4. Visualization of the App

To implement the application, the *HTML* file was additionally used in conjunction with the *CSS* file, using the following line of code:



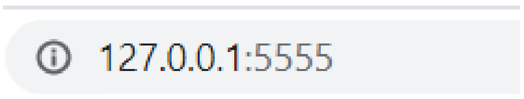
```
<link rel="stylesheet" href="static/css/style.css">
```

- *HTML* file builds a web form where the user can enter input values and get a prediction (filename - *web.html*);
- *CSS* file is for the stylistic decisions of the application (filename - *style.css*).

This can also be written to a single *HTML* file (as in our case).

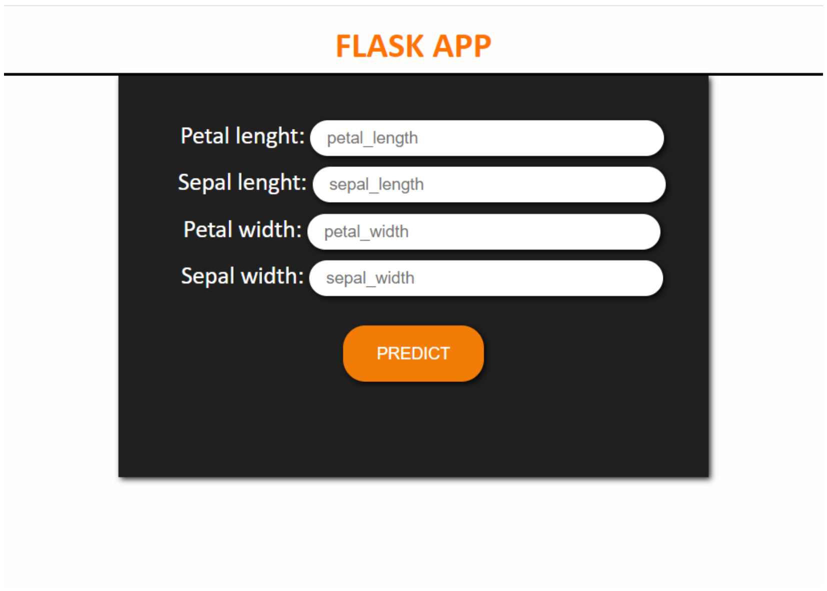
5. Web-application

The *web link* that took us to the browser directly to the *Flask* application.



127.0.0.1:5555

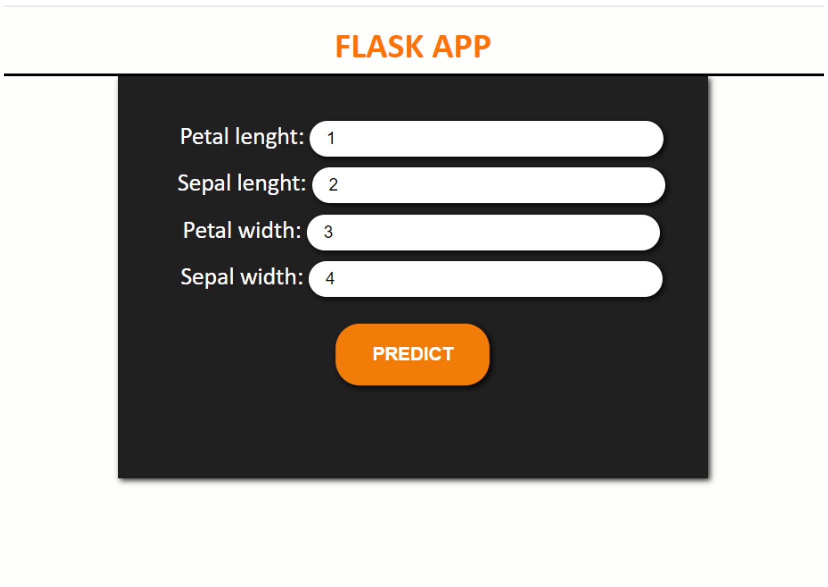
The *web form* of the *flask* application looks like:



FLASK APP

Petal lenght:	petal_length
Sepal lenght:	sepal_length
Petal width:	petal_width
Sepal width:	sepal_width

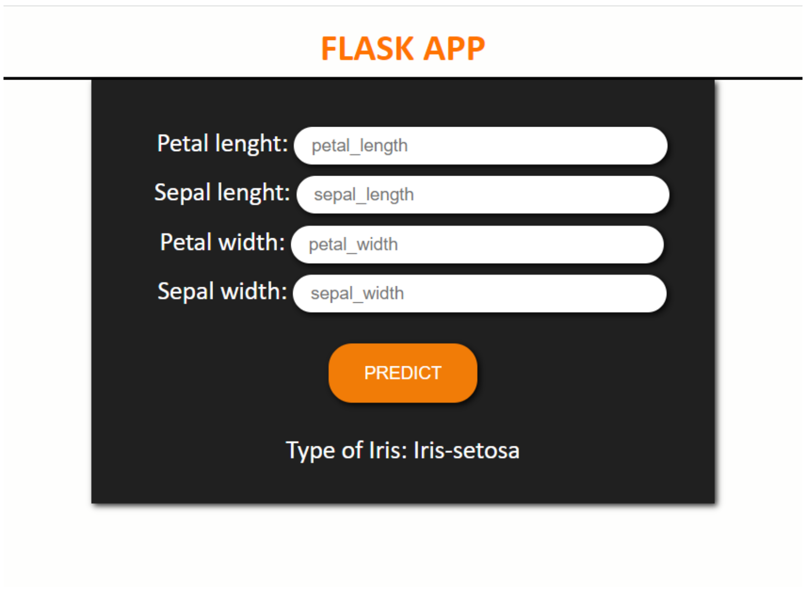
Let's fill in the *input data* and click on the *"predict"* button:



FLASK APP

Petal lenght:	1
Sepal lenght:	2
Petal width:	3
Sepal width:	4

Display the *result* for the *input* values using the web version of the *model*:

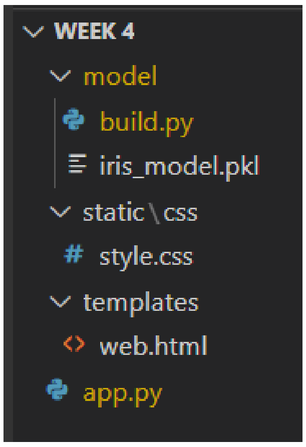


Type of Iris: Iris-setosa

The *type of iris* with the following parameters is called "*Iris-setosa*":

- petal length=1;
- petal width=2;
- sepal length=3;
- sepal width=4.

*The structure of the project is presented below, as well as with it and code of the Flask application, you can find the link on the github: *https://github.com/kkalyagina/Data-Glacier*



```
WEEK 4
  model
    build.py
    iris_model.pkl
  static\css
    style.css
  templates
    web.html
  app.py
```