Deployment on Flask

Name: Ignacio Solórzano

Batch code: LISP01

Submitted date: 22/03/2021

Submitted to: Data Glacier

Introduction

The objective of this work is to create and deploy (local) a machine learning model to make predictions (in this use case) via http.

This work is divided into 4 parts:

1- Serialization of Machine Learning Model
2- Installation of Flask and the libraries required
3- Modules of this project
4- Execution and deployment

1- Serialization of Machine Learning Model.

This model is a Random Forest Classifier aims to classify between 3 kinds of Iris flowers from the well-known **Iris Dataset**. The three kinds or classes are:

- Iris Setosa
- Iris Versicolour
- Iris Virginica

The prediction will be according to 4 features of the dataset:

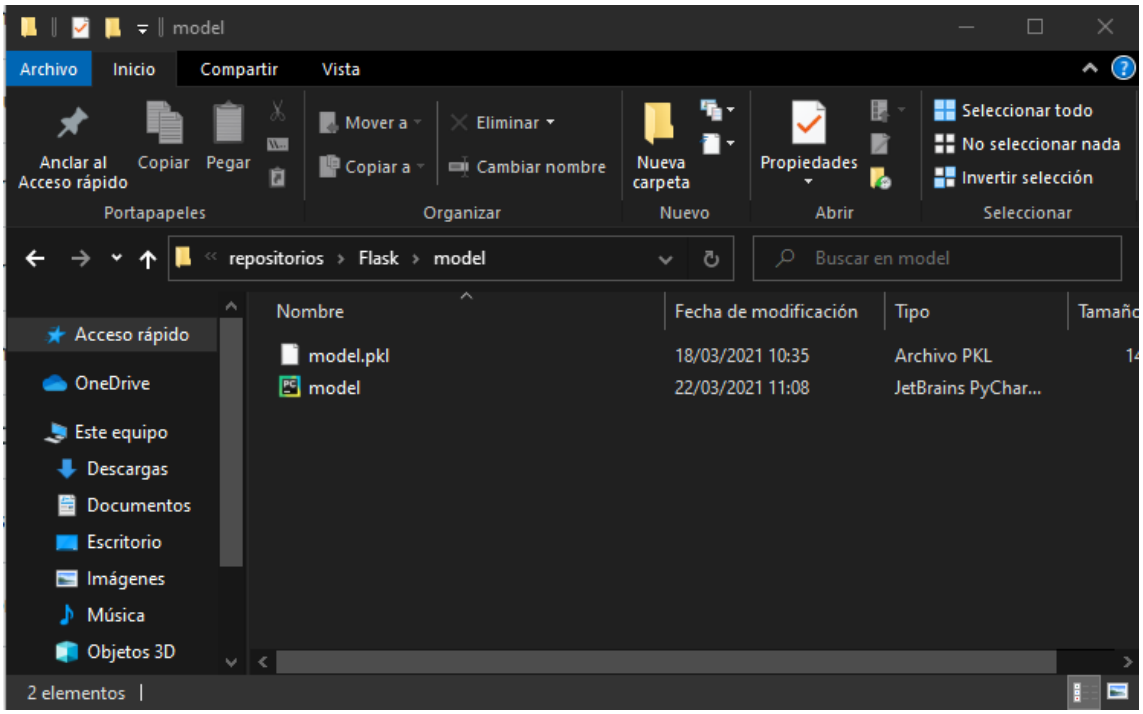
- Sepal Length and width.
- Petal Length and width.

Ones we are satisfied with our model performance we can serialize our model and save it in our project repository.

main.py | model.py | template.html | home.htm | result.html | demo2.css

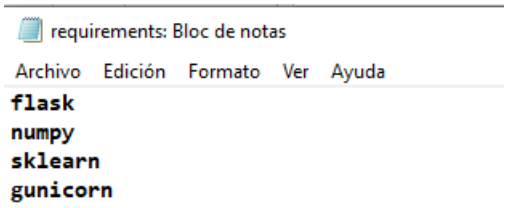
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

import pickle


X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)

clf = RandomForestClassifier()
print(clf.fit(X_train, y_train).score(X_test, y_test))

filename = 'model/model.pkl'
pickle.dump(clf, open(filename, 'wb'))
```

In our repository it should have saved in Pickle extension (.pkl)



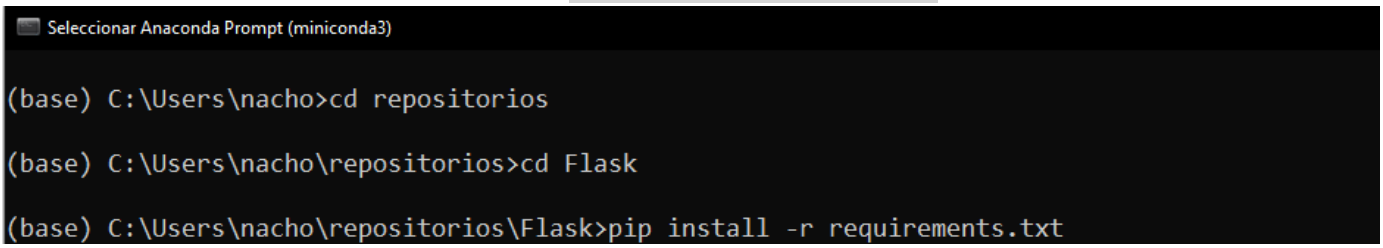
2- Installation of Flask and the libraries required

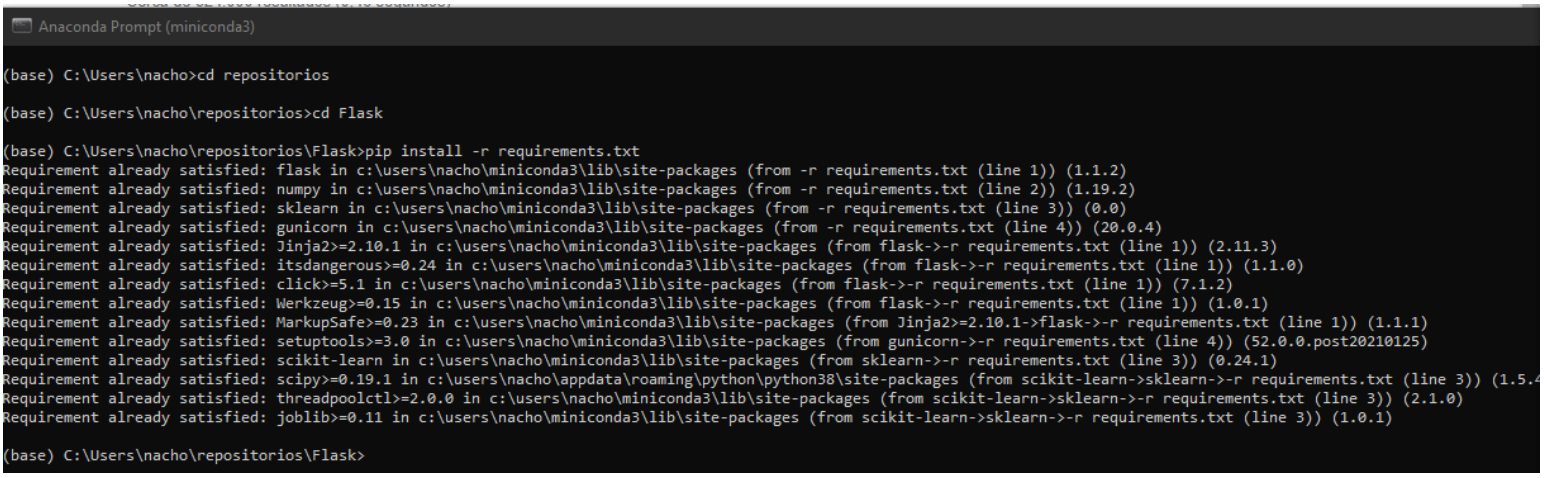
To make it faster, I created a .txt file in the repository named “requirements.txt” so we can call it in the command and install all the libraries required.



We need four libraries: Flask, NumPy, ScikitLearn and Gunicorn.

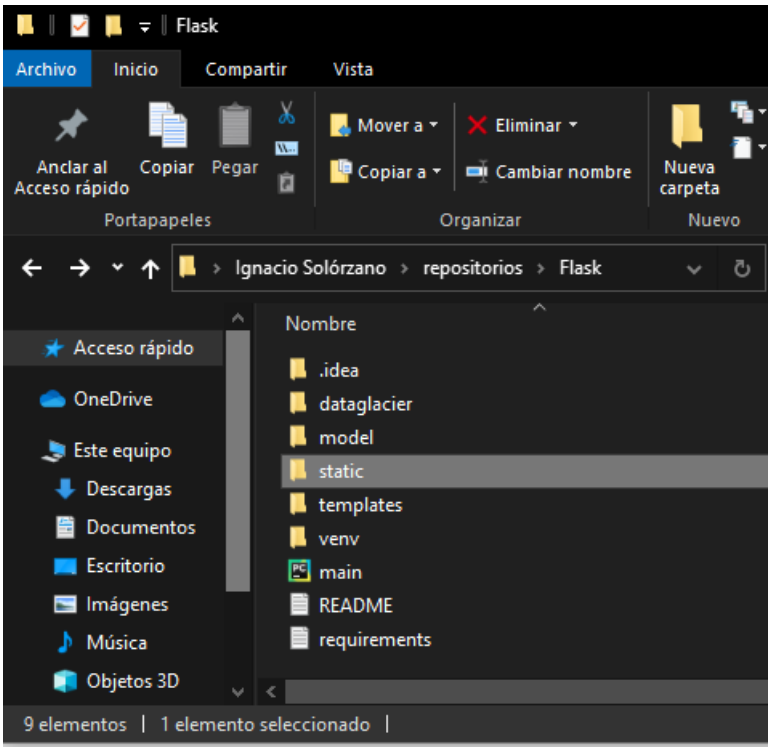
So, when we are in the command prompt, we go to our repository folder and install the libraries with the next command: “pip install -r requirements.txt”





Now, we are ready to the next step.

3- Modules of this project

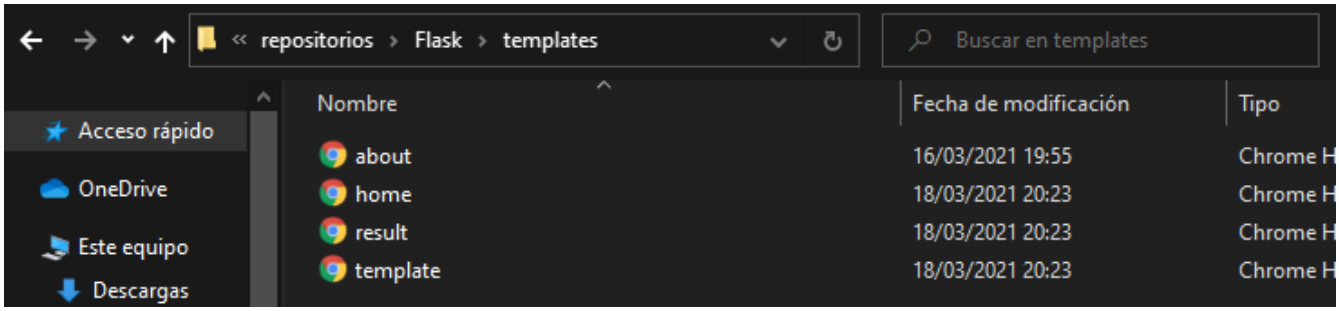


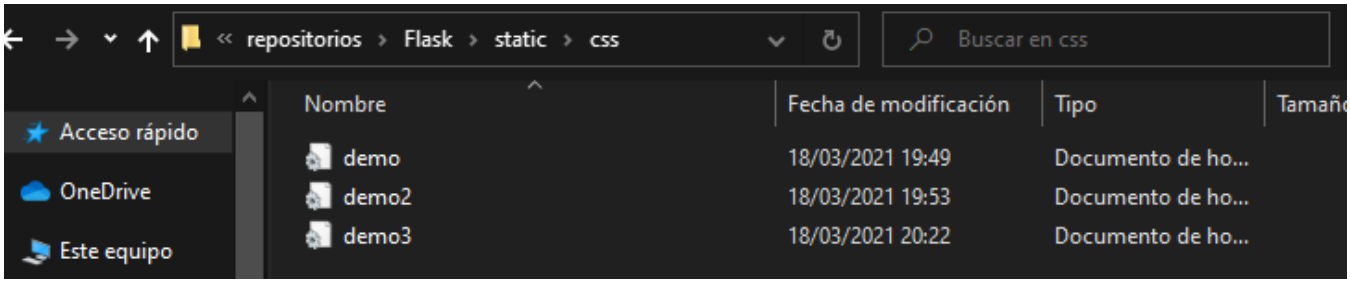
There are 4 main modules in this project. Main.py which has the flask constructor to execute each process.

The “model” folder which has both the ML model and the ML model serialization (model.pkl)

The “templates” folder which has all the .html files.

The “static” folder which has the .css files





1- Main.py

main.py | model.py | template.html | home.html | result.html

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

#Creating the instance of the class
app = Flask(__name__)

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

def ValuePredictor(to_predict_list):
    to_predict = np.array(to_predict_list).reshape(1, 4)
    loaded_model = pickle.load(open("model/model.pkl","rb"))
    result = loaded_model.predict(to_predict)
    return result[0]
@app.route('/result',methods = ['POST'])
def result():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        try:
            to_predict_list = list(map(float, to_predict_list))
            result = ValuePredictor(to_predict_list)
            if int(result)==0:
                prediction='Iris-Setosa'
            elif int(result)==1:
                prediction='Iris-Virginica'
            elif int(result)==2:
                prediction='Iris-Versicolour'
            else:
                prediction=f'{int(result)} Not-Defined'
        except ValueError:
            prediction='Data Format Error'

        return render_template("result.html", prediction=prediction)

if __name__=="__main__":

    app.run(port=5001)
```

2.1- Template.html

main.py | model.py | template.html | home.html | result.html | demo2.css

```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>ML API</title>

    <link rel="stylesheet" href="{{ url_for('static',    filename='css/demo2.css') }}">

</head>
  <body>
    <header>
      <div class="container">
        <h1 class="logo">First Web App</h1>
        <strong><nav>
          <ul class="menu">
            <li><a href="{{ url_for('home') }}">Home</a></li>

          </ul>
        </nav></strong>
      </div>
    </header>

  {% block content %}
  {% endblock %}

 </body>
</html>
```

2.2- Home.html

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>ML API</title>
    <link href="file:///C:/Users/nacho/repositorios/Flask/static/demo3.css" rel="'stylesheet" type="text/css">
   <link href="file:///C:/Users/nacho/repositorios/Flask/static/demo3.css" rel="'stylesheet" type="text/css">
   <link href="file:///C:/Users/nacho/repositorios/Flask/static/demo3.css" rel="'stylesheet" type="text/css">
   <link href="file:///C:/Users/nacho/repositorios/Flask/static/demo3.css" rel="'stylesheet" type="text/css">
   <link rel="stylesheet" href="{{ url_for('static',    filename='css/demo2.css') }}">

  </head>

<body>
  {% extends "template.html" %}
  {% block content %}
    <h3>Predictive Model of Iris Dataset</h3>

<div>
  <form action="/result" method="POST">

    <label for="sepal_length">Sepal length (in cm): </label>
    <input type="text" id="sepal_length" name="sepal_length">
    <br>
    <label for="sepal_width">Sepal width (in cm): </label>
    <input type="text" id="sepal_width" name="sepal_width">
    <br>
    <label for="petal_length">Petal length (in cm): </label>
    <input type="text" id="petal_length" name="petal_length">
    <br>
    <label for="petal_width">Petal width (in cm): </label>
    <input type="text" id="petal_width" name="petal_width">
    <br>
    <br>
    <input type="submit" value="Predict">
  </form>
</div>
  {% endblock %}
</body>
</html>
```

2.3- Result.html

```html
<!doctype html>
<html>

  <head>
   <link rel="stylesheet" href="{{ url_for('static',    filename='css/demo2.css') }}">

  </head>
   <body>
    {% extends "template.html" %}
    {% block content %}
        <h1>The Iris flower is:</h1>
        <h2> {{ prediction }}</h2>
    {% endblock %}
   </body>
</html>
```

And due to the extension of the css file, I am not presenting it in this documentation.

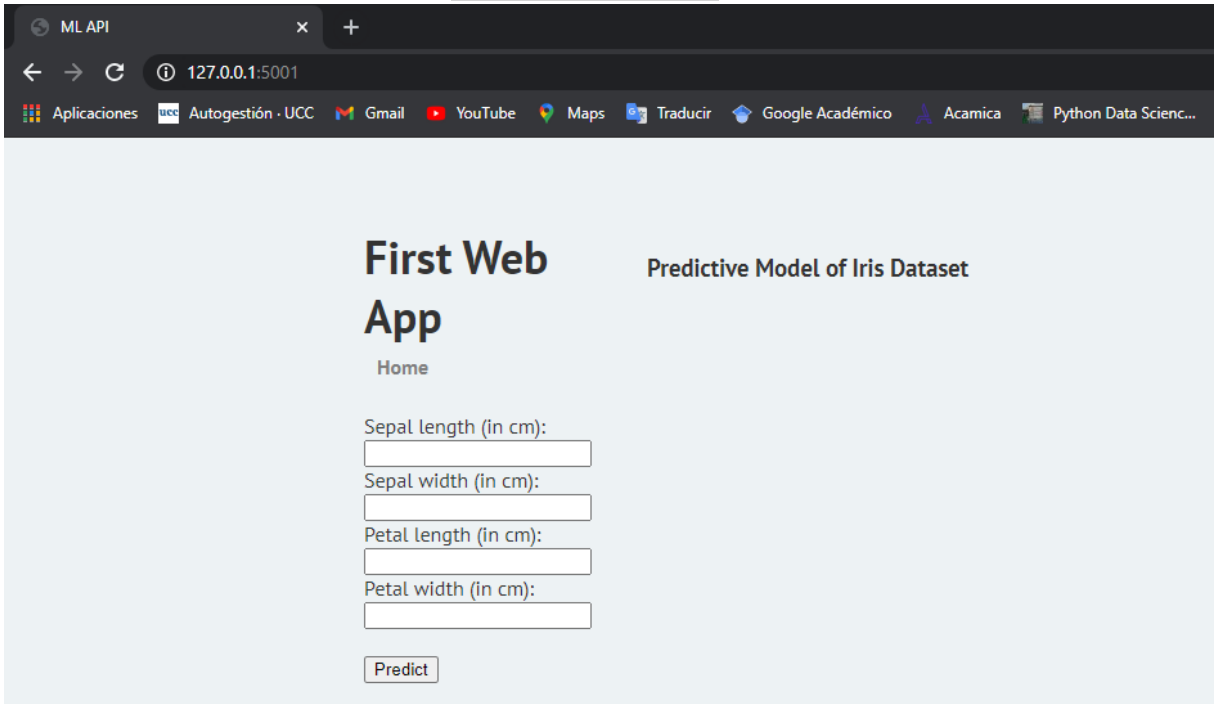
Now that we have each of the files needed, we can deploy our project locally.

4- Execution and Deployment

We must access to our repository in the command prompt and execute the file which has the Flask App.

```
(base) C:\Users\nacho\repositorios\Flask>python main.py
 * Serving Flask app "main" (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 http://127.0.0.1:5001/ (Press CTRL+C to quit)
```

Then, copy the link in the last line: http://127.0.0.1:5001/



Now, we can check for a prediction…