Model deployment on Flask

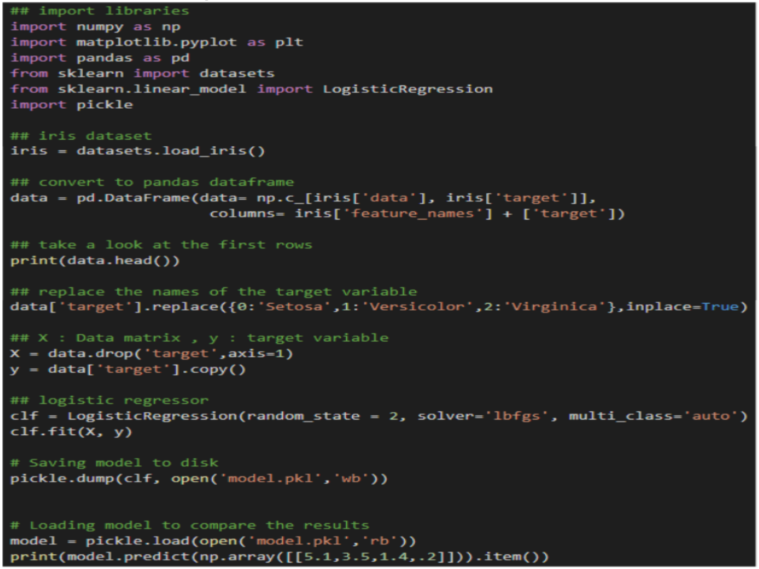
Christos Christoforou

Data Glacier Internship: Cohort Code LISUM15

November 22, 2022

Step 1: Create a model using a toy dataset. I chose the famous iris dataset and a logistic classifier to predict the species of the flower.

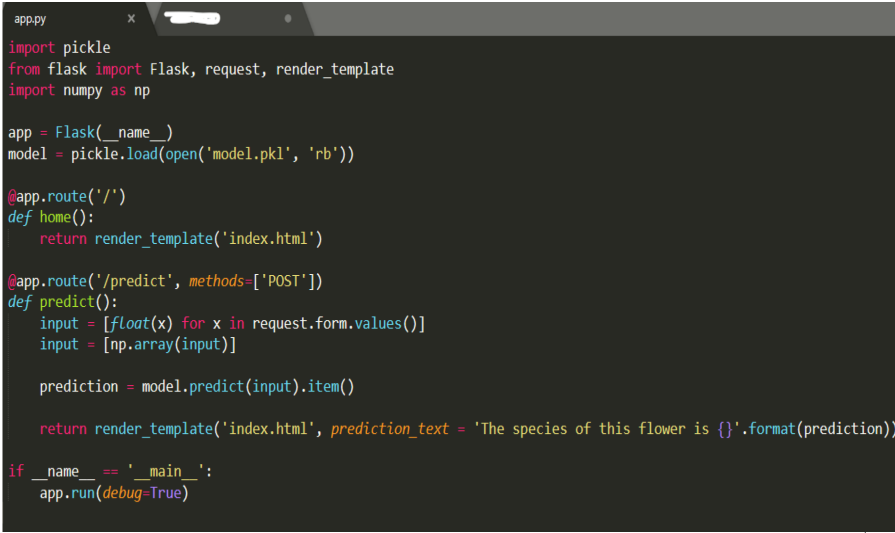
Step 1: Create a model using a toy dataset. I chose the famous iris dataset and a logistic classifier to predict the species of the flower.



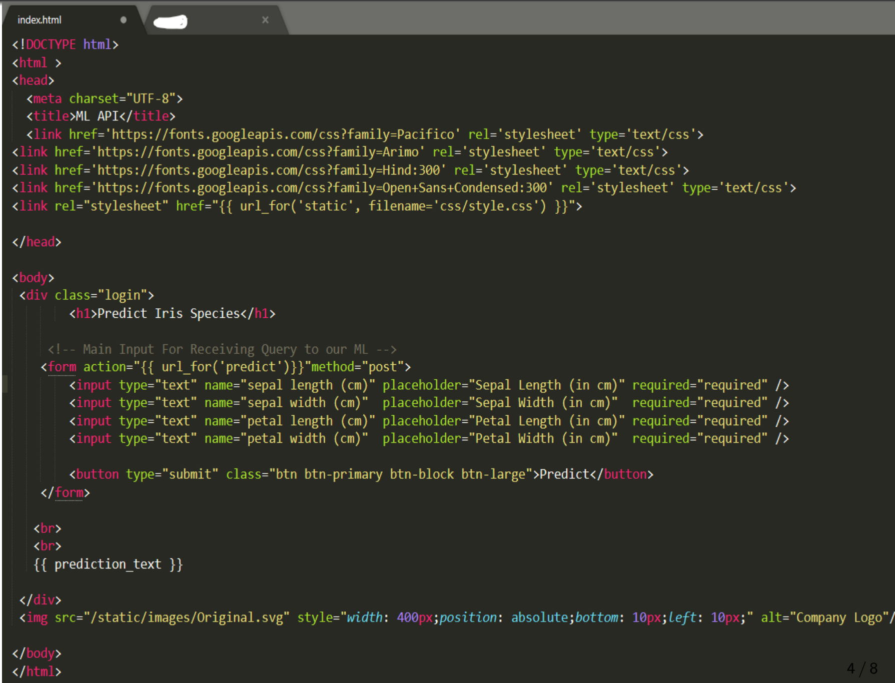
```
## import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
import pickle

## iris dataset
iris = datasets.load_iris()

## convert to pandas dataframe
data = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                     columns= iris['feature_names'] + ['target'])

## take a look at the first rows
print(data.head())

## replace the names of the target variable
data['target'].replace({0:'Setosa',1:'Versicolor',2:'Virginica'},inplace=True)

## X : Data matrix , y : target variable
X = data.drop('target',axis=1)
y = data['target'].copy()

## logistic regressor
clf = LogisticRegression(random_state = 2, solver='lbfgs', multi_class='auto')
clf.fit(X, y)

# Saving model to disk
pickle.dump(clf, open('model.pkl','wb'))


# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict(np.array([[5.1,3.5,1.4,.2]])).item())
```

Step 2: Open a text editor of your choice to create the flask app to run our model on, and the HTML code to structure a web page to display the app.

Step 2: Open a text editor of your choice to create the flask app to run our model on, and the HTML code to structure a web page to display the app.



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

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

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

@app.route('/predict', methods=['POST'])
def predict():
    input = [float(x) for x in request.form.values()]
    input = [np.array(input)]

    prediction = model.predict(input).item()

    return render_template('index.html', prediction_text = 'The species of this flower is {}'.format(prediction)

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



```html
<!DOCTYPE html>
<html >
<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'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
        <h1>Predict Iris Species</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="sepal length (cm)" placeholder="Sepal Length (in cm)" required="required" />
        <input type="text" name="sepal width (cm)"  placeholder="Sepal Width (in cm)"  required="required" />
        <input type="text" name="petal length (cm)" placeholder="Petal Length (in cm)" required="required" />
        <input type="text" name="petal width (cm)"  placeholder="Petal Width (in cm)"  required="required" />

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

   <br>
   <br>
   {{ prediction_text }}

 </div>
 <img src="/static/images/Original.svg" style="width: 400px;position: absolute;bottom: 10px;left: 10px;" alt="Company Logo"/

</body>
</html>
```

Step 3:

- Create a requirements text file.

Step 3:

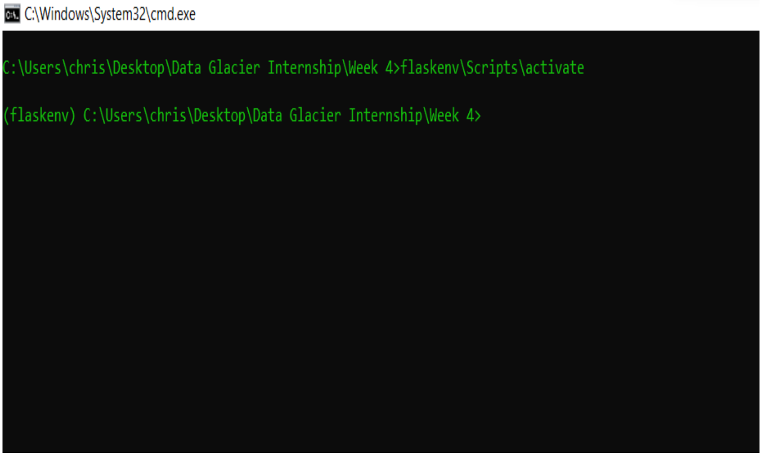
- Create a requirements text file.
- Create a virtual environment that supports the requirements you put in the text file.

Step 3:

- Create a requirements text file.
- Create a virtual environment that supports the requirements you put in the text file.
- Start your command-line interpreter (cmd) and activate the virtual environment you created.

Step 3:

- Create a requirements text file.
- Create a virtual environment that supports the requirements you put in the text file.
- Start your command-line interpreter (cmd) and activate the virtual environment you created.



C:\Windows\System32\cmd.exe

```
C:\Users\chris\Desktop\Data Glacier Internship\Week 4>flaskenv\Scripts\activate

(flaskenv) C:\Users\chris\Desktop\Data Glacier Internship\Week 4>
```

Step 4:

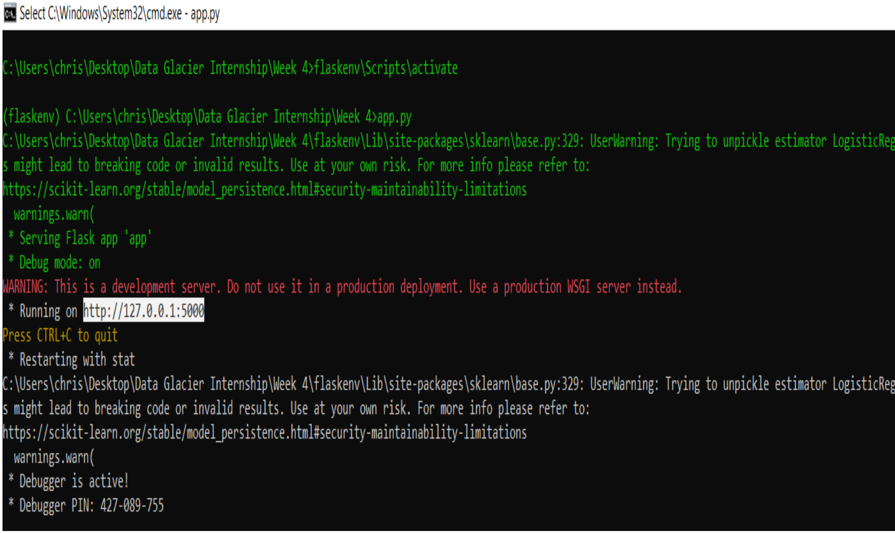
- On the command-line type the name of your flask app's file (app.py in this case) and hit enter.

Step 4:

- On the command-line type the name of your flask app's file (app.py in this case) and hit enter.
- Copy the URL (http://127.0.0.1:5000)

Step 4:

- On the command-line type the name of your flask app's file (app.py in this case) and hit enter.
- Copy the URL (http://127.0.0.1:5000)

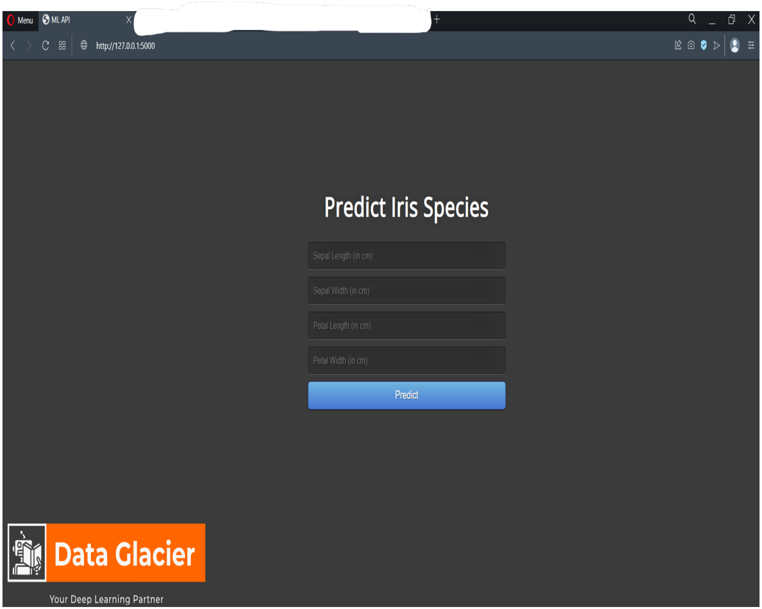


Select C:\Windows\System32\cmd.exe - app.py

```
C:\Users\chris\Desktop\Data Glacier Internship\Week 4>flaskenv\Scripts\activate

(flaskenv) C:\Users\chris\Desktop\Data Glacier Internship\Week 4>app.py
C:\Users\chris\Desktop\Data Glacier Internship\Week 4\flaskenv\Lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticReg
s might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
C:\Users\chris\Desktop\Data Glacier Internship\Week 4\flaskenv\Lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticReg
s might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 427-089-755
```

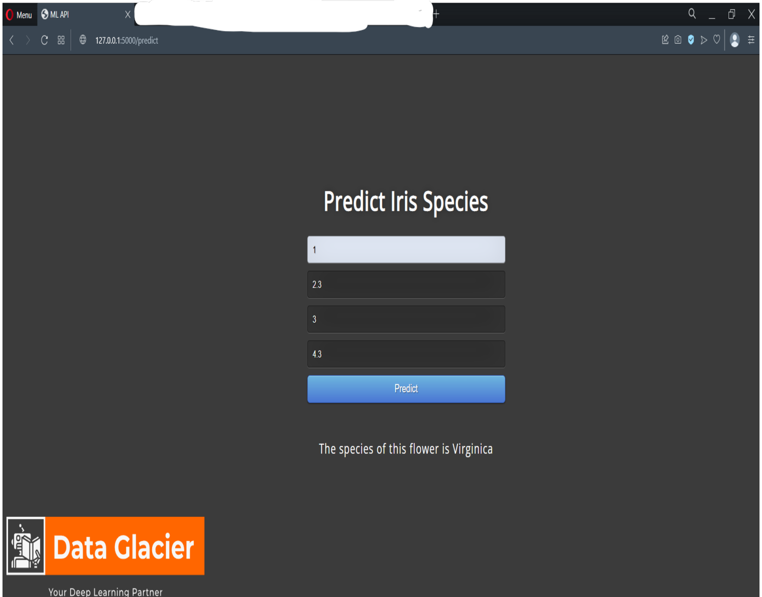
Step 5: Paste the URL on your browser to get to the web page you created.

Step 5: Paste the URL on your browser to get to the web page you created.



Step 6: Enter the Sepal and Petal lengths and widths and hit the predict button to predict the species of the flower.

Step 6: Enter the Sepal and Petal lengths and widths and hit the predict button to predict the species of the flower.



Predict Iris Species

1

2.3

3

4.3

The species of this flower is Virginica