

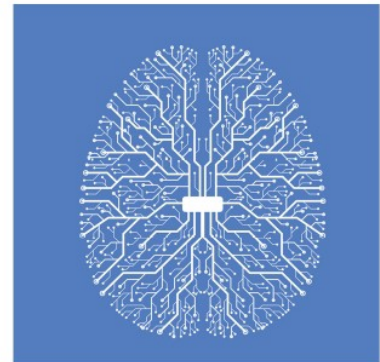
Deploy a machine learning model using flask



Hemang Vyas

Follow

Nov 30, 2018 · 5 min read



source

As a beginner in machine learning, it might be easy for anyone to get enough resources about all the algorithms for machine learning and deep learning but when I started to look for references to deploy ML model to production I did not find really any good resources which could help me to deploy my model as I am very new to this field. So,

when I succeeded to deploy my model using Flask as an API, I decided to write an article to help others to simply deploy their model. I hope it helps:)

In this article, we are going to use simple linear regression algorithm with scikit-learn for simplicity, we will use Flask as it is a very light web framework. We will create three files,

1. *model.py*

2. *server.py*

3. *request.py*

In a *model.py* file, we will develop and train our model, in a *server.py*, we will code to handle POST requests and return the results and finally in the *request.py*, we will send requests with the features to the server and receive the results.

Let's begin the coding part

1. *model.py*

As I mentioned above, in this file we will develop our ML model and train it. We will predict the salary of an employee based on his/her experience in the field. You can find the dataset [here](#).

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pickle
import requests
import json
```

Importing the libraries that we are going to use to develop our model. *numpy* and *pandas* to manipulate the matrices and data respectively, *sklearn.model_selection* for splitting data into train and test set and *sklearn.linear_model* to train our model using *LinearRegression*. *pickle* to save our trained model to the disk, *requests* to send requests to the server and *json* to print the result in our terminal.

```
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

We have imported the dataset using pandas and separated the features and label from the dataset.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.33, random_state = 0)
```

In this section, we have split our data into train and test size of 0.67 and 0.33 respectively using *train_test_split* from *sklearn*.

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
```

The object is instantiated as a *regressor* of the class *LinearRegression()* and trained using *X_train* and *y_train*. Latter the predicted results are stored in the *y_pred*.

```
pickle.dump(regressor, open('model.pkl','wb'))
```

We will save our trained model to the disk using the *pickle* library. *Pickle* is used to serializing and de-serializing a Python object structure. In which python object is converted into the byte stream. *dump()* method dumps the object into the file specified in the arguments.

In our case, we want to save our model so that it can be used by the server. So we will save our object *regressor* to the file named *model.pkl*.

We can again load the model by the following method,

```
model = pickle.load(open('model.pkl', 'rb'))  
print(model.predict([[1.8]]))
```

pickle.load() method loads the method and saves the deserialized bytes to *model*.

Predictions can be done using *model.predict()*.

For example, we can predict the salary of the employee who has experience of 1.8 years.

Here, our *model.py* is ready to train and save the model. The whole code of *model.py* is as follows.

```
# Importing the libraries  
import numpy as np  
import pandas as pd  
from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LinearRegression  
import pickle  
import requests  
import json  
  
# Importing the dataset  
dataset = pd.read_csv('Salary_Data.csv')  
X = dataset.iloc[:, :-1].values  
y = dataset.iloc[:, 1].values  
  
# Splitting the dataset into the Training set and Test set  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size  
= 1/3, random_state = 0)  
  
# Fitting Simple Linear Regression to the Training set  
regressor = LinearRegression()  
regressor.fit(X_train, y_train)  
  
# Predicting the Test set results  
y_pred = regressor.predict(X_test)  
  
# Saving model to disk  
pickle.dump(regressor, open('model.pkl', 'wb'))  
  
# Loading model to compare the results  
model = pickle.load(open('model.pkl', 'rb'))  
print(model.predict([[1.8]]))
```

2. server.py

In this file, we will use the flask web framework to handle the POST requests that we will get from the *request.py*.

Importing the methods and libraries that we are going to use in the code.

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

Here we have imported *numpy* to create the array of requested data, *pickle* to load our trained model to predict.

In the following section of the code, we have created the instance of the *Flask()* and loaded the model into the *model*.

```
app = Flask(__name__)

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

Here, we have bounded */api* with the method *predict()*. In which predict method gets the data from the json passed by the requestor. *model.predict()* method takes input from the json and converts it into 2D *numpy array* the results are stored into the variable named *output* and we return this variable after converting it into the json object using flask's *jsonify()* method.

```
@app.route('/api', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict([[np.array(data['exp'])]])
    output = prediction[0]
    return jsonify(output)
```

Finally, we will run our server by following code section. Here I have used port 5000 and have set *debug=True* since if we get any error we can debug it and solve it.

```
if __name__ == '__main__':  
    app.run(port=5000, debug=True)
```

Here, our server is ready to serve the requests. Here is the whole code of the *server.py*.

```
# Import libraries  
import numpy as np  
from flask import Flask, request, jsonify  
import pickle  
  
app = Flask(__name__)  
  
# Load the model  
model = pickle.load(open('model.pkl', 'rb'))  
  
@app.route('/api', methods=['POST'])  
def predict():  
    # Get the data from the POST request.  
    data = request.get_json(force=True)  
  
    # Make prediction using model loaded from disk as per the data.  
    prediction = model.predict([[np.array(data['exp'])]])  
  
    # Take the first value of prediction  
    output = prediction[0]  
  
    return jsonify(output)  
  
if __name__ == '__main__':  
    app.run(port=5000, debug=True)
```

3. *request.py*

As I mentioned earlier that *request.py* is going to request the server for the predictions.

Here is the whole code to make a request to the server.

```
import requests
```

```
url = 'http://localhost:5000/api'

r = requests.post(url,json={'exp':1.8,})
print(r.json())
```

We have used *requests* library to make post requests. *requests.post()* takes URL and the data to be passed in the POST request and the returned results from the servers are stored into the variable *r* and printed by *r.json()*.

Conclusion

We have created three files *model.py*, *server.py* and *request.py* to train and save a model, to handle the request, to make a request to the server respectively.

After coding all of these files, the sequence to execute the files should be *model.py*, *server.py*(in a separate terminal) and at the end *request.py*.

You can compare the results of prediction with a *model.py* as we printing the result at the end of the file.

You can find all the coding in my Github repository, flask-salary-predictor.

Don't hesitate to flow your ideas in the comment section below.

Thank you :)

[Machine Learning](#) [Flask](#) [Data Science](#) [Python](#) [Deployment](#)

[About](#) [Help](#) [Legal](#)