Integrate Flask With Scoring End Point on machine learning

IBM Data Science Experience (DSX) is an interactive, collaborative, cloud-based environment where data scientists can use multiple tools to achieve insights. Data scientists can use the best of open source, tap into IBM's unique features, grow their skills, and collaborate with teams. One of the many features of DSX provides the capability to create and train a machine learning model in DSX with little to no coding. This model can subsequently be saved and deployed to Watson Machine Learning on IBM Bluemix and called for scoring in real-time. This tutorial is a continuation of the logistic regression analysis tutorial at: https://datascience.ibm.com/docs/content/analyze-data/ml-example-log-regress.html which creates, trains, saves and deploys a logistic regression model that predicts the possibility for a tent purchase based on age, sex, marital status and job profession for an individual. In this tutorial, a very simple Python Flask web application front end will be created to call this deployed model in real time.

Table of contents

1. Test model access by using a notebook.

The first part of this notebook will confirm that the deployed model can be accessed via an external API. Please note that the credentials and endpoint used in this tutorial are only examples. You must substitute your own credentials and endpoint values.

1.1 Prerequisites.

This notebook and the corresponding tutorial sections require several IBM Bluemix services.

- An Apache Spark instance which was created with your Data Science Experience account
- An Object Storage instance which was also created with your Data Science Experience account
- An IBM Watson Machine Learning instance which will need to be created
- A Python Flask instance which will be created later as part of this tutorial

For more information about setting up your machine learning service, see Setting up your machine learning environment.

1.2 Retrieve your credentials.

To run the code samples in this notebook, you must supply information, such as the URL, username, password, and scoring endpoint variables from your environment. The details on how to do this are described in the following steps:

1. Log in to your **Bluemix** account and, from the dashboard, select the **IBM Watson Machine Learning** service that you created.
2. Click **Service credentials**.
3. Click **View credentials** and then copy the values from the **URL**, **username**, and **password** fields to the corresponding URL, username, and password variables in the code cells in this notebook.
4. To find the scoring endpoint, log in to **Data Science Experience**.
5. Click **Projects** and from the list, click your project.
6. On the **Analytics Assets** tab, in the **Models** section, click the model.
7. In the **Deployments** section, click the model deployment and then, on the **Details** tab, copy the **Scoring End Point** and paste it into the scoring_endpoint variable in this notebook.

1.3 Authenticate and import libraries.

After you update the following code cells with your specific information, you are ready to run the cells. Run the following cell to define your credentials to the Watson Machine Learning service and to import necessary libraries.

In [1]:

```
# @hidden_cell
# Copy and paste url, username and password from Watson Machine Learning
service credentials.
# Copy scoring_endpoint from Data Science Experience deployed model API
details.
url = 'enter_your_wml_instance_url'
username = 'enter_your_username'
password = 'enter_your_password'
scoring_endpoint = 'enter_your_online_scoring_end_point'
import urllib3, requests, json
```

To retrieve the token that you need to call the scoring API, run the following code:

In [2]:

```
# Retrieves token to be used to call scoring API
headers = urllib3.util.make_headers(basic_auth='{}:{}'.format(username,
password))
path = '{}/v3/identity/token'.format(url)
response = requests.get(path, headers=headers)
mltoken = json.loads(response.text).get('token')
```

To call the scoring endpoint with some initial sample data, run the following code:

In [3]:

```
# Call scoring endpoint with data payload
scoring_header = {'Content-Type': 'application/json', 'Authorization':
mltoken}
payload = {"fields":
["GENDER","AGE","MARITAL_STATUS","PROFESSION"],"values": [["M", 20,
"Single", "Student"]]}
scoring = requests.post(scoring_endpoint, json=payload,
headers=scoring_header)
print scoring.text
```

```
{
  "fields": ["GENDER", "AGE", "MARITAL_STATUS", "PROFESSION", "features", "
rawPrediction", "probability", "prediction", "nodeADP_class", "nodeADP_clas
ses"],
  "values": [["M", 20, "Single", "Student", [0.0, 1.9795602967407233, 1.567
9400843677074, 2.60755509295228], [1.4163195860557454, -1.4163195860557454]
, [0.8047607951511817, 0.19523920484881827], 0.0, 0.0, [0.0, 1.0]]]
```

}
The result predicts a probability for a tent purchase at 19.52% (your result may vary slightly) given the input sample data record of a male, 20 years old, single and a student. This confirms that the deployed machine learning model can be called by using an API.

2. Create a user interface

After you confirm access to the API by running the code cells, it's time to start work on a basic Web application. For this example, you will create a user interface by using Python Flask. The following techniques can also be applied to other web app frameworks. For this tutorial, a boilerplate Python Flask web app will be created on Bluemix and the supporting files will be downloaded. These files will be updated to integrate with your model and then pushed back up.

1. Start by deploying a Python Flask boilerplate. Sign on to your **Bluemix** account and click **Catalog** > **Apps** > **Boilerplates** > **Python Flask**.
2. In the **App name** field, enter a unique name for your application. This name identifies your application and will appear in the URL that users need to access the app.
3. Click **Create**. Please give this some time to start. The web app is running when there is a green dot next to the top title and it says **Running**. To access this web app, click on the **Visit App URL** link. The URL to access this Web application will be: http://.mybluemix.net
4. Download and unpack the command line interface and the starter code shown on this web page. Please spend some time on this page and become familiar with Cloud Foundry and Python Flask.
5. Now this boilerplate Python Flask Web application will be updated to call the deployed logistic regression model above.
6. In the folder where the unpacked starter code is located, create a file called MLTutorial.py and copy the following Python code into that file. As before, update the url, username, password, and scoring_endpoint fields to match with your deployed model. You may need to set pasting options to preserve the indentations which are critical for the following Python code. When completed, save this file. Note: For detailed information on coding in Flask, please refer to the following Web site: http://flask.pocoo.org

```
import urllib3, requests, json, os
from flask import Flask, render_template, request
from flask_bootstrap import Bootstrap
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField, RadioField, FloatField, IntegerField
from wtforms.validators import Required, Length, NumberRange
url = 'enter_your_wml_instance_url'
username = 'enter_your_username'
password = 'enter_your_password'
scoring_endpoint = 'enter_your_online_scoring_end_point'
app = Flask(__name__)
app.config['SECRET_KEY'] = 'secretpassw0rd'
bootstrap = Bootstrap(app)
class TentForm(FlaskForm):
  Sex = RadioField('Your Gender', coerce=str, choices=[('M','Male'),('F','Female')])
  Age = FloatField('Your Age', validators=[NumberRange(1,100)])
  Married = RadioField('Your Marital Status', coerce=str, choices=[('Married','Married'), \
    ('Single','Single'),('Unspecified','Unspecified')])
  Job = RadioField('Your Job', coerce=str, choices=[('Executive', 'Executive'), \
    ('Hospitality' ,'Hospitality'),('Other','Other'),('Professional','Professional'), \
    ('Retail','Retail'),('Retired','Retired'),('Sales','Sales'),('Student','Student'), \
    ('Trades','Trades')])
  submit = SubmitField('Submit')
@app.route('/', methods=['GET', 'POST'])
def index():
  form = TentForm()
```

```
  if form.validate_on_submit():
    Sex = form.Sex.data
    form.Sex.data = ''
    Age = form.Age.data
    form.Age.data = ''
    Married = form.Married.data
    form.Married.data = ''
    Job = form.Job.data
    form.Job.data = ''
    headers = urllib3.util.make_headers(basic_auth='{}:{}'.format(username, password))
    path = '{}/v3/identity/token'.format(url)
    response = requests.get(path, headers=headers)
    mltoken = json.loads(response.text).get('token')
    scoring_header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
    payload = {"fields": ["GENDER","AGE","MARITAL_STATUS","PROFESSION"], "values":
[["M",20,"Single","Student"]]}
    scoring = requests.post(scoring_endpoint, json=payload, headers=scoring_header)
    return render_template('score.html', form=form, scoring=scoring)
  return render_template('index.html', form=form)
port = os.getenv('PORT', '5000')
if __name__ == "__main__":
  app.run(host='0.0.0.0', port=int(port))
```

Setting up files

1. Edit the Procfile file and replace the contents with the following line of code.

2. web: python MLTutorial.py

   This points to the Python code to be executed that was created in the previous step. Save this file.

3. Add the following lines of code to the requirements.txt file. Flask should already be there. Save this file.

4. Flask==0.10.1
5. urllib3
6. requests
7. flask_bootstrap
8. flask_wtf
9. Two template web pages need to be added. Create a sub-folder called templates. In the templates folder, create a file called index.html and copy the following code into the file.

10. {% extends 'bootstrap/base.html' %}
11. {% import "bootstrap/wtf.html" as wtf %}
12. {% block title %}Tent Prediction{% endblock %}
13. {% block navbar %}
14. <nav class="navbar navbar-inverse" role="navigation">
15.
16. <a class="navbar-brand" href="#">Tent Predictiona>
17. div>
18. nav>
19. {% endblock %}
20. {% block content %}
21.
22. <form method="POST" action="">
23.
24.
25. <h3>To determine the probability of a tent purchase, please enter the following:h3>
26. div>

```
        div>
    <div class="row">
        <div class="col-md-8">
            <p>{{ form.Sex.label }} {{ wtf.form_field(form.Sex) }}p>
            <p>{{ form.Age.label }} {{ form.Age(size=2) }}
                {% for error in form.Age.errors %}
                  {{ error }}
            {% endfor %}p>
            <p>{{ form.Married.label }} {{ wtf.form_field(form.Married) }}p>
            <p>{{ form.Job.label }} {{ wtf.form_field(form.Job) }}p>
            <br>
            {{ form.submit() }}
            {{ form.hidden_tag() }}
        div>
    div>
div>
{% endblock %}
```

44. Save this file.

45. In the templates folder, create a file called score.html and copy the following lines of code into the file you create.

```
{% extends 'bootstrap/base.html' %}
{% import "bootstrap/wtf.html" as wtf %}
{% block title %}Tent Prediction{% endblock %}
{% block navbar %}
    <nav class="navbar navbar-inverse" role="navigation">
        <div class="container">
        <a class="navbar-brand" href="/">Tent Predictiona>
        div>
    nav>
{% endblock %}
{% block content %}
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                {{ scoring.json() }}
            div>
        div>
        <div class="row">
            <div class="col-md-12">
            <h4><a href="/">Try Again!h4>
            div>
        div>
    div>
{% endblock %}
```

70. Save this file.

Step 3: Publish

You can now publish this updated web application to Bluemix.

1. Go to the folder where the starter code is located. **Hint**: If you're in the templates folder, move up one folder.
2. Type cf login and press **Enter**. You are prompted for your email id. Type it in and press **Enter**.
3. Type your password and press **Enter**.
4. If prompted, type the number that corresponds to your organization and press **Enter**.

5. Type cf push and press **Enter**. If successful, the responses will display the **App started** message and the **OK** status as well as information about the instances and usage. Please give this some time to complete.
6. Once started, point your browser to the web application URL. The URL consists of the name that you used for the app your_app_name, inserted into the following URL: http://your_app_name.mybluemix.net
7. When the page renders, make the following selections: For gender, select **Male**; for the age, type **20**; for marital status, select **Single**; and for the profession, select **Student**.
8. Click **Submit**. If you see the boilerplate Web page, you must refresh the browser page.

The web page displays the raw JSON response from the deployed model. In the JSON are two sets of data, a list of fields and a list of corresponding values. If you find the value for **probability**, you will see that there is a 19.52% (your result may vary slightly) probability for a tent purchase given a single, 20 year old male that is a student. Note that this is the same value previously observed in the beginning part of this notebook. To observe probabilities for other variations, click **Try Again!**. It should be noted that a skilled web developer can take this raw data and present it elegantly.

4. Summary and next steps

Congratulations! You have completed this tutorial that demonstrates how a deployed model in IBM Watson Machine Learning can be called in real time for scoring. By using advanced web application development skills, the user interface can be significantly enhanced and other web app technologies leveraged.