Lab 03 - Adding Flask to our model to expose an API

In this lab we are going to add a REST API to our model.

We have seen the basics of building a model and how to pickle the model. We now need to wrap the model in an API.

This is the best way to expose a model. Having an API means that there is an abstraction layer away from the model itself which integrates with existing processes.

Step 1

If you have not used Flask before then you will need to import the library.

1. Pip install the following libraries

```
python -m pip install Flask==0.12
python -m pip install flask_restful==0.3.5
```

Once you have these both installed, we will be able to expose a REST API on top of our model code.

REST or Representational state transfer is an internet protocol for sending messages. It is used everywhere, most likely without your knowledge.

Flask is an open-source webserver.

Step 2

We will now instantiate an instance of Flask using the following syntax.

2. Add the following code to your python script. (see completed code for an example)

```
app = Flask(__name__)
api = Api(app)
```

Step 3

We will now need to a route path for flask. The is how the webserver will know which sections of code to execute.

Lab 03 - Adding Flask to our model to expose an API

Create a route for scoring.

3. Add the following code to your python script. (see completed code for an example)

```
@app.route('/score')
def apicall():
    <Your code to score>
```

Step 4

Create a route for training

4. Add the following code to your python script. (see completed code for an example)

```
@app.route('/train')
def apicalled():
```

Step 5

Finally, we need to tell flask to start working as a web server.

5. Add the following code to your python script. (see completed code for an example)

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5071, debug=True)
```

Completed code

At the end of this lab you have the following code:

```
from flask import Flask, request, jsonify
from flask_restful import Resource, Api
import _pickle as pickle
import numpy as np
from sklearn import datasets, linear_model

app = Flask(__name__)
api = Api(app)

PickleModelPath = './regression.pkl'

@app.route('/score')
def apicall():
    var1 = float(request.args.get('var1'))
    with open(PickleModelPath, 'rb') as k:
```

Lab 03 - Adding Flask to our model to expose an API

```
        PickleModel = pickle.load(k)
    Answer = PickleModel.predict(var1)
    return jsonify(Answer.tolist())


@app.route('/train')
def apicalled():
    # Load the diabetes dataset
    diabetes = datasets.load_diabetes()
    # Use only one feature
    diabetes_X = diabetes.data[:, np.newaxis, 2]
    # Split the data into training/testing sets
    diabetes_X_train = diabetes_X[:-20]
    diabetes_X_test = diabetes_X[-20:]
    # Split the targets into training/testing sets
    diabetes_y_train = diabetes.target[:-20]
    diabetes_y_test = diabetes.target[-20:]
    # Create linear regression object
    regr = linear_model.LinearRegression()
    # Train the model using the training sets
    regr.fit(diabetes_X_train, diabetes_y_train)
    # Make predictions using the testing set
    diabetes_y_pred = regr.predict(diabetes_X_test)
    with open(PickleModelPath, 'wb') as f:
            pickle.dump(regr, f)
    return "Model hase been retrained. Run /score to score model"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5071, debug=True)
```