Data Glacier Internship Week 4 Assignment
Submission Date: June 27th, 2022
By: Amir Shahcheraghian

Deploy Machine Learning on Flask

The Flask micro framework is used to develop small-scale websites using Python. Making Restful APIs with Flask is very easy.
Using Flask, I will model the Building Load prediction model, which is trained using Machine Learning Algorithms that help engineers model the prediction for building loads.

1. Libraries

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle
from sklearn import preprocessing
```

2. Load Dataset

As seen from the figure below, data consist of Electric and Gas consumption of the sample building (X values) and, finally, the total load of the building (Y value).

```
#load csv file
df = pd.read_csv('/Users/amirshahcheraghian/Desktop/ML Model Deployment Flask Jupyter/ENB2012_data.csv')

#select dependant and independant variables
X = df[['X1','X2','X3','X4','X5','X6']]
y = df[['Y1']]

df.head(10)
```

	X1	X2	X3	X4	X5	X6	Y1
0	0.98	514.5	294.0	110.25	7.0	2	15.55
1	0.98	514.5	294.0	110.25	7.0	3	15.55
2	0.98	514.5	294.0	110.25	7.0	4	15.55
3	0.98	514.5	294.0	110.25	7.0	5	15.55
4	0.90	563.5	318.5	122.50	7.0	2	20.84
5	0.90	563.5	318.5	122.50	7.0	3	21.46
6	0.90	563.5	318.5	122.50	7.0	4	20.71
7	0.90	563.5	318.5	122.50	7.0	5	19.68
8	0.86	588.0	294.0	147.00	7.0	2	19.50
9	0.86	588.0	294.0	147.00	7.0	3	19.95

3. Split the dataset into train and test

```
# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)
```

4. Feature Scaling

```
# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)
```

5. Fit the Model

```
# Fit the model
lab = preprocessing.LabelEncoder()
y_train = y_train.values.ravel()
y_transformed = lab.fit_transform(y_train)

classifier.fit(X_train, y_transformed.ravel())
```

6. Dump Pickle File

```
# Make pickle file of our model
pickle.dump(classifier, open("/Users/amirshahcheraghian/Desktop/ML Model Deployment Flask Jupyter/model.pkl", "wb
```

7. Deploy Flask

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

# Create flask app
flask_app = Flask(__name__)

#Load pickle model
model = pickle.load(open("/Users/amirshahcheraghian/Desktop/ML Model Deployment Flask Jupyter/model.pkl", "rb"))

@flask_app.route("/")
def Home():
    return render_template("/Users/amirshahcheraghian/Desktop/ML Model Deployment Flask Jupyter/template/index.ht

@flask_app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("/Users/amirshahcheraghian/Desktop/ML Model Deployment Flask Jupyter/template/index.ht

if __name__ == "__main__":
    flask_app.run(debug=True)
```

Batch Code:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle
from sklearn import preprocessing

#load csv file
df = pd.read_csv(' ENB2012_data.csv')

#select dependant and independant variables
X = df[['X1','X2','X3','X4','X5','X6']]
y = df[['Y1']]

df.head(10)

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)

#Instantiate the model
classifier = RandomForestClassifier()

# Fit the model
lab = preprocessing.LabelEncoder()
y_train = y_train.values.ravel()
y_transformed = lab.fit_transform(y_train)

classifier.fit(X_train, y_transformed.ravel())

# Make pickle file of our model
pickle.dump(classifier, open("model.pkl", "wb"))


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

```
# Create flask app
flask_app = Flask(__name__)

#Load pickle model
model = pickle.load(open("model.pkl", "rb"))

@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text = "The flower species is
{}".format(prediction))

if __name__ == "__main__":
    flask_app.run(debug=True)

myBat = open(r'Data_Glacier.bat','w+')
myBat.write('command to be included in the batch file')
myBat.close()
```