Data Glacier

Data Science Intern at Data Glacier

Week 4: Deployment on Flask

Name: Yusuf Yuhan

Batch Code: LISUM19

Date: 03/29/2023

Submitted to: Data Glacier

Introduction

a Python script that demonstrates how to build a simple machine learning model for predicting age based on the name of a person. The script uses the scikit-learn library to build a decision tree regression model and the joblib library to save and load the model to/from disk.

The script starts by defining the input features and target variable. In this case, the input feature is the name of a person and the target variable is their age. The names are encoded as numeric values using the LabelEncoder class from scikit-learn.

Next, a decision tree regression model is created using the DecisionTreeRegressor class from scikit-learn. The model is trained on the encoded names and corresponding ages using the fit() method.

The trained model is then saved to disk using the joblib.dump() method.

To demonstrate how to use the saved model to make predictions, the script loads the model from disk using the joblib.load() method.

The script then prompts the user to enter a name for which they want to predict the age. The entered name is encoded using the same LabelEncoder object as before, and a dictionary is created to map the encoded label back to the original name.

If the entered name is found in the dataset, the model is used to predict the age for that name and the predicted age is displayed. If the entered name is not found in the dataset, a message indicating that the name was not found is displayed.

Data Information

contains a small dataset of names and corresponding ages that is used to train and test the machine learning model. The dataset consists of four names ('John', 'Jane', 'Jack', 'Jill') and their corresponding ages (25, 30, 22, 28).

Building Model

```python
from flask import Flask, request, render_template
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import LabelEncoder
import joblib

# Define the input features and target variable
X = [['John'], ['Jane'], ['Jack'], ['Jill']]
y = [25, 30, 22, 28]

# Encode the names as numeric values
le = LabelEncoder()
X_encoded = le.fit_transform([name[0] for name in X])

# Create the model
model = DecisionTreeRegressor()

# Fit the model to the data
model.fit(X_encoded.reshape(-1, 1), y)

# Save the model to disk
joblib.dump(model, 'age_prediction_model.joblib')

# Load the model from disk
model = joblib.load('age_prediction_model.joblib')

# Define a dictionary to map the numeric values back to the original names
label_to_name_dict = {label: name for name, label in zip([name[0] for name in X], X_encoded)}

# Define the Flask application
app = Flask(__name__)

# Define the route to handle user input
@app.route('/', methods=['GET', 'POST'])
```

```
# Define the input features and target variable
X = [['John'], ['Jane'], ['Jack'], ['Jill']]
y = [25, 30, 22, 28]

# Encode the na
le = LabelEncod
X_encoded = le.fit_transform([name[0] for name in X])

# Create the model
model = DecisionTreeRegressor()

# Fit the model to the data
model.fit(X_encoded.reshape(-1, 1), y)

# Save the model to disk
joblib.dump(model, 'age_prediction_model.joblib')

# Load the model from disk
model = joblib.load('age_prediction_model.joblib')

# Define a dictionary to map the numeric values back to the original names
label_to_name_dict = {label: name for name, label in zip([name[0] for name in X], X_encoded)}

# Define the Flask application
app = Flask(__name__)

# Define the route to handle user input
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        # Get user input for name
        name = request.form['name']
```

(function) fit_transform: Any

See Real World Examples From GitHub

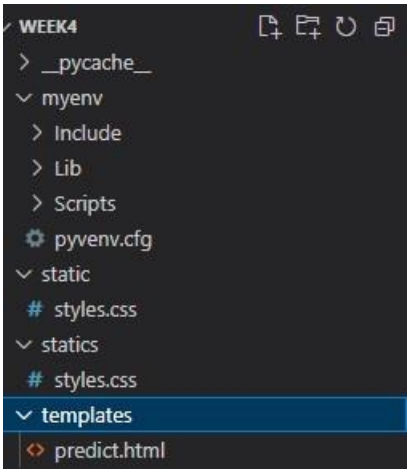
Save and Load the model

```
# Fit the model to the data
model.fit(X_encoded.reshape(-1, 1), y)

# Save the model to disk
joblib.dump(model, 'age_prediction_model.joblib')

# Load the model from disk
model = joblib.load('age_prediction_model.joblib')

# Define a dictionary to map the numeric values back to the original names
label_to_name_dict = {label: name for name, label in zip([name[0] for name in X], X_encoded)}
```

Deploy to Flask

```python
# Define a dictionary to map the numeric values back to the original names
label_to_name_dict = {label: name for name, label in zip([name[0] for name in X], X_encoded)}

# Define the Flask application
app = Flask(__name__)

# Define the route to handle user input
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        # Get user input for name
        name = request.form['name']

        # Get the numeric label for the input name
        label = le.transform([name])[0]

        if label not in X_encoded:
            error = 'Name not found in dataset.'
            return render_template('predict.html', error=error)
        else:
            # Use the model to predict the age
            age = model.predict([[label]])[0]
            name = label_to_name_dict[label]
            return render_template('predict.html', name=name, age=age)
    else:
        return render_template('predict.html')

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

Turn the Model into web application



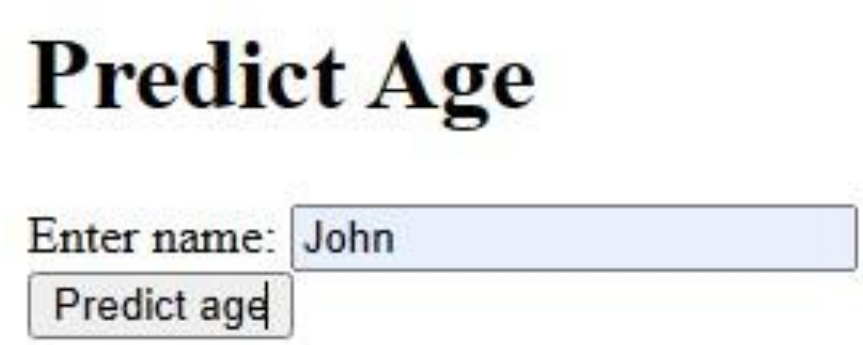
Index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Age Prediction</title>
</head>
<body>
    <h1>Age Prediction</h1>
    {% if error %}
        <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="name">Name:</label>
        <input type="text" id="name" name="nar
        <input type="submit" value="Predict">
    </form>
    {% if age %}
        <p>Age for {{ name }} is {{ age }}.</p
    {% endif %}

    <script>
        $(document).ready(function() {
  $('form').submit(function(event) {
    // Prevent form from submitting normally
    event.preventDefault();

    // Get the input data from the form
    var name = $('#name').val();

    // Send a POST request to the Flask server
    $.ajax({
      url: '/predict',
      type: 'POST',
      data: { name: name }
```

```
        <p>Age for {{ name }} is {{ age }}.</p>
    {% endif %}

    <script>
        $(document).ready(function() {
  $('form').submit(function(event) {
    // Prevent form from submitting normally
    event.preventDefault();

    // Get the input data from the form
    var name = $('#name').val();

    // Send a POST request to the Flask server
    $.ajax({
      url: '/predict',
      type: 'POST',
      data: { name: name },
      success: function(data) {
        // Update the HTML with the predicted age
        $('#age').text('Age for ' + name + ' is ' + data.age + '.');
      },
      error: function(xhr, status, error) {
        // Handle errors
        console.error(error);
      }
    });
  });
});
    </script>
</body>
</html>
```

Running The flask server

```
$ flask run
'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
 * Serving Flask app 'server.py'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a pr
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
 * Debugger is active!
 * Debugger PIN: 532-891-569
```



OUTput


Predict Age
Enter name: John
Predict age
Predicted age for John: 25 years.