

Iris Model Deployment

Name: Bilgan Kiris

Batch Code: LISUM38

Submission Date: October 30th, 2024

Submitted To: Data Glacier

Project Overview

This project focuses on building and deploying a Flask model for a toy dataset. Key steps include data loading, exploratory data analysis (EDA), model training, saving, and web deployment. The chosen dataset for this project is Iris dataset and model is build to determine which specie of the flower given features.

1. Loading the Dataset

The Iris dataset from scikit-learn was downloaded for data determination and model training.

```
[22]: # importing libraries
      from sklearn.datasets import load_iris
      from sklearn.model_selection import train_test_split
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.metrics import accuracy_score
      import pickle
```

```
[23]: # 1. Loading the dataset
      data = load_iris()
      X = data.data # Feature variables
      y = data.target # Target variable
```

2. Exploratory Data Analysis (EDA)

Basic statistical analysis were performed to understand the data structure.

```
[24]: # 2. Exploratory Data Analysis (EDA)
# For simplicity, we'll print dataset shape and a few sample records
print("Dataset Shape:", X.shape)
print("Sample Records:", X[:5])

Dataset Shape: (150, 4)
Sample Records: [[5.1 3.5 1.4 0.2]
 [4.9 3. 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5. 3.6 1.4 0.2]]
```

3. Model Building

A machine learning model was developed to classify iris species based on petal and sepal measurements. The accuracy of the model was also tested to verify the accuracy.

```
[25]: # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")

Model Accuracy: 1.00
```

4. Saving The Model

The trained model was saved using pickle library.

```
[26]: with open("iris_model.pkl", "wb") as file:
        pickle.dump(model, file)
        print("Model saved as iris_model.pkl")

Model saved as iris_model.pkl
```

5. Deploying the Model Using Flask

The Flask application was created along with an HTML interface to input features and display model predictions.

The app.py code:

```

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

app = Flask(__name__)

# Load the model
with open("iris_model.pkl", "rb") as file:
    model = pickle.load(file)

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():
    try:
        # Get feature inputs from form
        features = [float(request.form[feature]) for feature in ["sepal_length", "sepal_width", "petal_length", "petal_width"]]

        # Convert to numpy array and reshape for prediction
        input_features = np.array(features).reshape(1, -1)

        # Make prediction
        prediction = model.predict(input_features)[0]

        # Map prediction to species name
        species_map = {0: "Setosa", 1: "Versicolor", 2: "Virginica"}
        prediction_text = f"The predicted iris species is {species_map[prediction]}."

        return render_template("index.html", prediction_text=prediction_text)
    except Exception as e:
        return render_template("index.html", prediction_text=f"Error in processing prediction: {e}")

if __name__ == "__main__":
    app.run(debug=True, port=5002)

```

The HTML interface:

Iris Species Prediction

Sepal Length:

Sepal Width:

Petal Length:

Petal Width:

PREDICT

6. Testing

Iris Species Prediction

Sepal Length:

Sepal Width:

Petal Length:

Petal Width:

PREDICT

Prediction Result:

The predicted iris species is Setosa.

Iris Species Prediction

Sepal Length:

Sepal Width:

Petal Length:

Petal Width:

PREDICT

Prediction Result:

The predicted iris species is Versicolor.