Name: Naga Pavithra Jajala

Batch Code: LISUM44: 30 March (2025) – 30 June (2025)

Submission Date: April 28, 2025

Submitted To: Data Glacier Team

1. Model Training Section

localhost:8798/notebooks/dataglacier-task/VC/Week4/train_model.ipynb?

jupyter train_model Last Checkpoint: 5 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

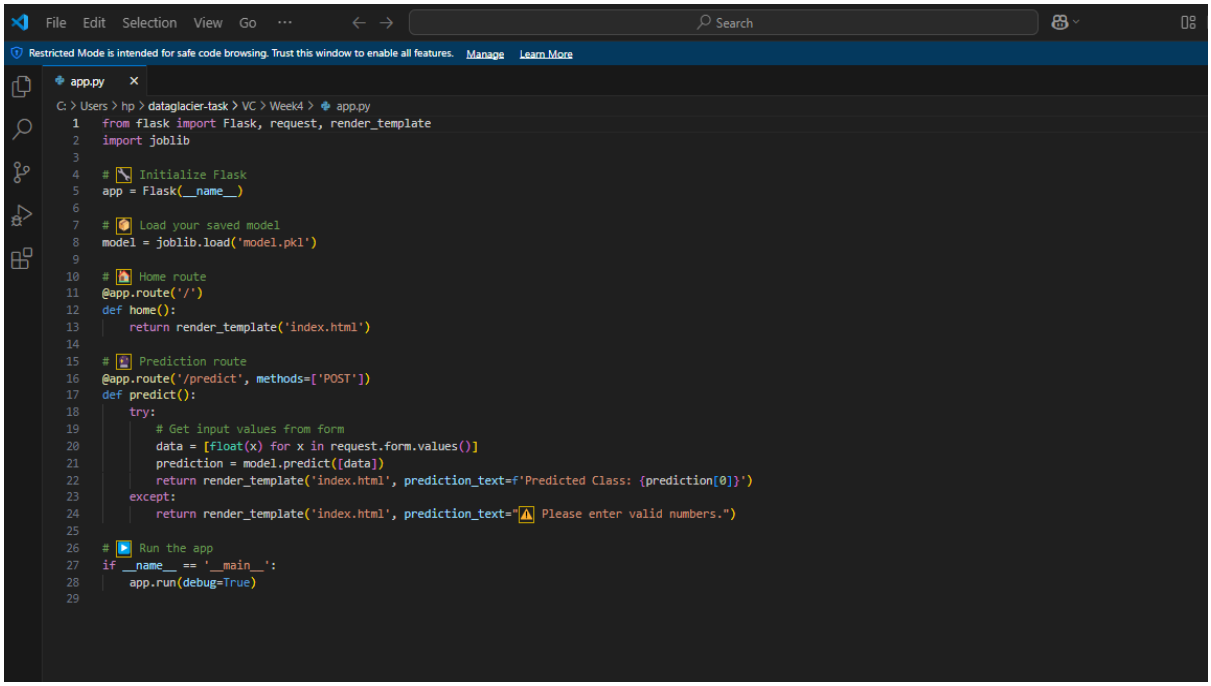
[1]:
```python
# 🎯 Import all required libraries
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import joblib  # ✅ Make sure joblib is imported

# 📊 Load and prepare data
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target

X = df[iris.feature_names]
y = df['target']

# ✂️ Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 🧠 Train the model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# 📈 Check accuracy
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)

# 💾 Save the trained model
joblib.dump(model, 'model.pkl')
print("Model saved successfully as model.pkl")
```

```
Model Accuracy: 1.0
Model saved successfully as model.pkl
```

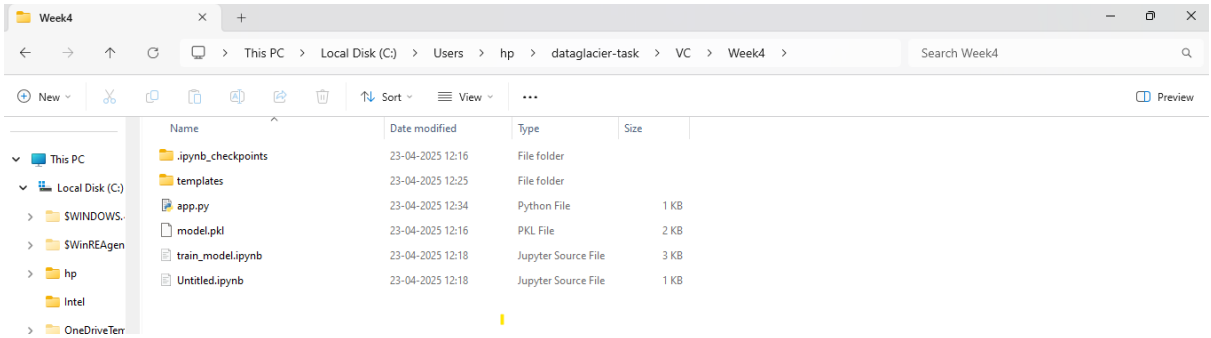
- Trained a Logistic Regression model on the Iris dataset.
- Achieved an accuracy of 100% on training data.
- Successfully saved the trained model as model.pkl using joblib.

2. Flask App Setup

Developed a Flask application (app.py) to load the trained machine learning model, collect user input via a form, and predict the Iris flower class.



```python
from flask import Flask, request, render_template
import joblib

# Initialize Flask
app = Flask(__name__)

# Load your saved model
model = joblib.load('model.pkl')

# Home route
@app.route('/')
def home():
    return render_template('index.html')

# Prediction route
@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get input values from form
        data = [float(x) for x in request.form.values()]
        prediction = model.predict([data])
        return render_template('index.html', prediction_text=f'Predicted Class: {prediction[0]}')
    except:
        return render_template('index.html', prediction_text="⚠ Please enter valid numbers.")

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



Organized the Flask project structure with:

- templates/ folder for the frontend HTML (index.html)
- model.pkl containing the saved trained model
- app.py for backend Flask server code.

3. Testing the Application

{Testing the Deployed Flask Application }

After setting up the Flask application and loading the trained Iris classification model, the application was run locally on the machine. The application was tested by providing sample inputs through the web form, and the output prediction was successfully received.

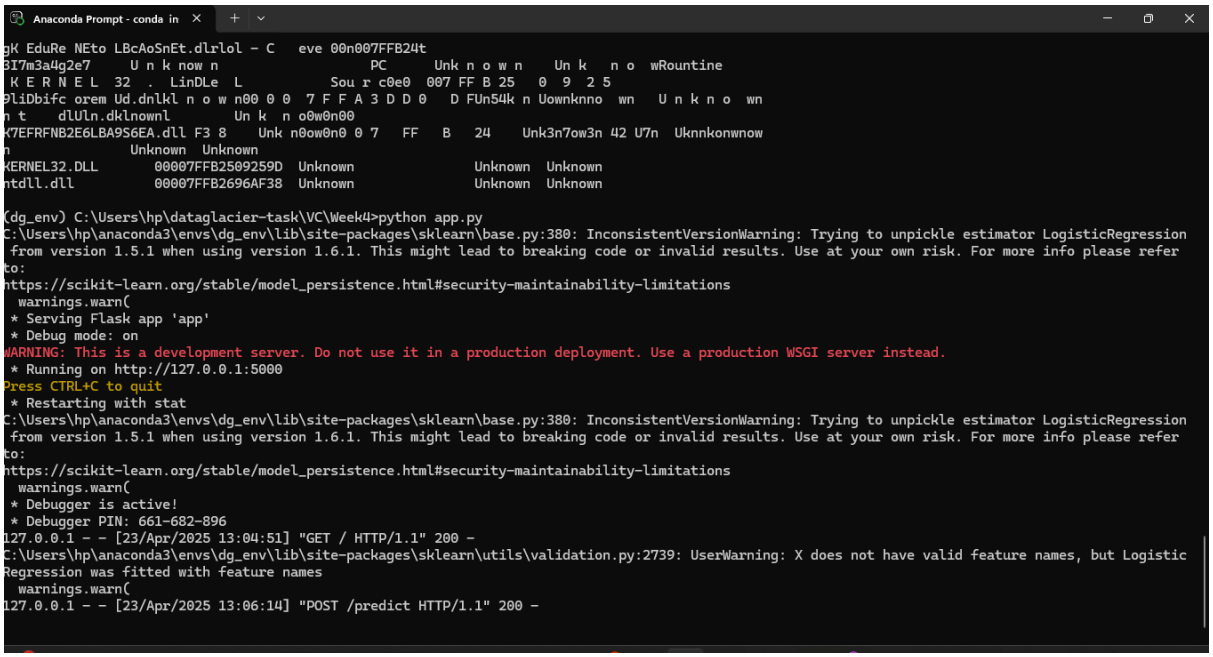


FIGURE1: **Flask application running successfully on localhost (http://127.0.0.1:5000) using Anaconda Prompt.**

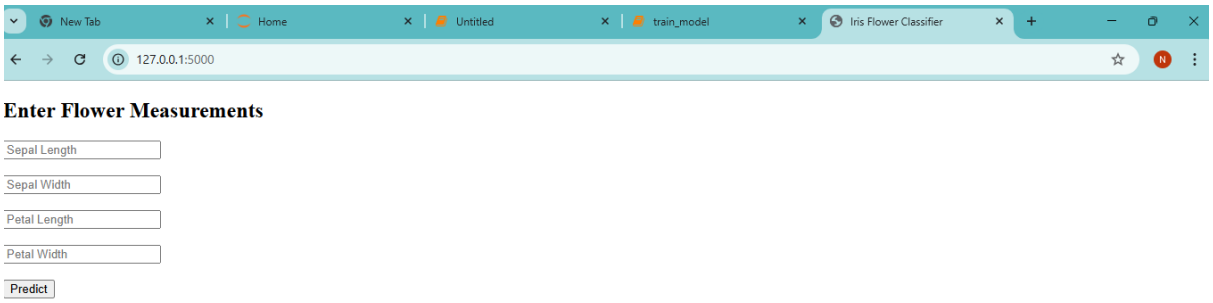


FIGURE 2: **Web interface of the Flask app displaying a form to input flower measurements (sepal and petal dimensions).**

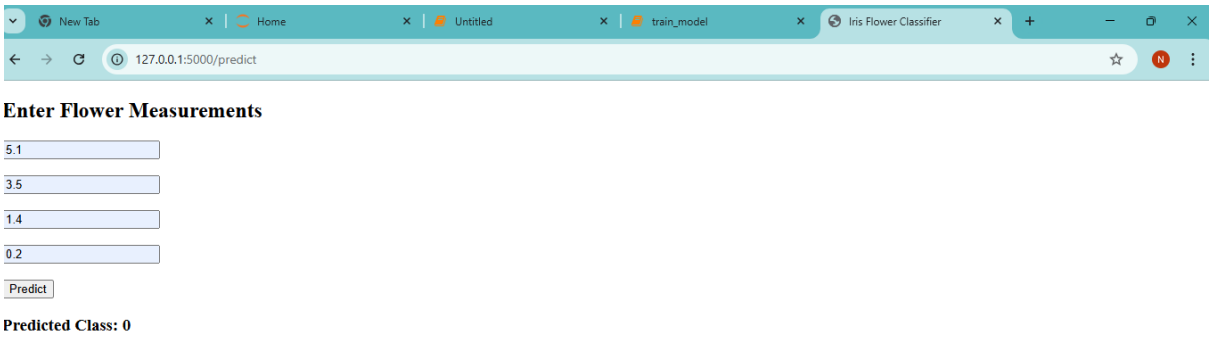


FIGURE 3**: Prediction result generated by the deployed model based on the provided input features.**

4. Conclusion

In this task, I successfully completed the end-to-end deployment of a machine learning model using Flask.
I trained a Logistic Regression model on the Iris dataset, achieving high accuracy.

After training, the model was saved and integrated into a Flask web application.
The application allowed users to input flower measurements and receive the predicted Iris class through a simple web interface.

This exercise helped me understand the real-world process of saving models, building backend Flask servers, connecting front-end forms, and testing locally.
I also gained experience in project structure management and deploying machine learning models into web applications.