

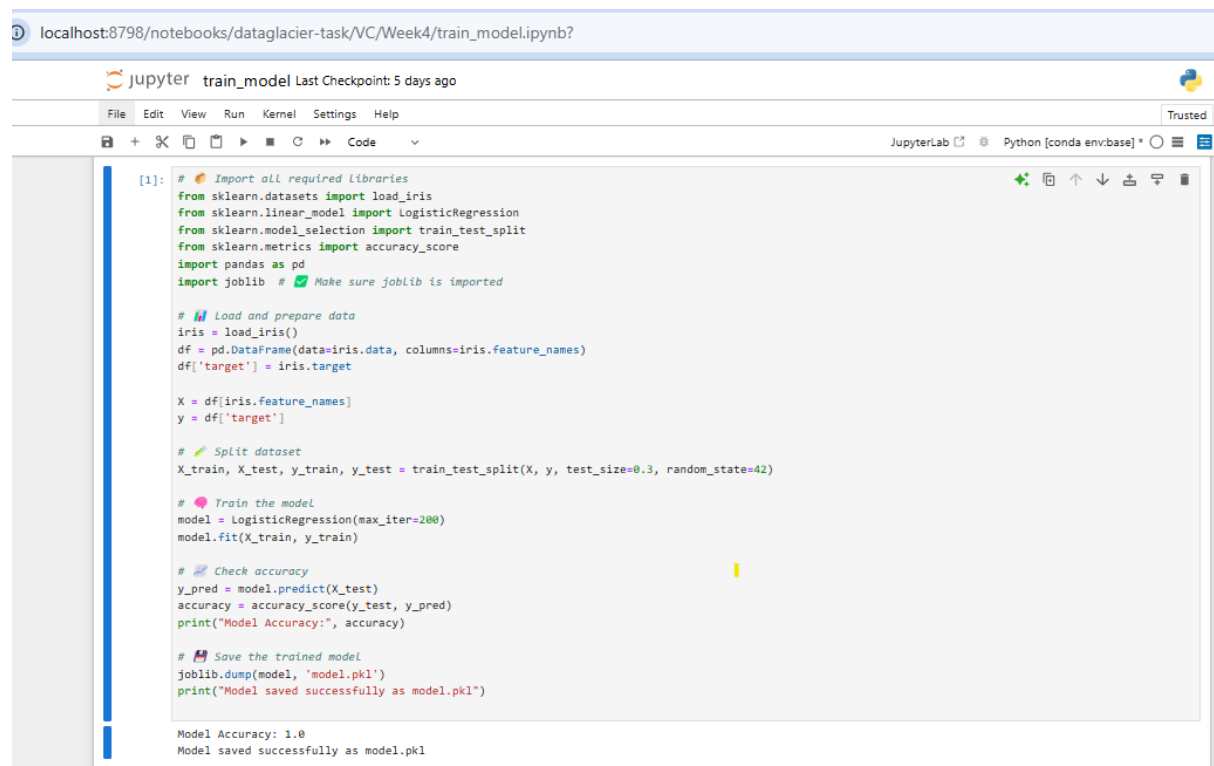
Name: Naga Pavithra Jajala

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

JupyterLab Python [conda env:base]

```
[1]: # 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)

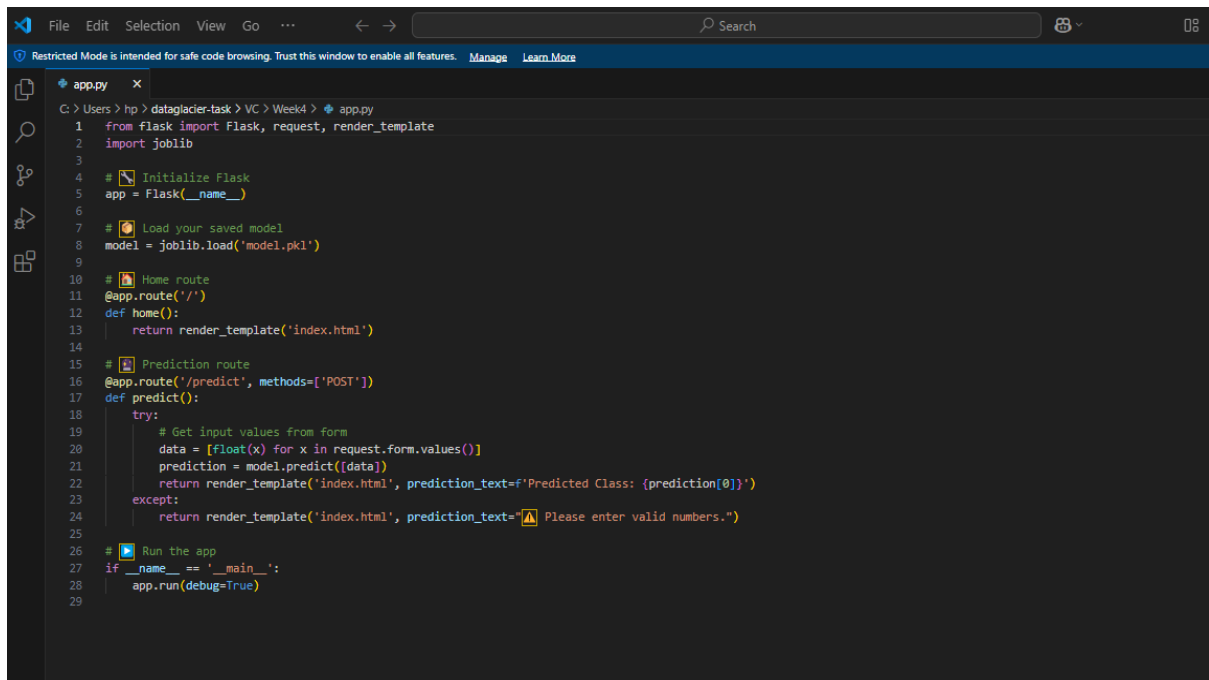
# Save the trained model
joblib.dump(model, 'model.pkl')
print("Model saved successfully as model.pkl")

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

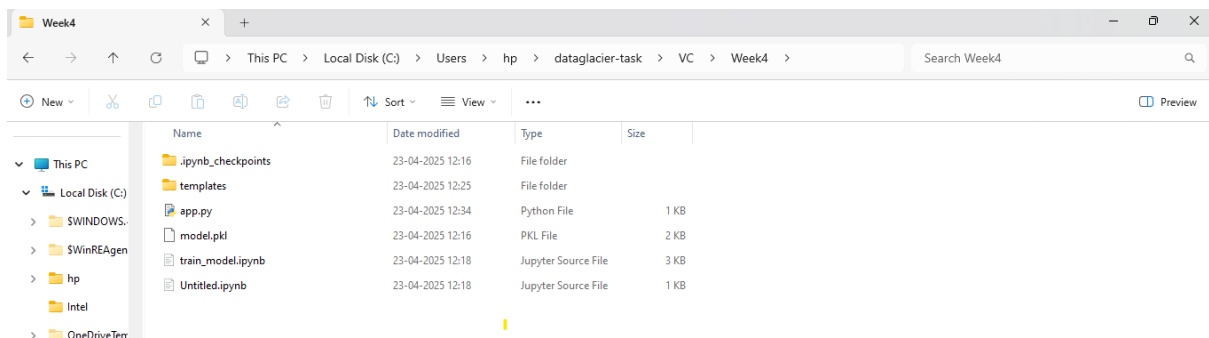
- Trained a Logistic Regression model on the Iris dataset.
- Achieved an accuracy of 100% on training data.
- Successfully saved the trained model as model.pkl using joblib.

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.



```
1 from flask import Flask, request, render_template
2 import joblib
3
4 # Initialize Flask
5 app = Flask(__name__)
6
7 # Load your saved model
8 model = joblib.load('model.pkl')
9
10 # Home route
11 @app.route('/')
12 def home():
13     return render_template('index.html')
14
15 # Prediction route
16 @app.route('/predict', methods=['POST'])
17 def predict():
18     try:
19         # Get input values from form
20         data = [float(x) for x in request.form.values()]
21         prediction = model.predict([data])
22         return render_template('index.html', prediction_text=f'Predicted Class: {prediction[0]}')
23     except:
24         return render_template('index.html', prediction_text="⚠ Please enter valid numbers.")
25
26 # Run the app
27 if __name__ == '__main__':
28     app.run(debug=True)
29
```



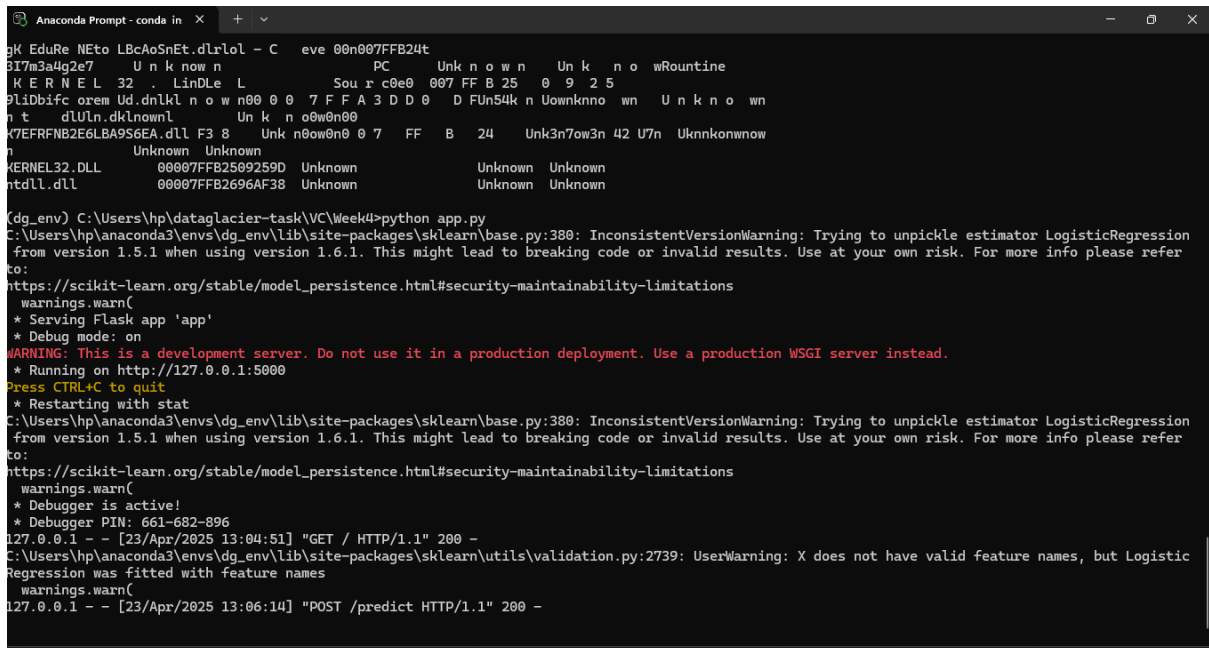
Organized the Flask project structure with:

- 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 }

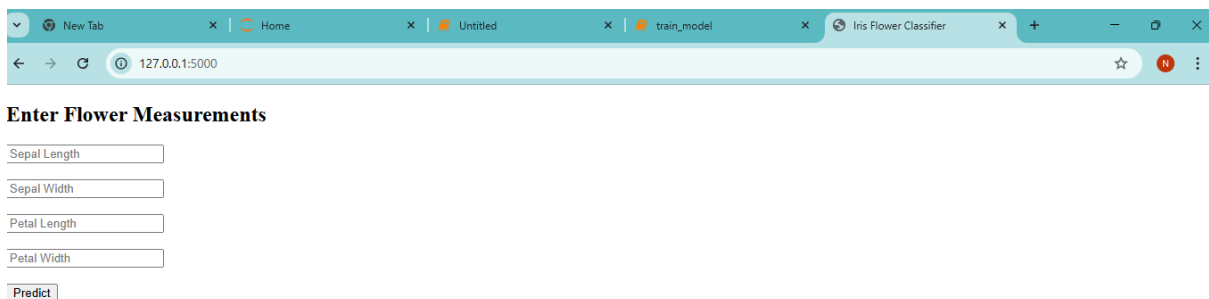
After setting up the Flask application and loading the trained Iris classification model, the application was run locally on the machine. The application was tested by providing sample inputs through the web form, and the output prediction was successfully received.



```
gk EduRe NEto LBcAoSnEt.dlrlol - C eve 00n007FFB24t
3I7m3a4g2e7 U n k n o w n PC Unk n o w n Un k n o w Rountine
K E R N E L 32 . LinDLe L Sou r c e 0 0 7 F F B 2 5 0 9 2 5
9lIdBifc orem Ud.dnkl n o w n 0 0 7 F F A 3 D D 0 D FUn54k n Uownknno wn Un k n o w n
n t dLUln.dklnownl Un k n o w n 0 0
Y7EFRFB2E6LBA9S6EA.dll F3 8 Unk n o w n 0 0 7 F F B 2 4 Unk3n7ow3n 42 U7n Uknknknwnow
n Unknown Unknown
KERNEL32.DLL 00007FFB2509259D Unknown Unknown Unknown
ntdll.dll 00007FFB2696AF38 Unknown Unknown Unknown

(dg_env) C:\Users\hp\dataglacier-task\VC\Week4>python app.py
C:\Users\hp\anaconda3\envs\dg_env\lib\site-packages\sklearn\base.py:380: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression
from version 1.5.1 when using version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer
to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
C:\Users\hp\anaconda3\envs\dg_env\lib\site-packages\sklearn\base.py:380: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression
from version 1.5.1 when using version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer
to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
* Debugger is active!
* Debugger PIN: 661-682-896
127.0.0.1 - - [23/Apr/2025 13:04:51] "GET / HTTP/1.1" 200 -
C:\Users\hp\anaconda3\envs\dg_env\lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but Logistic
Regression was fitted with feature names
warnings.warn(
127.0.0.1 - - [23/Apr/2025 13:06:14] "POST /predict HTTP/1.1" 200 -
```

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



New Tab x Home x Untitled x train_model x Iris Flower Classifier x + -

127.0.0.1:5000

Enter Flower Measurements

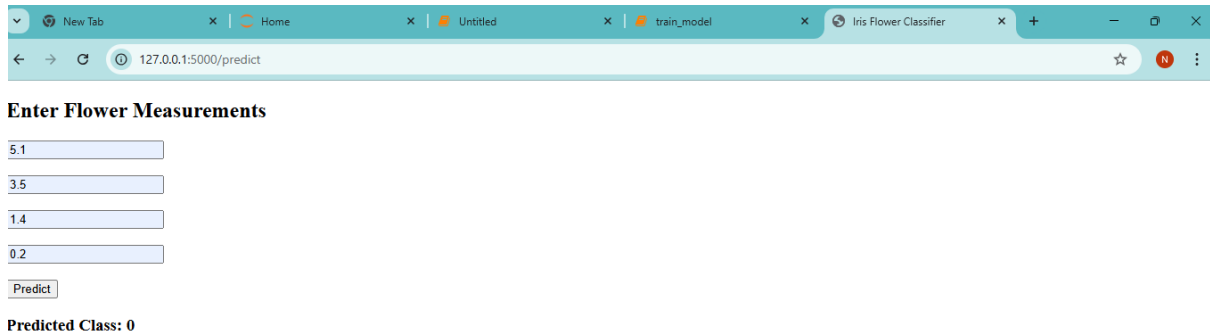
Sepal Length

Sepal Width

Petal Length

Petal Width

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



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/predict'. The page title is 'Iris Flower Classifier'. The main content area is titled 'Enter Flower Measurements' and contains four input fields with the following values: 5.1, 3.5, 1.4, and 0.2. Below these fields is a 'Predict' button. Underneath the button, the text 'Predicted Class: 0' is displayed.

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.