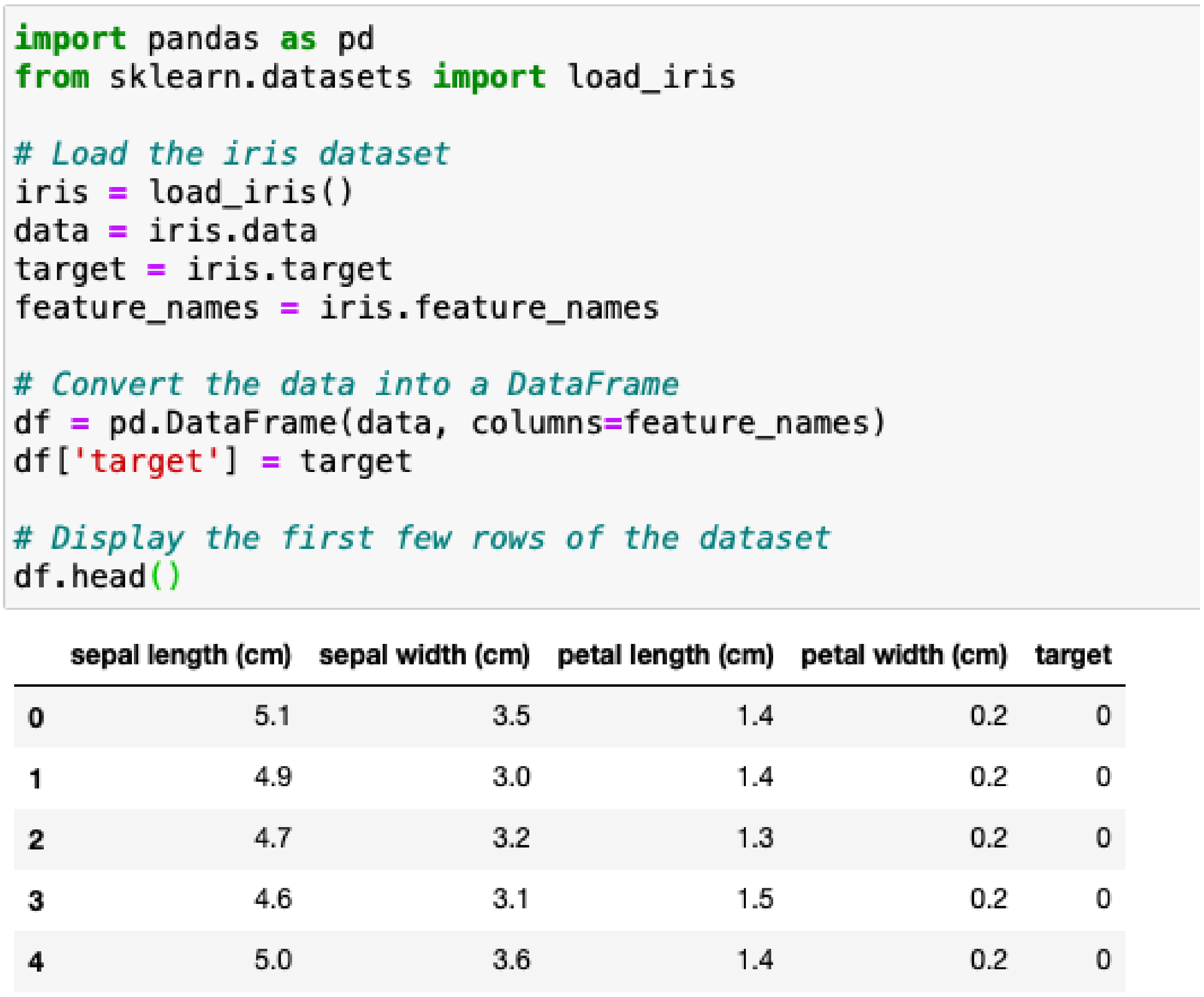
Week 4 - Report

Name: <Data Analyst Intern>
Internship Batch:<LISUM23:30 >
Data intake by:<Dingyun Hu>

Step 1: Load the data. I chose the iris dataset for this week’s task.

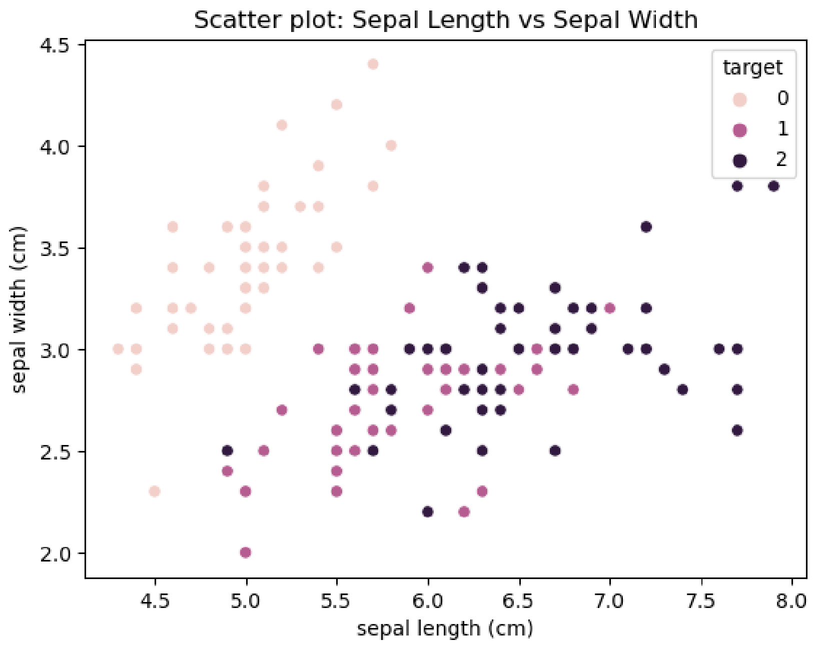


```python
import pandas as pd
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()
data = iris.data
target = iris.target
feature_names = iris.feature_names

# Convert the data into a DataFrame
df = pd.DataFrame(data, columns=feature_names)
df['target'] = target

# Display the first few rows of the dataset
df.head()
```

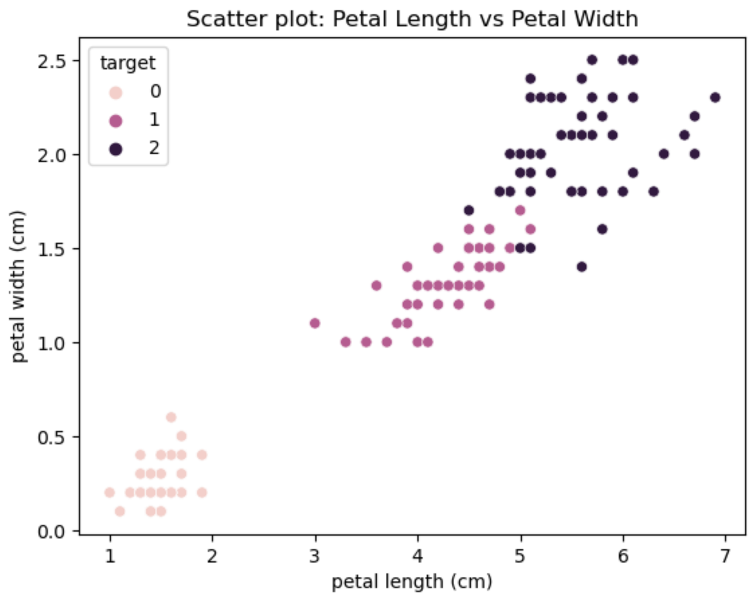
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Step 2: Simple EDA

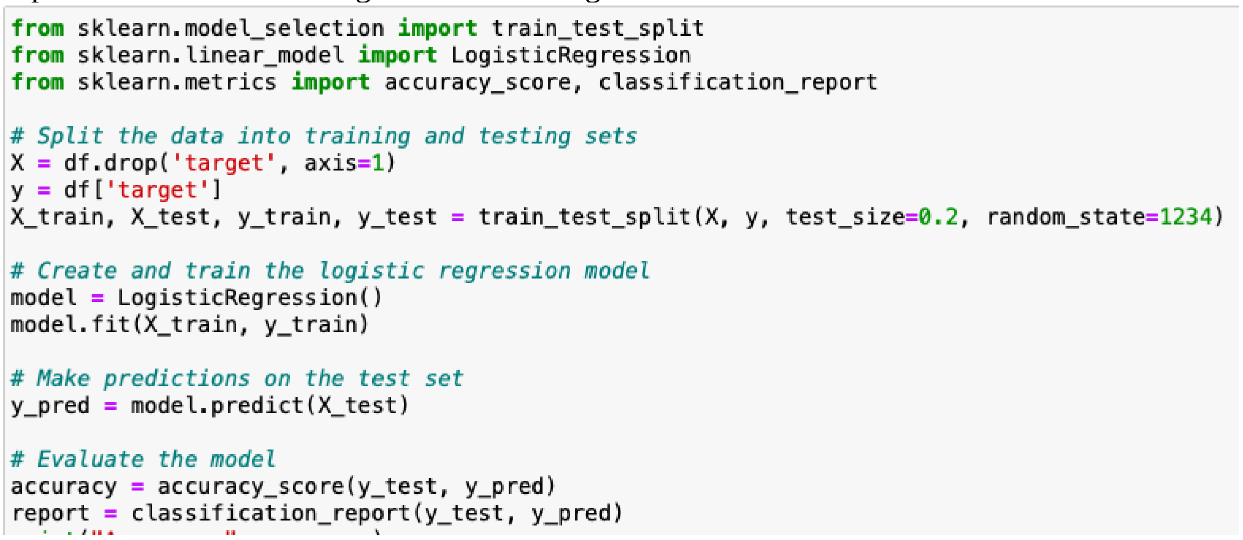
```python
# EDA: Scatter plot for Sepal Length vs Sepal Width
sns.scatterplot(data=df, x='sepal length (cm)', y='sepal width (cm)', hue='target')
plt.title('Scatter plot: Sepal Length vs Sepal Width')
plt.show()
```



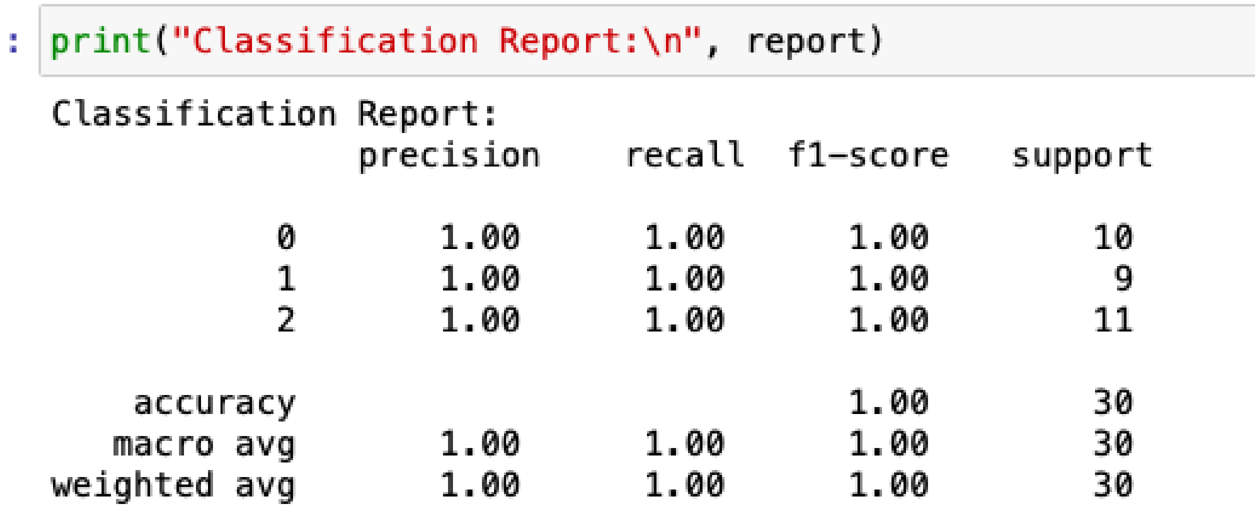
```python
# EDA: Scatter plot for Petal Length vs Petal Width
sns.scatterplot(data=df, x='petal length (cm)', y='petal width (cm)', hue='target')
plt.title('Scatter plot: Petal Length vs Petal Width')
plt.show()
```



Step 3: Build the model using machine learning.



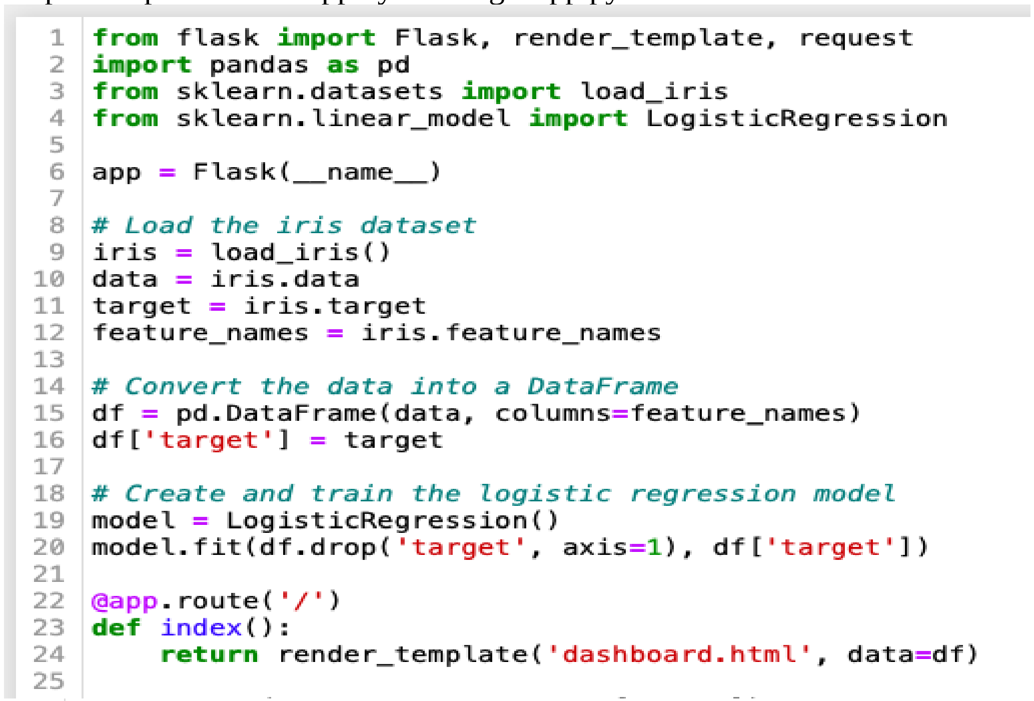
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Split the data into training and testing sets
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)

# Create and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```



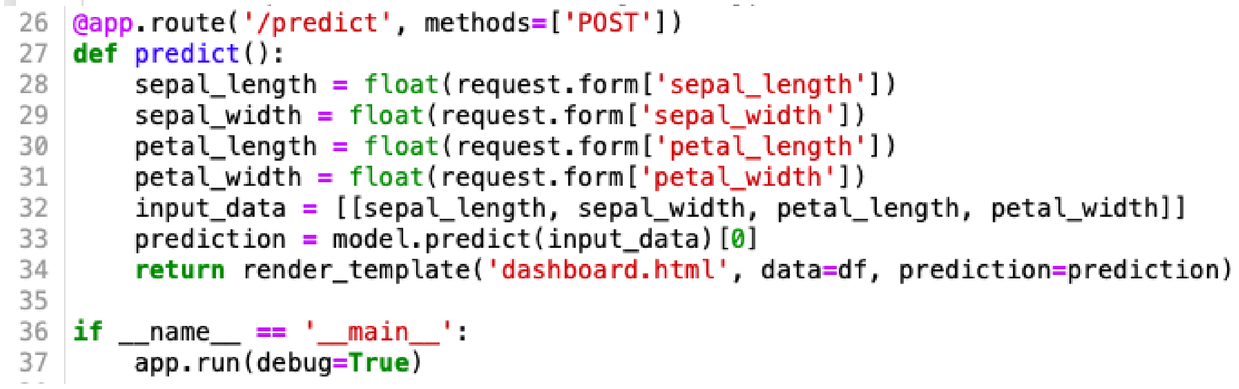
```
: print("Classification Report:\n", report)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Step 4: Prepare the webapp by creating a app.py file

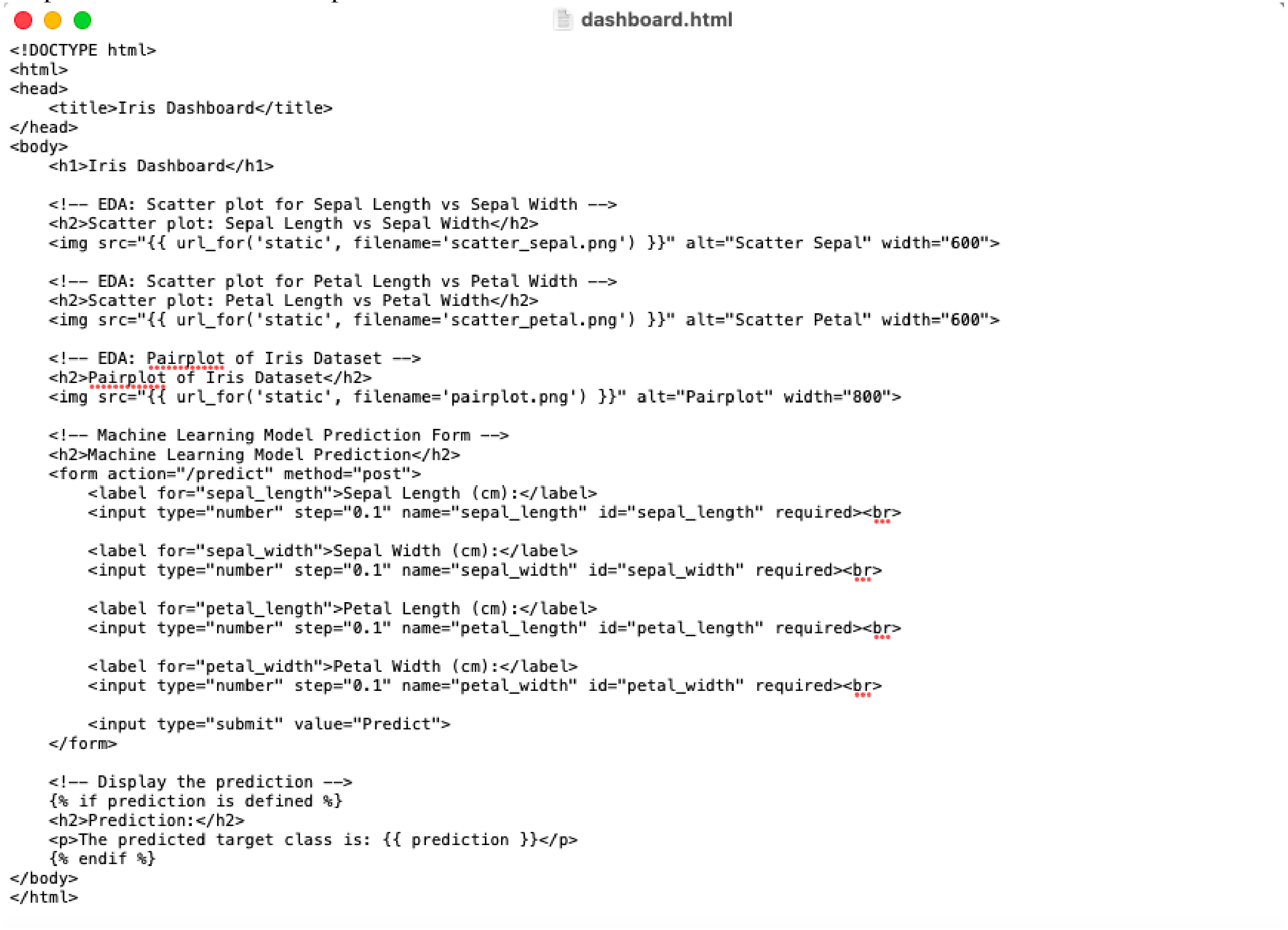


```python
from flask import Flask, render_template, request
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

app = Flask(__name__)

# Load the iris dataset
iris = load_iris()
data = iris.data
target = iris.target
feature_names = iris.feature_names

# Convert the data into a DataFrame
df = pd.DataFrame(data, columns=feature_names)
df['target'] = target

# Create and train the logistic regression model
model = LogisticRegression()
model.fit(df.drop('target', axis=1), df['target'])

@app.route('/')
def index():
    return render_template('dashboard.html', data=df)

```



```python
@app.route('/predict', methods=['POST'])
def predict():
    sepal_length = float(request.form['sepal_length'])
    sepal_width = float(request.form['sepal_width'])
    petal_length = float(request.form['petal_length'])
    petal_width = float(request.form['petal_width'])
    input_data = [[sepal_length, sepal_width, petal_length, petal_width]]
    prediction = model.predict(input_data)[0]
    return render_template('dashboard.html', data=df, prediction=prediction)

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

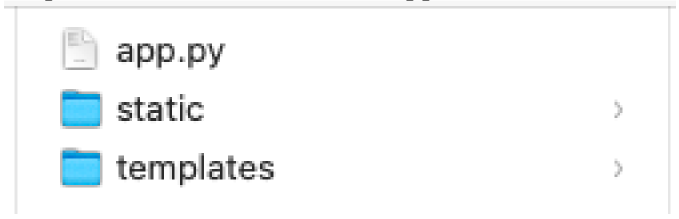
Step 5: Create HTML template called dashboard.html.



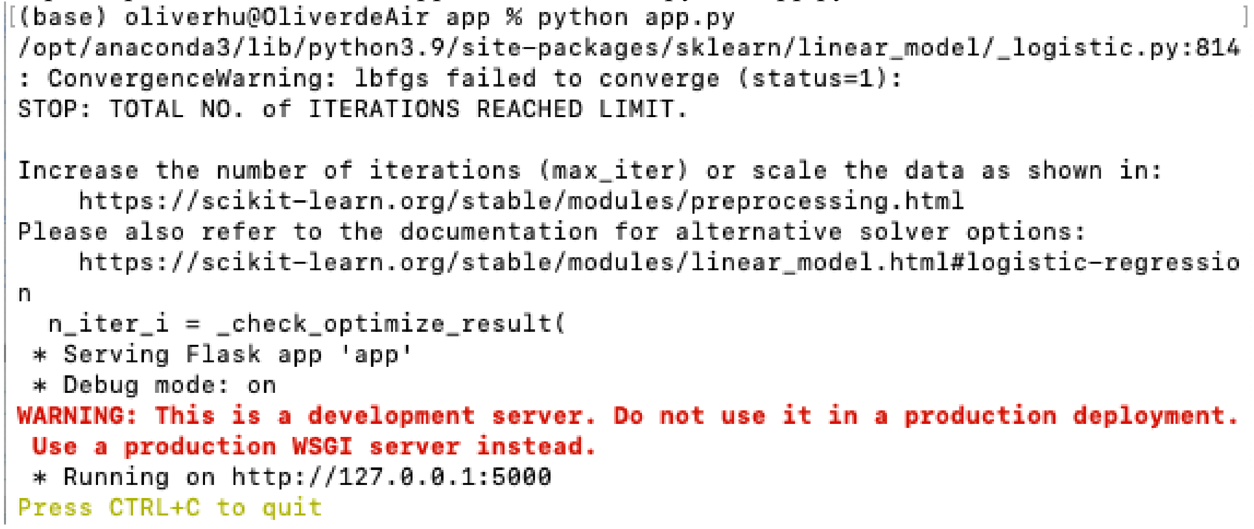
dashboard.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Iris Dashboard</title>
</head>
<body>
    <h1>Iris Dashboard</h1>

    <!-- EDA: Scatter plot for Sepal Length vs Sepal Width -->
    <h2>Scatter plot: Sepal Length vs Sepal Width</h2>
    <img src="{{ url_for('static', filename='scatter_sepal.png') }}" alt="Scatter Sepal" width="600">

    <!-- EDA: Scatter plot for Petal Length vs Petal Width -->
    <h2>Scatter plot: Petal Length vs Petal Width</h2>
    <img src="{{ url_for('static', filename='scatter_petal.png') }}" alt="Scatter Petal" width="600">

    <!-- EDA: Pairplot of Iris Dataset -->
    <h2>Pairplot of Iris Dataset</h2>
    <img src="{{ url_for('static', filename='pairplot.png') }}" alt="Pairplot" width="800">

    <!-- Machine Learning Model Prediction Form -->
    <h2>Machine Learning Model Prediction</h2>
    <form action="/predict" method="post">
        <label for="sepal_length">Sepal Length (cm):</label>
        <input type="number" step="0.1" name="sepal_length" id="sepal_length" required><br>

        <label for="sepal_width">Sepal Width (cm):</label>
        <input type="number" step="0.1" name="sepal_width" id="sepal_width" required><br>

        <label for="petal_length">Petal Length (cm):</label>
        <input type="number" step="0.1" name="petal_length" id="petal_length" required><br>

        <label for="petal_width">Petal Width (cm):</label>
        <input type="number" step="0.1" name="petal_width" id="petal_width" required><br>

        <input type="submit" value="Predict">
    </form>

    <!-- Display the prediction -->
    {% if prediction is defined %}
    <h2>Prediction:</h2>
    <p>The predicted target class is: {{ prediction }}</p>
    {% endif %}
</body>
</html>
```

Step 6: Create a folder for the app



Step 7: Open New Terminal at app folder, and run ‘python app.py’



```
[(base) oliverhu@OliverdeAir app % python app.py                              ]
/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814
: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n
  n_iter_i = _check_optimize_result(
 * 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
```

Step 8: Open http://127.0.0.1:5000 in browser. The app is shown in ‘Iris Dashboard_dh.pdf’ file.