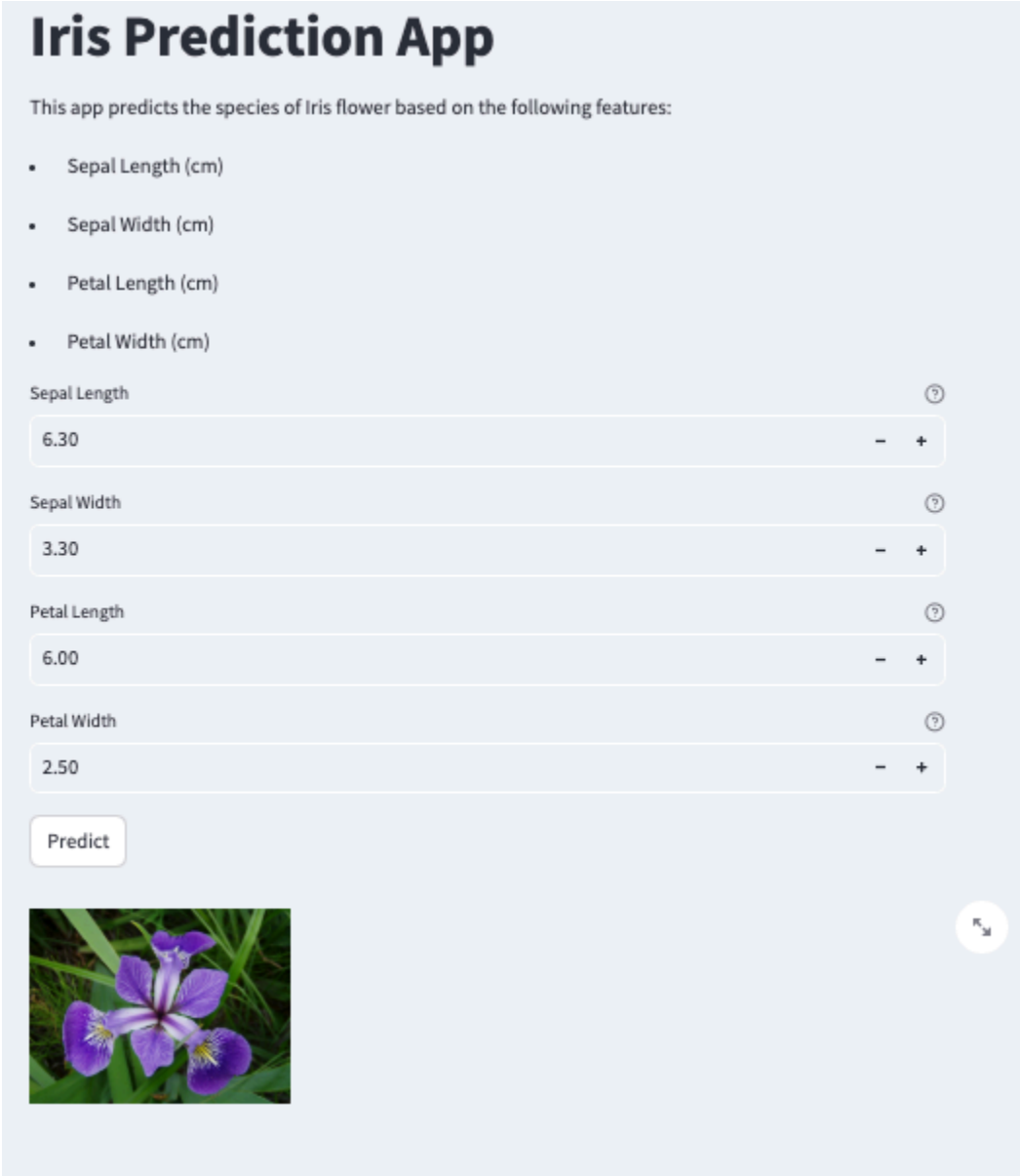
DATA-236 Sec 12 - Distributed Systems for Data Engineering
HOMEWORK 10
Nandhakumar Apparsamy
018190003

GitHub -
https://github.com/Nandha951/DATA-236-HW-10-Iris-Logistic-regression-Flask-Streamlit

Build two small apps — Flask and Streamlit — to deploy a machine learning model trained on a new dataset (of your choice) and stored in a .pkl file. Both apps should allow the user to input relevant data and receive a prediction.



localhost:8501

Iris Prediction App

This app predicts the species of Iris flower based on the following features:

- Sepal Length (cm)
- Sepal Width (cm)
- Petal Length (cm)
- Petal Width (cm)

Sepal Length

6.30

Sepal Width

3.30

Petal Length

6.00

Petal Width

2.50

Predict

The prediction is: virginica

localhost:8501

Iris Prediction App

This app predicts the species of Iris flower based on the following features:

- Sepal Length (cm)
- Sepal Width (cm)
- Petal Length (cm)
- Petal Width (cm)

Sepal Length

5.10

Sepal Width

3.50

Petal Length

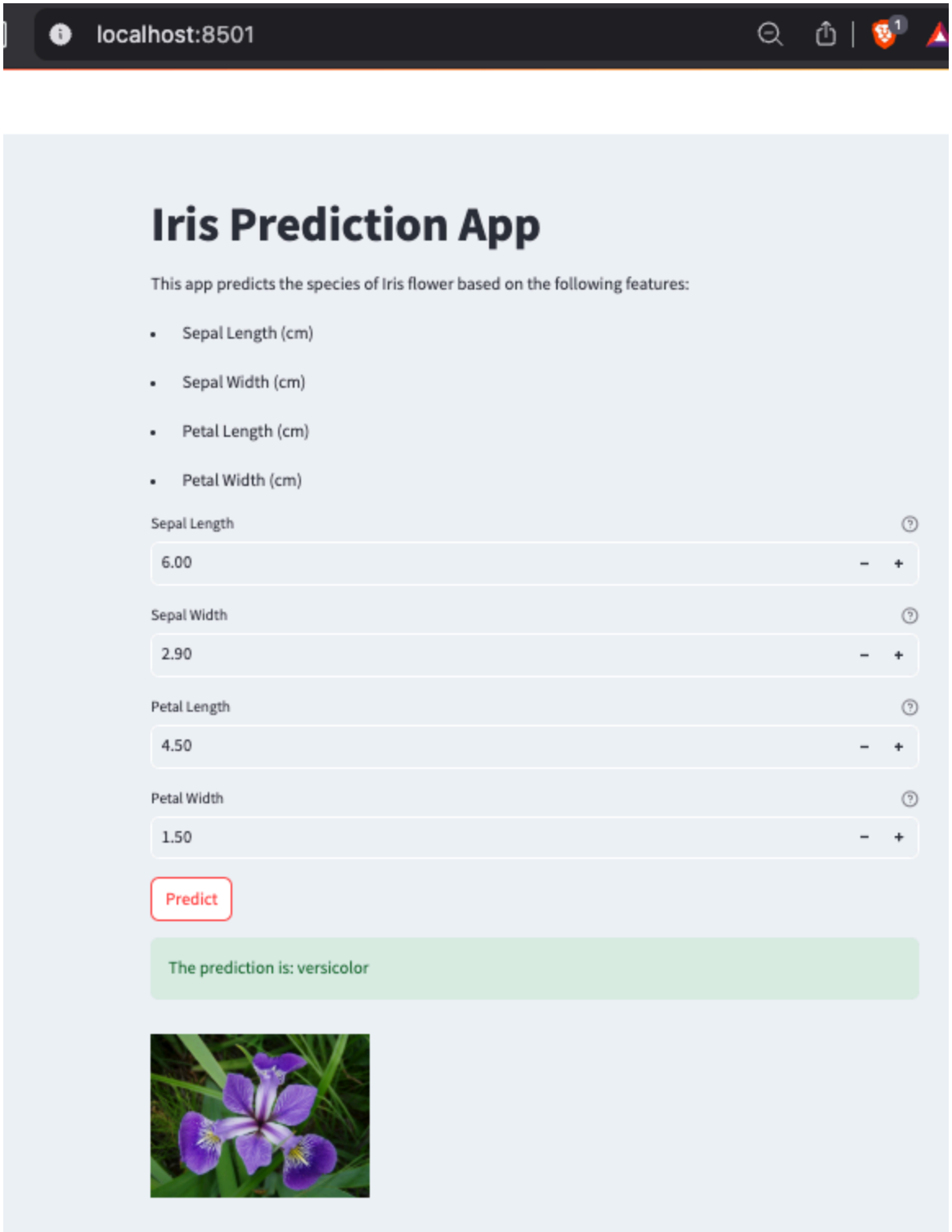
1.40

Petal Width

0.20

Predict

The prediction is: setosa



You must do the following:

● Choose or create a .pkl file for a new regression or classification task.

- You may use a pre-trained .pkl file or train your own model briefly.

ml_deployment > model_training.py

```python
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# 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)

# Train the model
model = LogisticRegression(random_state=42, solver='liblinear', multi_class='ovr')
model.fit(X_train, y_train)

# Save the model to a file
filename = 'model.pkl'
pickle.dump(model, open(filename, 'wb'))
print(f"Model saved to {filename}")
```

Create a Flask API (5 Marks)

- Load the model using pickle.
- Create a /predict endpoint that accepts POST requests with appropriate input JSON.
- Validate the input and return a clean JSON with the prediction and add error handling

ml_deployment > flask_app.py

```python
import flask
from flask import request, jsonify
import pickle
import numpy as np

app = flask.Flask(__name__)
app.config["DEBUG"] = True

# Load the model
try:
    model = pickle.load(open('model.pkl', 'rb'))
except Exception as e:
    print(f"Error loading the model: {e}")
    model = None

@app.route('/predict', methods=['POST'])
def predict():
    if model is None:
        return jsonify({'error': 'Model not loaded'}), 500

    try:
        data = request.get_json()
        # Validate input data
        if not all(key in data for key in ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']):
            return jsonify({'error': 'Invalid input: Missing features'}), 400

        sepal_length = data['sepal_length']
        sepal_width = data['sepal_width']
        petal_length = data['petal_length']
        petal_width = data['petal_width']

        if not all(isinstance(value, (int, float)) for value in [sepal_length, sepal_width, petal_length, petal_width]):
            return jsonify({'error': 'Invalid input: Features must be numbers'}), 400

        # Prepare input for the model
        input_array = np.array([[sepal_length, sepal_width, petal_length, petal_width]])

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

        # Map the prediction to the species name
        species_mapping = {0: 'setosa', 1: 'versicolor', 2: 'virginica'}
        species_name = species_mapping.get(prediction, 'unknown')

        # Return the prediction as JSON
        return jsonify({'prediction': species_name})

    except Exception as e:
        return jsonify({'error': str(e)}), 500

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

Create a Streamlit UI (5 Marks)

- Load the same .pkl file using joblib or pickle.
- Create input fields for the user (use number_input, text_input, etc.).
- Display the prediction when a button is clicked.

● Add styling or images relevant to your prediction task.

ml_deployment > streamlit_app.py

```python
import streamlit as st
import pickle

# Load the model
try:
    model = pickle.load(open('model.pkl', 'rb'))
except Exception as e:
    st.error(f"Error loading the model: {e}")
    model = None

st.title("Iris Prediction App")

st.write("This app predicts the species of Iris flower based on the following features:")
st.write("- Sepal Length (cm)")
st.write("- Sepal Width (cm)")
st.write("- Petal Length (cm)")
st.write("- Petal Width (cm)")

# Input fields
sepal_length = st.number_input("Sepal Length", value=5.1, help="Example: 5.1")
sepal_width = st.number_input("Sepal Width", value=3.5, help="Example: 3.5")
petal_length = st.number_input("Petal Length", value=1.4, help="Example: 1.4")
petal_width = st.number_input("Petal Width", value=0.2, help="Example: 0.2")

# Prediction button
if st.button("Predict"):
    if model is not None:
        # Make prediction
        input_array = [[sepal_length, sepal_width, petal_length, petal_width]]
        prediction = model.predict(input_array)[0]

        # Map the prediction to the species name
        species_mapping = {0: 'setosa', 1: 'versicolor', 2: 'virginica'}
        species_name = species_mapping.get(prediction, 'unknown')

        # Display the prediction
        st.success(f"The prediction is: {species_name}")
    else:
        st.error("Model not loaded. Please check the model file.")

# Add styling
st.markdown("""
    <style>
    .stApp {
        background-color: #f0f2f6;
    }
    </style>
    """, unsafe_allow_html=True)

# Add image
st.image("https://upload.wikimedia.org/wikipedia/commons/4/41/Iris_versicolor_3.jpg", width=200)
```

Submission Requirements:

● model.pkl (your trained model)

● flask_app.py

- streamlit_app.py

Marking Scheme (10 Marks):

- Flask app works correctly- 3
- Flask returns valid prediction JSON- 2
- Streamlit app works and takes input- 3
- UI shows prediction correctly- 2