Name: CHUKWUJEKWU JOSEPH EZEMA

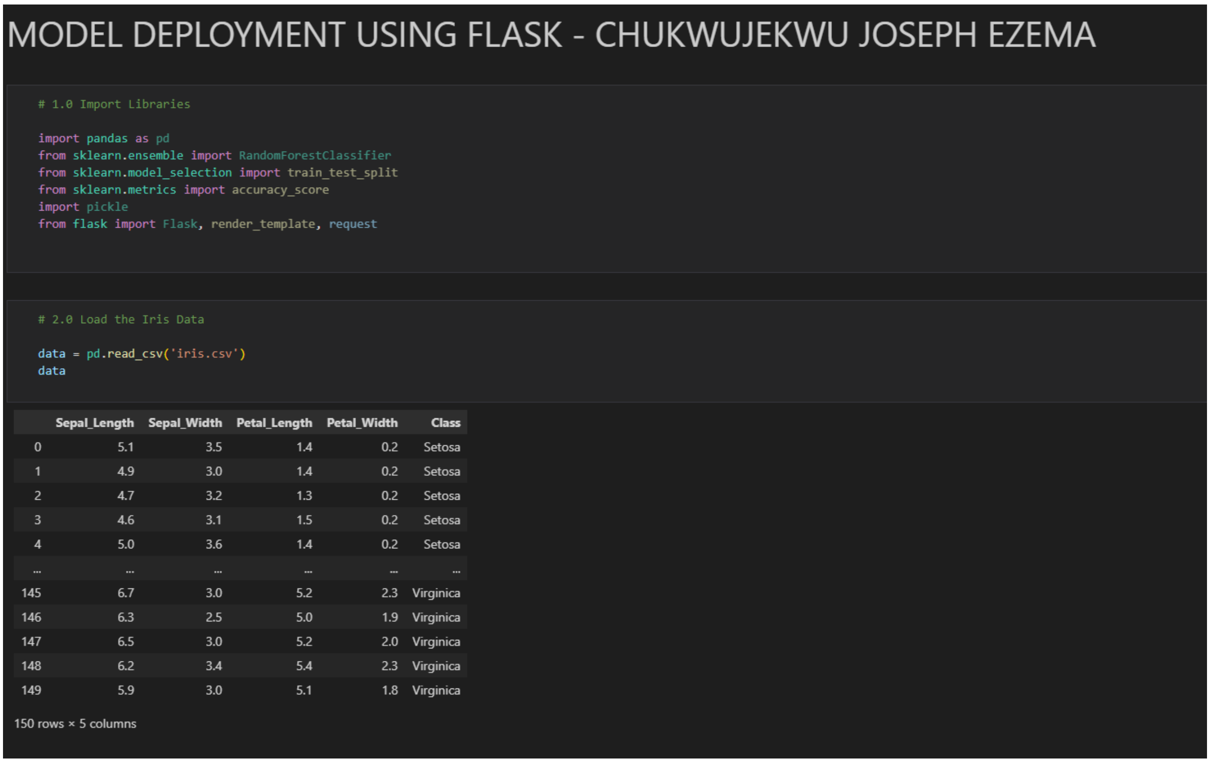
Batch Code: LISUM18
Submission Date: March 17th, 2023

Steps followed in Heroku Deployment using Flask.

1.0 **Choosing a Toy Dataset (Iris Flower Dataset)**

Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
5.1	3.5	1.4	0.2	Setosa
4.9	3	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Setosa
5	3.6	1.4	0.2	Setosa
5.4	3.9	1.7	0.4	Setosa
4.6	3.4	1.4	0.3	Setosa
5	3.4	1.5	0.2	Setosa
4.4	2.9	1.4	0.2	Setosa
4.9	3.1	1.5	0.1	Setosa
5.4	3.7	1.5	0.2	Setosa
4.8	3.4	1.6	0.2	Setosa
4.8	3	1.4	0.1	Setosa
4.3	3	1.1	0.1	Setosa
5.8	4	1.2	0.2	Setosa
5.7	4.4	1.5	0.4	Setosa
5.4	3.9	1.3	0.4	Setosa
5.1	3.5	1.4	0.3	Setosa
5.7	3.8	1.7	0.3	Setosa

2.0 Pre-processing and Modelling



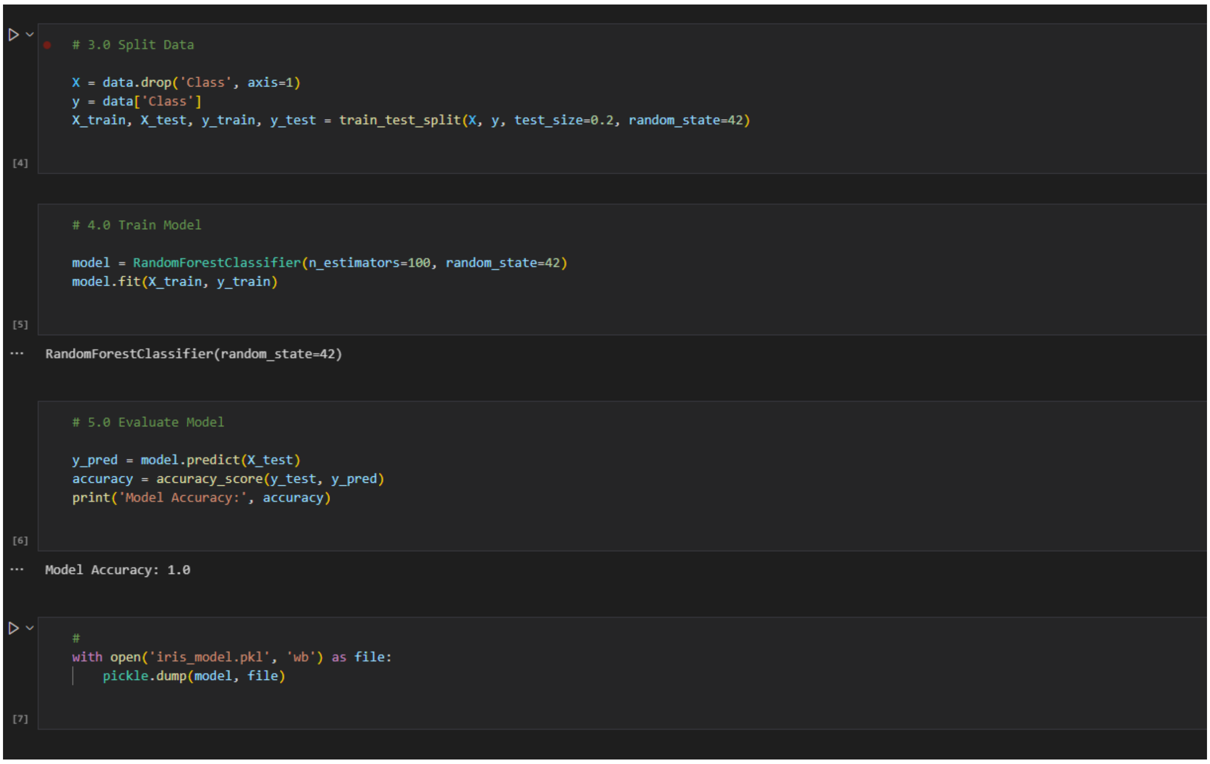
MODEL DEPLOYMENT USING FLASK - CHUKWUJEKWU JOSEPH EZEMA

```python
# 1.0 Import Libraries

import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pickle
from flask import Flask, render_template, request
```

```python
# 2.0 Load the Iris Data

data = pd.read_csv('iris.csv')
data
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

150 rows × 5 columns



```
# 3.0 Split Data

X = data.drop('Class', axis=1)
y = data['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[4]

```
# 4.0 Train Model

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

[5]

```
RandomForestClassifier(random_state=42)
```

```
# 5.0 Evaluate Model

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print('Model Accuracy:', accuracy)
```

[6]

```
Model Accuracy: 1.0
```

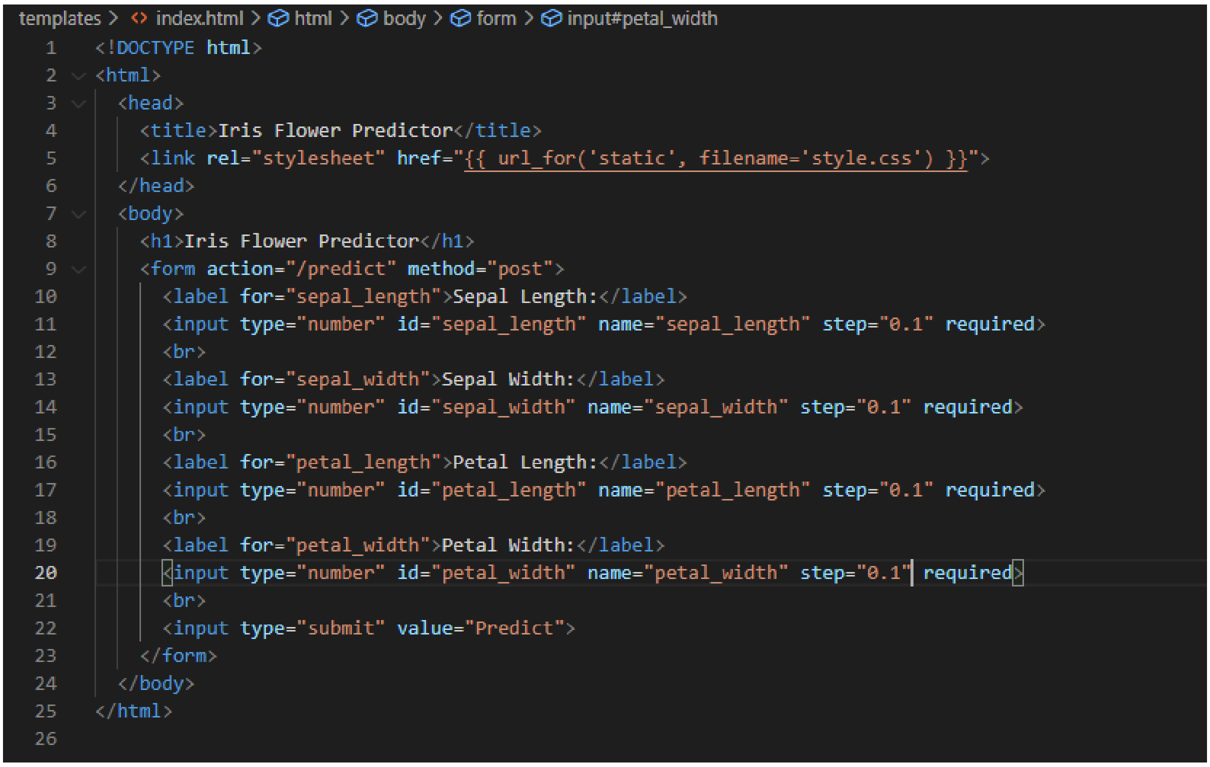
```
#
with open('iris_model.pkl', 'wb') as file:
    pickle.dump(model, file)
```

[7]

After building the model, I dumped it using pickle for flask application. I also saved the model to a python file (model.py)

3.0 Created HTMLs and CSS Files for Web Page Deployment

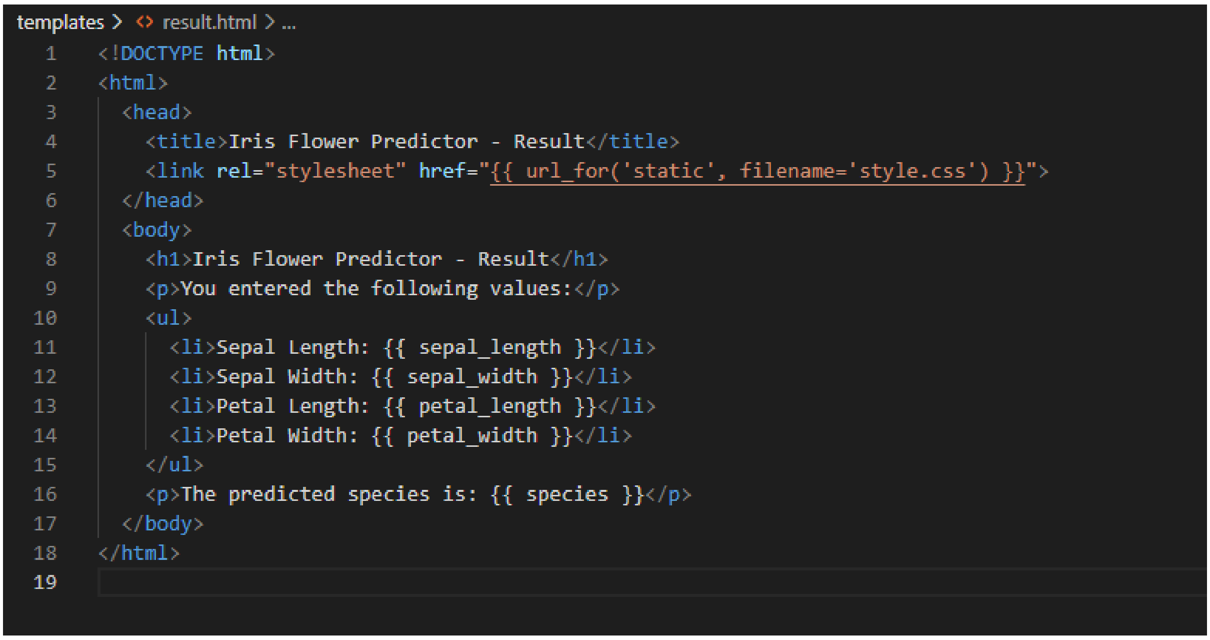
I. **index.html – for inputting the values to predict.**



templates > index.html > html > body > form > input#petal_width

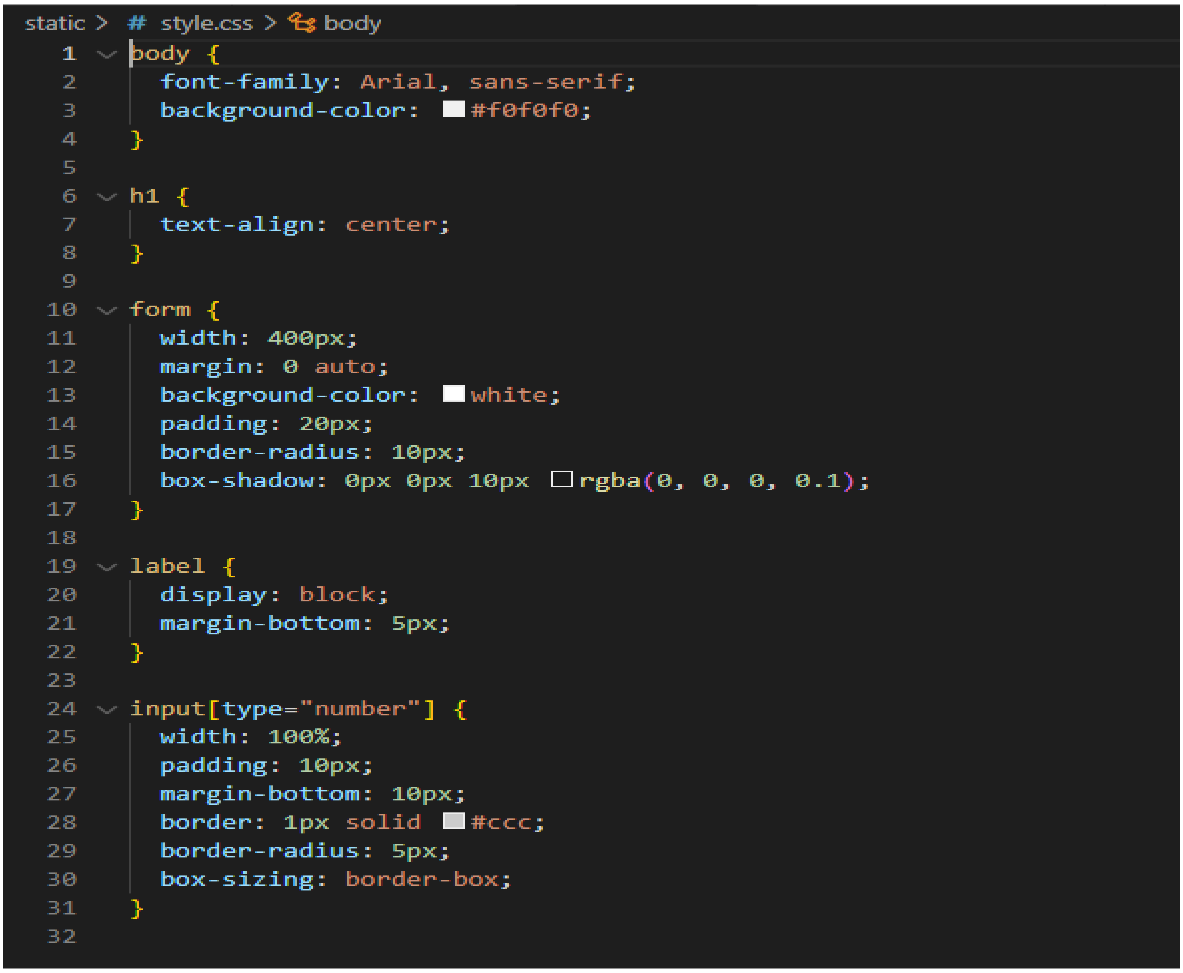
```
<!DOCTYPE html>
<html>
  <head>
    <title>Iris Flower Predictor</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>
    <h1>Iris Flower Predictor</h1>
    <form action="/predict" method="post">
      <label for="sepal_length">Sepal Length:</label>
      <input type="number" id="sepal_length" name="sepal_length" step="0.1" required>
      <br>
      <label for="sepal_width">Sepal Width:</label>
      <input type="number" id="sepal_width" name="sepal_width" step="0.1" required>
      <br>
      <label for="petal_length">Petal Length:</label>
      <input type="number" id="petal_length" name="petal_length" step="0.1" required>
      <br>
      <label for="petal_width">Petal Width:</label>
      <input type="number" id="petal_width" name="petal_width" step="0.1" required>
      <br>
      <input type="submit" value="Predict">
    </form>
  </body>
</html>

```

II. result.html – for showing the predicted results.



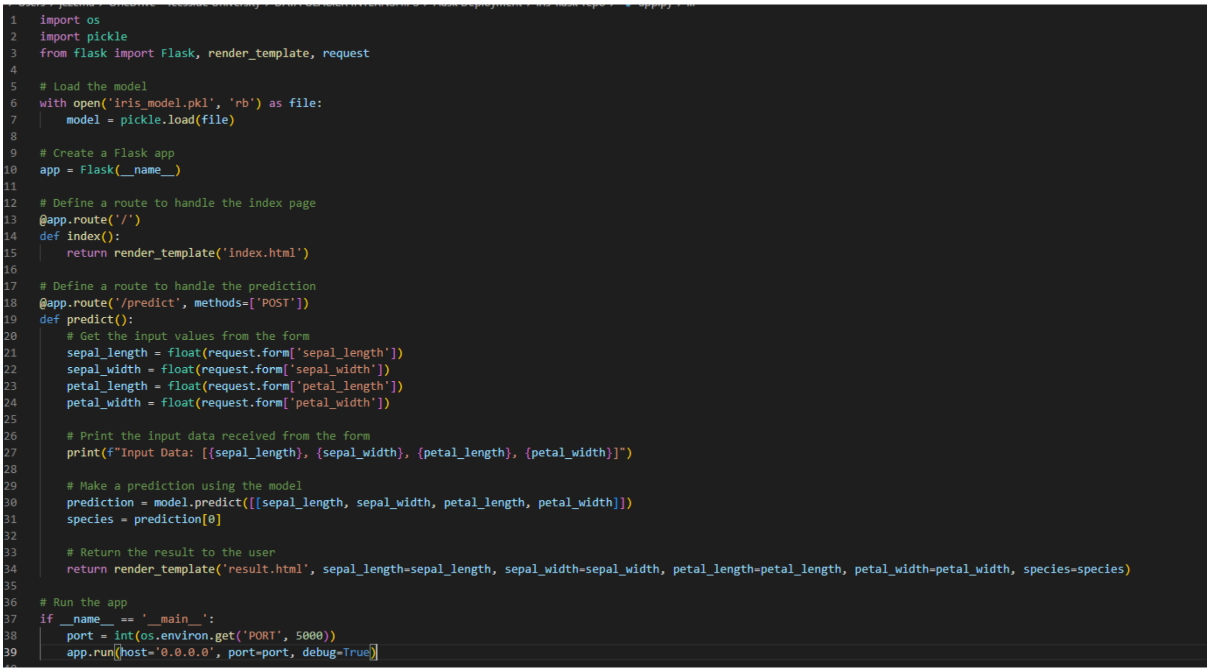
```html
<!DOCTYPE html>
<html>
  <head>
    <title>Iris Flower Predictor - Result</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>
    <h1>Iris Flower Predictor - Result</h1>
    <p>You entered the following values:</p>
    <ul>
      <li>Sepal Length: {{ sepal_length }}</li>
      <li>Sepal Width: {{ sepal_width }}</li>
      <li>Petal Length: {{ petal_length }}</li>
      <li>Petal Width: {{ petal_width }}</li>
    </ul>
    <p>The predicted species is: {{ species }}</p>
  </body>
</html>

```

III. style.css – web page formatting



```css
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}

h1 {
  text-align: center;
}

form {
  width: 400px;
  margin: 0 auto;
  background-color: white;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

label {
  display: block;
  margin-bottom: 5px;
}

input[type="number"] {
  width: 100%;
  padding: 10px;
  margin-bottom: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  box-sizing: border-box;
}

```

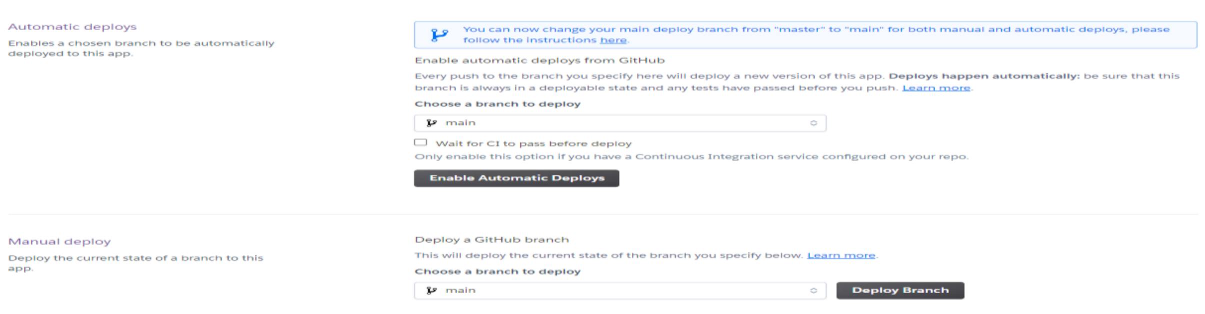
4.0 Created a Flask App



```python
import os
import pickle
from flask import Flask, render_template, request

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

# Create a Flask app
app = Flask(__name__)

# Define a route to handle the index page
@app.route('/')
def index():
    return render_template('index.html')

# Define a route to handle the prediction
@app.route('/predict', methods=['POST'])
def predict():
    # Get the input values from the form
    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'])

    # Print the input data received from the form
    print(f"Input Data: [{sepal_length}, {sepal_width}, {petal_length}, {petal_width}]")

    # Make a prediction using the model
    prediction = model.predict([[sepal_length, sepal_width, petal_length, petal_width]])
    species = prediction[0]

    # Return the result to the user
    return render_template('result.html', sepal_length=sepal_length, sepal_width=sepal_width, petal_length=petal_length, petal_width=petal_width, species=species)

# Run the app
if __name__ == '__main__':
    port = int(os.environ.get('PORT', 5000))
    app.run(host='0.0.0.0', port=port, debug=True)
```

Added port for through which the Heroku app can run.

5.0 App Deployment on Heroku using app creation from the website



Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy: main

□ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

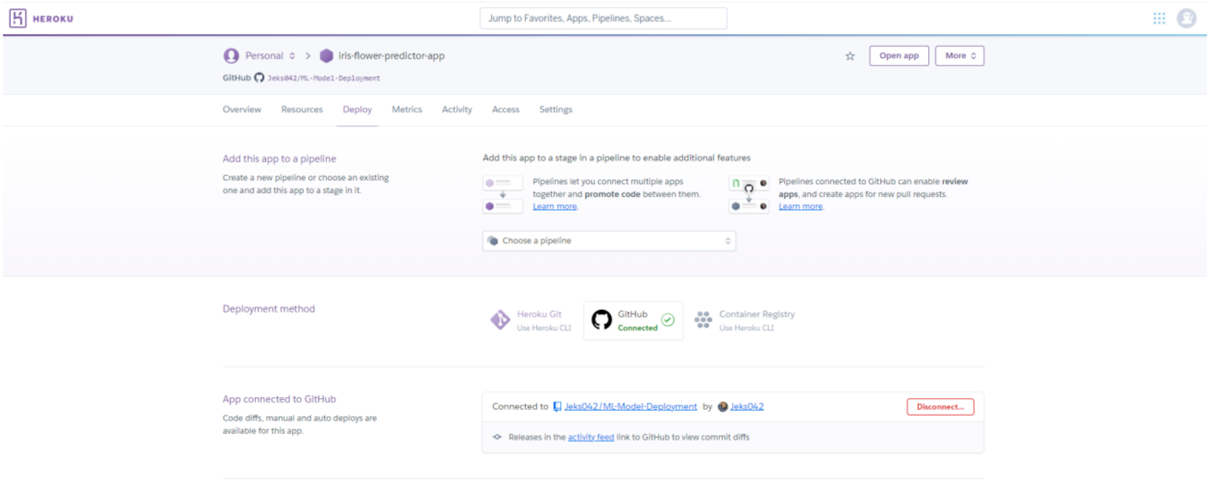
Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy: main



GitHub Jeks042/ML-Model-Deployment

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Deployment method

App connected to GitHub

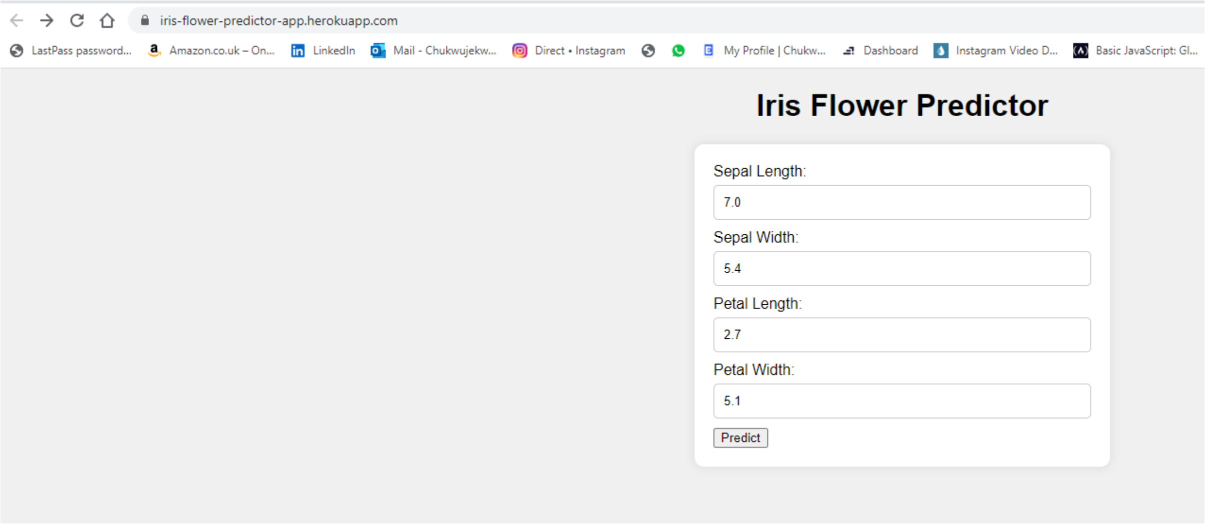
Code diffs, manual and auto deploys are available for this app.

Connected to Jeks042/ML-Model-Deployment by Jeks042

Releases in the activity feed link to GitHub to view commit diffs

- Connected the GitHub repository where I have put my project files.
- Added Procfile for declaring the port route.
- Updated the dependencies (requirements.txt)

6.0 **Web Page for Testing from the web (Heroku): https://iris-flower-predictor-app.herokuapp.com/**



Iris Flower Predictor

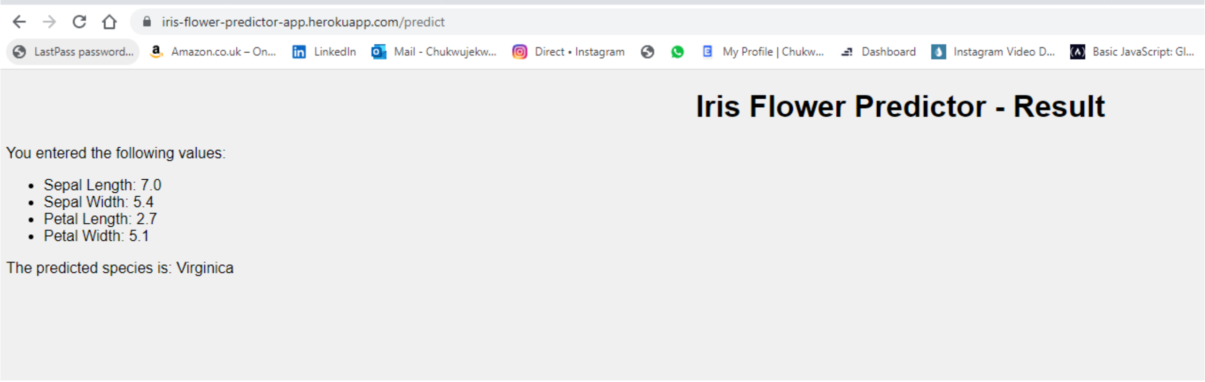
Sepal Length: 7.0

Sepal Width: 5.4

Petal Length: 2.7

Petal Width: 5.1

7.0 Web Page for Predicted Result



Iris Flower Predictor - Result

You entered the following values:

- Sepal Length: 7.0
- Sepal Width: 5.4
- Petal Length: 2.7
- Petal Width: 5.1

The predicted species is: Virginica

SUMMARY:

This project involved creating a machine learning model to predict the species of an iris flower based on its sepal and petal dimensions. The iris flower dataset was used to train and test the model, which was built using Python's scikit-learn library. A Flask web application was created to allow users to input the dimensions of an iris flower and get a predicted species as output. Two HTML files were created for the web application: index.html for inputting values and result.html for displaying the predicted result. The app was then deployed on the heroku using the app creation channel on the website. A screenshot of the web page for testing, with input values of Sepal Length 7.0, Sepal Width 5.4, Petal Length 2.7, and Petal Width 5.1 was shown. The predicted result for these values was Virginica, which was displayed on the web page for predicted result. Overall, this project introduced machine learning modelling and web application deployment on Heroku using flask app.