Name: Tony Nguyen

Batch Code: LISUM30

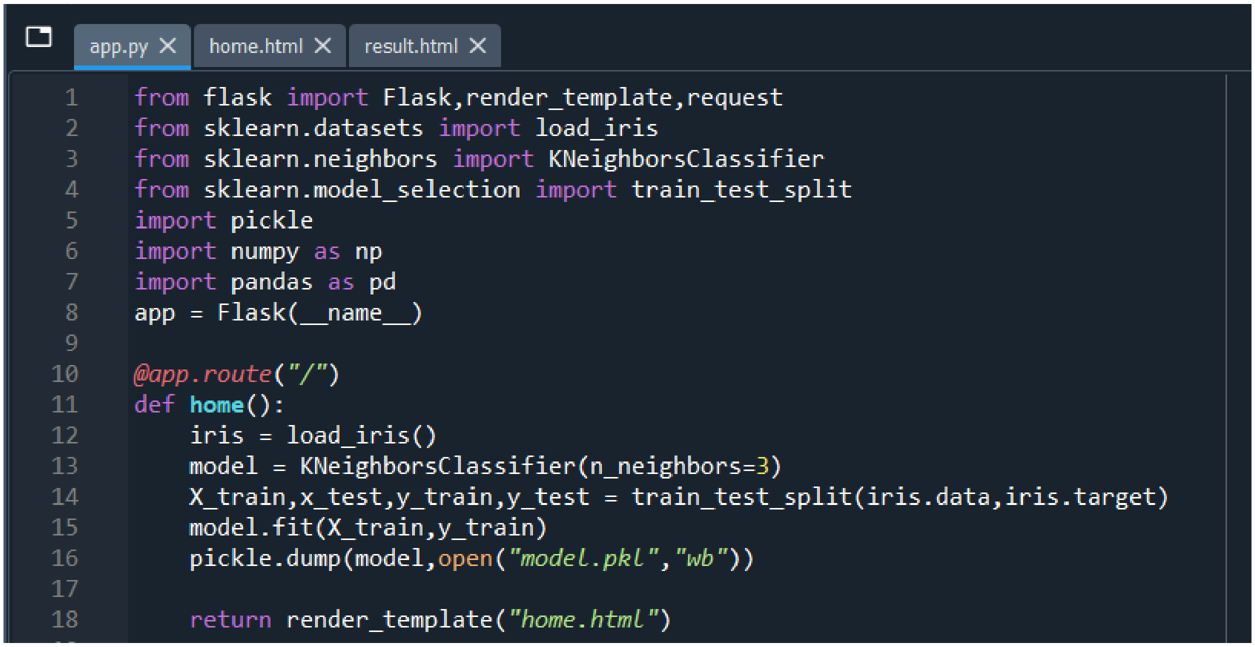
Submission date: 2/27/2024

Submitted to: Data Glacier

Deployment on Flask

Step 1:

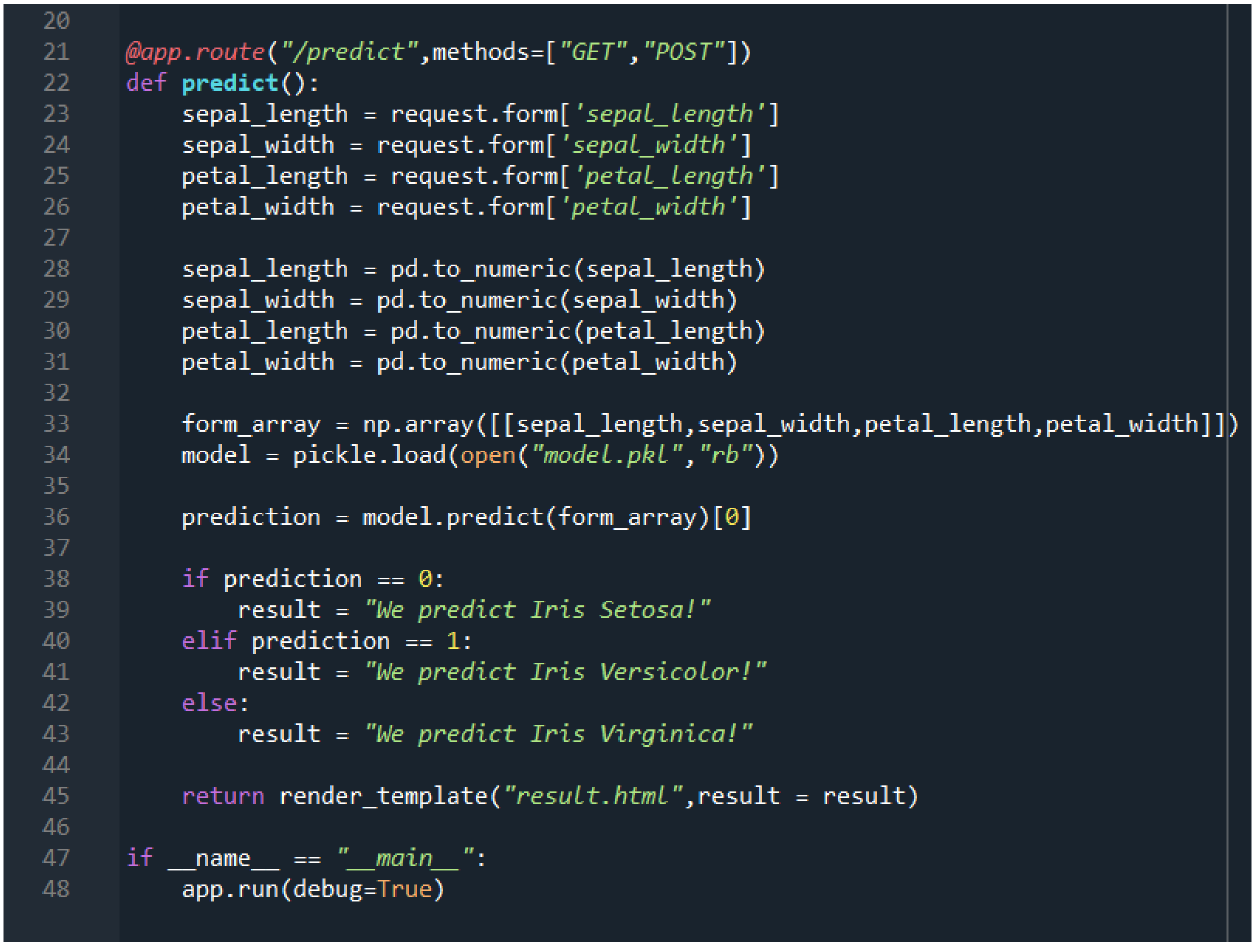
Develop Model – Predict the type of iris flower based on the length and width of the sepal and petal.



```python
from flask import Flask,render_template,request
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import pickle
import numpy as np
import pandas as pd
app = Flask(__name__)

@app.route("/")
def home():
    iris = load_iris()
    model = KNeighborsClassifier(n_neighbors=3)
    X_train,x_test,y_train,y_test = train_test_split(iris.data,iris.target)
    model.fit(X_train,y_train)
    pickle.dump(model,open("model.pkl","wb"))

    return render_template("home.html")
```

Step 2:

Saving model and Deployment – Uses *pickle* library to save trained model. Takes given input from users and predicts iris flower based on the inputs given.

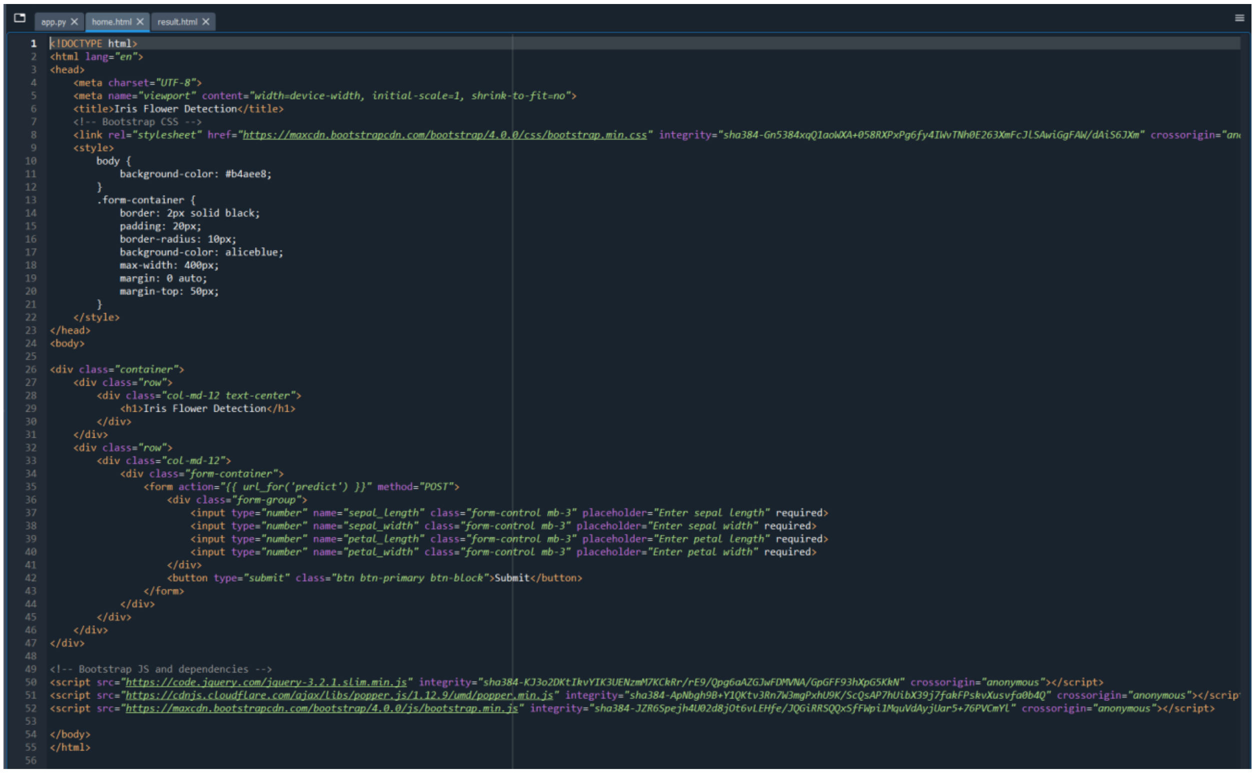


```python

@app.route("/predict",methods=["GET","POST"])
def predict():
    sepal_length = request.form['sepal_length']
    sepal_width = request.form['sepal_width']
    petal_length = request.form['petal_length']
    petal_width = request.form['petal_width']

    sepal_length = pd.to_numeric(sepal_length)
    sepal_width = pd.to_numeric(sepal_width)
    petal_length = pd.to_numeric(petal_length)
    petal_width = pd.to_numeric(petal_width)

    form_array = np.array([[sepal_length,sepal_width,petal_length,petal_width]])
    model = pickle.load(open("model.pkl","rb"))

    prediction = model.predict(form_array)[0]

    if prediction == 0:
        result = "We predict Iris Setosa!"
    elif prediction == 1:
        result = "We predict Iris Versicolor!"
    else:
        result = "We predict Iris Virginica!"

    return render_template("result.html",result = result)

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

Step 3:

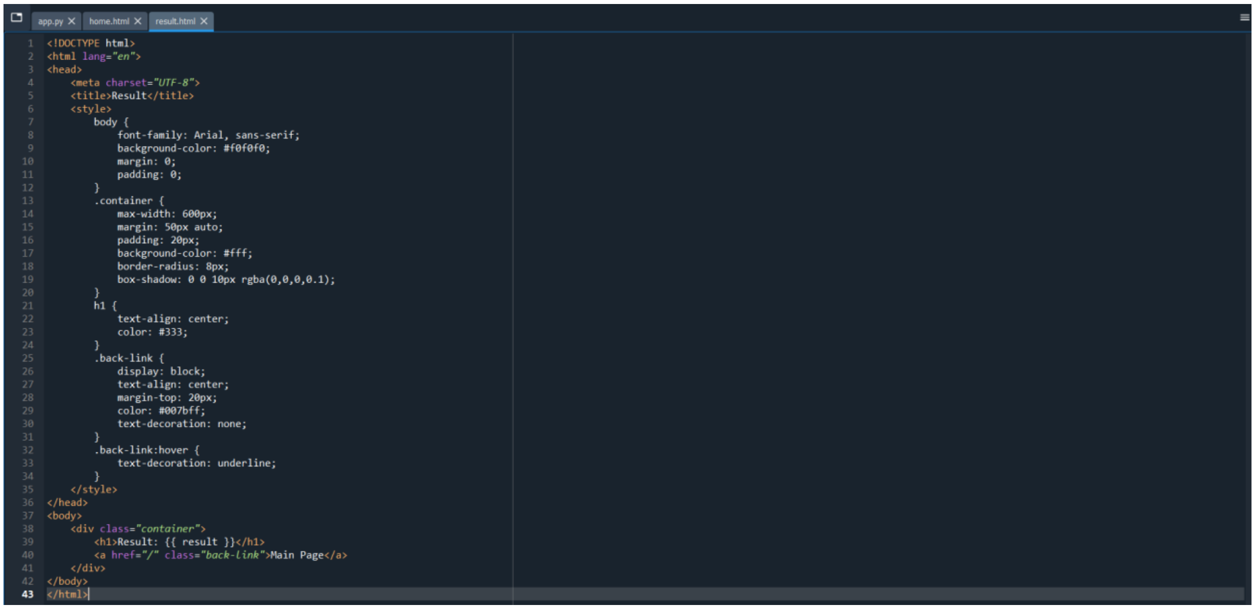
Creating home page and result page formats

Main Page:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Iris Flower Detection</title>
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="an
    <style>
        body {
            background-color: #b4aee8;
        }
        .form-container {
            border: 2px solid black;
            padding: 20px;
            border-radius: 10px;
            background-color: aliceblue;
            max-width: 400px;
            margin: 0 auto;
            margin-top: 50px;
        }
    </style>
</head>
<body>

<div class="container">
    <div class="row">
        <div class="col-md-12 text-center">
            <h1>Iris Flower Detection</h1>
        </div>
    </div>
    <div class="row">
        <div class="col-md-12">
            <div class="form-container">
                <form action="{{ url_for('predict') }}" method="POST">
                    <div class="form-group">
                        <input type="number" name="sepal_length" class="form-control mb-3" placeholder="Enter sepal length" required>
                        <input type="number" name="sepal_width" class="form-control mb-3" placeholder="Enter sepal width" required>
                        <input type="number" name="petal_length" class="form-control mb-3" placeholder="Enter petal length" required>
                        <input type="number" name="petal_width" class="form-control mb-3" placeholder="Enter petal width" required>
                    </div>
                    <button type="submit" class="btn btn-primary btn-block">Submit</button>
                </form>
            </div>
        </div>
    </div>
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></scrip
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

</body>
</html>

```

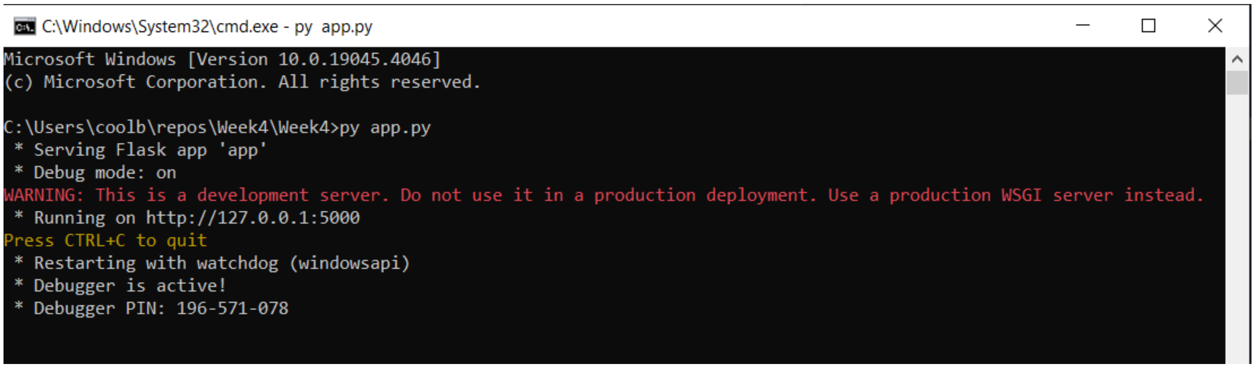
Results Page:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Result</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 50px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h1 {
            text-align: center;
            color: #333;
        }
        .back-link {
            display: block;
            text-align: center;
            margin-top: 20px;
            color: #007bff;
            text-decoration: none;
        }
        .back-link:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Result: {{ result }}</h1>
        <a href="/" class="back-link">Main Page</a>
    </div>
</body>
</html>
```

Step 4:

Python app.py file in CMD.

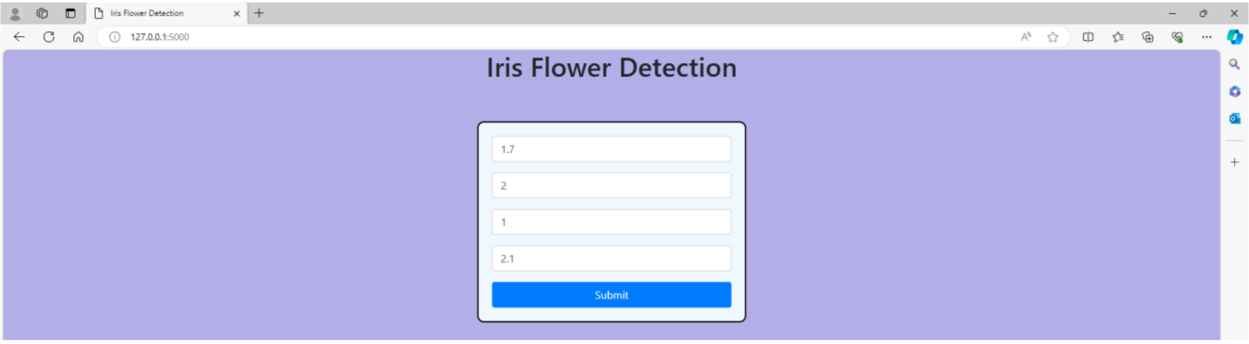


```
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\coolb\repos\Week4\Week4>py app.py
 * 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 watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 196-571-078
```

Step 5:

Web App – Using the URL and testing the application.

Main Page:



Iris Flower Detection

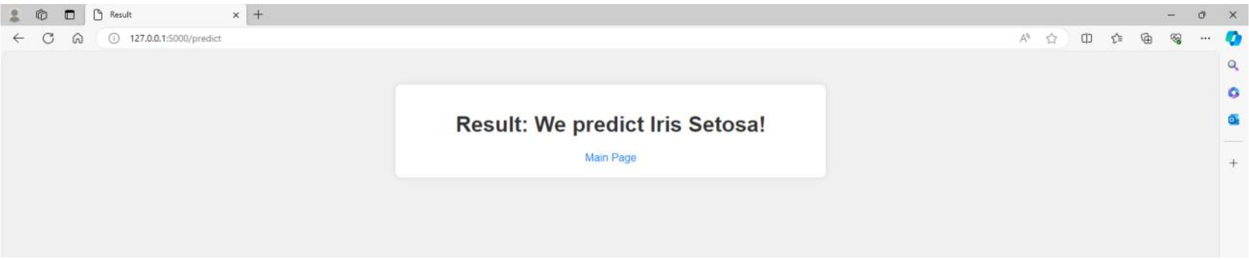
1.7

2

1

2.1

Results Page:

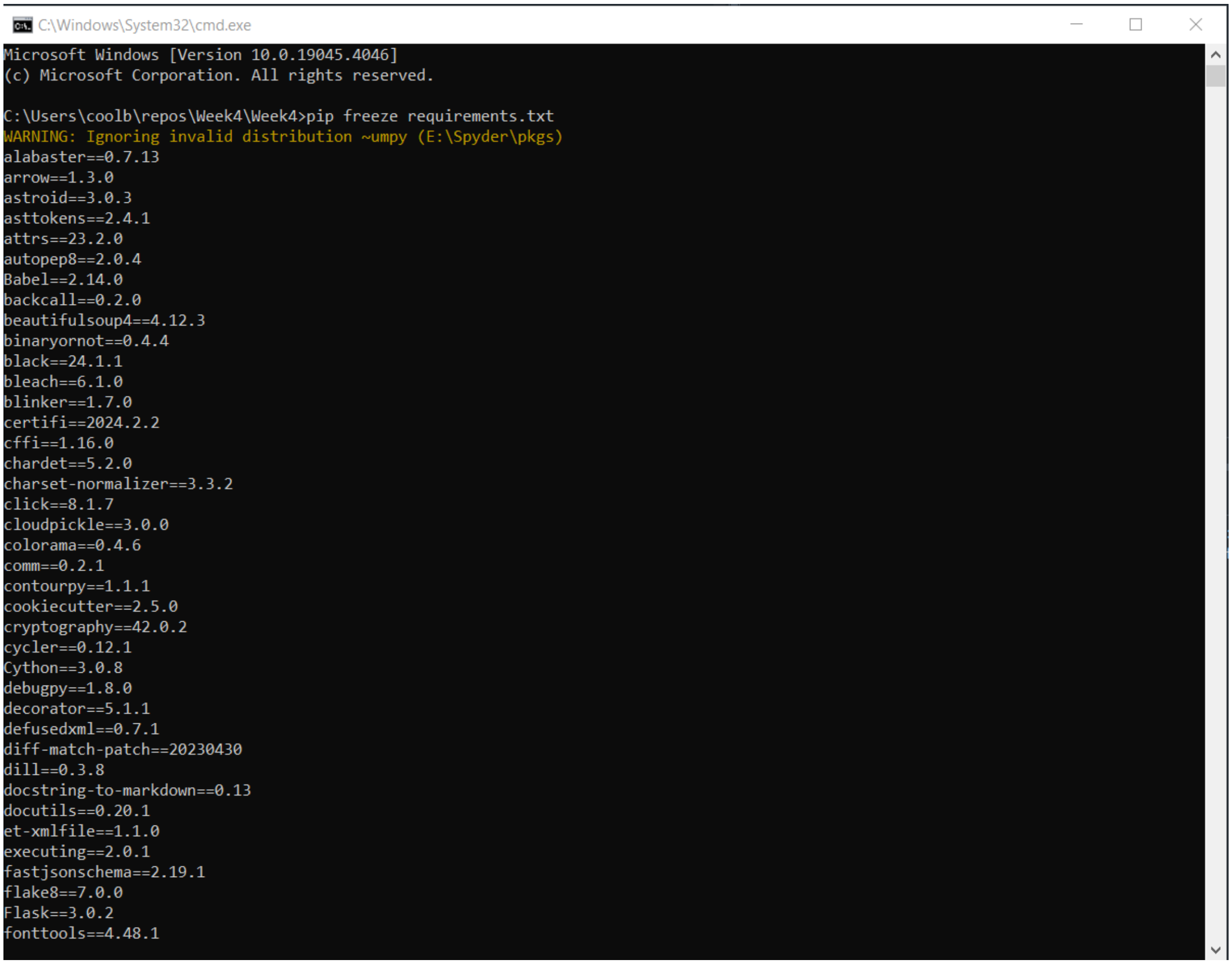


Result: We predict Iris Setosa!

Main Page

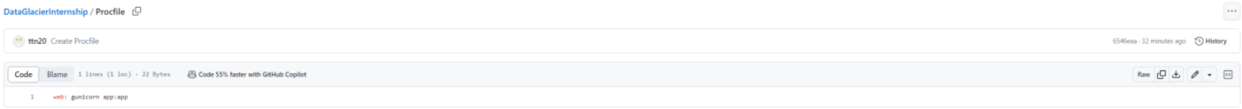
Step 6:

Heroku app Setup – Using ‘pip freeze requirements.txt’ in CMD, create new file that has the dependencies for the flask app. Also create Procfile file for startup.



```
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\coolb\repos\Week4\Week4>pip freeze requirements.txt
WARNING: Ignoring invalid distribution ~umpy (E:\Spyder\pkgs)
alabaster==0.7.13
arrow==1.3.0
astroid==3.0.3
asttokens==2.4.1
attrs==23.2.0
autopep8==2.0.4
Babel==2.14.0
backcall==0.2.0
beautifulsoup4==4.12.3
binaryornot==0.4.4
black==24.1.1
bleach==6.1.0
blinker==1.7.0
certifi==2024.2.2
cffi==1.16.0
chardet==5.2.0
charset-normalizer==3.3.2
click==8.1.7
cloudpickle==3.0.0
colorama==0.4.6
comm==0.2.1
contourpy==1.1.1
cookiecutter==2.5.0
cryptography==42.0.2
cycler==0.12.1
Cython==3.0.8
debugpy==1.8.0
decorator==5.1.1
defusedxml==0.7.1
diff-match-patch==20230430
dill==0.3.8
docstring-to-markdown==0.13
docutils==0.20.1
et-xmlfile==1.1.0
executing==2.0.1
fastjsonschema==2.19.1
flake8==7.0.0
Flask==3.0.2
fonttools==4.48.1
```

Note: List goes on



DataGlacierInternship / Procfile
ttn20 Create Procfile
6546eaa · 32 minutes ago
History
Code
Blame
1 lines (1 loc) · 22 Bytes
Code 55% faster with GitHub Copilot
Raw

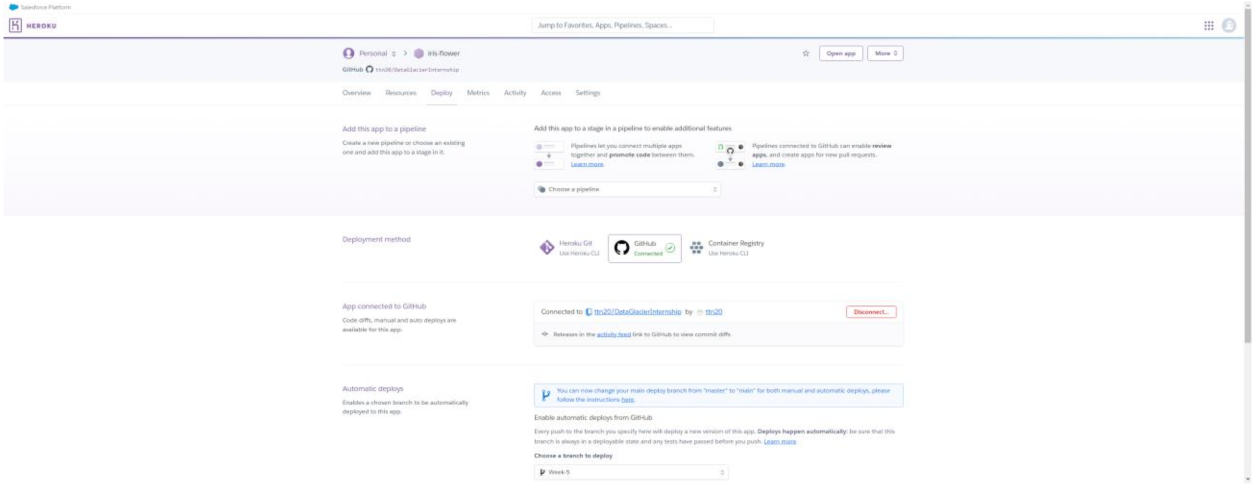
```
web: gunicorn app:app
```

Step 7:

Upload files required to Github - https://github.com/ttn20/DataGlacierInternship/tree/Week-5

Step 8:

Heroku App and Github linkage



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.

Choose a pipeline

Deployment method

App connected to GitHub

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

Connected to ttn20/DataGlacierInternship by ttn20

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

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

Week-5

Step 9:

Deploy model on Heroku