

# Deploying a Model

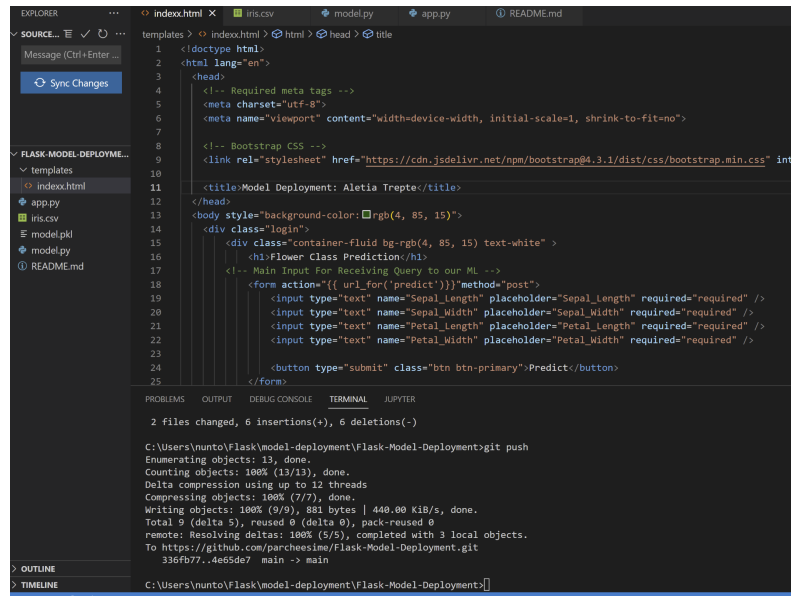
Name: Aletia Trepte

Batch: LISUM11: 30

Date: 07/28/2022

To: Glacier Data

Readme file



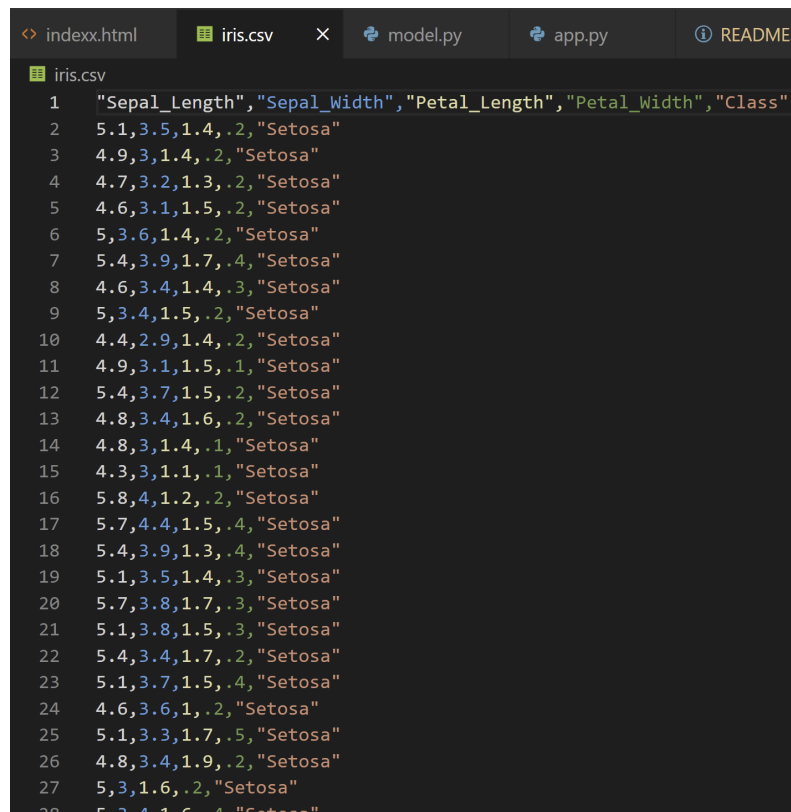
```
index.html | iris.csv | model.py | app.py | README.md

1 <!doctype html>
2 <html lang="en">
3 <head>
4 <!-- Required meta tags -->
5 <meta charset="utf-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7
8 <!-- Bootstrap CSS -->
9 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" int
10
11 <title>Model Deployment: Aletia Trepte</title>
12 </head>
13 <body style="background-color: #rgb(4, 85, 15)">
14 <div class="login">
15 <div class="container-fluid bg-rgb(4, 85, 15) text-white">
16 <h1>Flower Class Prediction</h1>
17 <!-- Main Input For Receiving Query to our ML -->
18 <form action="{url_for('predict')}}" method="post">
19 <input type="text" name="Sepal_Length" placeholder="Sepal_Length" required="required" />
20 <input type="text" name="Sepal_Width" placeholder="Sepal_Width" required="required" />
21 <input type="text" name="Petal_Length" placeholder="Petal_Length" required="required" />
22 <input type="text" name="Petal_Width" placeholder="Petal_Width" required="required" />
23
24 <button type="submit" class="btn btn-primary">Predict</button>
25 </form>

2 files changed, 6 insertions(+), 6 deletions(-)

C:\Users\nunto\Flask\model-deployment\Flask-Model-Deployment>git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (9/9), 881 bytes | 440.00 KiB/s, done.
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 3 local objects.
To https://github.com/parcheesime/Flask-Model-Deployment.git
336fb77..4e65de7 main -> main
```

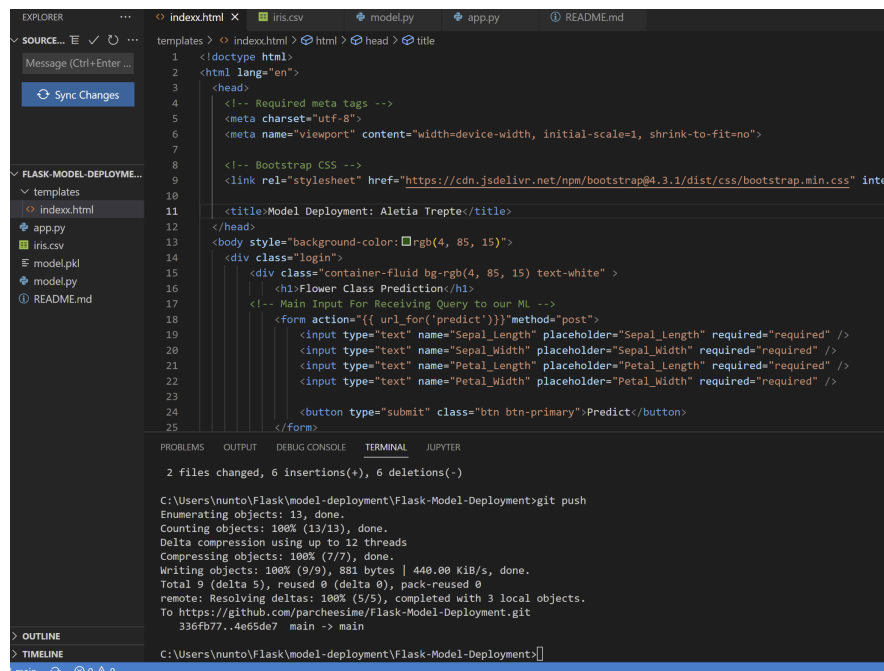
Iris dataset



```
iris.csv

1 "Sepal_Length","Sepal_Width","Petal_Length","Petal_Width","Class"
2 5.1,3.5,1.4,.2,"Setosa"
3 4.9,3,1.4,.2,"Setosa"
4 4.7,3.2,1.3,.2,"Setosa"
5 4.6,3.1,1.5,.2,"Setosa"
6 5,3.6,1.4,.2,"Setosa"
7 5.4,3.9,1.7,.4,"Setosa"
8 4.6,3.4,1.4,.3,"Setosa"
9 5,3.4,1.5,.2,"Setosa"
10 4.4,2.9,1.4,.2,"Setosa"
11 4.9,3.1,1.5,.1,"Setosa"
12 5.4,3.7,1.5,.2,"Setosa"
13 4.8,3.4,1.6,.2,"Setosa"
14 4.8,3,1.4,.1,"Setosa"
15 4.3,3,1.1,.1,"Setosa"
16 5.8,4,1.2,.2,"Setosa"
17 5.7,4.4,1.5,.4,"Setosa"
18 5.4,3.9,1.3,.4,"Setosa"
19 5.1,3.5,1.4,.3,"Setosa"
20 5.7,3.8,1.7,.3,"Setosa"
21 5.1,3.8,1.5,.3,"Setosa"
22 5.4,3.4,1.7,.2,"Setosa"
23 5.1,3.7,1.5,.4,"Setosa"
24 4.6,3.6,1,.2,"Setosa"
25 5.1,3.3,1.7,.5,"Setosa"
26 4.8,3.4,1.9,.2,"Setosa"
27 5,3,1.6,.2,"Setosa"
28 5,3,4,1.6,.4,"Setosa"
```

## Index.html



The screenshot shows the VS Code editor with the `index.html` file open. The file contains HTML code for a web page titled "Model Deployment: Aletia Trepte". It includes a Bootstrap CSS link and a form for predicting flower classes based on sepal and petal dimensions. The terminal window at the bottom shows the output of a `git push` command, indicating a successful push to the repository.

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <!-- Required meta tags -->
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7
8   <!-- Bootstrap CSS -->
9   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-ggkYb9VibgetRMHt2nhtNVvW9sFpohsF8fb68C7c7qT1giVg347qD802Ve7Wp7s" crossorigin="anonymous">
10
11   <title>Model Deployment: Aletia Trepte</title>
12 </head>
13 <body style="background-color: rgb(4, 85, 15)">
14   <div class="login">
15     <div class="container-fluid bg-rgb(4, 85, 15) text-white">
16       <h1>Flower Class Prediction</h1>
17       <!-- Main Input For Receiving Query to our ML -->
18       <form action="{{ url_for('predict')}}" method="post">
19         <input type="text" name="Sepal_Length" placeholder="Sepal_Length" required="required" />
20         <input type="text" name="Sepal_Width" placeholder="Sepal_Width" required="required" />
21         <input type="text" name="Petal_Length" placeholder="Petal_Length" required="required" />
22         <input type="text" name="Petal_Width" placeholder="Petal_Width" required="required" />
23         <button type="submit" class="btn btn-primary">Predict</button>
24       </form>
25     </div>
26   </div>
27 </body>
28 </html>
```

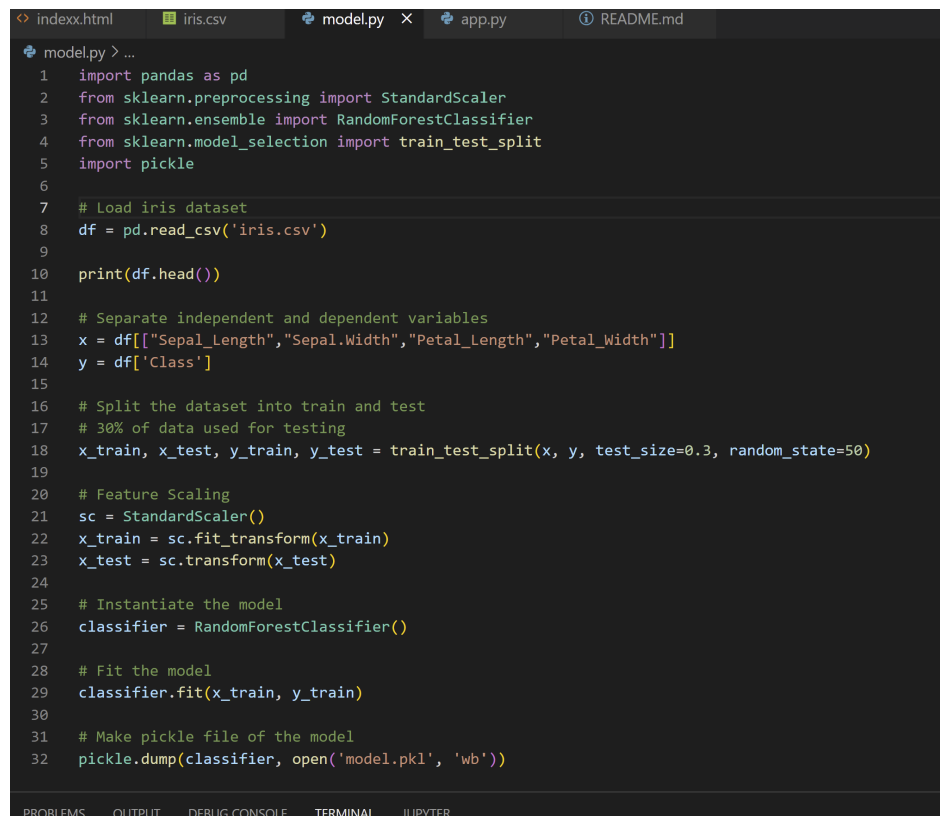
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

2 files changed, 6 insertions(+), 6 deletions(-)

C:\Users\nunto\Flask\model-deployment\Flask-Model-Deployment>git push  
Enumerating objects: 13, done.  
Counting objects: 100% (13/13), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (7/7), done.  
Writing objects: 100% (9/9), 881 bytes | 440.00 KiB/s, done.  
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (5/5), completed with 3 local objects.  
To https://github.com:parcheesine/Flask-Model-Deployment.git  
336fb77..4e65de7 main -> main

C:\Users\nunto\Flask\model-deployment\Flask-Model-Deployment>]

## model.py

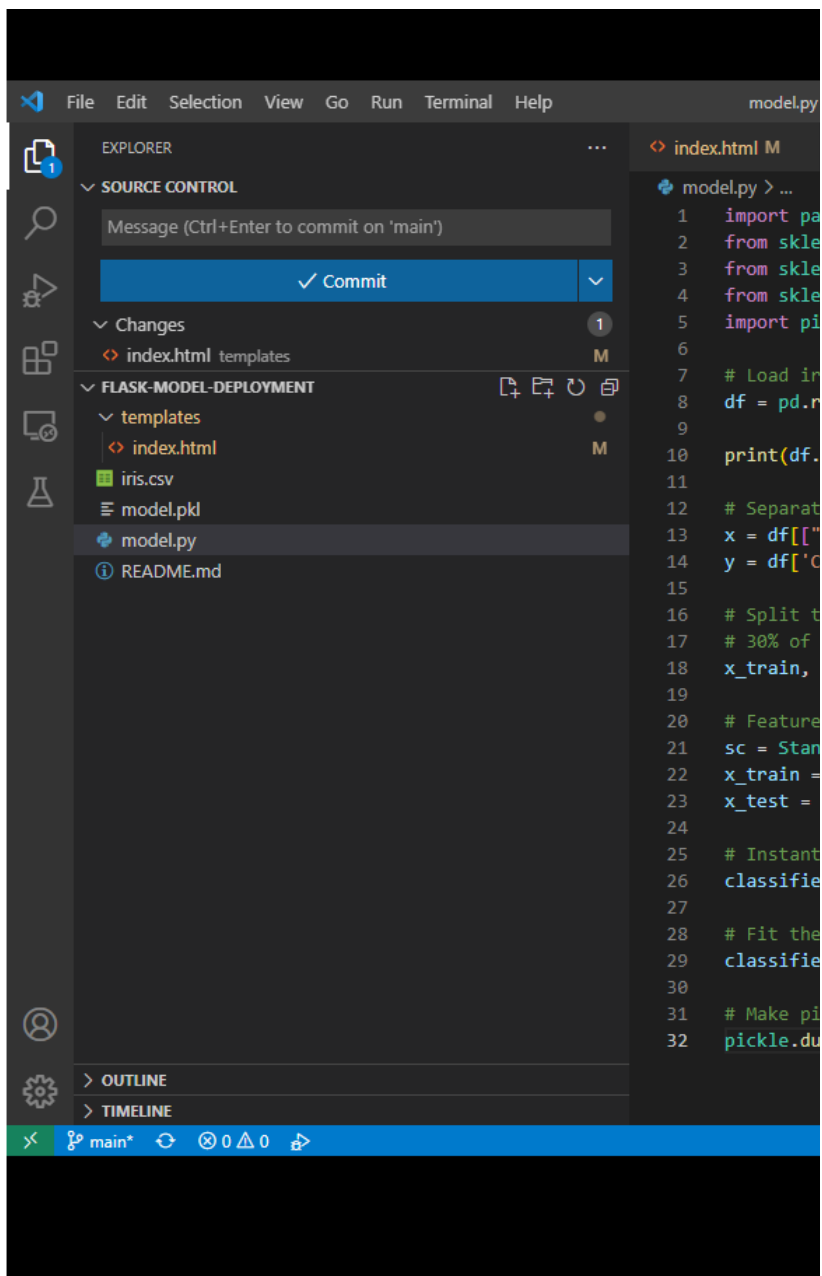


The screenshot shows the VS Code editor with the `model.py` file open. The file contains Python code for loading the Iris dataset, splitting it into training and testing sets, scaling the features, and training a Random Forest Classifier. The code also includes a step to save the trained model as a pickle file.

```
1 import pandas as pd
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.model_selection import train_test_split
5 import pickle
6
7 # Load iris dataset
8 df = pd.read_csv('iris.csv')
9
10 print(df.head())
11
12 # Separate independent and dependent variables
13 x = df[["Sepal_Length", "Sepal_Width", "Petal_Length", "Petal_Width"]]
14 y = df['Class']
15
16 # Split the dataset into train and test
17 # 30% of data used for testing
18 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=50)
19
20 # Feature Scaling
21 sc = StandardScaler()
22 x_train = sc.fit_transform(x_train)
23 x_test = sc.transform(x_test)
24
25 # Instantiate the model
26 classifier = RandomForestClassifier()
27
28 # Fit the model
29 classifier.fit(x_train, y_train)
30
31 # Make pickle file of the model
32 pickle.dump(classifier, open('model.pkl', 'wb'))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

## Pickle File

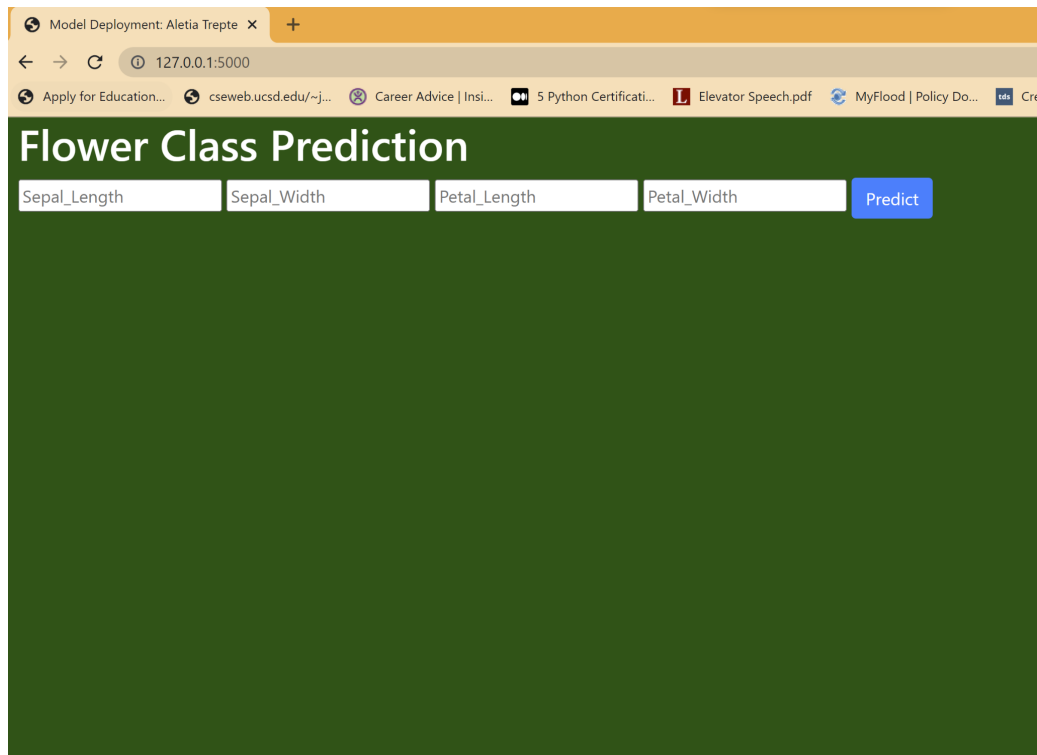


## Deploying the model

app.py

```
index.html  iris.csv  model.py  app.py  README.md
app.py > predict
1 # Credit for help with Model Deployment: https://clarusway.com/model-deployment-with-flask-part-1/
2 # Import Libraries
3 import numpy as np
4 from flask import Flask, request, render_template
5 import pickle
6
7 # create Flask app
8
9 app= Flask(__name__)
10
11 # load Pickle model
12
13 model = pickle.load(open("model.pkl", "rb"))
14
15 # define Home page
16
17 @app.route("/")
18 def Home():
19     return render_template("indexx.html")
20
21 # prediction page
22
23 @app.route("/predict", methods=["POST"])
24 def predict():
25     float_features = [float(x) for x in request.form.values()]
26     features = [np.array(float_features)]
27     prediction = model.predict(features)
28
29     return render_template("indexx.html", prediction_text="The flower species is {}".format(prediction))
30
31 if __name__ == "__main__":
32     app.run(debug=True)
```

## Browser



Run app.py

```
10
11 # load Pickle model
12
13 model = pickle.load(open("model.pkl", "rb"))
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

336fb77..4e65de7 main -> main

C:\Users\nunto\Flask\model-deployment\Flask-Model-Deployment>python app.py

- \* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
- \* Restarting with stat
- \* Debugger is active!
- \* Debugger PIN: 115-590-296

█

Web page output

Model Deployment: Aletia Trepte x +

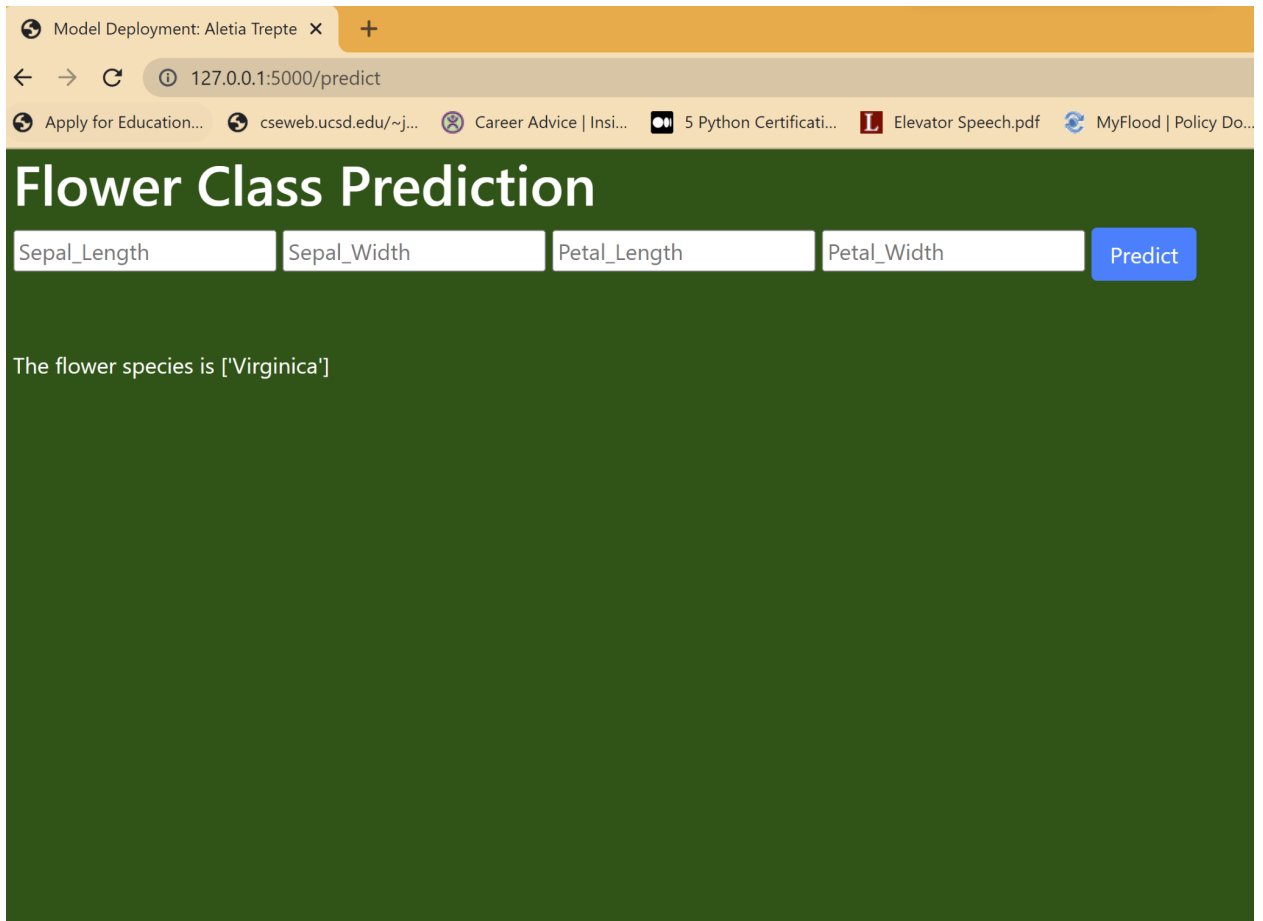
← → ↻ 127.0.0.1:5000

Apply for Education... cseweb.ucsd.edu/~j... Career Advice | Insi... 5 Python Certificati... Elevator Speech.pdf MyFlood | Policy Do... info Cre

## Flower Class Prediction

<input type="text" value="Sepal_Length"/>	<input type="text" value="Sepal_Width"/>	<input type="text" value="Petal_Length"/>	<input type="text" value="Petal_Width"/>	<input type="button" value="Predict"/>
---	--	---	--	--

Get prediction



The screenshot shows a web browser window with a single tab titled "Model Deployment: Aletia Trepte". The address bar displays "127.0.0.1:5000/predict". The browser's bookmark bar contains several links, including "Apply for Education...", "cseweb.ucsd.edu/~j...", "Career Advice | Insi...", "5 Python Certificati...", "Elevator Speech.pdf", and "MyFlood | Policy Do...". The main content area has a dark green background. At the top, the heading "Flower Class Prediction" is displayed in white. Below the heading, there are four white input fields labeled "Sepal\_Length", "Sepal\_Width", "Petal\_Length", and "Petal\_Width". To the right of these fields is a blue button labeled "Predict". Below the input fields, a white text label states "The flower species is ['Virginica']".

# Flower Class Prediction

Sepal\_Length Sepal\_Width Petal\_Length Petal\_Width Predict

The flower species is ['Virginica']