



**Data Glacier**

Your Deep Learning Partner

# CLOUD AND API DEPLOYMENT

NAME - JOHN ESHO

BATCH CODE - LISUM12

DATE - 12/09/2022

SUBMITTED TO - DATA GLACIER

# DATA INFORMATION

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa

# INDEX.HTML

```
1 <!DOCTYPE html>
2 <html >
3 <!--From https://codepen.io/frytyler/pen/EGdtg-->
4 <head>
5   <meta charset="UTF-8">
6   <title>ML API</title>
7   <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
8   <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
9   <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
10  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
11
12 </head>
13
14 <body>
15   <div class="login">
16     <h1>Flower Class Prediction</h1>
17
18     <!-- Main Input For Receiving Query to our ML -->
19     <form action="{{ url_for('predict')}}"method="post">
20       <input type="text" name="Sepal_Length" placeholder="Sepal_Length" required="required" />
21       <input type="text" name="Sepal_Width" placeholder="Sepal_Width" required="required" />
22       <input type="text" name="Petal_Length" placeholder="Petal_Length" required="required" />
23       <input type="text" name="Petal_Width" placeholder="Petal_Width" required="required" />
24
25       <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
26     </form>
27
28     <br>
29     <br>
30     {{ prediction_text }}
31
32   </div>
33
34
35 </body>
36 </html>
```

# BUILD MODEL

- import libraries
- load dataset
- select independent and dependent variables
- split the dataset
- feature scaling
- instantiate model
- fit model
- create pickle file

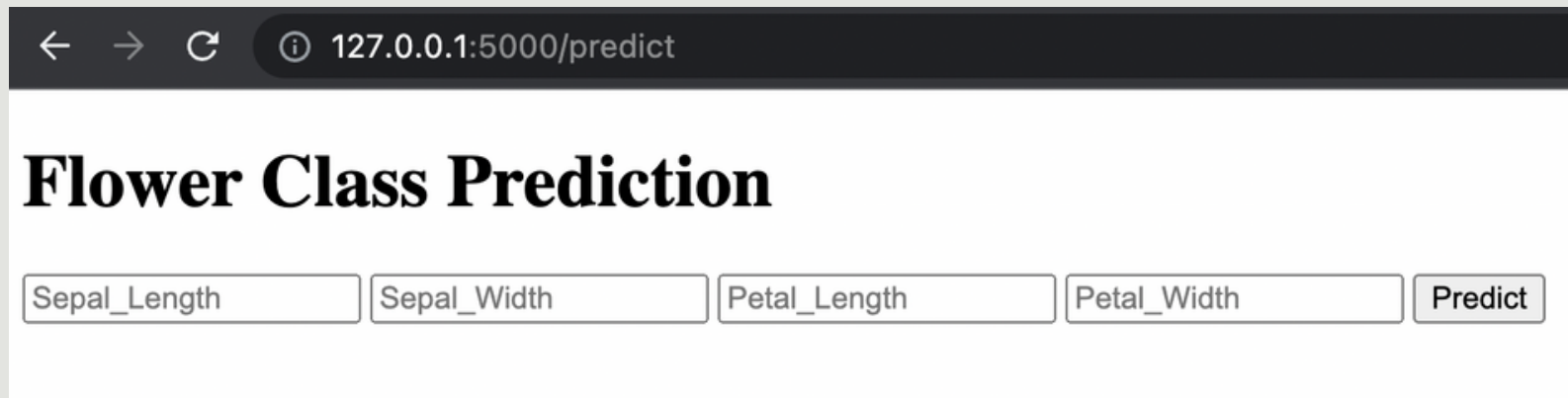
```
1 # import libraries
2 import pickle
3 import pandas as pd
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.ensemble import RandomForestClassifier
6 from sklearn.model_selection import train_test_split
7
8 # load the dataset
9 df = pd.read_csv("Iris.csv")
10 # print(df)
11
12 # select independent and dependent variables
13 X = df[["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm"]]
14 y = df["Species"]
15
16
17 # split the dataset
18 X_train, X_test, y_train, y_test = train_test_split(
19     X, y, test_size=0.3, random_state=42)
20
21 # feature scaling
22 sc = StandardScaler()
23 X_train = sc.fit_transform(X_train)
24 X_test = sc.fit_transform(X_test)
25
26 # instantiate model
27 classifier = RandomForestClassifier()
28
29 # fit model
30 classifier.fit(X_train, y_train)
31
32
33 # create pickle file
34 pickle.dump(classifier, open("model.pkl", "wb"))
```

# CREATE APP

- create flask app
- load pickle model

```
1 # import libraries
2 import pickle
3 import numpy as np
4 from flask import Flask, request, jsonify, render_template
5
6 # create flask app
7 app = Flask(__name__)
8
9 # load the pickle model
10 model = pickle.load(open("model.pkl", "rb"))
11
12 @app.route("/")
13 def Home():
14     return render_template("index.html")
15
16 @app.route("/predict", methods = ["POST"])
17 def predict():
18     float_features = [float(x) for x in request.form.values()]
19     features = [np.array(float_features)]
20     prediction = model.predict(features)
21
22     return render_template("index.html", prediction_text = "The flower species is {}".format(prediction))
23
24 if __name__ == "__main__":
25     app.run(debug=True)
```

# WEB APP



The screenshot shows a web browser window with a dark address bar. The address bar contains navigation icons (back, forward, refresh) and the URL `127.0.0.1:5000/predict`. The main content area has a white background and features the title "Flower Class Prediction" in a large, bold, black serif font. Below the title, there is a horizontal row of four text input fields labeled "Sepal\_Length", "Sepal\_Width", "Petal\_Length", and "Petal\_Width". To the right of these fields is a button labeled "Predict".

← → ↻ ⓘ 127.0.0.1:5000/predict


## Flower Class Prediction

# CREATE PROCFILE



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

# REQUIREMENTS.TXT



```
1  Flask==2.1.2
2  gunicorn==20.1.0
3  itsdangerous==2.1.2
4  Jinja2==3.1.2
5  MarkupSafe==2.1.1
6  Werkzeug==2.1.2
7  numpy>=1.22.4
8  scipy>=1.8.1
9  scikit-learn>=1.1.1
10 matplotlib>=3.5.2
11 pandas>=1.4.2
```

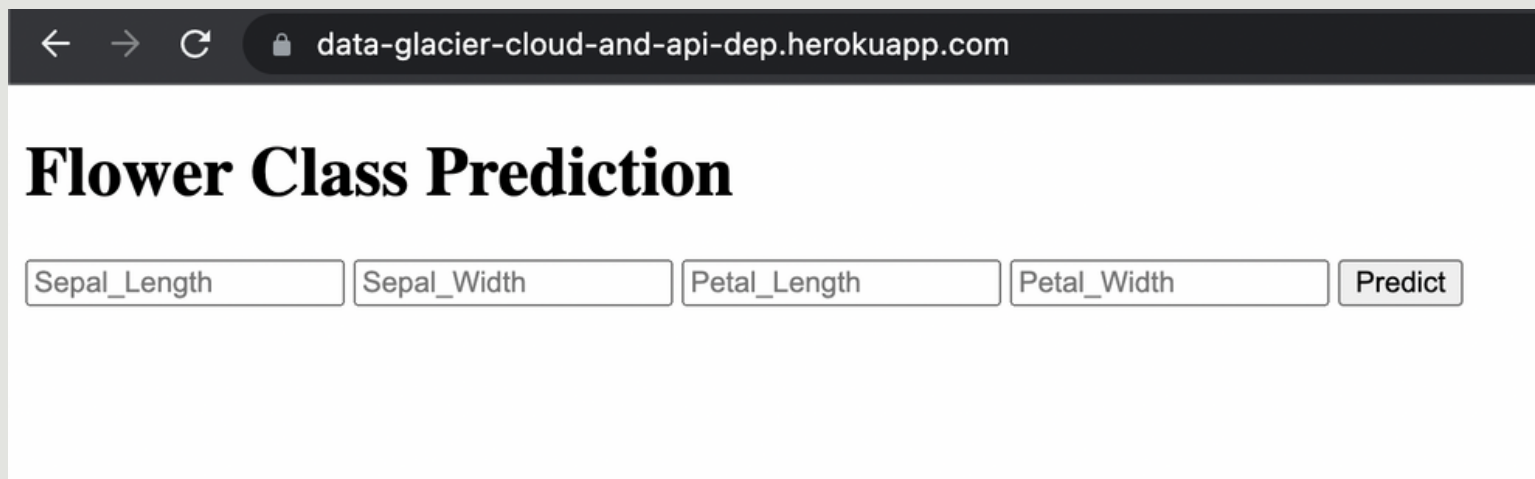


# DEPLOY ON HEROKU

- create new app
- connect to github
- deploy branch

The screenshot shows the Heroku dashboard for an app named "data-glacier-cloud-and-api-dep". The interface includes a navigation bar with "Overview", "Resources", "Deploy", "Metrics", "Activity", "Access", and "Settings". The "Deploy" tab is active, showing options to add the app to a pipeline or stage. Below this, the "Deployment method" section shows "Heroku Git" as the selected method, with "GitHub Connected" status. The "App connected to GitHub" section shows the app is connected to the "johnesho/data-glacier-cloud-and-api-deployment" repository. The "Automatic deploys" section shows the "main" branch selected for automatic deployment. The "Manual deploy" section shows the "main" branch selected for manual deployment. The "Deploy a GitHub branch" section shows the "main" branch selected for deployment. The "Receive code from GitHub" section shows the build ID "355cf800". The "Build main" section shows the release phase. The "Release phase" section shows the deployment status. The "Deploy to Heroku" section shows the deployment status. The "Your app was successfully deployed." message is displayed at the bottom.

# APP ON HEROKU



A screenshot of a web browser displaying a web application. The browser's address bar shows the URL `data-glacier-cloud-and-api-dep.herokuapp.com`. The page title is "Flower Class Prediction". Below the title, there is a form with four input fields labeled "Sepal\_Length", "Sepal\_Width", "Petal\_Length", and "Petal\_Width", followed by a "Predict" button.

← → ↻ 🔒 data-glacier-cloud-and-api-dep.herokuapp.com

## Flower Class Prediction