

# Week 5: Cloud and API deployment

Name, Batch code: LISUM10: 30

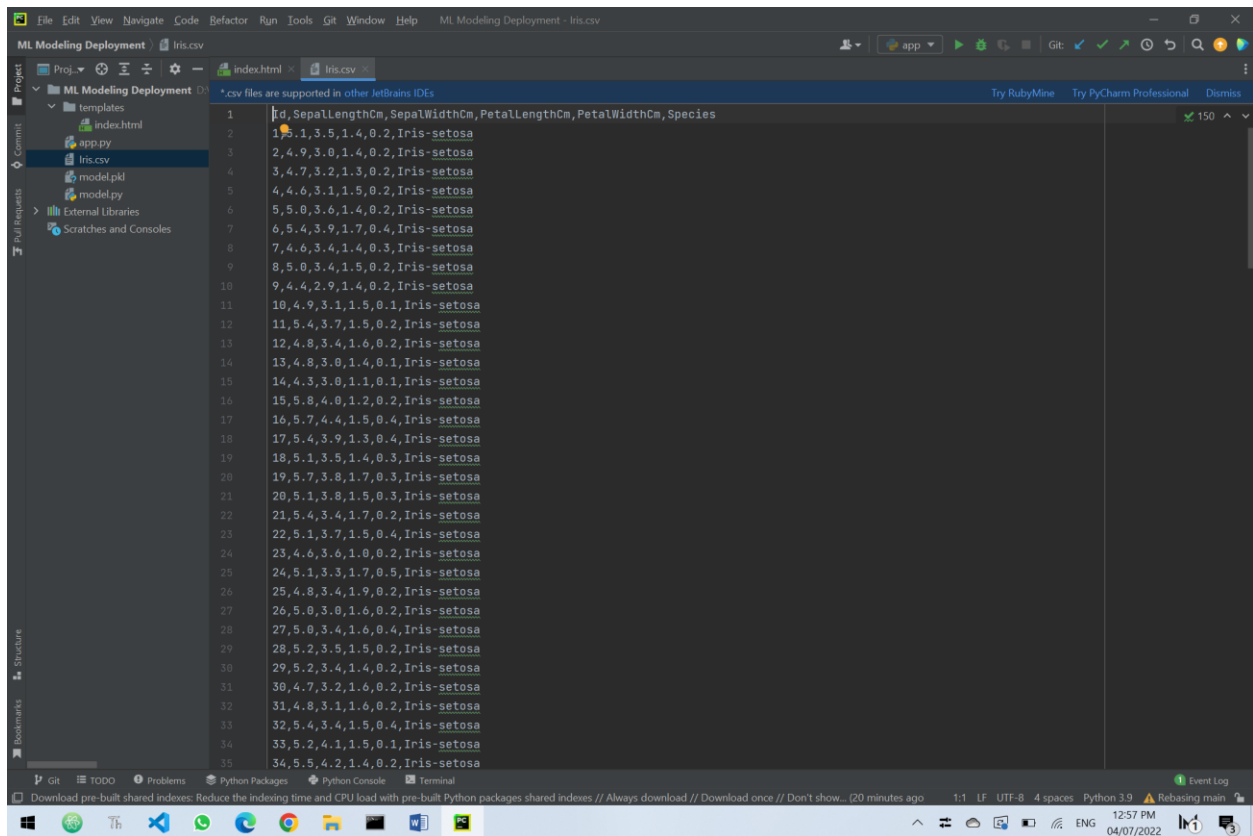
Submission date: 27/07/2022

Submitted to : Data Glacier

Done by: Shaimaa Al-khawlani

## Snapshot of each step of deployment:

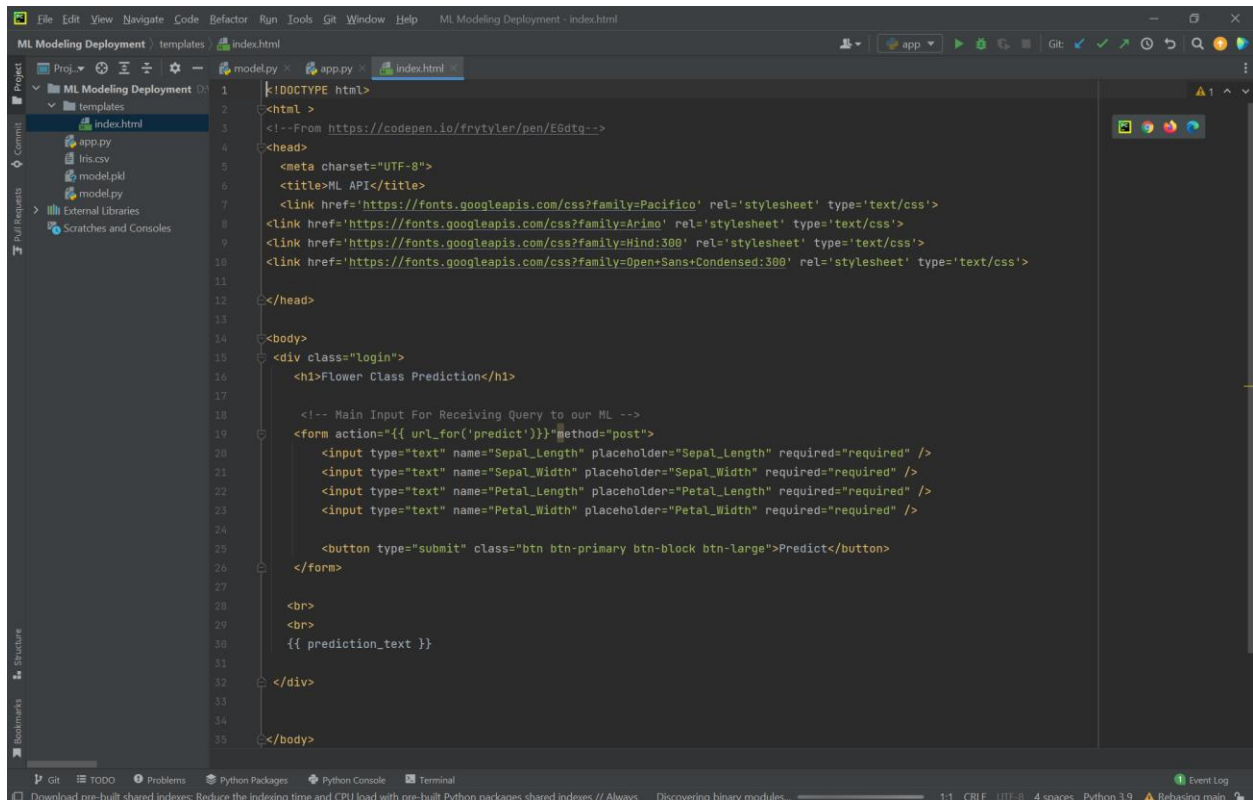
### 1. Download toy data.



The screenshot shows an IDE window titled "ML Modeling Deployment - Iris.csv". The left sidebar displays a project structure with files: index.html, app.py, Iris.csv, model.pkl, and model.py. The main editor area shows the content of Iris.csv, which is a CSV file containing 150 rows of data. The first row is the header: "Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species". The subsequent rows contain numerical values for the first five features, followed by the species name "Iris-setosa". The status bar at the bottom indicates the file encoding is UTF-8, the line length is 150, and the current cursor position is at line 1, column 1.

```
1 Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, Species
2 1,5.1,3.5,1.4,0.2,Iris-setosa
3 2,4.9,3.0,1.4,0.2,Iris-setosa
4 3,4.7,3.2,1.3,0.2,Iris-setosa
5 4,4.6,3.1,1.5,0.2,Iris-setosa
6 5,5.0,3.6,1.4,0.2,Iris-setosa
7 6,5.4,3.9,1.7,0.4,Iris-setosa
8 7,4.6,3.4,1.4,0.3,Iris-setosa
9 8,5.0,3.4,1.5,0.2,Iris-setosa
10 9,4.4,2.9,1.4,0.2,Iris-setosa
11 10,4.9,3.1,1.5,0.1,Iris-setosa
12 11,5.4,3.7,1.5,0.2,Iris-setosa
13 12,4.8,3.4,1.6,0.2,Iris-setosa
14 13,4.8,3.0,1.4,0.1,Iris-setosa
15 14,4.3,3.0,1.1,0.1,Iris-setosa
16 15,5.8,4.0,1.2,0.2,Iris-setosa
17 16,5.7,4.4,1.5,0.4,Iris-setosa
18 17,5.4,3.9,1.3,0.4,Iris-setosa
19 18,5.1,3.5,1.4,0.3,Iris-setosa
20 19,5.7,3.8,1.7,0.3,Iris-setosa
21 20,5.1,3.8,1.5,0.3,Iris-setosa
22 21,5.4,3.4,1.7,0.2,Iris-setosa
23 22,5.1,3.7,1.5,0.4,Iris-setosa
24 23,4.6,3.6,1.0,0.2,Iris-setosa
25 24,5.1,3.3,1.7,0.5,Iris-setosa
26 25,4.8,3.4,1.9,0.2,Iris-setosa
27 26,5.0,3.0,1.6,0.2,Iris-setosa
28 27,5.0,3.4,1.6,0.4,Iris-setosa
29 28,5.2,3.5,1.5,0.2,Iris-setosa
30 29,5.2,3.4,1.4,0.2,Iris-setosa
31 30,4.7,3.2,1.6,0.2,Iris-setosa
32 31,4.8,3.1,1.6,0.2,Iris-setosa
33 32,5.4,3.4,1.5,0.4,Iris-setosa
34 33,5.2,4.1,1.5,0.1,Iris-setosa
35 34,5.5,4.2,1.4,0.2,Iris-setosa
```

## 2. Create HTML index page which contains the input texts of each feature in the toy data.



The screenshot shows a code editor with a dark theme. The file explorer on the left shows a project named 'ML Modeling Deployment' with a 'templates' folder containing 'index.html'. The main editor area displays the content of 'index.html'. The code is an HTML document with a head section including a meta charset, a title 'ML API', and four Google Fonts links (Pacifico, Arimo, Hind:300, and Open+Sans+Condensed:300). The body section has a 'login' class and a heading 'Flower Class Prediction'. Below the heading is a form for receiving queries to the ML model. The form has four text inputs for 'Sepal\_Length', 'Sepal\_Width', 'Petal\_Length', and 'Petal\_Width', each with a placeholder and 'required' attribute. A 'submit' button labeled 'Predict' is at the bottom of the form. Below the form are two blank lines and a placeholder for 'prediction\_text'.

```
1 <!DOCTYPE html>
2 <html>
3 <!-- From https://codepen.io/frtytyler/pen/E6dtg-->
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>
```

## 3. Create the model.py

- Read CSV file.
- Select independent and dependent variables.
- Split dataset into train and test
- Feature scaling.
- Instantiate the model.
- Fit the model
- Create pickle file.

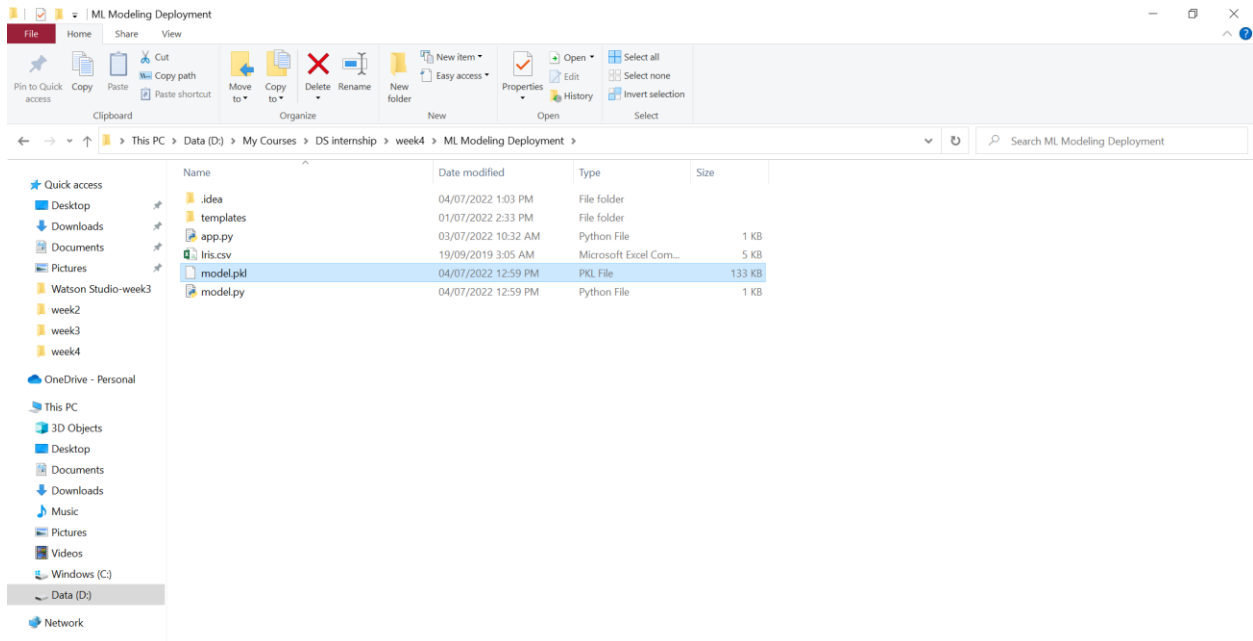
The screenshot displays a Jupyter Notebook environment with a dark theme. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, and Help. The title bar indicates the current file is 'ML Modeling Deployment - model.py'. The left sidebar shows the Project Explorer with a tree view of the file structure: 'ML Modeling Deployment' (containing 'templates', 'index.html', 'app.py', 'iris.csv', 'model.pkl', and 'model.py'), 'External Libraries', and 'Scratches and Consoles'. The 'model.py' file is selected and open in the main editor. The code in the notebook is as follows:

```
4 from sklearn.model_selection import train_test_split
5 import pickle
6
7 # read the csv file
8
9 iris_df = pd.read_csv("iris.csv")
10 print(iris_df.head())
11
12 # select independent and dependent variable
13 x = iris_df[["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm"]]
14 y = iris_df["Species"]
15
16 # split the dataset into train and test
17 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=58)
18
19 # feature scaling
20 sc = StandardScaler()
21 x_train = sc.fit_transform(x_train)
22 x_test = sc.transform(x_test)
23
24 # instantiate the model
25 classifier = RandomForestClassifier()
26
27 # fit the model
28 classifier.fit(x_train, y_train)
29
30 # make pickle file of our model
31 pickle.dump(classifier, open("model.pkl", "wb"))
```

Below the code editor, the 'Run' output shows the execution of the script, displaying the first five rows of the Iris dataset:

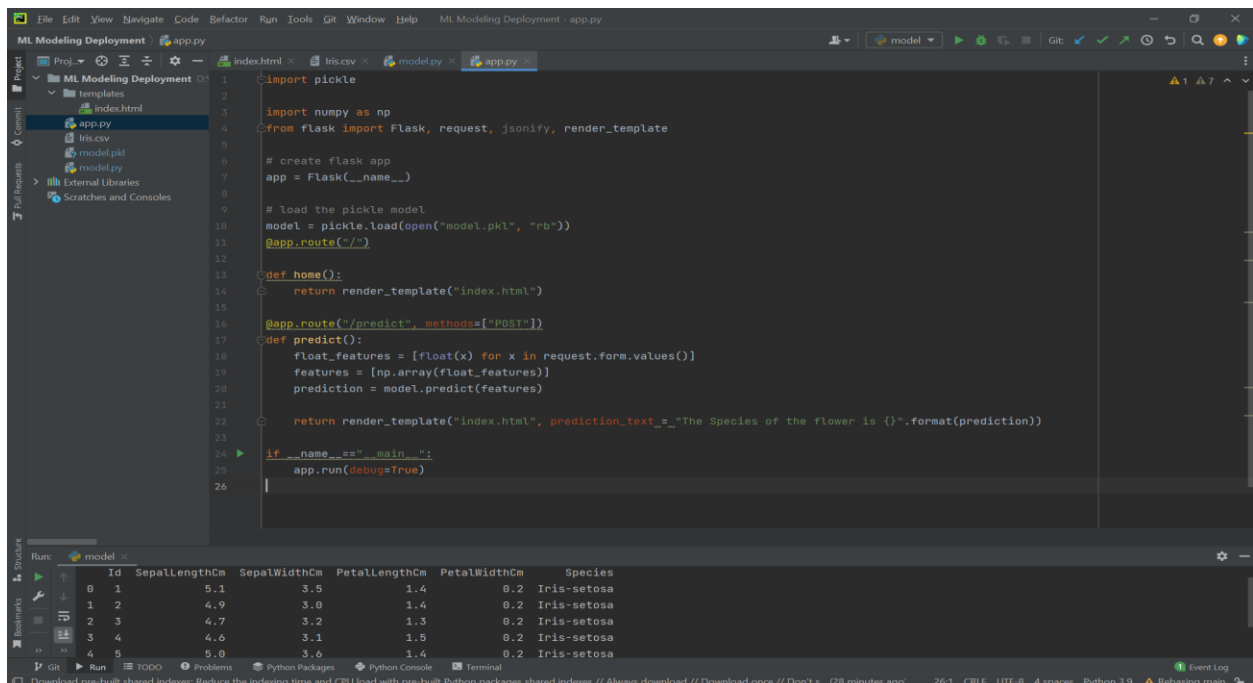
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

The bottom status bar shows icons for Git, Run, TODO, Problems, Python Packages, Python Console, Terminal, and Event Log.



#### 4. Create app.py

- Create flask app
- Load pickle model





## Flower Class Prediction

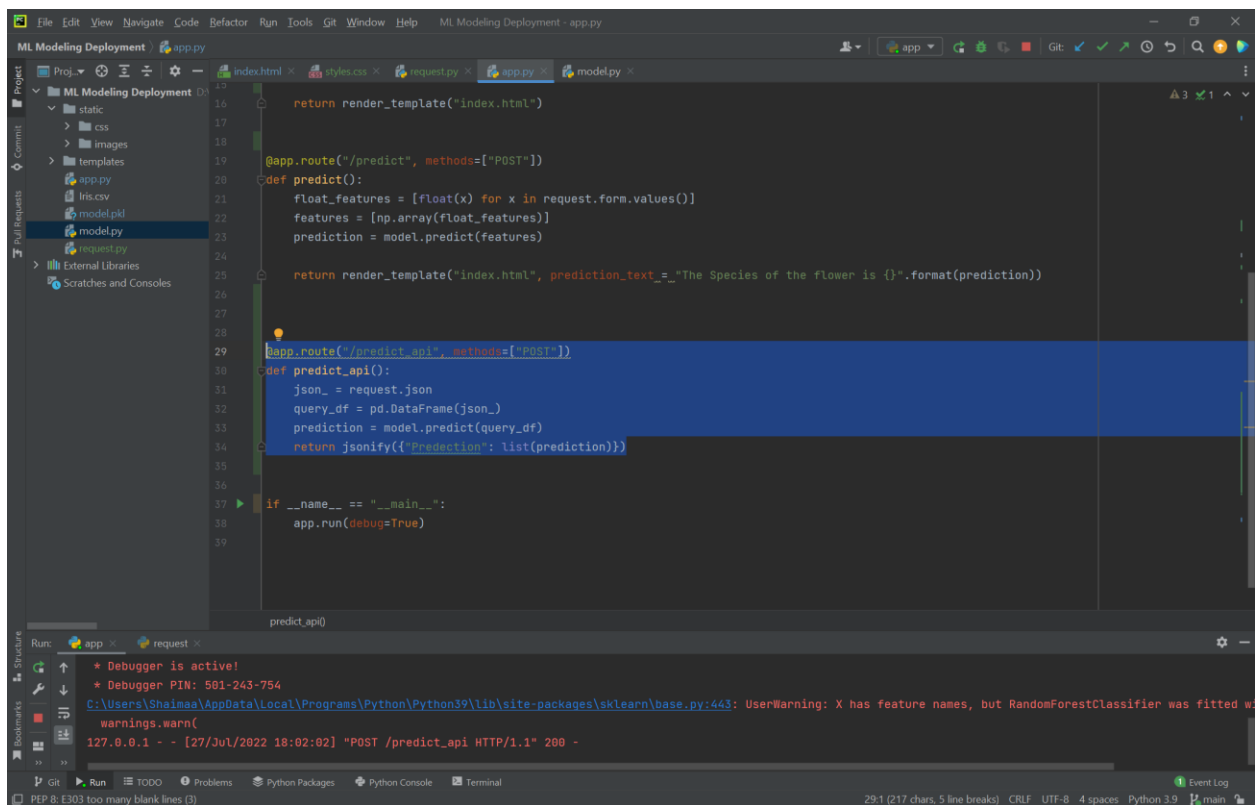
Sepal_Length
Sepal_Width
Petal_Length
Petal_Width
Predict

The Species of the flower is ['Iris-virginica']

# API deployment

## 1. Define API Method

```
@app.route("/predict_api", methods=["POST"])
def predict_api():
    json_ = request.json
    query_df = pd.DataFrame(json_)
    prediction = model.predict(query_df)
    return jsonify({"Prededction": list(prediction)})
```



## 2. Test API With Postman

The screenshot displays the Postman application interface. The top bar includes the Postman logo, menu options (File, Edit, View, Help), and navigation links (Home, Workspaces, Reports, Explore). A search bar and utility buttons (Invite, Upgrade) are also present.

The left sidebar shows the 'My Workspace' section with a 'New Collection' button. Below it, a list of collections is visible, including 'Iris model test'.

The main workspace area shows a REST client setup for a POST request to `localhost:5000/predict_api`. The request body is a JSON array of two objects:

```
[{"SepalLengthCm":0, "SepalWidthCm":0, "PetalLengthCm":0, "PetalWidthCm":0}, {"SepalLengthCm":3, "SepalWidthCm":5, "PetalLengthCm":1.2, "PetalWidthCm":1}]
```

The response is displayed in the 'Body' tab, showing a JSON object with a 'Prededction' (sic) field:

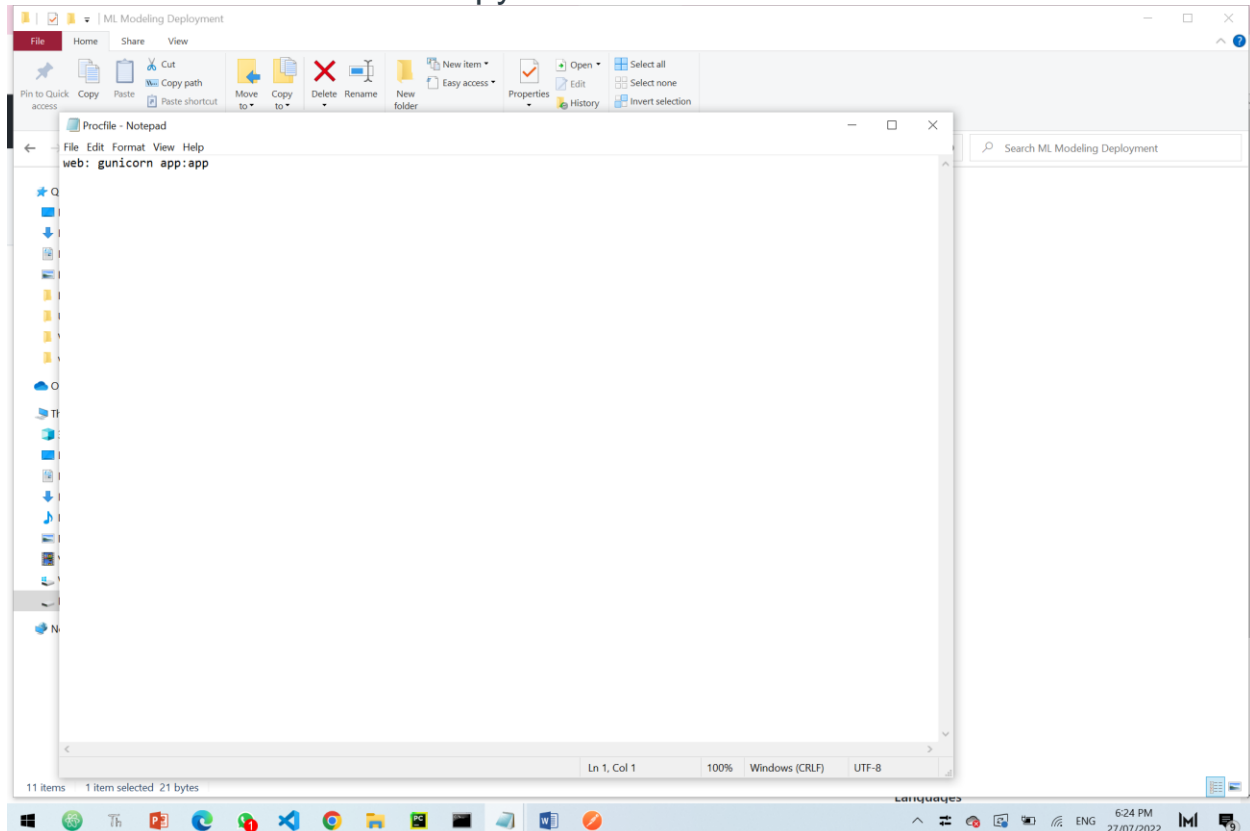
```
{ "Prededction": [ "Iris-virginica", "Iris-virginica" ] }
```

The status bar at the bottom indicates the request was successful with a status of 200 OK, a time of 533 ms, and a size of 234 B. The bottom of the interface includes a 'Find and Replace' bar and a 'Console' panel.

Deploy the model on any open source cloud Heroku

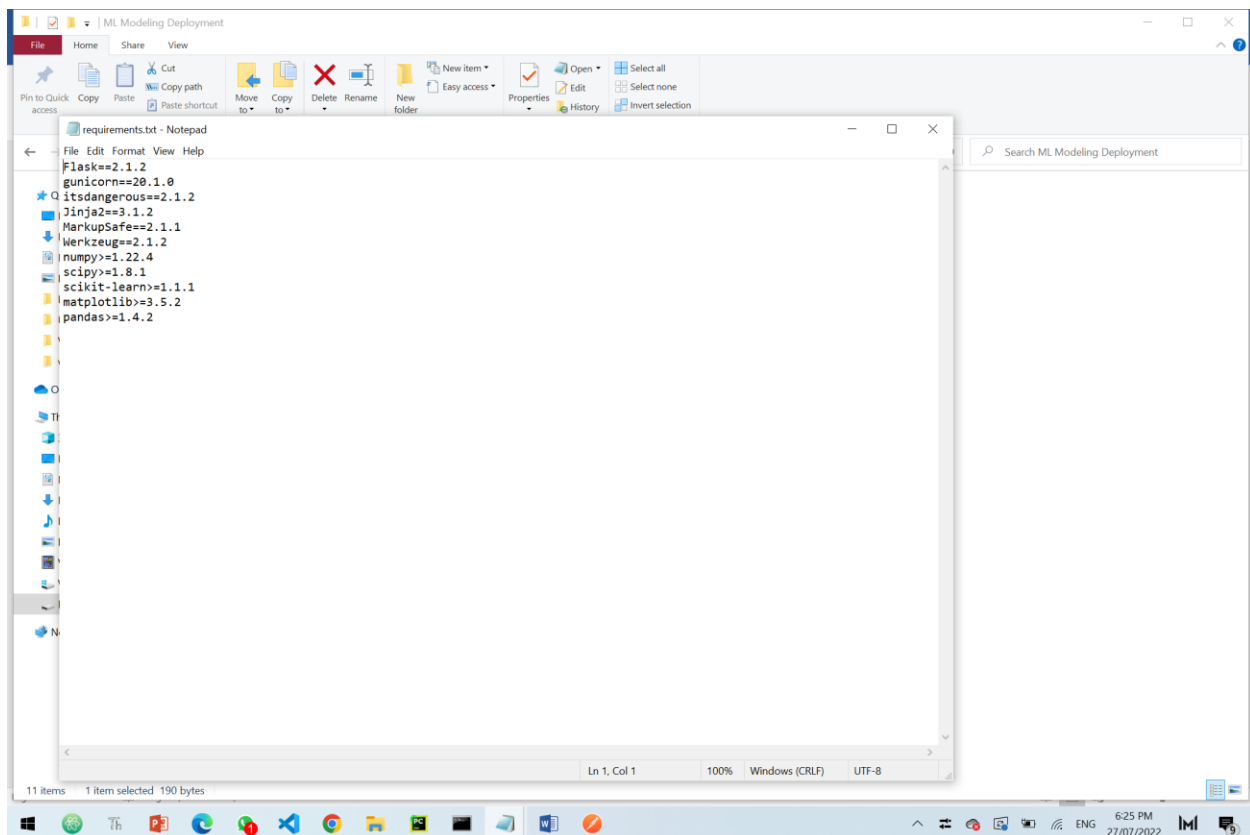
## 1. Create the following files:

- Profile: contains python file that will be executed.

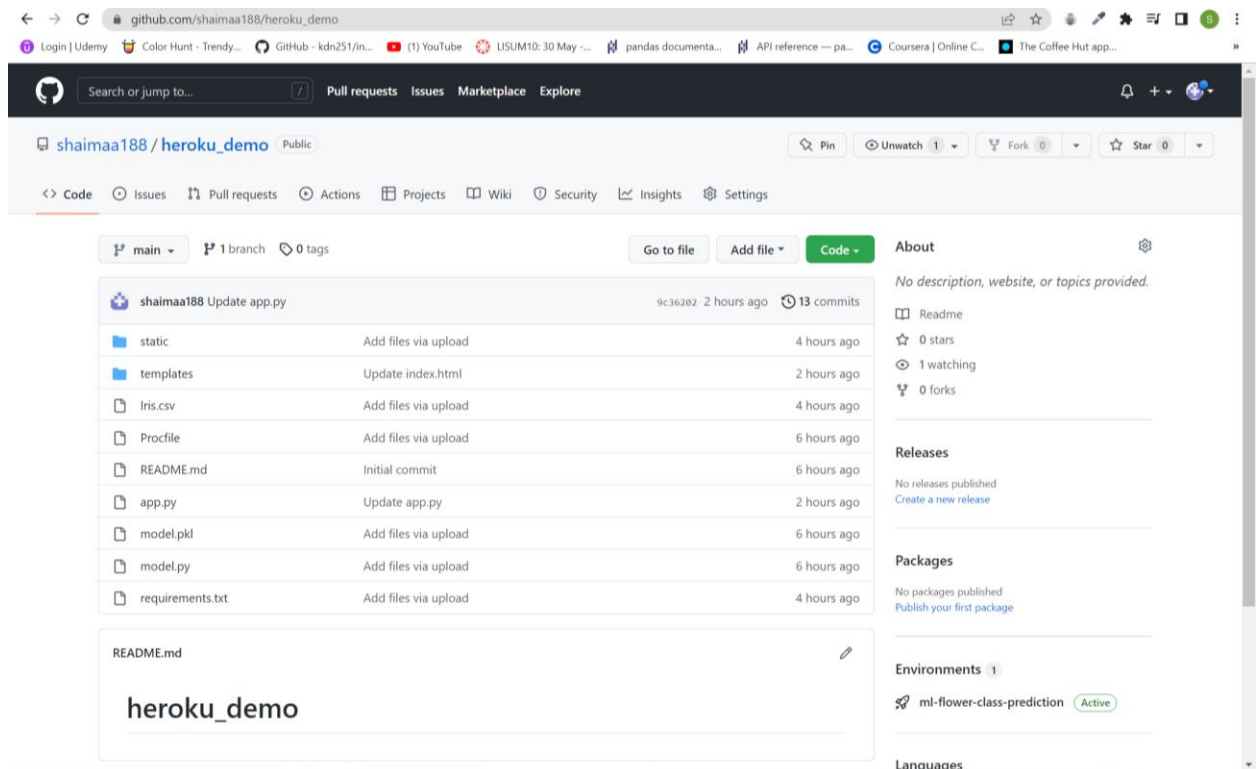


- Requirements: contains the packages installed with versions that will be read by heroku

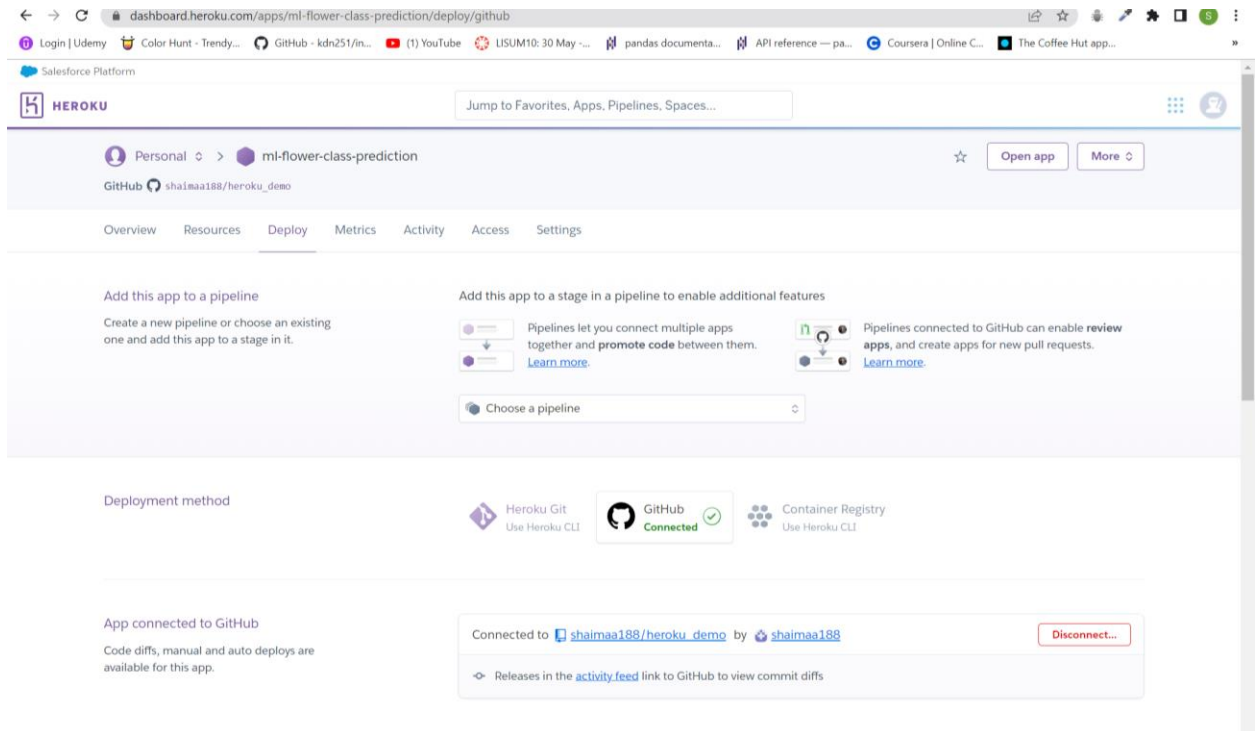




## 2. Upload files of the model in GitHub



### 3. Sign in Heroku website and connect with GitHub and the required repostiry.



### 4. Choose Deploy branch

← → ↻ dashboard.heroku.com/apps/ml-flower-class-prediction/deploy/github

Login | Udem... Color Hunt - Trendy... GitHub - kdn251/in... (1) YouTube LISUM10: 30 May ... pandas documenta... API reference — pa... Coursera | Online C... The Coffee Hut app...

Salesforce Platform

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Enable Automatic Deploys

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

Deploy Branch

Receive code from GitHub ✓

Build main 9c362022

-----> Building on the Heroku-20 stack

-----> Using buildpack: heroku/python

-----> Python app detected

-----> No Python version was specified. Using the same version as the last build: python-3.10.5

-----> To use a different version, see: <https://devcenter.heroku.com/articles/python-runtimes>

-----> No change in requirements detected, installing from cache

-----> Using cached install of python-3.10.5

-----> Installing pip 22.1.2, setuptools 60.10.0 and wheel 0.37.1

☒ Autoscroll with output

[View build log](#)

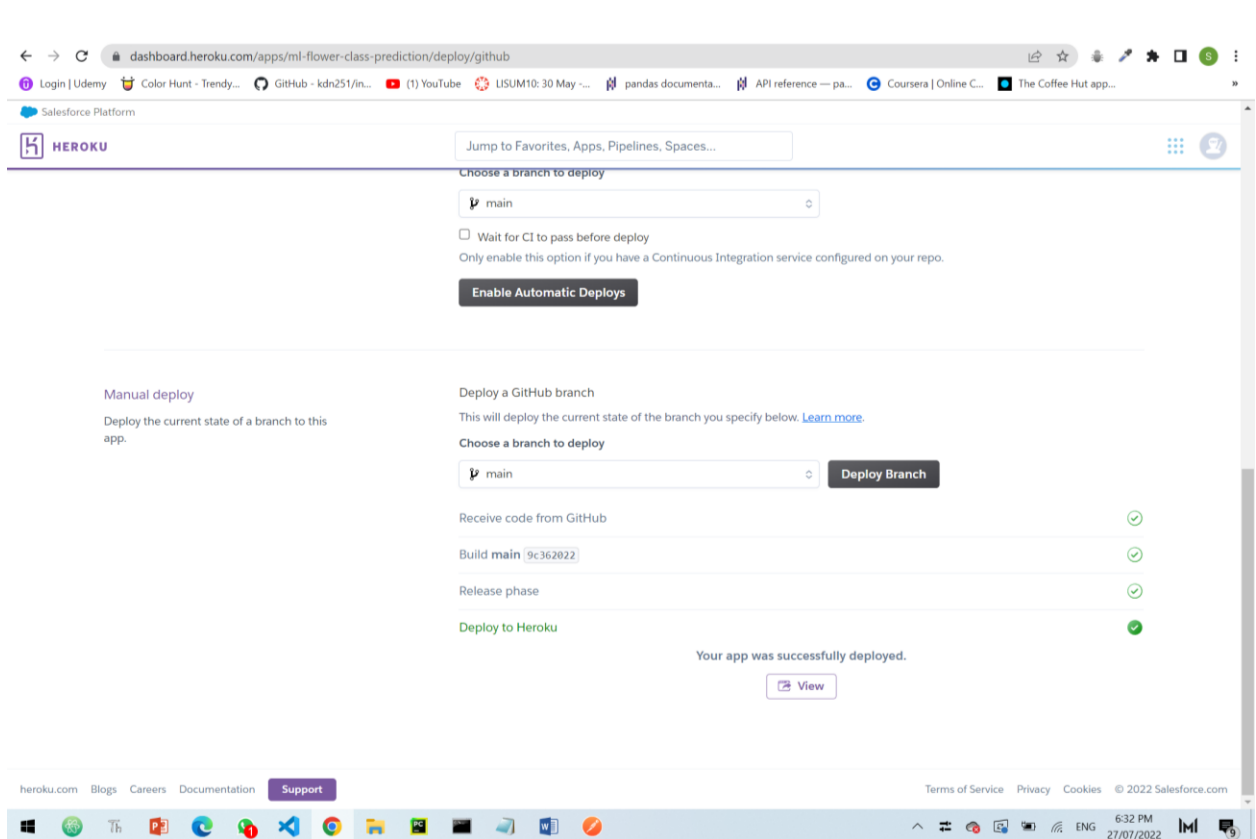
Release phase

Deploy to Heroku

heroku.com Blogs Careers Documentation **Support**

Terms of Service Privacy Cookies © 2022 Salesforce.com

Ready




5. Access the web page with below link

<https://ml-flower-class-prediction.herokuapp.com/>

Week 5: Cloud and API deployment | ml-flower-class-prediction - GitHub | Flower Class ML | shaimaa188/heroku\_demo

ml-flower-class-prediction.herokuapp.com

Login | Udem... | Color Hunt - Trendy... | GitHub - kdn251/in... | (1) YouTube | LISUM10: 30 May ~... | pandas documenta... | API reference — pa... | Coursera | Online C... | The Coffee Hut app...


 **Data Glacier**  
Your Deep Learning Partner

## Flower Class Prediction

Sepal_Length
Sepal_Width
Petal_Length
Petal_Width
Predict

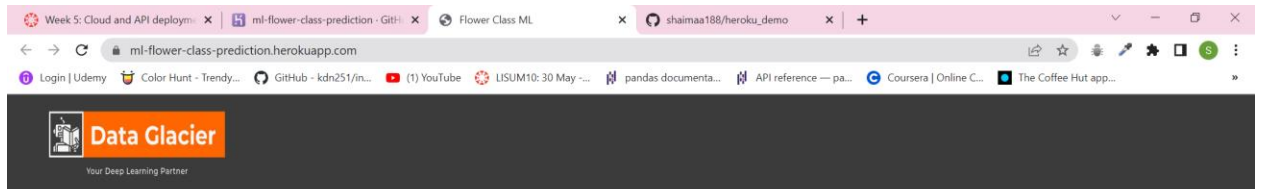
ml-flower-class-prediction.herokuapp.com

Login | Udem... | Color Hunt - Trendy... | GitHub - kdn251/in... | (1) YouTube | LISUM10: 30 May ~... | pandas documenta... | API reference — pa... | Coursera | Online C... | The Coffee Hut app...

 **Data Glacier**  
Your Deep Learning Partner

## Flower Class Prediction

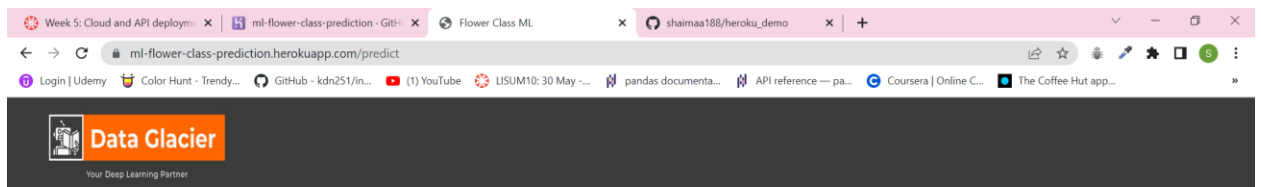
1.2
5.1
0
2.0
Predict



## Flower Class Prediction

1.2
5.1
0
2.0
Predict

The result displayed successfully



## Flower Class Prediction

Sepal_Length
Sepal_Width
Petal_Length
Petal_Width
Predict

The Species of the flower is ['Iris-virginica']