

Name: **Mario Rodrigues Peres**  
Batch code: **LISUM19**  
Submission date: **March 26th, 2023**  
Submitted to: **Glacier Data Scientist Internship**

## INTRODUCTION

In this project, a ML model to predict the flower iris species was developed and deployed as a web app and API using the Flask for the app development and Heroku for deployment. All the steps involved in the constructions of the scripts and deploy of model can be found in the following sections.

## STEPS FOR WEB/API FLASK/HEROKU APP DEPLOYMENT (ML MODEL)

### 1. MODEL DEVELOPMENT

Creation of a support vector machine (SVM) model, which was saved as model.pkl. In this first step, the dataset Iris from the Sklearn was used as the toy data to create a ML model using the support vector machine applied to classification. The data consisted of three categories of species of iris flower and four predictors, including the sepal length, sepal width, petal length and petal width.

```
model.py x
1  ##### Development of Predictive Classification Model using SVM for the Iris dataset #####
2
3  # This model was created based on the iris dataset, which is provided with the package sklearn.
4  # There are four variables, including the sepal length, sepal width, petal length and petal width (all in cm) and 150 records.
5  # In total there are three different species, which includes setosa, versicolor and virginica.
6
7  # Load packages
8  import random
9  import pickle
10 import numpy as np
11 import matplotlib.pyplot as plt
12 from sklearn import datasets
13 from sklearn.svm import SVC
14 from sklearn.metrics import accuracy_score
15
16 # Load the diabetes dataset
17 iris_X, iris_y = datasets.load_iris(return_X_y=True)
18
19 # Creating random indexes for the training set
20 train_indexes = random.sample(range(0, 150), 120)
21
22 # Split the data into training/testing sets
23 iris_X_train = iris_X[train_indexes,:]
24 iris_X_test = np.delete(iris_X, train_indexes, 0)
25
26 # Split the targets into training/testing sets
27 iris_y_train = iris_y[train_indexes]
28 iris_y_test = np.delete(iris_y, train_indexes)
29
30 # Create support vector machine (SVC) classifier
31 svn = SVC()
32
33 # Train the model using the training sets
34 svn.fit(iris_X_train, iris_y_train)
35
36 # Make predictions using the testing set
37 iris_y_pred = svn.predict(iris_X_test)
38
39
40 accuracy = accuracy_score(iris_y_test, iris_y_pred)
41
42 # The accuracy of the model
43 print("Accuracy of Model: \n", accuracy)
44
45 # Saving the model using pickle
46 with open("model.pkl", "wb") as f:
47     pickle.dump(svn, f)
```

The dataset was loaded and randomly separated into train and test datasets. Following that, the SVC function was used to create an instance of the model, which was fitted with the train data. After training, the model was tested with the test data and the accuracy calculated (94.6%). Finally, the model was serialized and saved as “model.pkl” to be used for the app development.

## 2. APP/API DEVELOPMENT USING FLASK

### 2.1. Creation of the app.py file

```
1  # Importing required packages
2  import numpy as np
3  import pandas as pd
4  from flask import Flask, request, render_template, jsonify, request
5  import pickle
6
7  # Creating the Flask application
8  app=Flask(__name__)
9
10 # Importing the model previous created
11 model = pickle.load(open("model.pkl", "rb"))
12
13 # Creating the application routes including the home and predict paths
14 @app.route("/")
15 def home():
16     return render_template("index.html")
17
18 @app.route("/predict", methods=["POST"])
19 def predict():
20
21     # For rendering results on HTML GUI
22     features = [[float(x) for x in request.form.values()]]
23     prediction = model.predict(features)
24
25     # Result of the prediction
26     output = round(prediction[0], 1)
27
28     # Creating a text for the flower species
29     if output == 0 :
30         text = "This plant is from the Setosa species"
31     elif output == 1:
32         text = "This plant is from the Versicolor species"
33     elif output == 1:
34         text = "This plant is from the Virginica species"
35     else:
36         text = "This plant is unkown!"
37
38     # Rendering the results of the prediction
39     return render_template("index.html", prediction_text="{}".format(text))
40
41 @app.route('/predict_api/')
42 def price_predict():
43     model = pickle.load(open('model.pkl', 'rb'))
44     sepal_length = request.args.get('sepal_length')
45     sepal_width = request.args.get('sepal_width')
46     petal_length = request.args.get('petal_length')
47     petal_width = request.args.get('petal_width')
48
49     test_df = pd.DataFrame({'Sepal length':[sepal_length], 'Sepal width':[sepal_width], \
50                             'Petal length':[petal_length], 'Petal width':[petal_width]})
51
52     pred = model.predict(test_df)
53
54     if pred == 0:
55         pred_plant = "Setosa"
56     elif pred == 1:
57         pred_plant = "Versicolor"
58     elif pred == 2:
59         pred_plant = "Virginica"
60     else:
61         pred_plant = "Not known"
62
63     return jsonify({'Plant specie': pred_plant})
64
65 # Initializing the application
66 if __name__ == "__main__":
67     app.run(port=5000, debug=True)
```

The “app.py” file is the application itself. An instance of the Flask was created and the model, previously created, was loaded. Following that, the routes for the home and predict pages as well as the route for the use of the API (predict\_api) were set up. As the outcome of the model was numeric, some text showing the name of the species was prepared, so that the predictions return the name of the species. The initialization code was created and the file saved as “app.py”

## 2.2. Creation of the “index.html” script:

```
1  <!DOCTYPE html>
2  <html >
3  <head>
4      <meta charset="/UTF-8">
5      <title>API_SVC_IRIS</title>
6      <style>
7          body {
8              background-color: royalblue;
9          }
10         h1, h3 {
11             color: white;
12             text-align: center;
13         }
14         div {
15             color: white;
16             text-align: center;
17             font-size: 28px
18         }
19         p {
20             color: white;
21             text-align: left;
22             text-indent: 215px;
23             font-family: verdana;
24             font-size: 20px;
25         }
26         input {
27             color: black;
28             text-align: center;
29             font-family: verdana;
30             font-size: 16px;
31         }
32         button {
33             background-color: black;
34             color: white;
35             font-family: verdana;
36             font-size: 20px;
37         }
38     </style>
39 </head>
40 <body>
41     <div>
42         <h1>Predict Plants Species</h1>
43         <h3>Based on sepal length and width, and petal length and width.</h3>
44         <br>
45         <p>Please enter the details of the plant:</p>
46         <!-- Main input for receiving query to our ML -->
47         <form action="{{ url_for('predict') }}" method="POST">
48             <input type="text" name="sepal_lenght" placeholder="Sepal Lenght" required="required"/>
49             <input type="text" name="sepal_width" placeholder="Sepal Width" required="required"/>
50             <input type="text" name="petal_length" placeholder="Petal Length" required="required"/>
51             <input type="text" name="petal_width" placeholder="Petal Width" required="required"/>
52             <br>
53             <br>
54             <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
55         </form>
56         <br>
57         <br>
58         <div>{{ prediction_text }}</div>
59     </div>
60 </body>
61 </html>
```

The index.html file is the front end of our application. A very simple HTML code was developed to show an interface for the user to insert the measurement of the sepal and petal of iris flower. Moreover, it was created a button for the prediction, so that when the button is pressed, the route is directed to the /predict/, and the prediction is displayed right below, in the same page.

A title and some style for the different sections of the web app home page was created. Following that, the headings and a paragraph were inserted, right before a form that was developed, so that the user can input the values for the sepal length, sepal width, petal length and petal width. This file was saved as index.html and placed in the templates folder. In total there were 3 files in the project's folder, including the model.pkl, the app.py and the index.html.

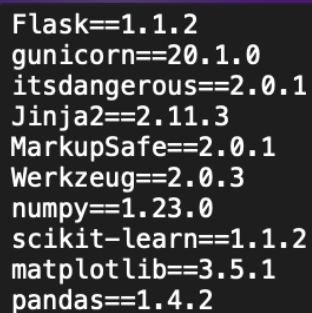
At this point, we could test the App by running it in the local machine. The App was already ready for deployment. The model deployment is shown in the following sections of this document.

### 3. APP/API DEPLOY USING HEROKU

The deployment was performed with Heroku and using the data uploaded to a GitHub account. First of all, two more files were created: requirements.txt and Procfile. These are requirements to deploy the model using Heroku.

#### 3.1. Creation of the requirements.txt:

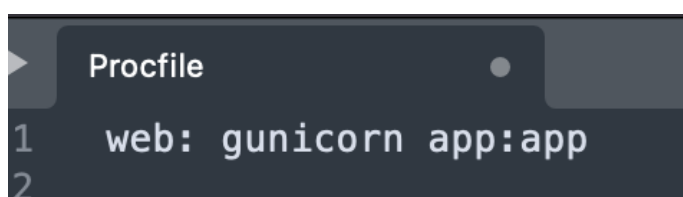
The requirement.txt was create using the package pipreqs, which automatically generates the requirement.txt with the packages used in the project. As there was not all the packages necessary in the produced file, a few more were added, so that the list was complete.

A screenshot of a code editor showing the contents of a requirements.txt file. The text is as follows:

```
Flask==1.1.2
gunicorn==20.1.0
itsdangerous==2.0.1
Jinja2==2.11.3
MarkupSafe==2.0.1
Werkzeug==2.0.3
numpy==1.23.0
scikit-learn==1.1.2
matplotlib==3.5.1
pandas==1.4.2
```

#### 3.2. Creation of the Procfile:

The Procfile is a file that specifies the commands that are executed by an Heroku app on startup. The content of the file is shown in the figure below.

A screenshot of a code editor showing the contents of a Procfile. The text is as follows:

```
Procfile
1 web: gunicorn app:app
2
```

With all the necessary files needed for the deployment of the ML web application were organized in the project folder as shown in the next section.

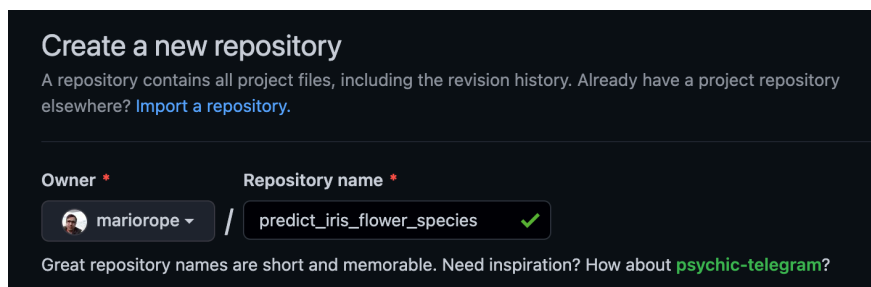
### 3.3. Files and folders organization:

app.py	Today 09:19	2 KB	Python Source
model.pkl	22 March 2023 18:46	4 KB	Document
Procfile	Yesterday 12:13	21 bytes	Document
requirements.txt	Yesterday 12:31	164 bytes	Plain Text
templates	Today 09:31	--	Folder
index.html	Yesterday 15:21	1 KB	HTML text

The directory templates holds the html files, whereas the main directory holds all other files, including the app.py, model.pkl, model.py, Procfile and requirements.txt.

### 3.4. Upload the files to GitHub:

In order to upload the files, we need a GitHub account. Next, we create a new repository to hold the project files.



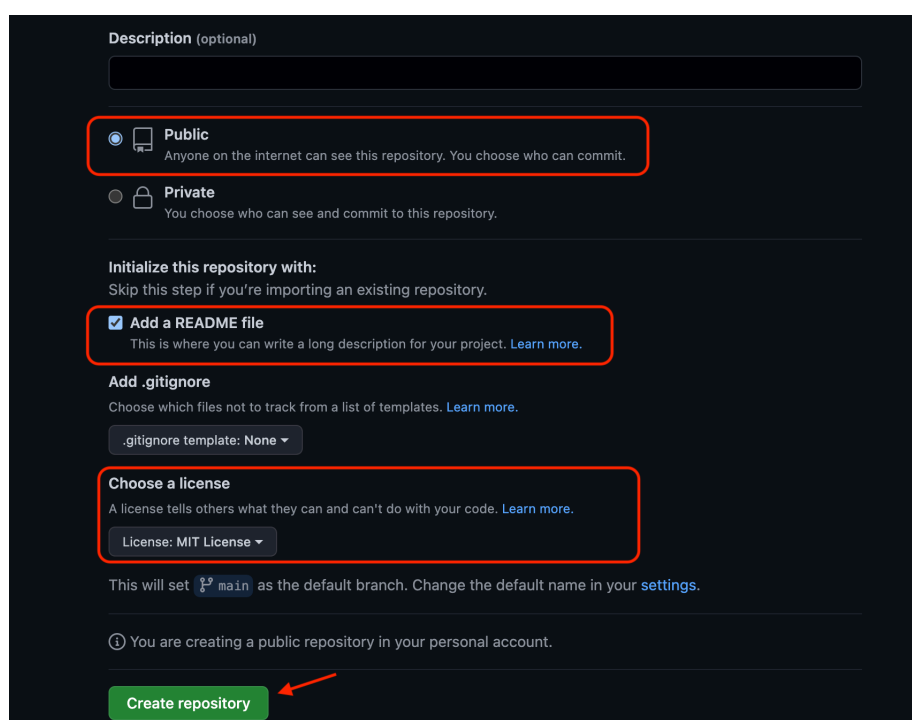
**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner <sup>\*</sup> mariorope / Repository name <sup>\*</sup> predict\_iris\_flower\_species ✓

Great repository names are short and memorable. Need inspiration? How about [psychic-telegram](#)?

We should give a name to the repository (figure above) and select a few options below, which in this case we left it as a public repository, and included the readme file and MIT license as shown in the next figure:



Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License

This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

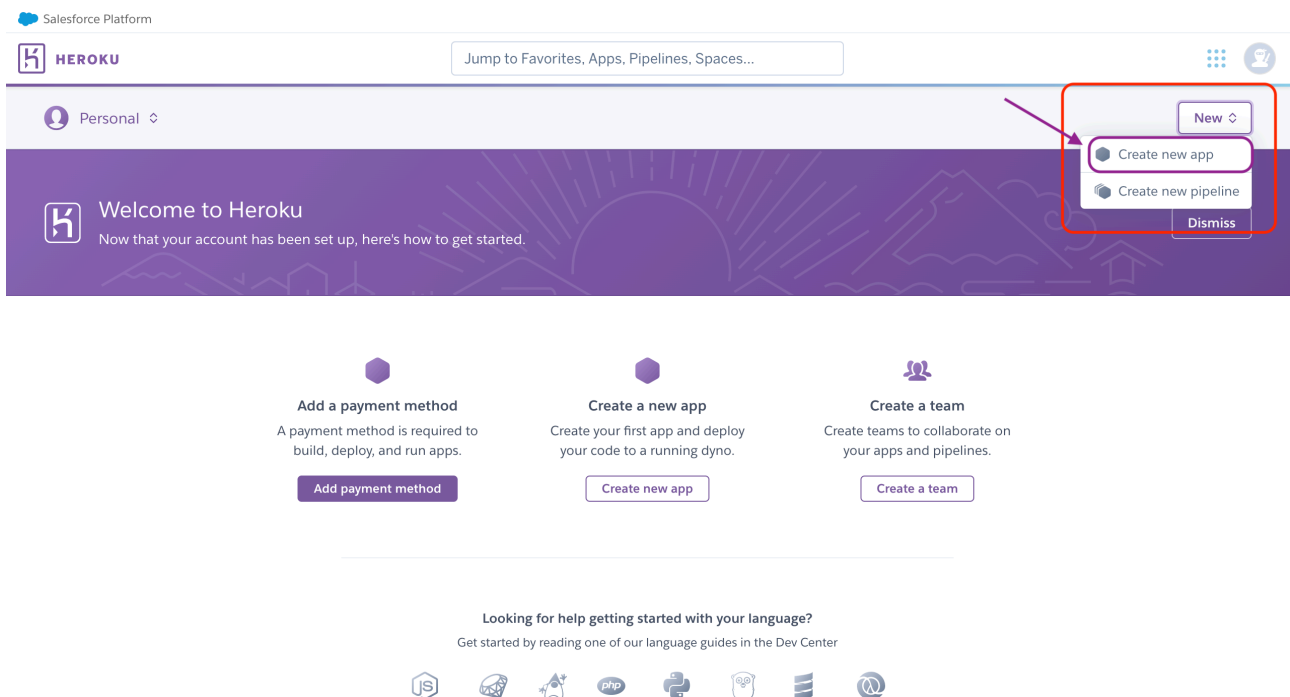
**Create repository**

Finally, after clicking the 'Create repository' icon, a new repository is created. All the files and folds for the app deployment were then uploaded to this repository (next figure)

templates	Add files via upload	now
LICENSE	Initial commit	1 minute ago
Procfile	Add files via upload	now
README.md	Initial commit	1 minute ago
app.py	Add files via upload	now
model.pkl	Add files via upload	now
requirements.txt	Add files via upload	now

### 3.5. Deploy of the ML App/API in Heroku

You need to have an account in Heroku or create a new one. As soon as we login in the Heroku web page, there is an option on the right-top side of the page presented by the icon "NEW" (figure below). Click this button and choose the option "Create new app".



If it is the first time we are logging in, we have to include a credit card, although it is said that the first deployment will not be charged. After inserting the new card details, we can finally go through the steps previously explained. In the new window that opens after clicking to create a new app, we have to insert a name for the app, which in this case was "predict-iris-flower-species", and choose a region, which in this case was left the default option - USA (next picture). The button "Create App" was clicked.

Create New App

App name

predict-iris-flower-species

✓

predict-iris-flower-species is available

Choose a region

United States

⌵

Add to pipeline...

Create app

Cancel

After that, in the following window, in the deployment method tab, the GitHub option was selected. Then Heroku was authorized to connect with the Heroku and the repository that Heroku would access was inserted and the button to search for it clicked (figure below).

Deployment method

Heroku Git  
Use Heroku CLI

GitHub  
Connect to GitHub

Container Registry  
Use Heroku CLI

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

mariope

predict-iris-flower-species

Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access.](#)

Following that, the name of the repository appears right below the search section with a new icon to connect to the GitHub repository. The connection button was clicked.

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

mariope

predict-iris-flower-species

Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access.](#)

mariope/predict\_iris\_flower\_species

Connect

After clicking the connect button, some new information in the connect to GitHub tab showing that the Heroku is connected to the GitHub repository appeared (figure below).

App connected to GitHub

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

Connected to [mariope/predict\\_iris\\_flower\\_species](#) by mariope

Disconnect...

Releases in the [activity feed](#) link to GitHub to view commit diffs

There are two options for deploy (automatic and manual). In the Manual Deploy, the main branch was selected and the button deploy branch was clicked.

Salesforce Platform

HEROKU

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

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.

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

When the Deploy Branch button is clicked, the deployment starts and Heroku download, install all the dependencies and create the necessary files for the deployment of the ML App/API. When the deployment is concluded, the following message (next figure) appears at the bottom of the page, showing that the deployment was successful.

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 29dad6cd

Release phase

Deploy to Heroku

Your app was successfully deployed.

View

The web App can be accessed by clicking the button “View” in the bottom of the page or by going to the address created by Heroku that can be seen in the settings section. The address of the Web App is as follow:

<https://predict-iris-flower-species.herokuapp.com/>



## USING THE WEB AND API FUNCTIONALITIES OF THIS APPLICATION

To access the Web App, just insert the url shown in the end of the previous section in your browser. In the home page, just add the measurements of the sepal\_length, sepal\_width, petal\_length and petal width, and click predict. The model makes the prediction in presents the outcome right below the predict button.

To use the API, insert the url "[https://predict-iris-flower-species.herokuapp.com/predict\\_api/](https://predict-iris-flower-species.herokuapp.com/predict_api/)" in Postman and pass the four required measurements (sepal length and width, and petal length and width) and the outcome of the prediction will be presented. Moreover you can also use the mentioned url with the arguments following the address and insert it into the web browser and the result will be displayed. You can see an example of the full url including the API call and the required arguments.

E.g., [http://predict-iris-flower-species.herokuapp.com/predict\\_api/?sepal\\_length=3&sepal\\_width=2.5&petal\\_length=2&petal\\_width=2](http://predict-iris-flower-species.herokuapp.com/predict_api/?sepal_length=3&sepal_width=2.5&petal_length=2&petal_width=2)