Deployment on Flask

Name: Buse Küçükçoban

Batch code: LISP01

Submission date: 22-03-2021

Submitted to: Data Glacier

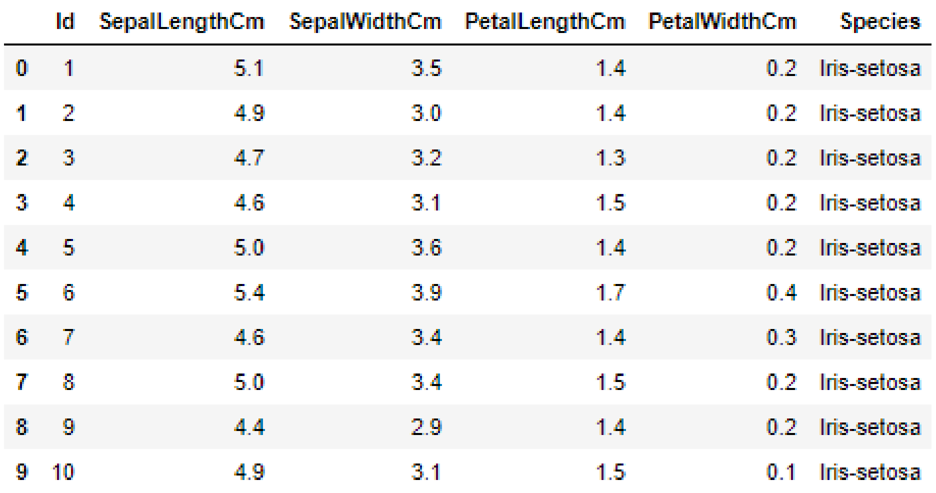
This article demonstrated to deploy machine learning models on Flask.

Project has three parts :

1. model.py — This contains code for the machine learning model to predict species when user get input.
2. app.py — This contains Flask APIs that receives sales details through GUI or API calls, computes the predicted value based on our model and returns it.
3. HTML/CSS — This contains the HTML template and CSS styling to allow user to enter flower details and displays the species.

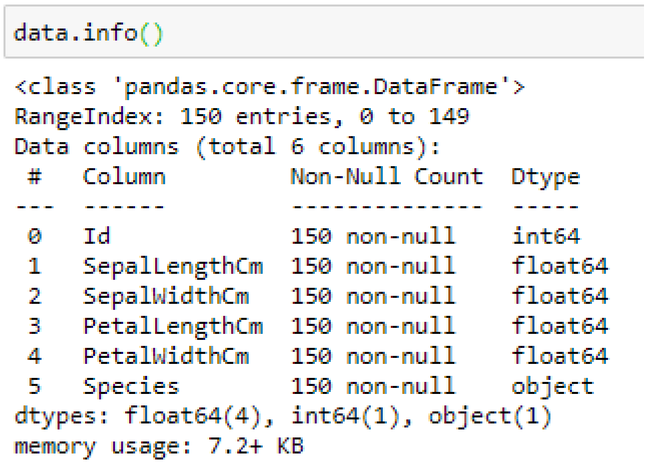
1.Build Machine Learning Model

I used “Iris” dataset for this project which has six columns :



	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

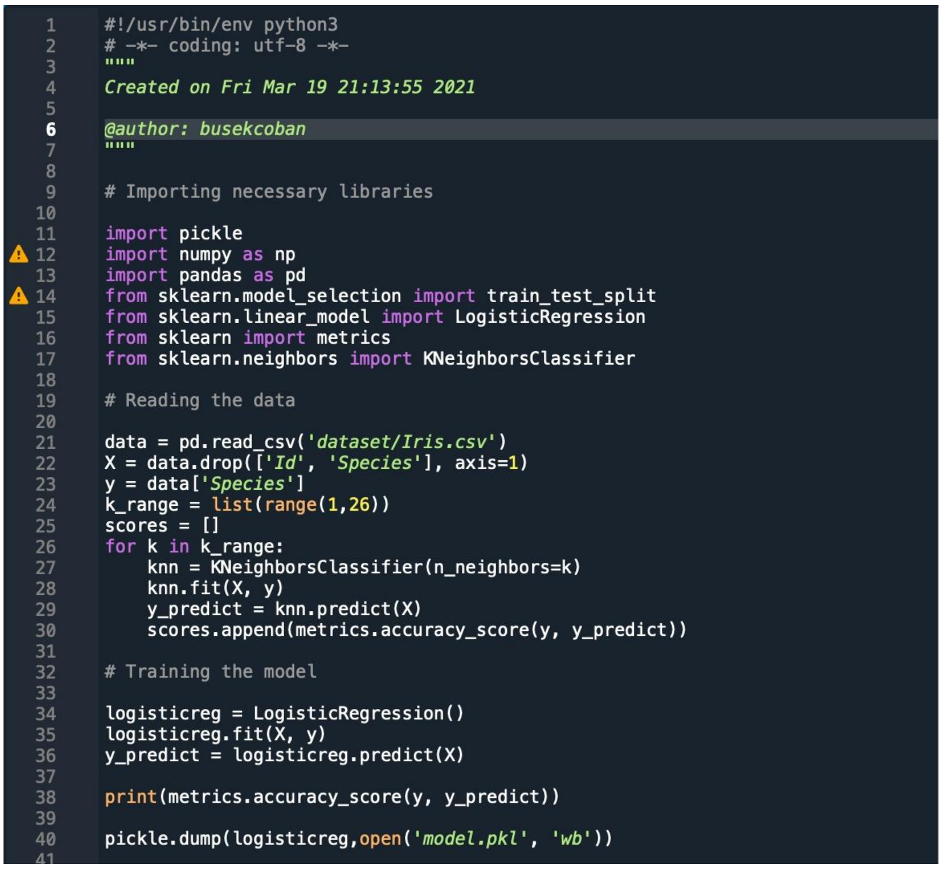
Some info about data:



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Now make a machine learning model to predict species when user get input. I used logistic regression as the machine learning algorithm.

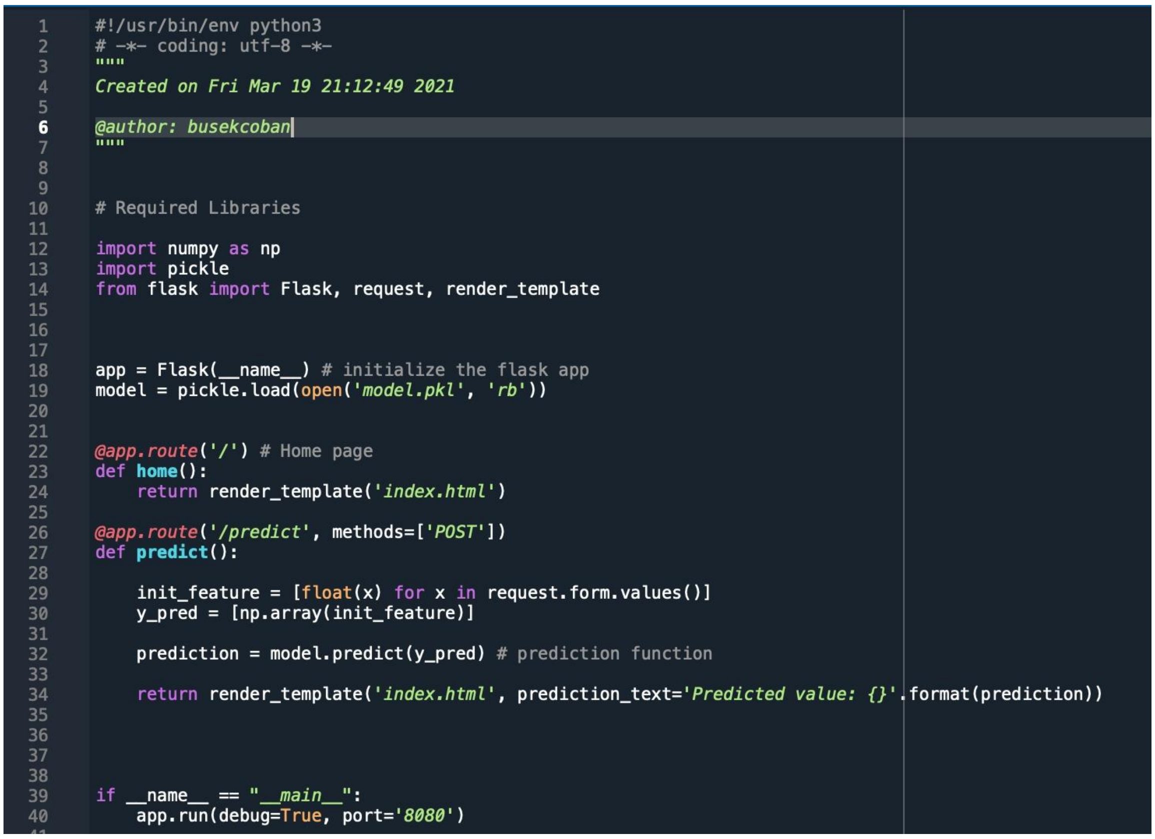


```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 19 21:13:55 2021

@author: busekcoban
"""

# Importing necessary libraries

import pickle
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier

# Reading the data

data = pd.read_csv('dataset/Iris.csv')
X = data.drop(['Id', 'Species'], axis=1)
y = data['Species']
k_range = list(range(1,26))
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X, y)
    y_predict = knn.predict(X)
    scores.append(metrics.accuracy_score(y, y_predict))

# Training the model

logisticreg = LogisticRegression()
logisticreg.fit(X, y)
y_predict = logisticreg.predict(X)

print(metrics.accuracy_score(y, y_predict))

pickle.dump(logisticreg,open('model.pkl', 'wb'))
```

2. Create app.py file

The next part was to make an API which receives sales details through GUI and computes the species on our model. For this Ide- serialized the pickled model in the form of python object.

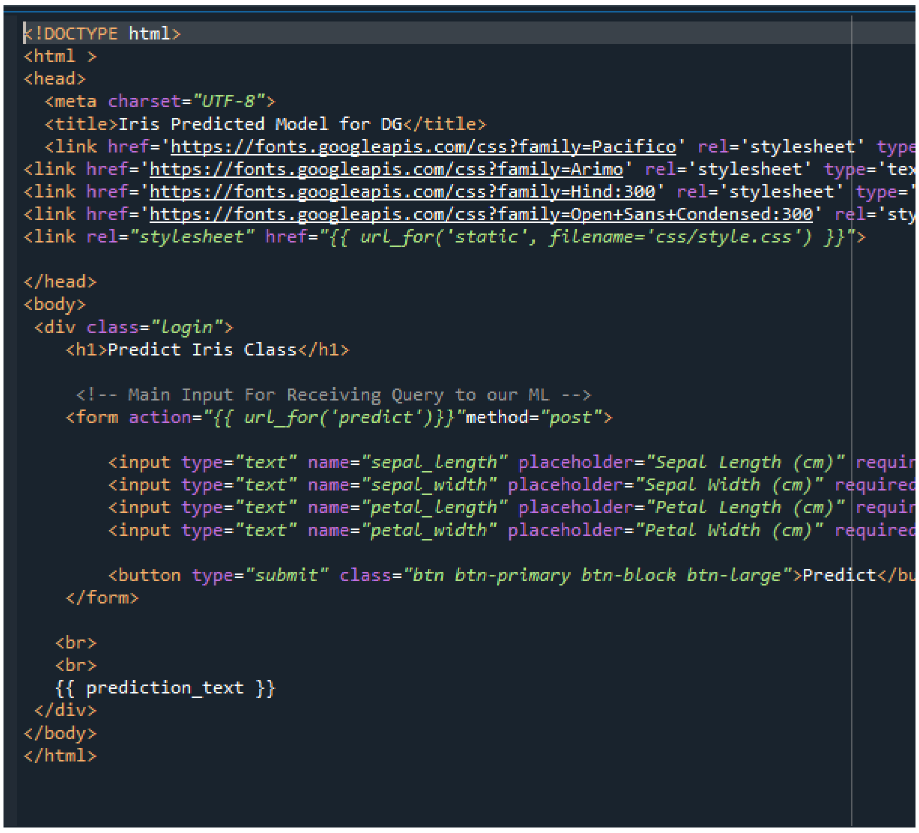
The results can be shown by making another POST request to /results. It receives JSON inputs, uses the trained model to make a prediction and returns that prediction in JSON format which can be accessed through the API endpoint.



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 19 21:12:49 2021

@author: busekcoban
"""


# Required Libraries

import numpy as np
import pickle
from flask import Flask, request, render_template



app = Flask(__name__) # initialize the flask app
model = pickle.load(open('model.pkl', 'rb'))


@app.route('/') # Home page
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():

    init_feature = [float(x) for x in request.form.values()]
    y_pred = [np.array(init_feature)]

    prediction = model.predict(y_pred) # prediction function

    return render_template('index.html', prediction_text='Predicted value: {}'.format(prediction))




if __name__ == "__main__":
    app.run(debug=True, port='8080')
```

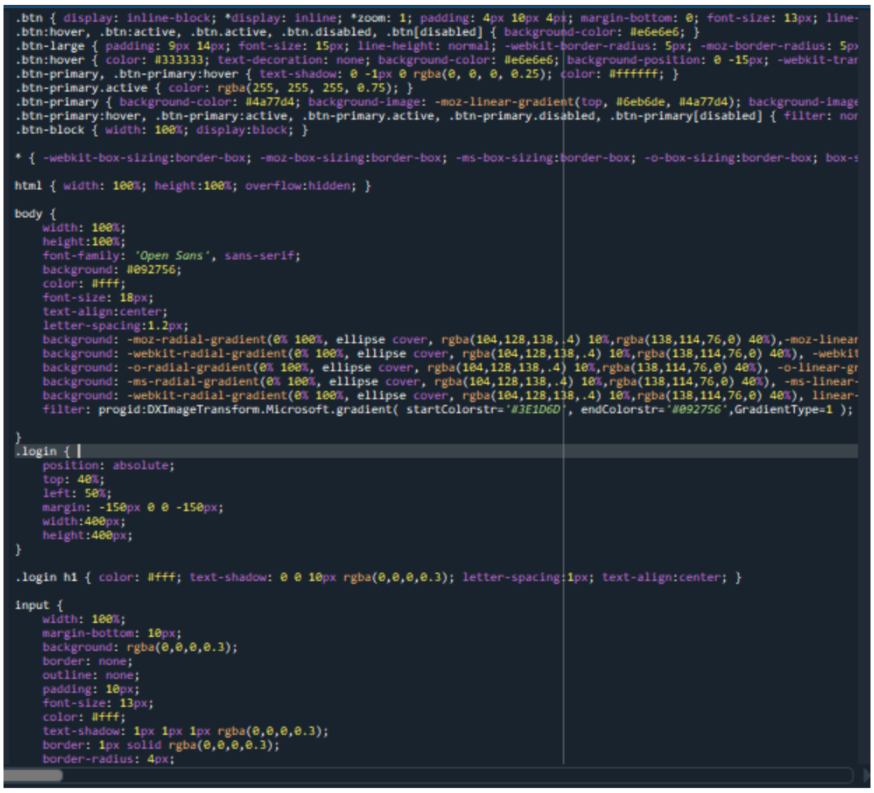
3.Create HTML file

Using HTML for the user to input the values. There are 4 fields which need to be filled by the user: **Sepal Length, Sepal Width, Petal Length, Petal Width**



```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Iris Predicted Model for DG</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='tex
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='sty
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>
<body>
 <div class="login">
    <h1>Predict Iris Class</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">

        <input type="text" name="sepal_length" placeholder="Sepal Length (cm)" requir
        <input type="text" name="sepal_width" placeholder="Sepal Width (cm)" required
        <input type="text" name="petal_length" placeholder="Petal Length (cm)" requir
        <input type="text" name="petal_width" placeholder="Petal Width (cm)" required

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</bu
    </form>

   <br>
   <br>
   {{ prediction_text }}
 </div>
</body>
</html>
```

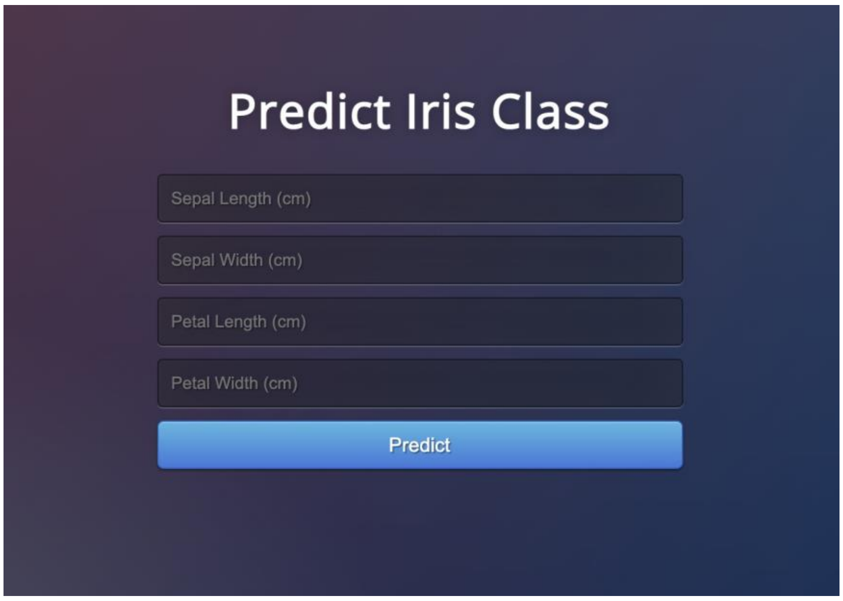
4.Using CSS for the input button and the background.



```css
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13px; line-
.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-radius: 5px
.btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-position: 0 -15px; -webkit-tran
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
.btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4); background-image
.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] { filter: non
.btn-block { width: 100%; display:block; }

* { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box; -o-box-sizing:border-box; box-s

html { width: 100%; height:100%; overflow:hidden; }

body {
    width: 100%;
    height:100%;
    font-family: 'Open Sans', sans-serif;
    background: #092756;
    color: #fff;
    font-size: 18px;
    text-align:center;
    letter-spacing:1.2px;
    background: -moz-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%),-moz-linear
    background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -webkit
    background: -o-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -o-linear-gr
    background: -ms-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -ms-linear-
    background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), linear-
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3E1D6D', endColorstr='#092756',GradientType=1 );

}
.login {
    position: absolute;
    top: 40%;
    left: 50%;
    margin: -150px 0 0 -150px;
    width:400px;
    height:400px;
}

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px; text-align:center; }

input {
    width: 100%;
    margin-bottom: 10px;
    background: rgba(0,0,0,0.3);
    border: none;
    outline: none;
    padding: 10px;
    font-size: 13px;
    color: #fff;
    text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
    border: 1px solid rgba(0,0,0,0.3);
    border-radius: 4px;
```

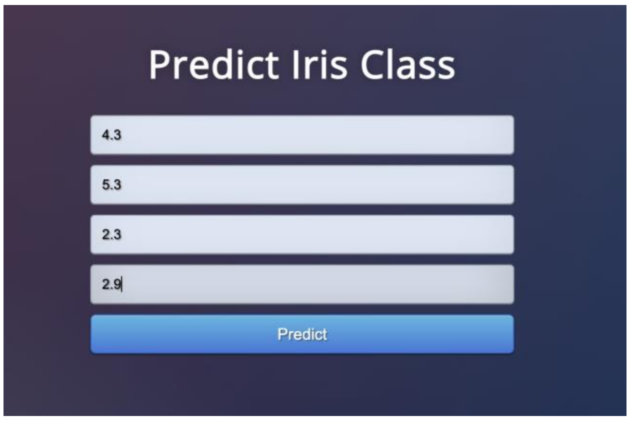
Predict species and send results

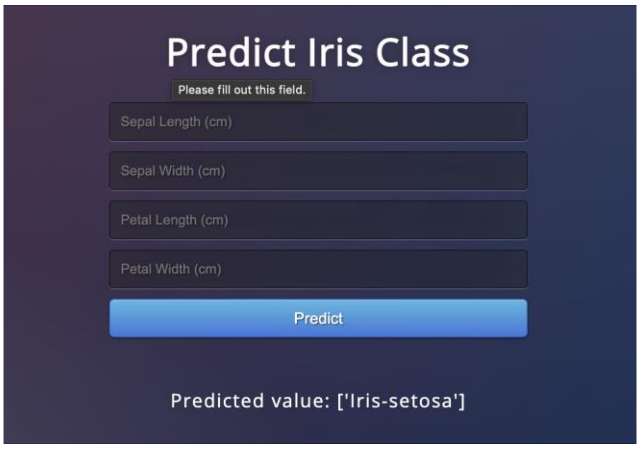
Open in web-browser, and the GUI as shown below should appear



For example, when we input Sepal length = 4.3, Sepal width = 5.3, Petal Length = 2.3, Petal width = 2.9

Result:





Predicted value: ['Iris-setosa']

Sources:

https://www.analyticsvidhya.com/blog/2020/04/how-to-deploy-machine-learning-model-flask/

https://flask.palletsprojects.com/en/1.1.x/tutorial/deploy/