Name: Ian Kihara Wangui

Batch Code: LISUM 19

Submission date: 23/03/2023

Submitted to: Data Glacier

---------- **Local Deployment** ----------

This guide illustrates local deployment on your machine. I will consider that you are using Visual Studio Code, in case you are not, no problem at all, the first thing is to create a virtual environment in the folder containing your code. Install virtualenv:

pip install virtualenv

In the folder of your project, if not:

cd my-project

Activate the Virtual Environment

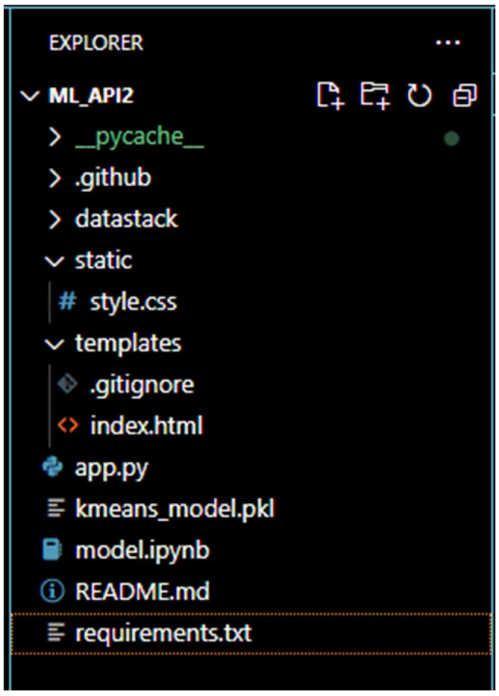
virtualenv --python C:\Path\To\Python\python.exe venv

Then, activate the Virtual Environment:

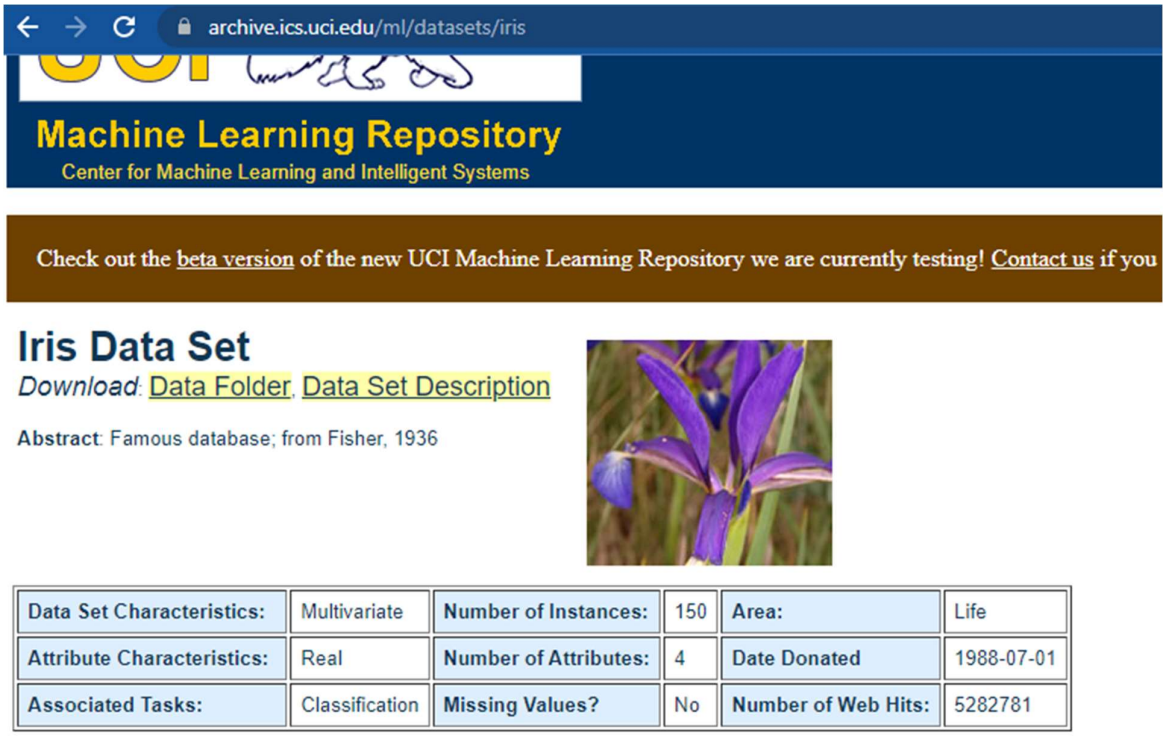
.\venv\Scripts\activate

If you need to install packages from a determined project type: **pip install -r requirements.txt**

All set to start the development, the final folder structure should look like this:



This application uses the iris dataset of scikit-learn.(https://archive.ics.uci.edu/ml/datasets/iris)



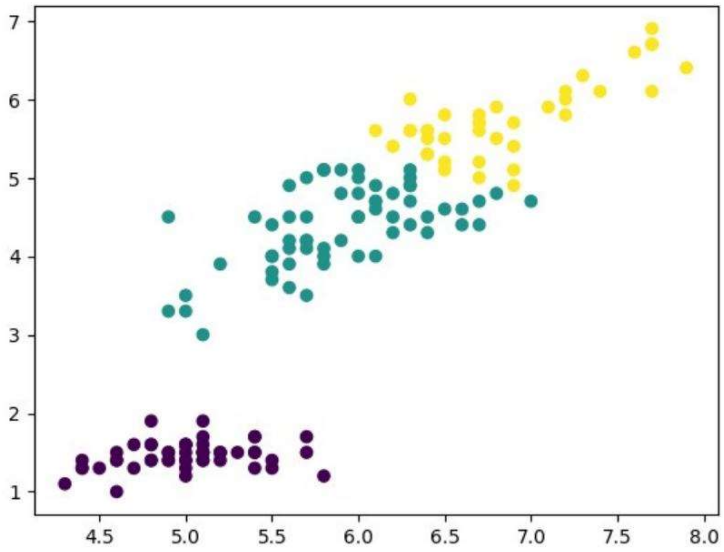
Data Set Characteristics:	Multivariate	Number of Instances:	150	Area:	Life
Attribute Characteristics:	Real	Number of Attributes:	4	Date Donated	1988-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	5282781

It is time to start working on our machine learning models; here is the snapshot code for **model.ipynb** file:

```
# Importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle
from sklearn.cluster import KMeans
from sklearn import datasets

# Loading Iris dataset
samples = datasets.load_iris()
X = samples.data
y = samples.target

# KMeans clustering
model = KMeans(n_clusters=3)
model.fit(samples.data)
labels = model.predict(samples.data)
print(labels)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1
 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2
2 2
 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2
1 2
 2 1]
```

```
# Scatter plot
xs = samples.data[:,0]
ys = samples.data[:,2]
plt.scatter(xs, ys, c=labels);
```




```
# Sample data for a prediction
new_sample = [[5.7, 4.4, 1.5, 0.4]]
print(new_sample)
```

```
[[5.7, 4.4, 1.5, 0.4]]
```

```
# Make a prediction
new_label = model.predict(new_sample)
print(new_label)
```

```
[0]
```

```
# Evaluating the clustering with cross tablulation
species = y[:]
df = pd.DataFrame({'labels': labels, 'species': species})
ct = pd.crosstab(df['labels'], df['species'])
print(ct)
```

```
species   0   1   2
labels
0        50   0   0
1         0  48  14
2         0   2  36
```

```
# Save model to disk using pickle
with open('kmeans_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

Ok, after the model is trained you should save a pickle file of it, then we will work on the app.py file:

app.py

```python
import numpy as np
from sklearn.datasets import load_iris
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('kmeans_model.pkl', 'rb'))
iris = load_iris()

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

@app.route('/predict', methods=['POST'])
def predict():
    '''
    Render the prediction results on HTML
    '''
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    output = iris.target_names[prediction[0]]

    return render_template('index.html', prediction_text='Iris type should be {}'.format(output))

if __name__ == "__main__":
    app.run(debug=True)
```

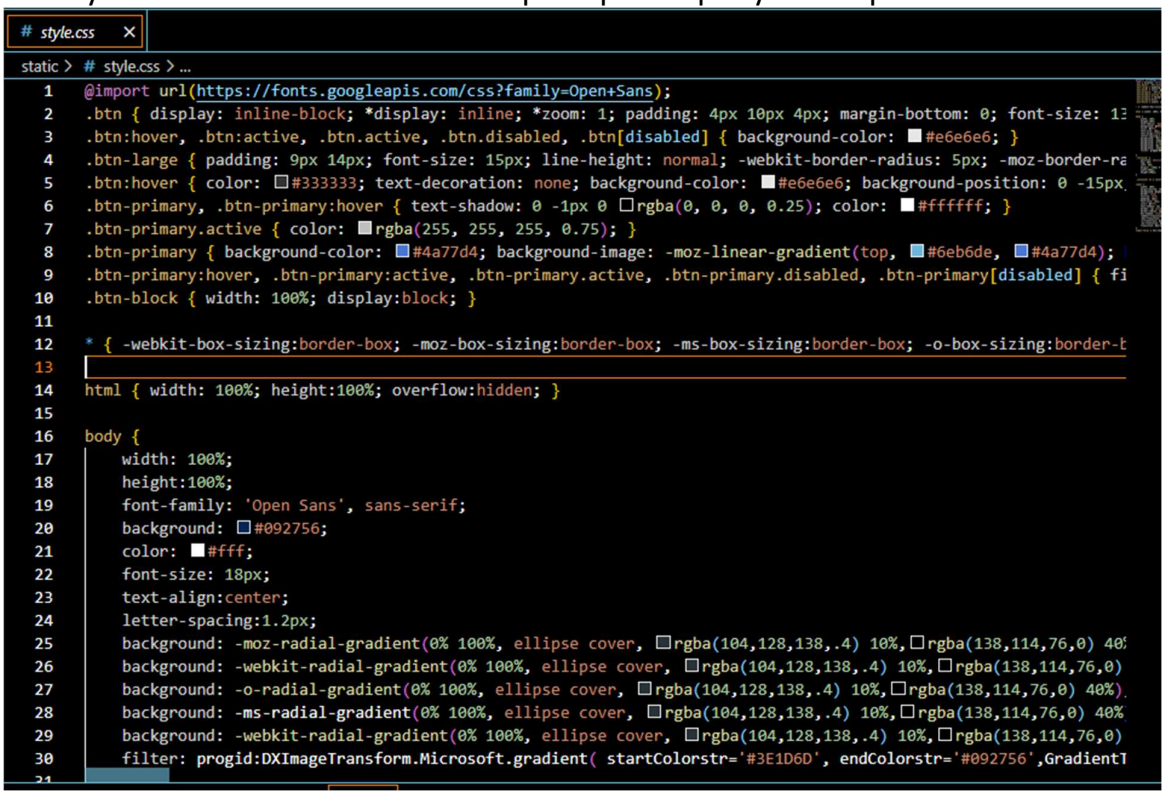
All right, go to the template folder and open a new file named index.html; here is the following code:

index.html

templates > index.html > html > body > div.container

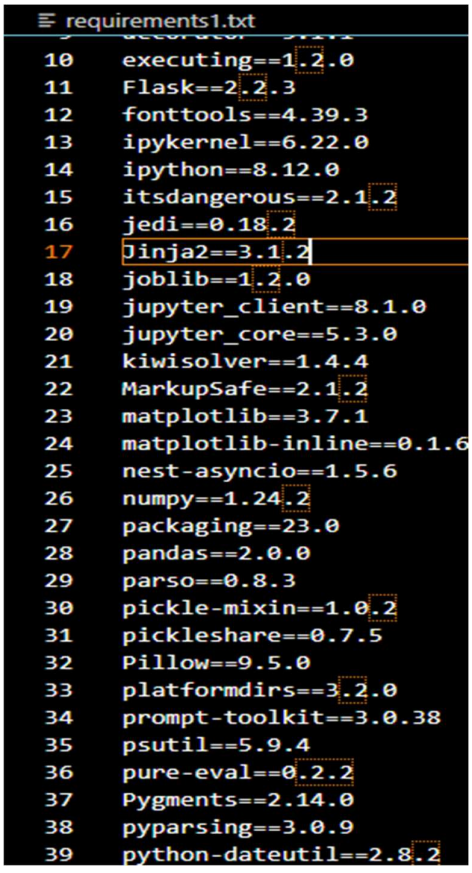
```html
<html >

<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body>
<div class="container">
  <h1>Predict Iris Type</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
      <input type="text" name="petal_length" placeholder="Petal Length" required="required" />
        <input type="text" name="petal_width" placeholder="Petal Width" required="required" />
    <input type="text" name="sepal_length" placeholder="Sepal Length" required="required" />
      <input type="text" name="sepal_width" placeholder="Sepal Width" required="required" />

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

   <br>
   <br>
   {{ prediction_text }}
```

To style the website I used Bootstrap to speed up my development.



```
static > # style.css > ...
1  @import url(https://fonts.googleapis.com/css?family=Open+Sans);
2  .btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13
3  .btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
4  .btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-ra
5  .btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-position: 0 -15px
6  .btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
7  .btn-primary.active { color: rgba(255, 255, 255, 0.75); }
8  .btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4);
9  .btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] { fi
10 .btn-block { width: 100%; display:block; }
11
12 * { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box; -o-box-sizing:border-b
13
14 html { width: 100%; height:100%; overflow:hidden; }
15
16 body {
17     width: 100%;
18     height:100%;
19     font-family: 'Open Sans', sans-serif;
20     background: #092756;
21     color: #fff;
22     font-size: 18px;
23     text-align:center;
24     letter-spacing:1.2px;
25     background: -moz-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40
26     background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0)
27     background: -o-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%),
28     background: -ms-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%
29     background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0)
30     filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3E1D6D', endColorstr='#092756',GradientT
```

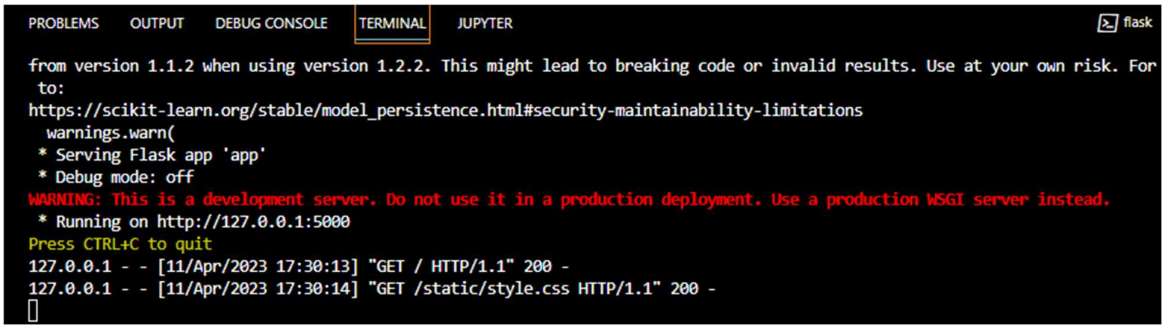
Type the following commands to generate a list of packages to install when you finish the deployment in the terminal: **pip freeze > requirements.txt**



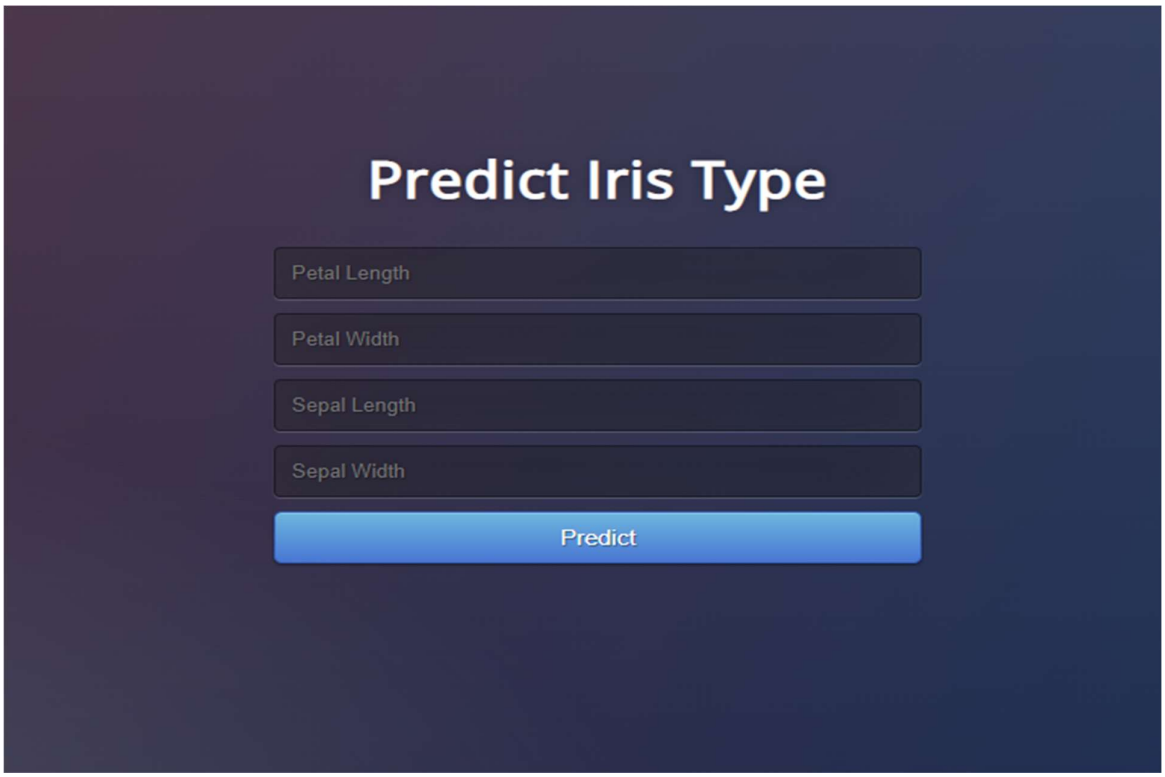
```
requirements1.txt
10  executing==1.2.0
11  Flask==2.2.3
12  fonttools==4.39.3
13  ipykernel==6.22.0
14  ipython==8.12.0
15  itsdangerous==2.1.2
16  jedi==0.18.2
17  Jinja2==3.1.2
18  joblib==1.2.0
19  jupyter_client==8.1.0
20  jupyter_core==5.3.0
21  kiwisolver==1.4.4
22  MarkupSafe==2.1.2
23  matplotlib==3.7.1
24  matplotlib-inline==0.1.6
25  nest-asyncio==1.5.6
26  numpy==1.24.2
27  packaging==23.0
28  pandas==2.0.0
29  parso==0.8.3
30  pickle-mixin==1.0.2
31  pickleshare==0.7.5
32  Pillow==9.5.0
33  platformdirs==3.2.0
34  prompt-toolkit==3.0.38
35  psutil==5.9.4
36  pure-eval==0.2.2
37  Pygments==2.14.0
38  pyparsing==3.0.9
39  python-dateutil==2.8.2
```

At this point, we can run the following command to run the flask application:

flask - -app app run



Go to http://127.0.0.1:5000/ in your browser, you see the website, once you do some request, the terminal should change to show the **API request**, you can notice the methods **POST** and **GET** and the **http** response **200** for successful anything different, it points an error.



Finally, after you run all codes, you need to deactivate the virtual environment; here is the command:

deactivate