

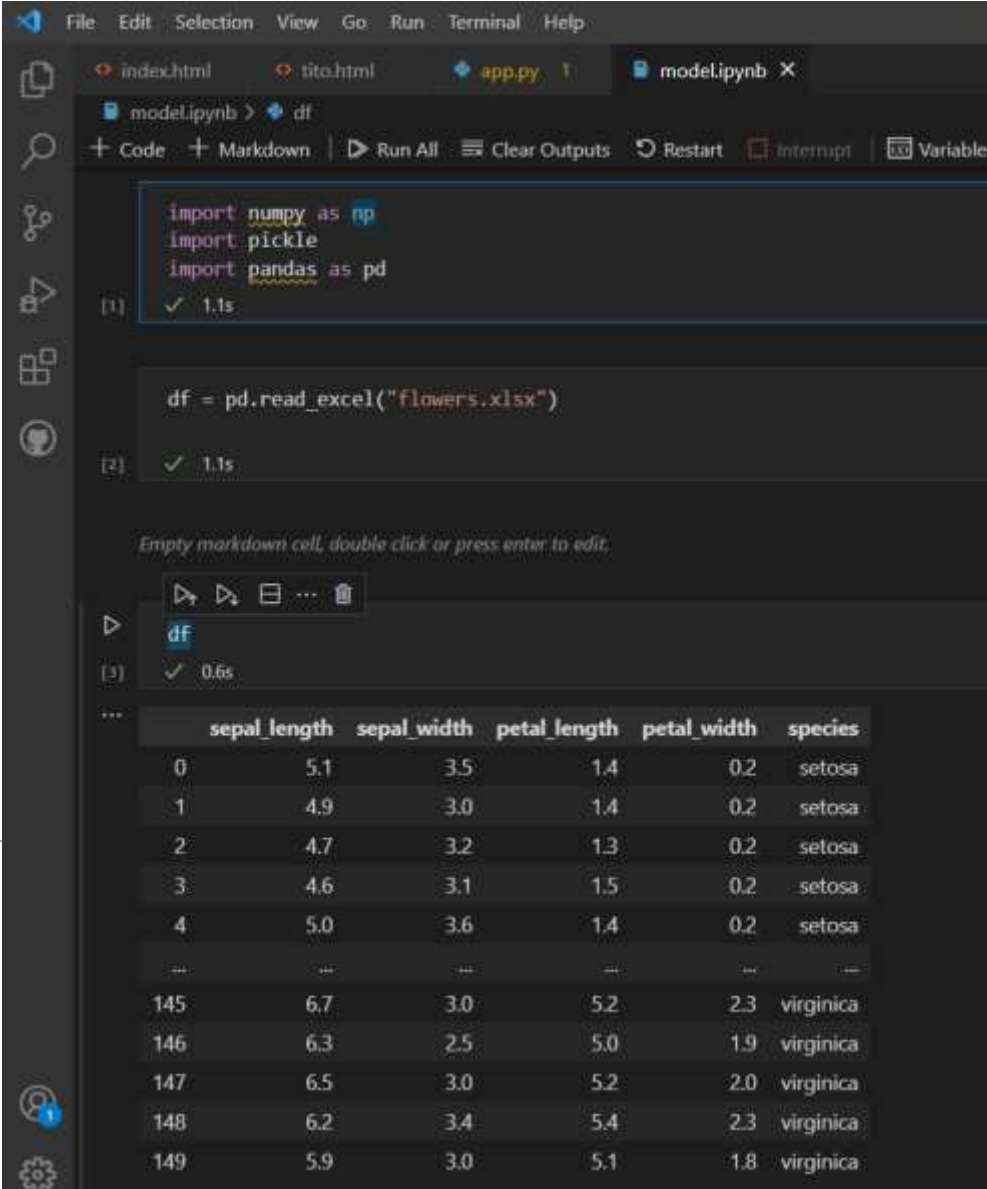
Data Glacier

Deployment on Flask

- Name : Mohamad Eyad Abras
- Batch Code: LISUM12
- Submission Date: 28/08/2022
- Submitted to: Github (week 4 Folder)

Step #1 :

Getting the data and preparing the model
(flowers classification Model)



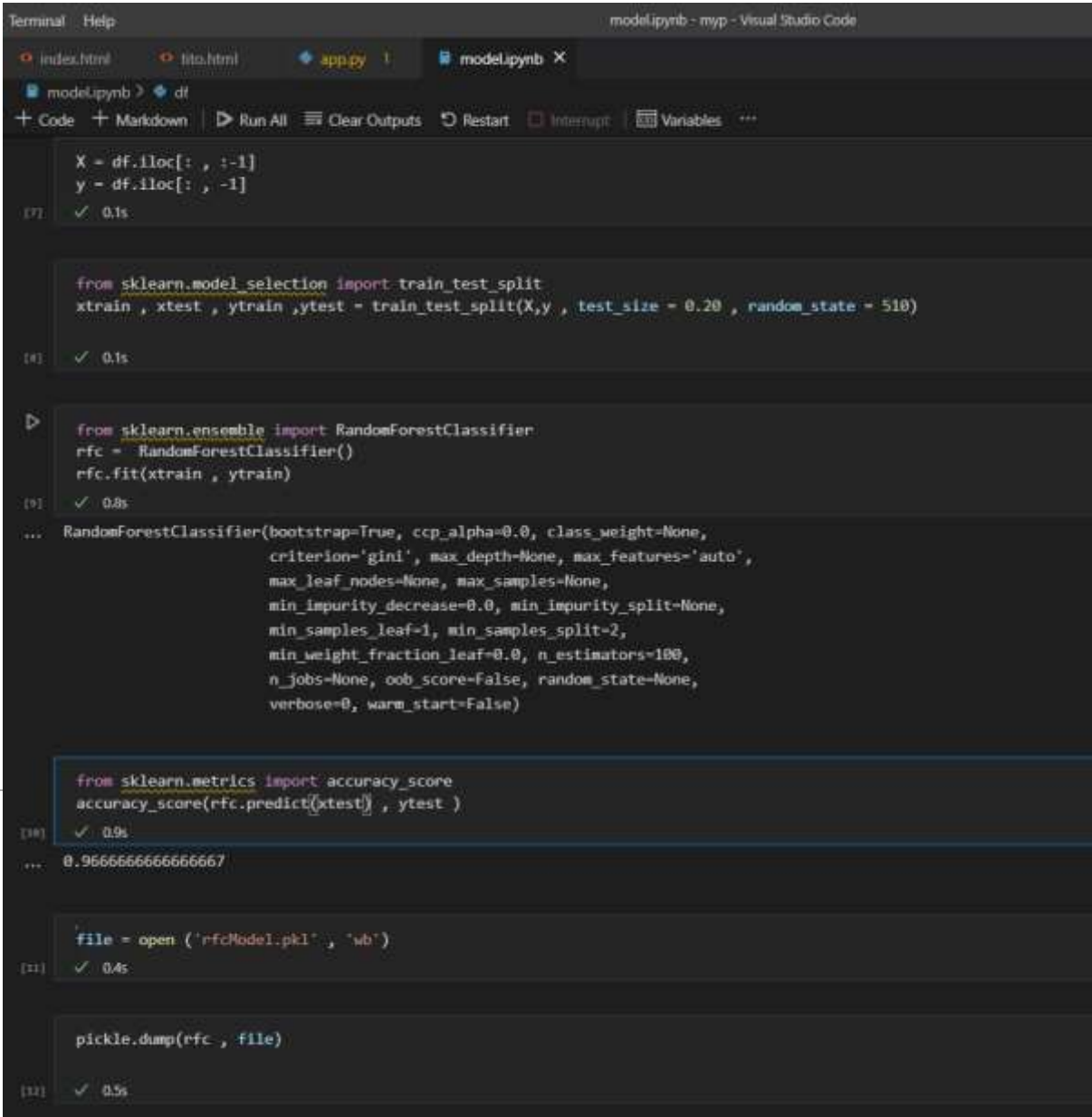
```
import numpy as np
import pickle
import pandas as pd
```

```
df = pd.read_excel("flowers.xlsx")
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

Step #2 :

Finishing and Pickling the Model
(in this case the model is Random Forest Classifier)



```
Terminal Help model.ipynb - myp - Visual Studio Code
index.html tito.html app.py 1 model.ipynb X
model.ipynb > dt
+ Code + Markdown | ▶ Run All | Clear Outputs | Restart | Interrupt | Variables ...

X = df.iloc[:, :-1]
y = df.iloc[:, -1]
[7] ✓ 0.1s

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.20, random_state = 510)
[8] ✓ 0.1s

▶ from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(xtrain, ytrain)
[9] ✓ 0.8s
... RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
criterion='gini', max_depth=None, max_features='auto',
max_leaf_nodes=None, max_samples=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100,
n_jobs=None, oob_score=False, random_state=None,
verbose=0, warm_start=False)

from sklearn.metrics import accuracy_score
accuracy_score(rfc.predict(xtest), ytest)
[10] ✓ 0.9s
... 0.9666666666666667

file = open('rfcModel.pkl', 'wb')
[11] ✓ 0.4s

pickle.dump(rfc, file)
[12] ✓ 0.5s
```

Step #3 :

Preparing the HTML file for rendering

```
terminal  Help  index.html - myp - Visual Studio Code

index.html x  titio.html  app.py 1  model.ipynb

templates > index.html > html > body#home > div#home > form > div
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head id = "test">
5      <h1 id = "test">Welcome!!</h1>
6      <meta charset="UTF-8">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Document</title>
9  </head>
10
11 <body id = "home">
12     <div id="home">
13         <form action="{{ url_for('predict')}}" method="post">
14             <h3> Enter the sepal length </h3><input id="fielddo" name="sepal_length" required="required "><br>
15             <h3> Enter the sepal width</h3><input id="fielddo" name="sepal_width" required="required">
16             <h3>Enter the petal length</h3><input id="fielddo" name="petal_length" required="required">
17             <h3>Enter the petal width</h3><input id="fielddo" name="petal_width" required="required">
18
19             <div>
20                 <br><button id="sub" type="submit ">what is the flower type </button>
21                 <h3 id = 'test'>{{ prediction_text }}</h3>
22             </div>
23
24         </form>
25
26
27         <div id = 'test' >
28             <form action="{{ url_for('About')}}" method="post">
29                 <button type="submit ">{{ info_f }} </button>
30             </form>
31         </div>
32     </div>
33 </body>
34
35
36
37
38 <style>
39 #fielddo {
40     border-radius: 14px;
41     font-size: 20px;
```

Step #4:

Working with flask :

Loading the Model from the .pkl files :

```
7
8 app = Flask(__name__)
9 model = pickle.load(open('rfcModel.pkl' , 'rb'))
10 @app.route('/',methods=['GET'])
11 def Home():
12     return render_template('index.html')
```

Preparing the home page with Flask :

```
@app.route('/',methods=['GET'])
def Home():
    return render_template('index.html')
```

Adding the main function of the app (predict) :

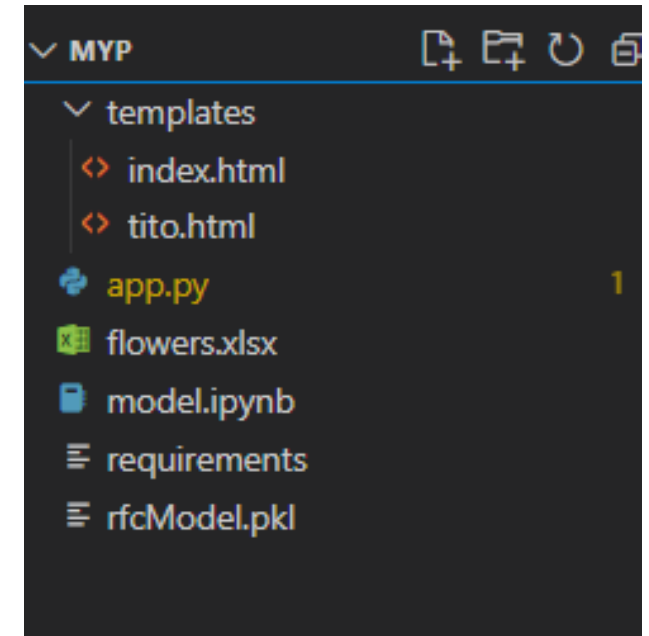
```
@app.route('/predict' , methods = ['POST' , 'GET'])
def predict():
    if request.method == 'POST':
        sepal_length = float(request.form['sepal_length'])
        sepal_width=float(request.form['sepal_width'])
        petal_length = float(request.form['petal_length'])
        petal_width=float(request.form['petal_width'])

        prediction=model.predict([[sepal_length ,sepal_width , petal_length, petal_width ]])
        if prediction ==0:
            return render_template('index.html',prediction_text="it's Setosa" , info_f = "more about Setosa ?")
        elif prediction == 1:
            return render_template('index.html',prediction_text="it's Versicolor" , info_f = "more about Versicolor ?")
        elif prediction == 2:
            return render_template('index.html',prediction_text="it's Virginica" , info_f = "more about Virginica ?")
        else:
            return render_template('index.html',prediction_text="hata olustu")

    else:
        return render_template('index.html')
```

Step #5 : launching

Now that we have all the required files



We are ready to launch

```
45  
46  
47 if __name__=="__main__":  
48     app.run(debug=True)
```

Step #6:

Activating the virtual environment and running the app

Running the App.py file

```
Anaconda Prompt (anaconda3)

(base) C:\Users\HP>cd desktop/myp

(base) C:\Users\HP\Desktop\myp>activate myenv

(myenv) C:\Users\HP\Desktop\myp>dir
Volume in drive C has no label.
Volume Serial Number is F4AB-AAB0

Directory of C:\Users\HP\Desktop\myp

08/08/2021  10:40 AM    <DIR>          .
08/08/2021  10:40 AM    <DIR>          ..
08/08/2021  03:23 PM             1,506 app.py
08/08/2021  12:55 AM             13,442 flowers.xlsx
08/08/2021  10:53 AM             13,781 model.ipynb
08/08/2021  10:40 AM                23 Procfile
08/08/2021  10:49 AM            158,792 rfcModel.pkl
08/08/2021  12:13 PM    <DIR>          templates
                    5 File(s)        187,544 bytes
                    3 Dir(s)  44,225,155,072 bytes free

(myenv) C:\Users\HP\Desktop\myp>python app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
```

It's ready to classify

Welcome!!

Enter the sepal length

Enter the sepal width

Enter the petal length

Enter the petal width

what is the flower type



It's Setosa!!

Welcome!!

Enter the sepal length

5

Enter the sepal width

2.3

Enter the petal length

1.4

Enter the petal width

6

what is the flower type

it's Setosa