



Data Science Intern at Data Glacier

Week 4: Deployment on Flask

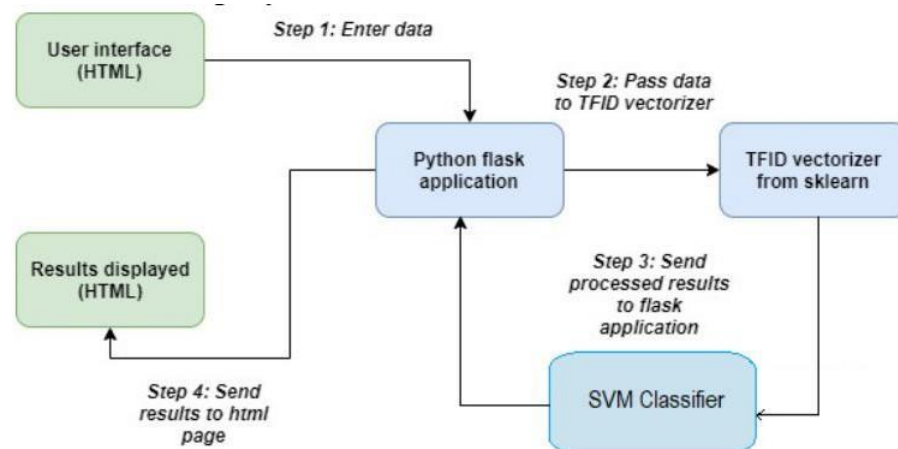
Name: Pravallika sheshabhatte

Batch Code: LISUM21

Date: 28 May 2023

Submitted to: Data Glacier

In this project, we are going to deploying machine learning model using the Flask Framework.



Create an API for the model, using Flask, the Python micro-framework for building web applications. This API allows us to utilize predictive capabilities through HTTP requests.

Data Information

- I have taken the toy data set as mentioned in the assignment i.e., IRIS data set.
- The collection is composed of one CSV file per dataset.

Building a Model

Import Required Libraries and Dataset

In this part, we import libraires and dataset which contain the information of five most commented video.

```
app.py x model.py x
61 plt.scatter(x['sepal.width'], x['petal.width'], c = colors[i], label=var
62 plt.xlabel("Sepal Width")
63 plt.ylabel("Petal Width")
64 plt.legend()
65
66 df.corr()
67 corr = df.corr()
68 fig, ax = plt.subplots(figsize=(5,4))
69 sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')
70
71 #Training the model
72 from sklearn.model_selection import train_test_split
73 # train = 70
74 # test = 30
75 X = df.drop(columns=['variety'])
76 Y = df['variety']
77 x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
78
79 from sklearn.linear_model import LogisticRegression
80 model = LogisticRegression()
81 model.fit(x_train, y_train)
82 print("Accuracy: ",model.score(x_test, y_test) * 100)
83
84
85 from sklearn.neighbors import KNeighborsClassifier
86 model = KNeighborsClassifier()
87 model.fit(x_train, y_train)
88
89 print("Accuracy: ",model.score(x_test, y_test) * 100)
90
91
92
93 from sklearn.tree import DecisionTreeClassifier
```

Name	Type	Size	Value
ax	axes._subplots.AxesSubplot	1	AxesSubplot object of matplotlib.axes._subplots module
colors	list	3	['pink', 'brown', 'blue']
corr	DataFrame	(4, 4)	Column names: sepal.length, sepal.width, petal.length, petal.width
df	DataFrame	(150, 5)	Column names: sepal.length, sepal.width, petal.length, petal.width, va ...
fig	figure.Figure	1	Figure object of matplotlib.figure module
i	int	1	2
variety	list	3	['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']

```
Help Variable Explorer Plots Files
Console I/A x
Out[26]: sepal.length sepal.width petal.length petal.width
sepal.length 1.000000 -0.117570 0.871754 0.817941
sepal.width -0.117570 1.000000 -0.428440 -0.366126
petal.length 0.871754 -0.428440 1.000000 0.962865
petal.width 0.817941 -0.366126 0.962865 1.000000

In [26]: corr = df.corr()
In [27]: fig, ax = plt.subplots(figsize=(5,4))
In [28]: sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')
Out[28]: <AxesSubplot:~>
In [29]:
```

```
app.py X model.py X
61 plt.scatter(x['sepal.width'], x['petal.width'], c = colors[i], label=var
62 plt.xlabel("Sepal Width")
63 plt.ylabel("Petal Width")
64 plt.legend()
65
66 df.corr()
67 corr = df.corr()
68 fig, ax = plt.subplots(figsize=(5,4))
69 sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')
70
71 #Training the model
72 from sklearn.model_selection import train_test_split
73 # train = 70
74 # test = 30
75 X = df.drop(columns=['variety'])
76 Y = df['variety']
77 x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
78
79 from sklearn.linear_model import LogisticRegression
80 model = LogisticRegression()
81 model.fit(x_train, y_train)
82 print("Accuracy: ",model.score(x_test, y_test) * 100)
83
84
85 from sklearn.neighbors import KNeighborsClassifier
86 model = KNeighborsClassifier()
87 model.fit(x_train, y_train)
88 print("Accuracy: ",model.score(x_test, y_test) * 100)
89
90 from sklearn.tree import DecisionTreeClassifier
```

Name	Type	Size	Value
x	DataFrame	(0, 5)	Column names: sepal.length, s...
X	DataFrame	(150, 4)	Column names: sepal.length, s...
x_test	DataFrame	(45, 4)	Column names: sepal.length, s...
x_train	DataFrame	(105, 4)	Column names: sepal.length, s...
Y	Series	(150,)	Series object of pandas.core.series module
y_test	Series	(45,)	Series object of pandas.core.series module
y_train	Series	(105,)	Series object of pandas.core.series module

Help Variable Explorer Plots Files

Console I/A X

```
In [32]: x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
In [33]: from sklearn.linear_model import LogisticRegression
In [34]: model = LogisticRegression()
In [35]: model.fit(x_train, y_train)
Out[35]: LogisticRegression()
In [36]: print("Accuracy: ",model.score(x_test, y_test) * 100)
Accuracy: 97.77777777777777
In [37]:
```

IPython Console History

```
app.py X model.py X
75 X = df.drop(columns=['variety'])
76 Y = df['variety']
77 x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
78
79 from sklearn.linear_model import LogisticRegression
80 model = LogisticRegression()
81 model.fit(x_train, y_train)
82 print("Accuracy: ",model.score(x_test, y_test) * 100)
83
84
85 from sklearn.neighbors import KNeighborsClassifier
86 model = KNeighborsClassifier()
87 model.fit(x_train, y_train)
88 print("Accuracy: ",model.score(x_test, y_test) * 100)
89
90
91 from sklearn.tree import DecisionTreeClassifier
92 model = DecisionTreeClassifier()
93 model.fit(x_train, y_train)
94 print("Accuracy: ",model.score(x_test, y_test) * 100)
95
96 # Saving model to disk
97 import pickle
98 filename = 'savedmodel.pkl'
99 pickle.dump(model, open(filename, 'wb'))
100 # Loading model to compare the results
101 x_test.head()
102 load_model = pickle.load(open(filename, 'rb'))
103 load_model.predict([[6.0, 2.2, 4.0, 1.0]])
104
105
106
```

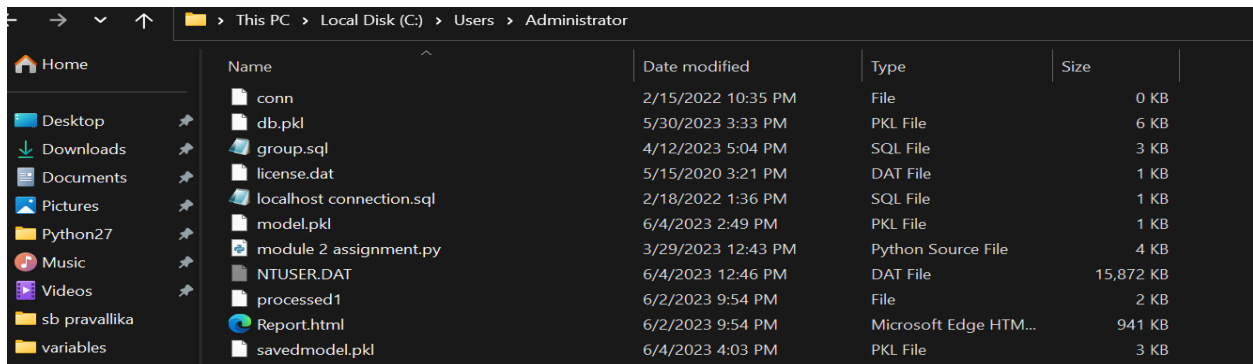
Name	Type	Size	Value
variety	list	3	['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']
x	DataFrame	(0, 5)	Column names: sepal.length, sepal...
X	DataFrame	(150, 4)	Column names: sepal.length, sepal...
x_test	DataFrame	(45, 4)	Column names: sepal.length, sepal...
x_train	DataFrame	(105, 4)	Column names: sepal.length, sepal...
Y	Series	(150,)	Series object of pandas.core.series module
y_test	Series	(45,)	Series object of pandas.core.series module

Help Variable Explorer Plots Files

Console I/A X

```
In [46]: filename = 'savedmodel.pkl'
In [47]: pickle.dump(model, open(filename, 'wb'))
In [48]: x_test.head()
Out[48]:
   sepal.length  sepal.width  petal.length  petal.width
30           4.8           3.1           1.6           0.2
99           5.7           2.8           4.1           1.3
34           4.9           3.1           1.5           0.2
23           5.1           3.3           1.7           0.5
38           4.4           3.0           1.3           0.2
In [49]:
```

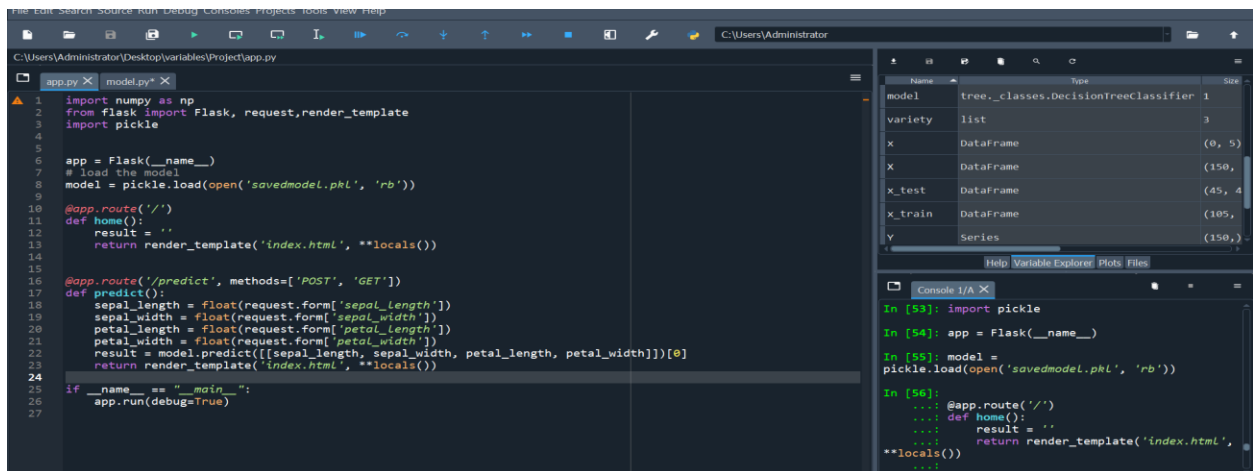
After executing the program in spyder in local disc C the *savemodel.pkl* is created .



Name	Date modified	Type	Size
conn	2/15/2022 10:35 PM	File	0 KB
db.pkl	5/30/2023 3:33 PM	PKL File	6 KB
group.sql	4/12/2023 5:04 PM	SQL File	3 KB
license.dat	5/15/2020 3:21 PM	DAT File	1 KB
localhost connection.sql	2/18/2022 1:36 PM	SQL File	1 KB
model.pkl	6/4/2023 2:49 PM	PKL File	1 KB
module 2 assignment.py	3/29/2023 12:43 PM	Python Source File	4 KB
NTUSER.DAT	6/4/2023 12:46 PM	DAT File	15,872 KB
processed1	6/2/2023 9:54 PM	File	2 KB
Report.html	6/2/2023 9:54 PM	Microsoft Edge HTM...	941 KB
savemodel.pkl	6/4/2023 4:03 PM	PKL File	3 KB

App.py

The *app.py* file contains the main code that will be executed by the Python interpreter to run the Flask web application, it included the ML code for classifying SD.



```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5
6 app = Flask(__name__)
7 # load the model
8 model = pickle.load(open('savemodel.pkl', 'rb'))
9
10 @app.route('/')
11 def home():
12     result = ''
13     return render_template("index.html", **locals())
14
15
16 @app.route('/predict', methods=['POST', 'GET'])
17 def predict():
18     sepal_length = float(request.form['sepal_length'])
19     sepal_width = float(request.form['sepal_width'])
20     petal_length = float(request.form['petal_length'])
21     petal_width = float(request.form['petal_width'])
22     result = model.predict([[sepal_length, sepal_width, petal_length, petal_width]])[0]
23     return render_template("index.html", **locals())
24
25 if __name__ == "__main__":
26     app.run(debug=True)
27
```

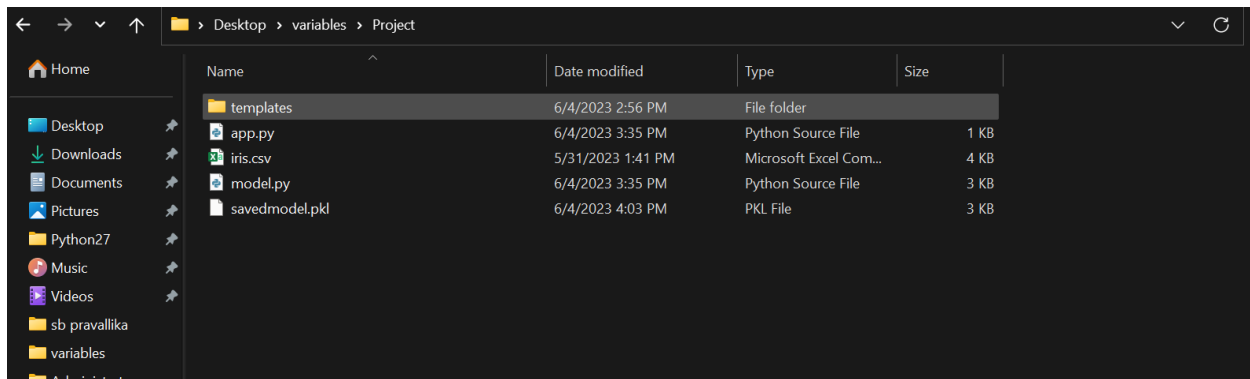
Variable Explorer:

Name	Type	Size
model	tree_classes.DecisionTreeClassifier	1
variety	list	3
x	DataFrame	(0, 5)
X	DataFrame	(150, 4)
x_test	DataFrame	(45, 4)
X_train	DataFrame	(105, 4)
Y	Series	(150, 1)

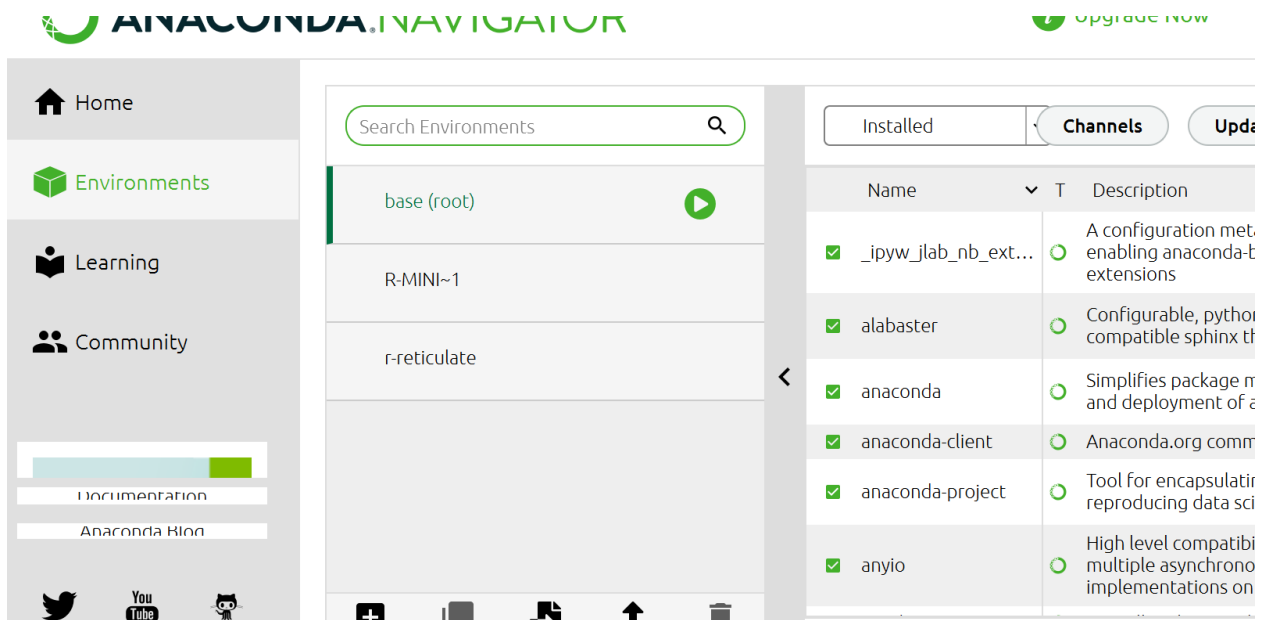
Console I/O:

```
In [53]: import pickle
In [54]: app = Flask(__name__)
In [55]: model = pickle.load(open('savemodel.pkl', 'rb'))
In [56]:
```

- We ran our application as a single module; thus we initialized a new Flask instance with the argument `__name__` to let Flask know that it can find the HTML template folder (*templates*) in the same directory where it is located.
- Next, we used the route decorator (`@app.route('/')`) to specify the URL that should trigger the execution of the home function.
- Our *home* function simply rendered *index.html* HTML file, which is located in the *templates* folder.

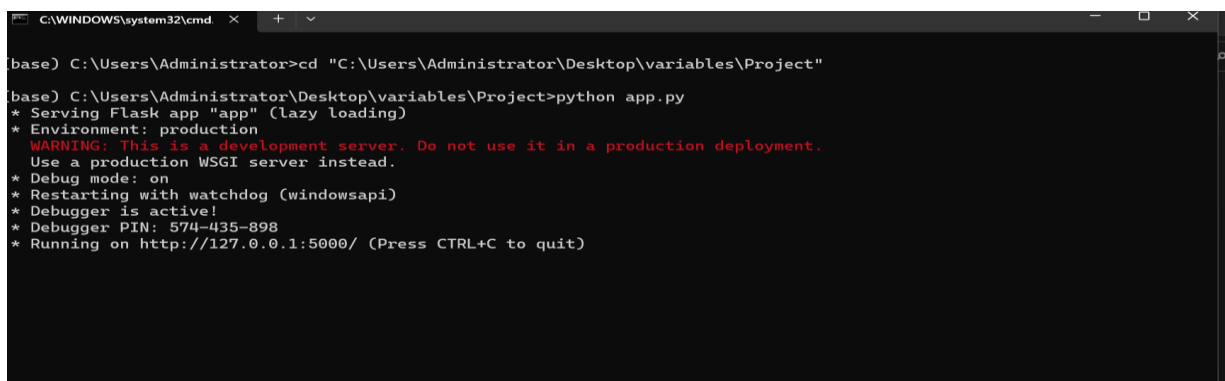


Go to ANACONDA NAVIGATOR and open terminal give the commands:



Running Procedure

Once we have done all of the above, we can start running the API by either double click *app.py*, or executing the command from the Terminal:

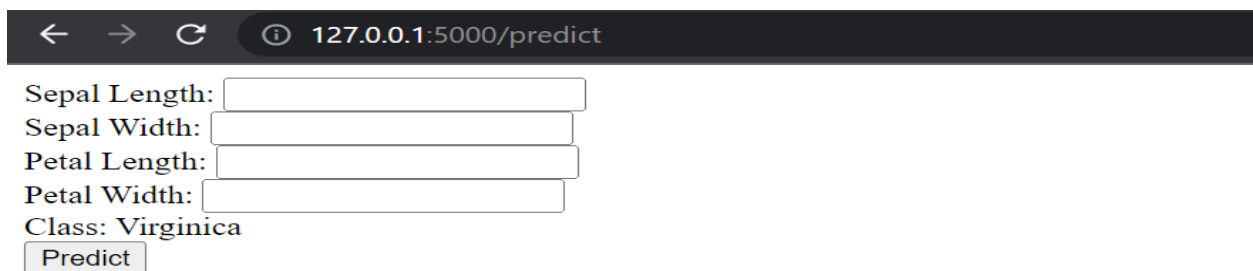


Now we could open a web browser and navigate to <http://127.0.0.1:5000/> we should see a simple website.



A screenshot of a web browser window. The address bar shows '127.0.0.1:5000'. The page contains a form with four input fields labeled 'Sepal Length:', 'Sepal Width:', 'Petal Length:', and 'Petal Width:'. Below these fields is a 'Class:' label and a 'Predict' button.

After entering the input click the predict button now, we can see the result of our input.



A screenshot of the same web browser window after clicking the 'Predict' button. The address bar now shows '127.0.0.1:5000/predict'. The form fields are still present, but the 'Class:' label now displays the text 'Virginica'. The 'Predict' button remains at the bottom.