DATA SCIENCE INTERN AT DATA GLACIER

WEEK 4: DEPLOYMENT ON FLASK

Name: Shubh Goyal

Batch Code: LISUM20

Date: 28 April 2022

Submitted to: Data Glacier

Table Of Content

INTRODUCTION

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

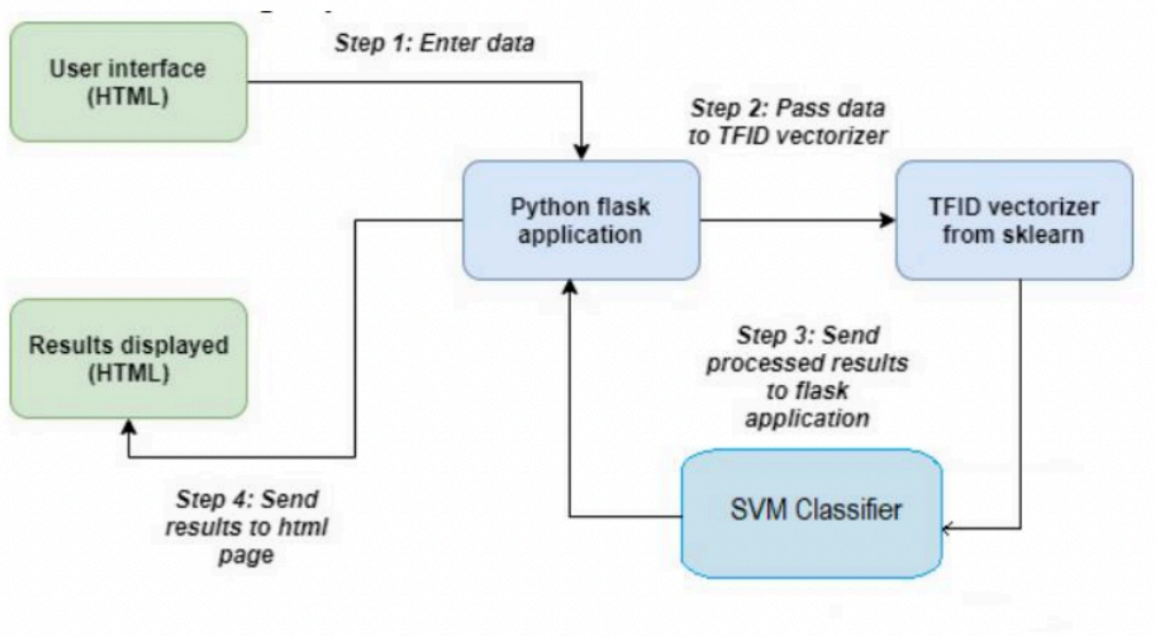
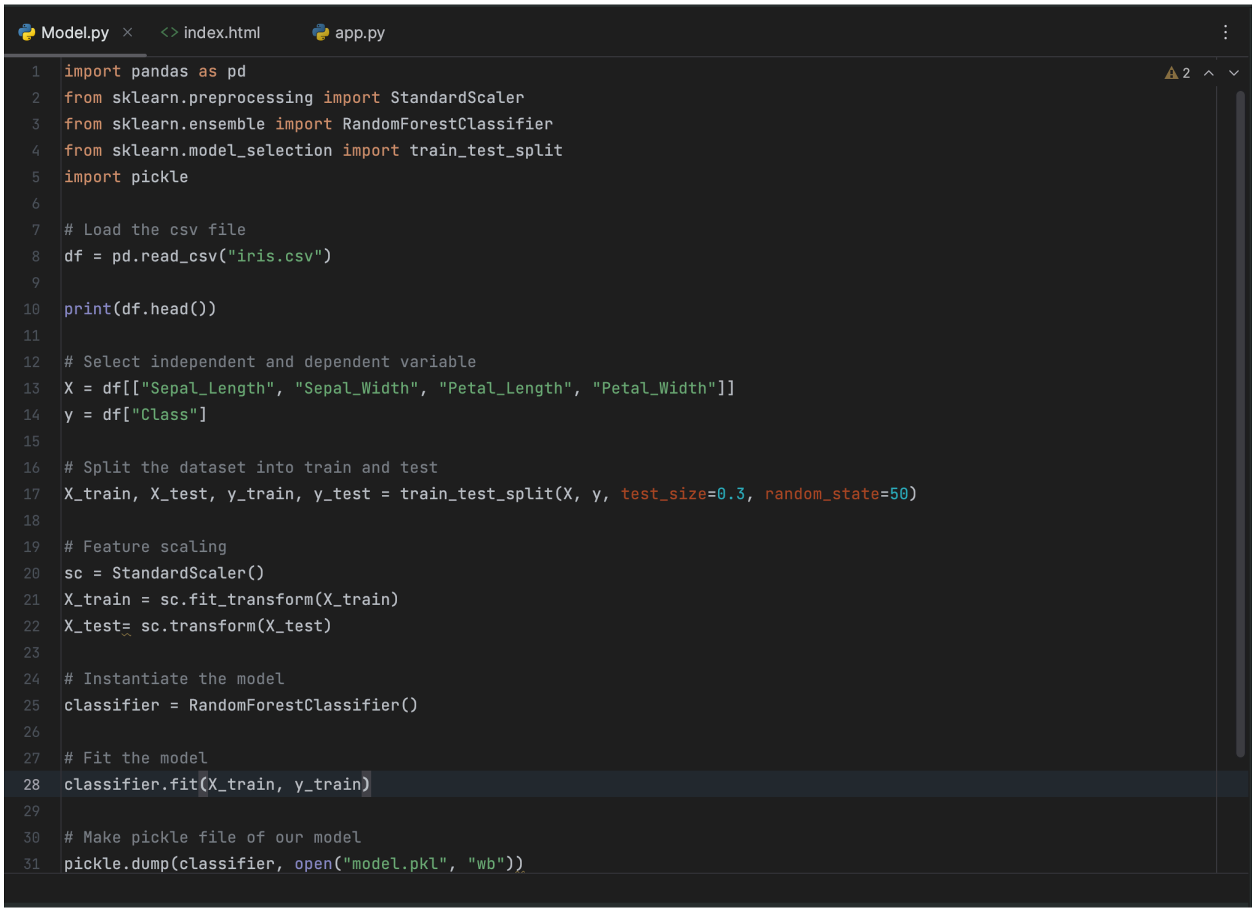

User interface (HTML)
Step 1: Enter data
Step 2: Pass data to TFID vectorizer
Python flask application
TFID vectorizer from sklearn
Results displayed (HTML)
Step 3: Send processed results to flask application
SVM Classifier
Step 4: Send results to html page

Figure 1.1: Application Workflow

we will focus on both: building a machine learning model, then 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.

BUILDING MACHINE LEARNING MODEL



```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle

# Load the csv file
df = pd.read_csv("iris.csv")

print(df.head())

# Select independent and dependent variable
X = df[["Sepal_Length", "Sepal_Width", "Petal_Length", "Petal_Width"]]
y = df["Class"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)

# Instantiate the model
classifier = RandomForestClassifier()

# Fit the model
classifier.fit(X_train, y_train)

# Make pickle file of our model
pickle.dump(classifier, open("model.pkl", "wb"))
```

TURNING MODEL INTO FLASK FRAMEWORK

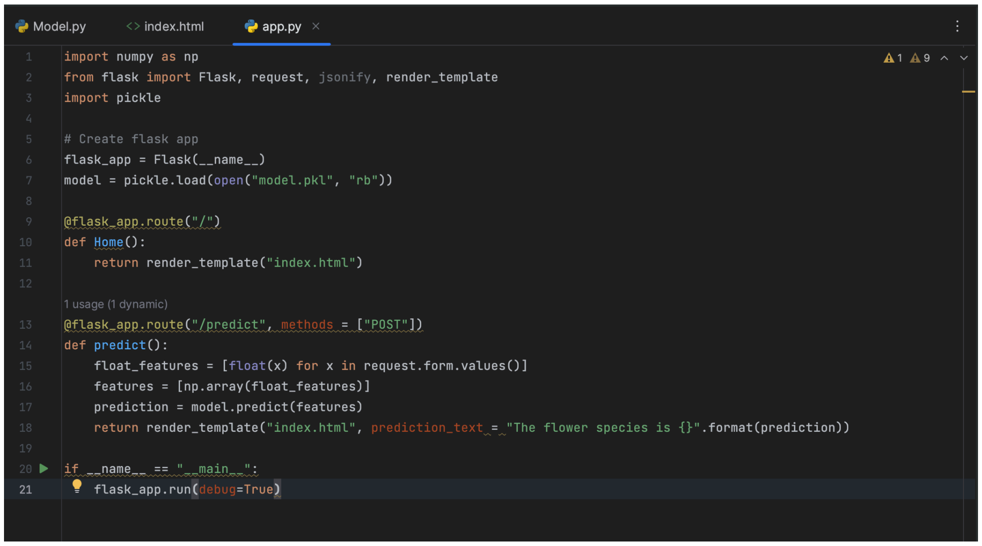
First, we create a folder for this project called YouTube Spam Filtering, this is the directory tree inside the folder. We will explain each file "ML MODEL DEPLOYMENT"

APPLICATION FOLDER FILE DIRECTORY

- App.py
- Templates/
  - Index.html
- Model.pkl
- Iris.csv

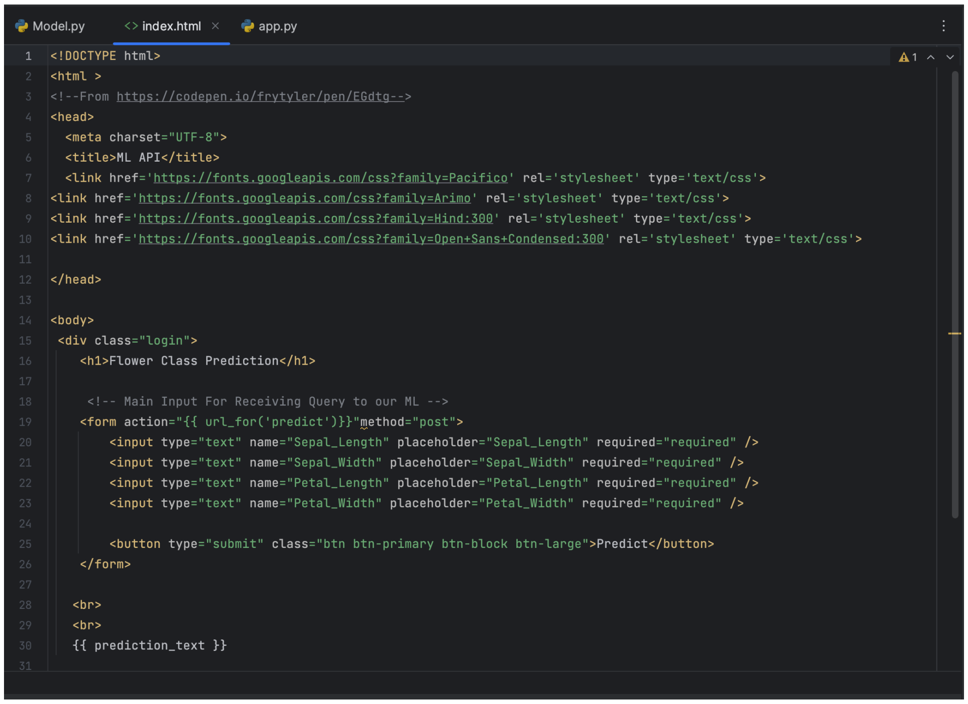
• APP.PY

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.



```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text = "The flower species is {}".format(prediction))

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

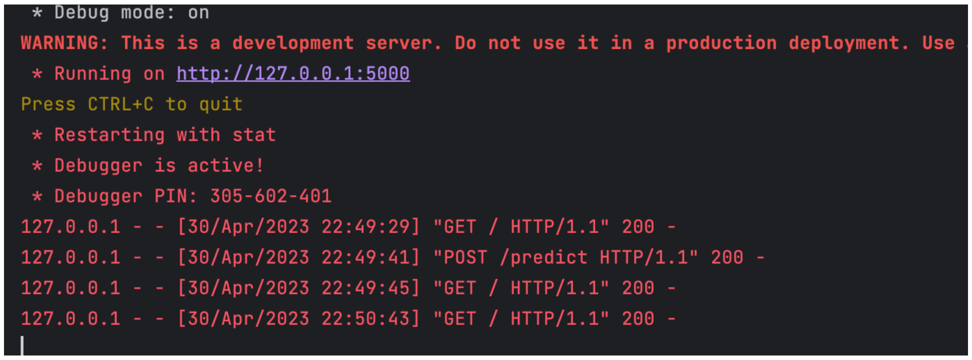
• INDEX.HTML



```html
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<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'>

</head>

<body>
 <div class="login">
    <h1>Flower Class Prediction</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" required="required" />
        <input type="text" name="Sepal_Width" placeholder="Sepal_Width" required="required" />
        <input type="text" name="Petal_Length" placeholder="Petal_Length" required="required" />
        <input type="text" name="Petal_Width" placeholder="Petal_Width" required="required" />

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

   <br>
   <br>
   {{ prediction_text }}

```

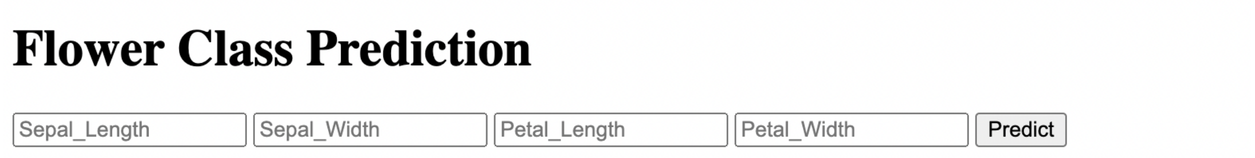
• RUNNING PROCEDURE

Once we have done all of the above, we can start running the API by either double click app.py.



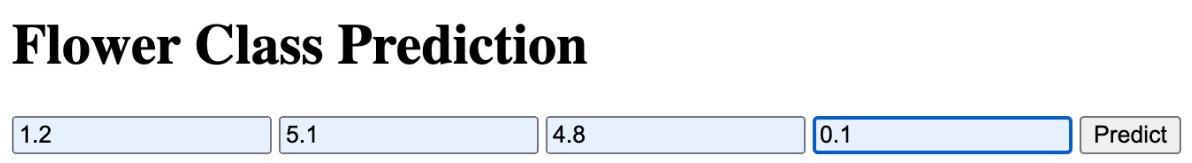
```
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 305-602-401
127.0.0.1 - - [30/Apr/2023 22:49:29] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2023 22:49:41] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2023 22:49:45] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2023 22:50:43] "GET / HTTP/1.1" 200 -
```

When you navigate to " http://127.0.0.1:5000" You will see the below page



Flower Class Prediction

Enter the information:

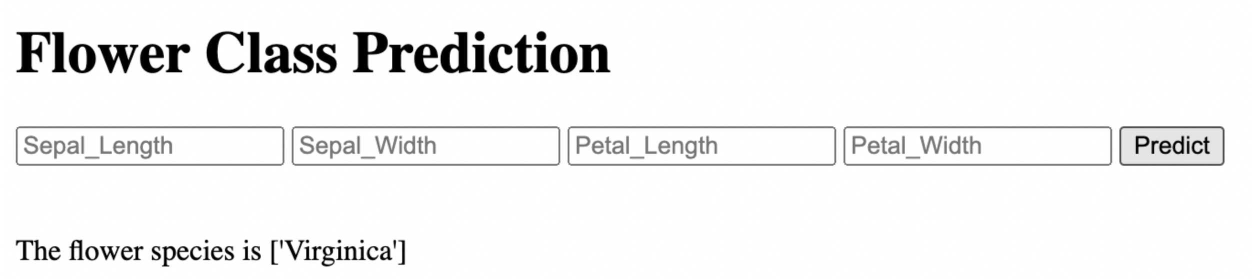


Flower Class Prediction

1.2	5.1	4.8	0.1

After you enter the data, click on "Predict" Button

Result:



Flower Class Prediction

The flower species is ['Virginica']