Name: Swapnil Vishwakarma

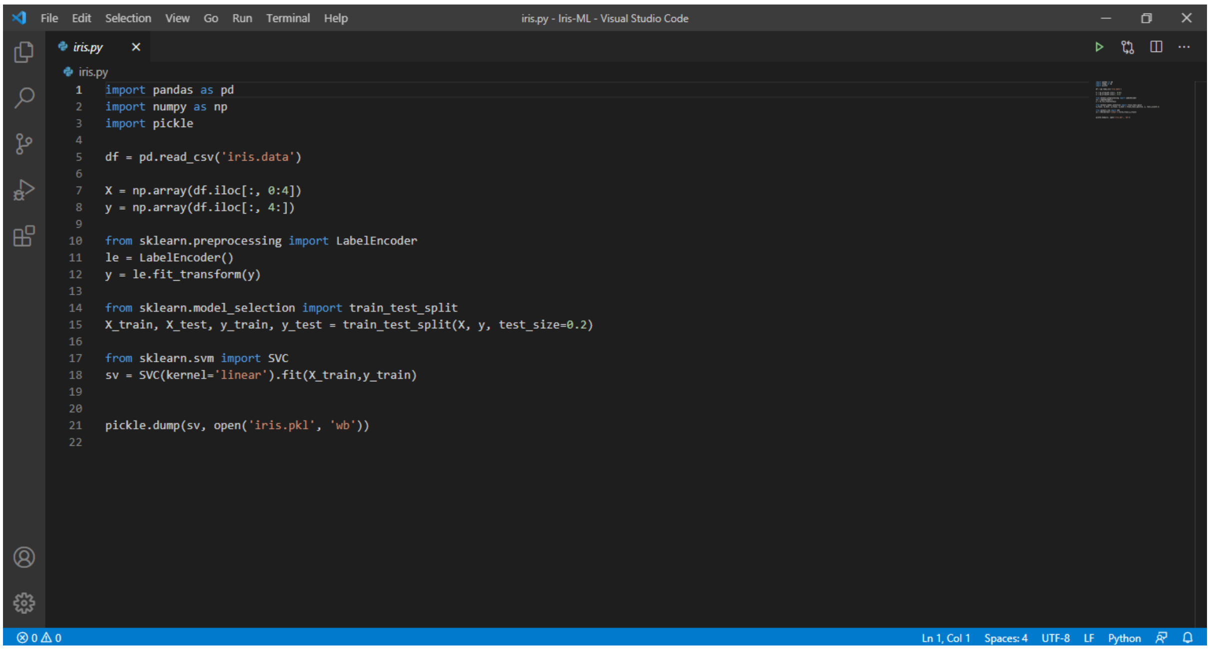
Batch Code: LISP01

Submission Date: 24-03-2021

Submitted to: Data Glacier

DEPLOYMENT PROCESS:

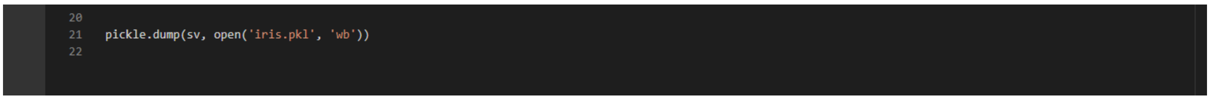
Step 1) Create a Machine Learning Model



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

df = pd.read_csv('iris.data')

X = np.array(df.iloc[:, 0:4])
y = np.array(df.iloc[:, 4:])

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

from sklearn.svm import SVC
sv = SVC(kernel='linear').fit(X_train,y_train)


pickle.dump(sv, open('iris.pkl', 'wb'))

```

I am using the iris dataset from UCI Machine Learning Repository and using Support Vector Classifier to train my model.

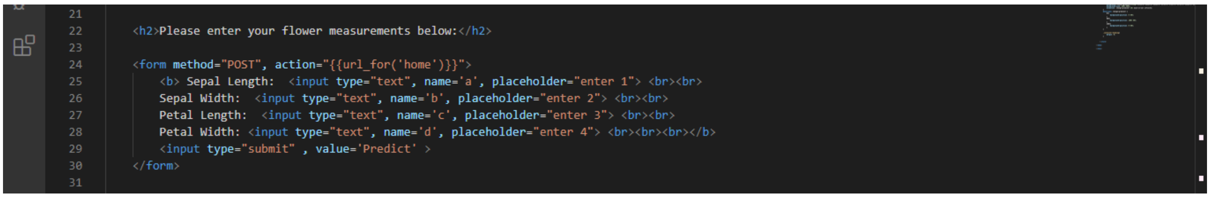
Step 2) Serialization using Pickle



```python

pickle.dump(sv, open('iris.pkl', 'wb'))

```

Using pickle.dump() to perform serialization using python’s inbuilt module pickle.

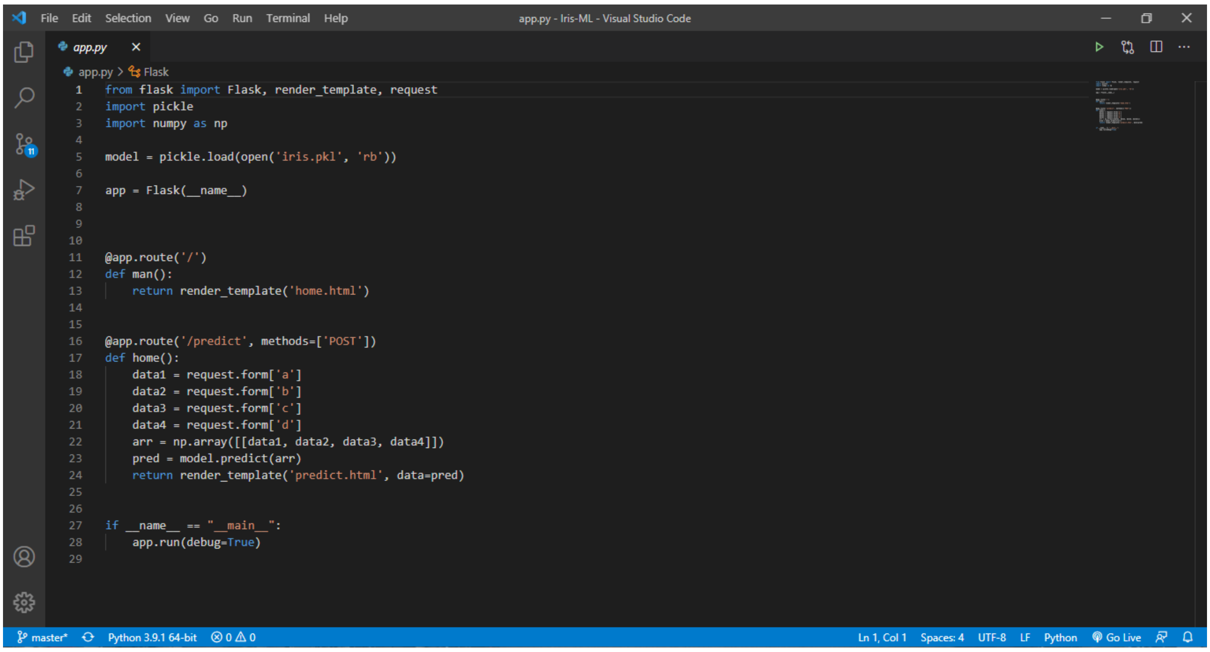
Step 3) Creating HTML Form



```
<h2>Please enter your flower measurements below:</h2>

<form method="POST", action="{{url_for('home')}}">
    <b> Sepal Length:  <input type="text", name='a', placeholder="enter 1"> <br><br>
    Sepal Width:  <input type="text", name='b', placeholder="enter 2"> <br><br>
    Petal Length:  <input type="text", name='c', placeholder="enter 3"> <br><br>
    Petal Width: <input type="text", name='d', placeholder="enter 4"> <br><br><br></b>
    <input type="submit" , value='Predict' >
</form>
```

To predict the class labels, the data is collected from new input values provided in the form and then use the model to predict the output and return the result in the form. Hence, an HTML form is used to display the result in the browser.

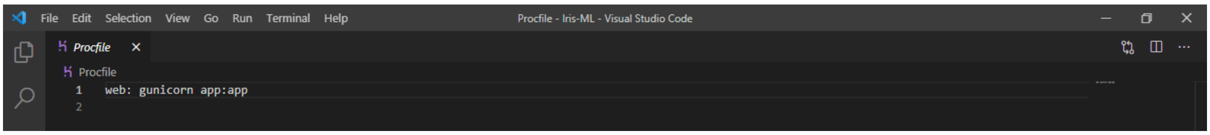
Step 4) Create Flask App



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

model = pickle.load(open('iris.pkl', 'rb'))

app = Flask(__name__)



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


@app.route('/predict', methods=['POST'])
def home():
    data1 = request.form['a']
    data2 = request.form['b']
    data3 = request.form['c']
    data4 = request.form['d']
    arr = np.array([[data1, data2, data3, data4]])
    pred = model.predict(arr)
    return render_template('predict.html', data=pred)


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

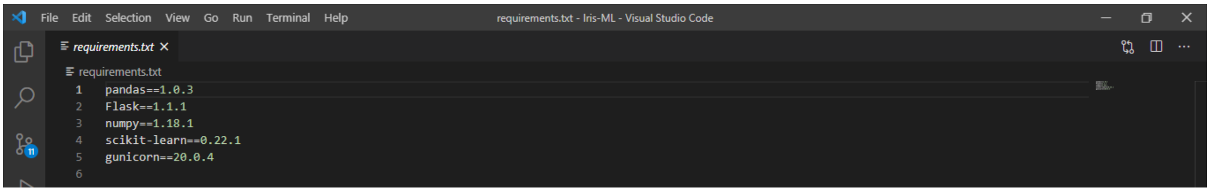
To host the HTML form, a Flask web app is created where the pickle file is read using pickle.load(). A home() fuction is created which takes the input from homepage (HTML homepage), the model will predict the class label and return the result.

Step 5) Create configuration files



Procfile

```
web: gunicorn app:app
```



requirements.txt

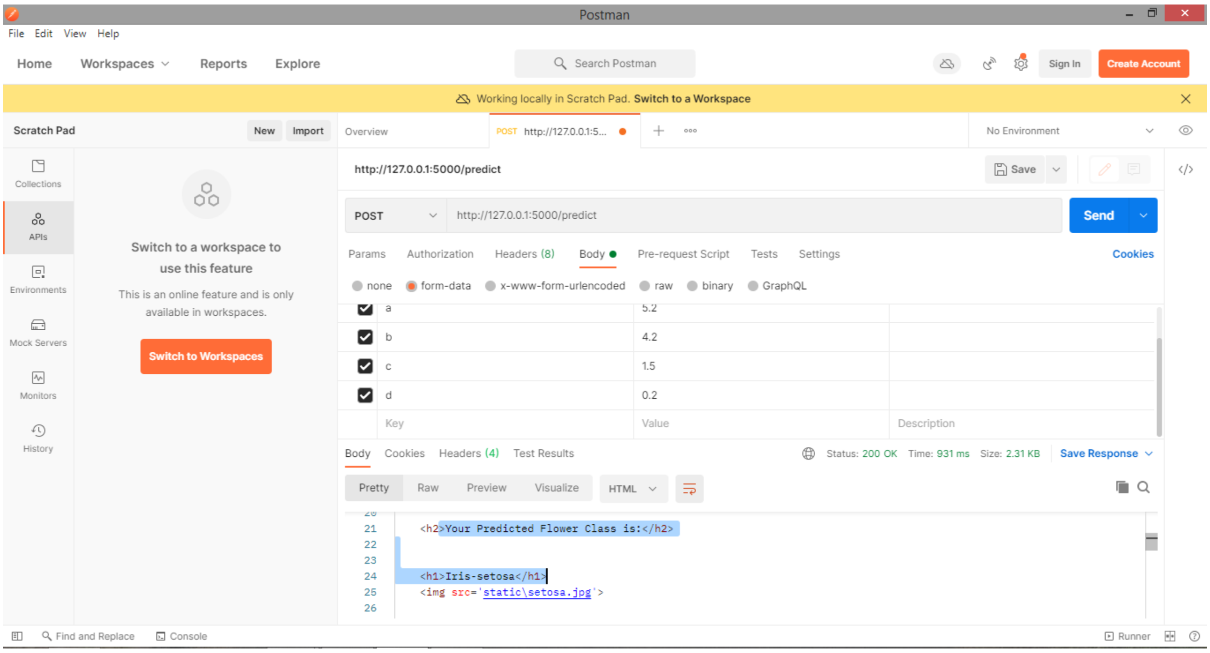
```
pandas==1.0.3
Flask==1.1.1
numpy==1.18.1
scikit-learn==0.22.1
gunicorn==20.0.4
```

For deployment, Procfile and requirements.txt files are created.

Procfile uses Gunicorn which is a pure-Python HTTP server for WSGI applications and it acts as a liaison in between the web application and the webserver.

Requirements.txt file contains all libraries and their dependencies.

Step 6) REST API using Postman



Postman
Working locally in Scratch Pad. Switch to a Workspace
Switch to a workspace to use this feature
This is an online feature and is only available in workspaces.
Switch to Workspaces

POST http://127.0.0.1:5000/predict

Body: ● form-data

	Key	Value	Description
☑	a	5.2	
☑	b	4.2	
☑	c	1.5	
☑	d	0.2	

Status: 200 OK Time: 931 ms Size: 2.31 KB

```
<h2>Your Predicted Flower Class is:</h2>


<h1>Iris-setosa</h1>
<img src='static\setosa.jpg'>
```

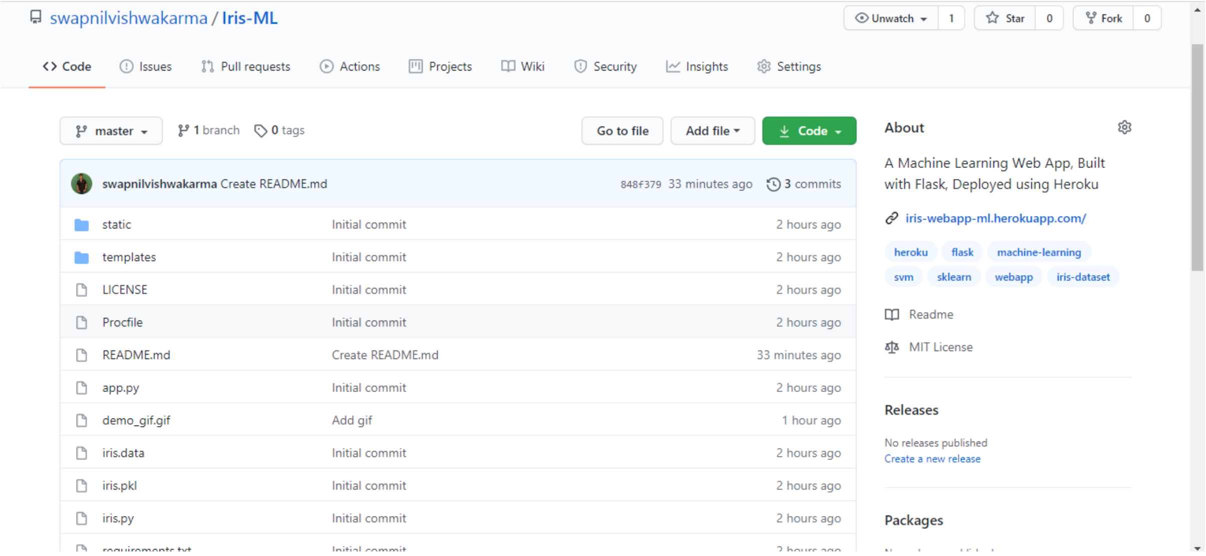
In the terminal run app.py and wait until it is up and running in your localhost. Now open Postman and do the following:

a) Change the method to POST
b) Enter localhost:5000/predict as the URL
c) Inside the Body tab choose form-data

d) Enter some key-value for prediction

Now hit Send and you’ll get the prediction back from the model in HTML.

Step 7) Commit files in Github Repo



swapnilvishwakarma Create README.md 848f379 33 minutes ago 3 commits

static	Initial commit	2 hours ago
templates	Initial commit	2 hours ago
LICENSE	Initial commit	2 hours ago
Procfile	Initial commit	2 hours ago
README.md	Create README.md	33 minutes ago
app.py	Initial commit	2 hours ago
demo_gif.gif	Add gif	1 hour ago
iris.data	Initial commit	2 hours ago
iris.pkl	Initial commit	2 hours ago
iris.py	Initial commit	2 hours ago
requirements.txt	Initial commit	2 hours ago

About

A Machine Learning Web App, Built with Flask, Deployed using Heroku

iris-webapp-ml.herokuapp.com/

heroku flask machine-learning svm sklearn webapp iris-dataset

Readme

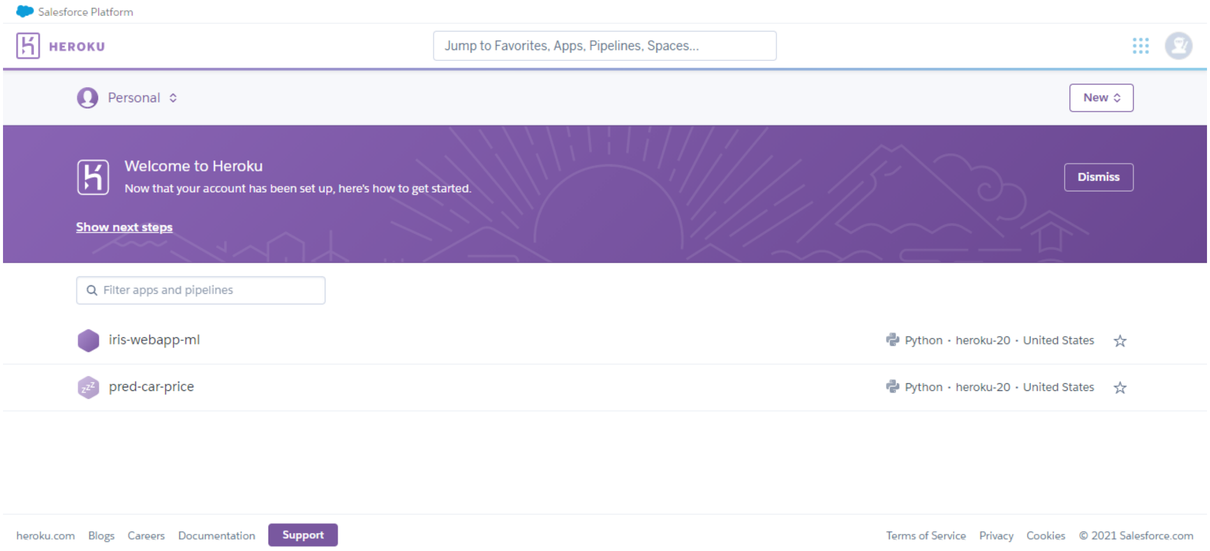
MIT License

Releases

No releases published
Create a new release

Packages

Step 8) Link Github Repo to Heroku and Deploy



Welcome to Heroku

Now that your account has been set up, here's how to get started.

iris-webapp-ml	Python · heroku-20 · United States
pred-car-price	Python · heroku-20 · United States

After creating a free account on Heroku, connect it to your Github.

To deploy a new app, click on create new app and connect to the Github repo which you want to deploy and click deploy branch. Now the web app is ready publically available.