CSE5121 - DevOps & MLOps

Activity 2

Create an ML model for the diabetes data and deploy using Docker.

Prerequisites:

VS Code
VS Code Extension – Python
Docker
Docker Hub

Step 1

Creation of ML Model and Pickle(.pkl) file

Note: After the successful completion, model.pkl will be generated.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pickle

df=pd.read_csv('diabetes.csv')

X=df.iloc[:,:-1]
y=df.iloc[:,-1]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)

classifier=RandomForestClassifier()
classifier.fit(X_train,y_train)

y_pred=classifier.predict(X_test)

from sklearn.metrics import accuracy_score
score=accuracy_score(y_test,y_pred)


print(score)

pickle_out = open("classifier.pkl","wb")
pickle.dump(classifier, pickle_out)
pickle_out.close()

classifier.predict([[2,3,4,1]])
```

Step 2

Creation of UI and Web Framework using Swagger

```
from flask import Flask, request
import numpy as np
import pickle
import pandas as pd
import flasgger
from flasgger import Swagger

app=Flask(__name__)
Swagger(app)

pickle_in = open("classifier.pkl","rb")
classifier=pickle.load(pickle_in)

@app.route('/')
def welcome():
    return "Welcome All"

@app.route('/predict',methods=["Get"])
def predict_note_authentication():

    """Diabetes predictor
    This is using docstrings for specifications.
    ---
    parameters:
      - name: Glucose
        in: query
        type: number
        required: true
      - name: Bp
        in: query
        type: number
        required: true
      - name: Insulin
        in: query
        type: number
        required: true
      - name: BMI
        in: query
        type: number
        required: true
    responses:
        200:
            description: The Prediction is

    """
    Glucose=request.args.get("Glucose")
    Bp=request.args.get("Bp")
    Insulin=request.args.get("Insulin")
    BMI=request.args.get("BMI")
    prediction=classifier.predict([[Glucose,Bp,Insulin,BMI]])
    print(prediction)
    return "Prediction is "+str(prediction)

@app.route('/predict_file',methods=["POST"])
def predict_note_file():
    """Diabetes predictor
    This is using docstrings for specifications.
    ---
    parameters:
      - name: file
        in: formData
        type: file
        required: true

    responses:
        200:
            description: The Prediction is

    """
    df_test=pd.read_csv(request.files.get("file"))
    print(df_test.head())
    prediction=classifier.predict(df_test)

    return str(list(prediction))

if __name__=='__main__':
    app.run(host='0.0.0.0',port=8000)
```

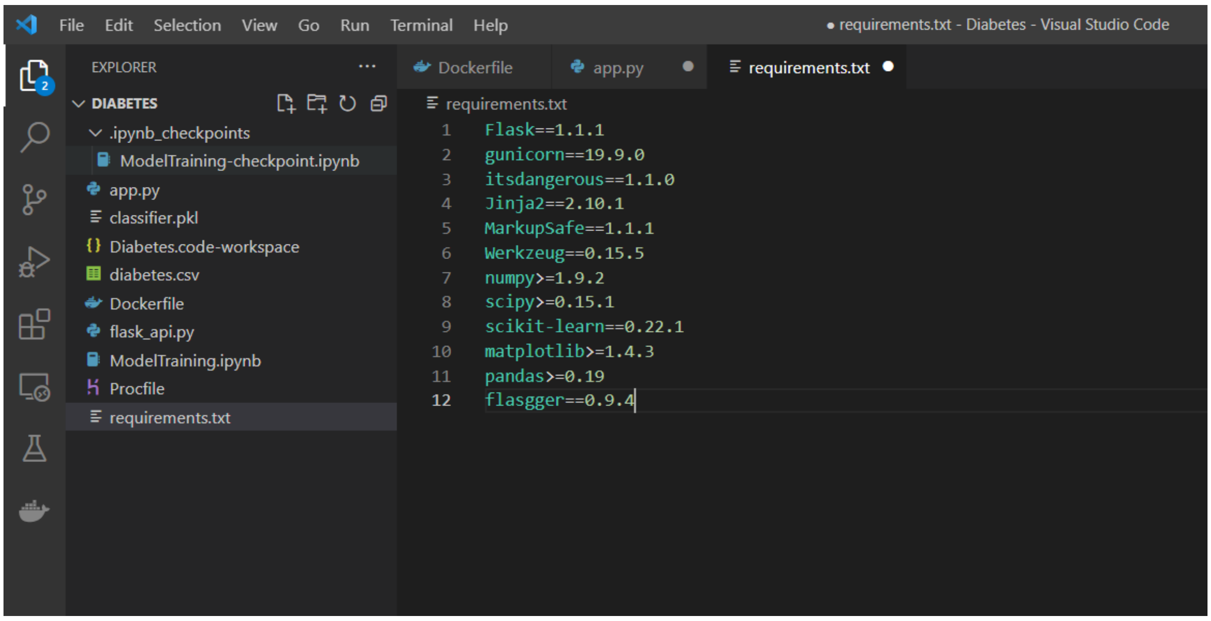
Step 3

Generate requirement file

Note
a. Run the below script in VS Code terminal window of your project directory.
b. After the successful completion, it generates a requirements.txt file.

i.pip install pipreqs

ii. pipreqs .



```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy>=1.9.2
scipy>=0.15.1
scikit-learn==0.22.1
matplotlib>=1.4.3
pandas>=0.19
flasgger==0.9.4
```

Step 4

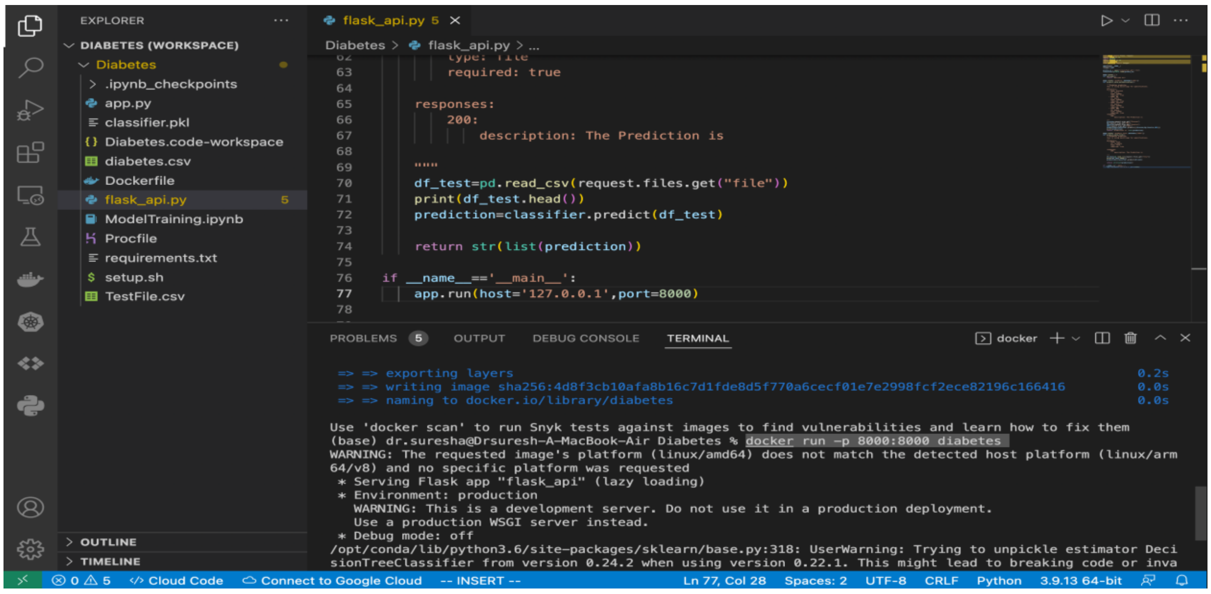
Creation of Docker file.

```
FROM continuumio/anaconda3:4.4.0
COPY . /usr/app/
EXPOSE 5000
WORKDIR /usr/app/
RUN pip install -r requirements.txt
CMD python flask_api.py
```

Step 5

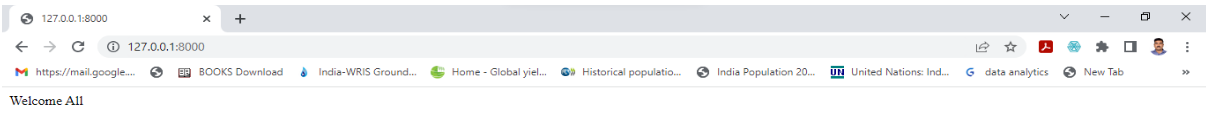
Run the following script in VS Code terminal

```
docker run -p 8000:8000 diabetes
```

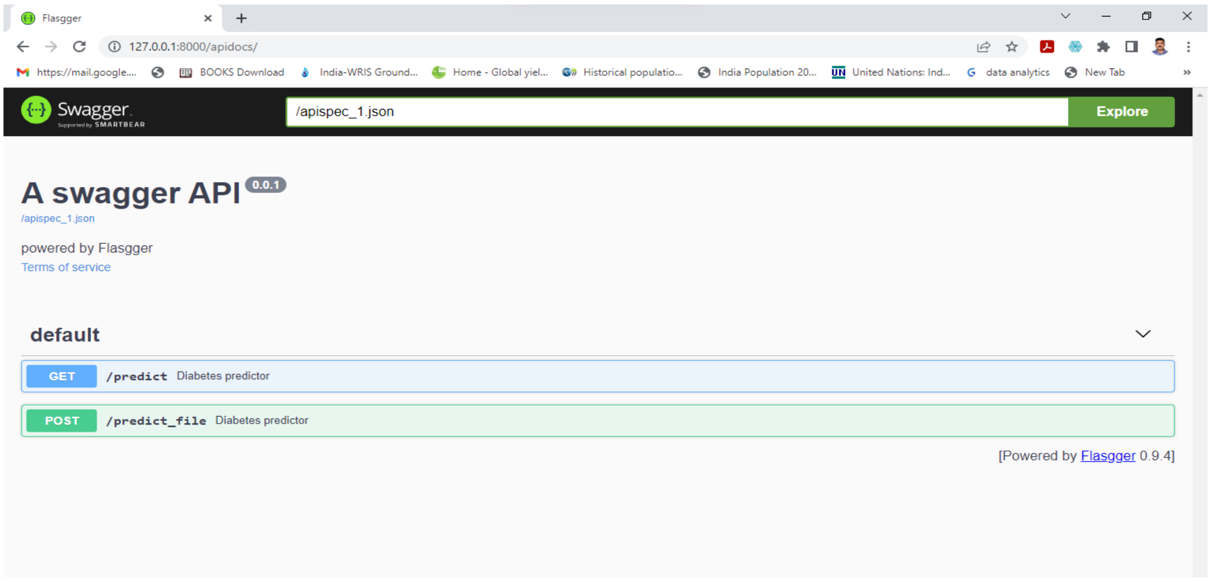


```
            required: true

        responses:
            200:
                description: The Prediction is

        """
        df_test=pd.read_csv(request.files.get("file"))
        print(df_test.head())
        prediction=classifier.predict(df_test)

        return str(list(prediction))

    if __name__=='__main__':
        app.run(host='127.0.0.1',port=8000)
```

```
 => => exporting layers                                                                        0.2s
 => => writing image sha256:4d8f3cb10afa8b16c7d1fde8d5f770a6cecf01e7e2998fcf2ece82196c166416  0.0s
 => => naming to docker.io/library/diabetes                                                    0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) dr.suresha@Drsuresh-A-MacBook-Air Diabetes % docker run -p 8000:8000 diabetes
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm
64/v8) and no specific platform was requested
 * Serving Flask app "flask_api" (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: off
/opt/conda/lib/python3.6/site-packages/sklearn/base.py:318: UserWarning: Trying to unpickle estimator Deci
sionTreeClassifier from version 0.24.2 when using version 0.22.1. This might lead to breaking code or inva
```



Welcome All


Flasgger

127.0.0.1:8000/apidocs/

Swagger

/apispec_1.json

Explore

A swagger API 0.0.1

/apispec_1.json

powered by Flasgger

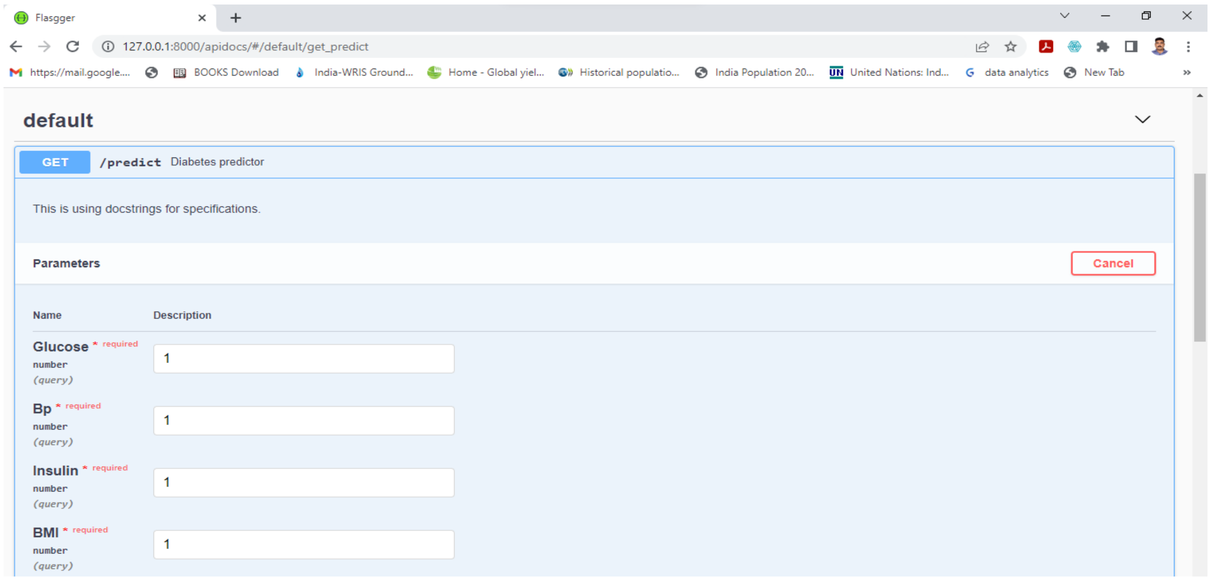
Terms of service

default

GET /predict Diabetes predictor

POST /predict_file Diabetes predictor

[Powered by Flasgger 0.9.4]


Flasgger

127.0.0.1:8000/apidocs/#/default/get_predict

default

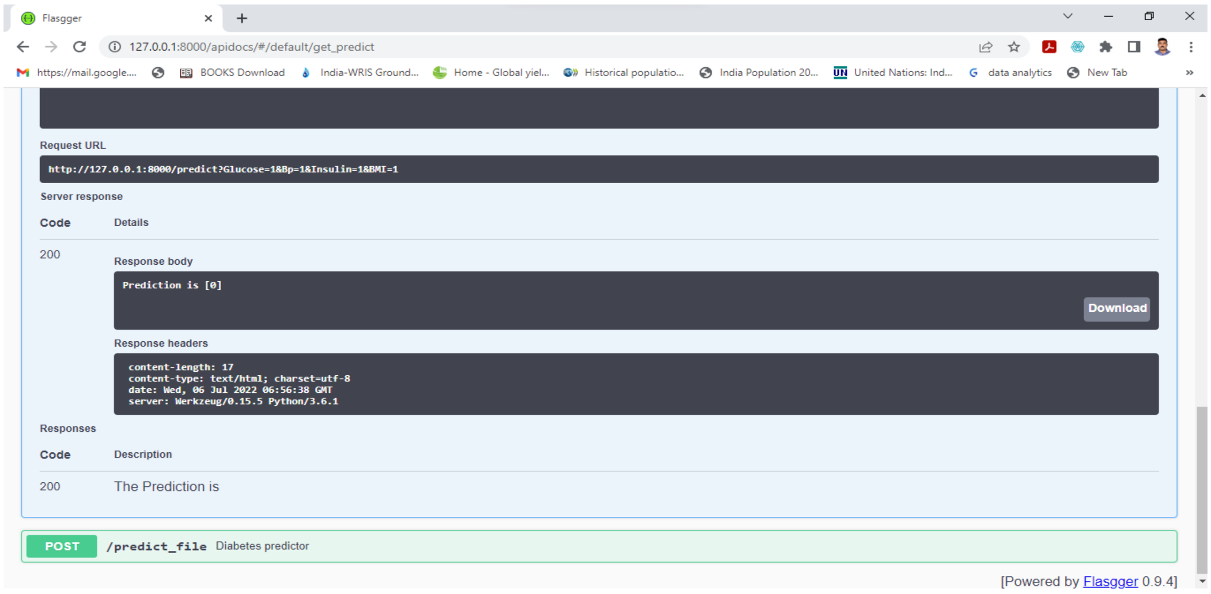
GET /predict Diabetes predictor

This is using docstrings for specifications.

Parameters

Cancel

Name	Description
Glucose * required number (query)	1
Bp * required number (query)	1
Insulin * required number (query)	1
BMI * required number (query)	1


Request URL

```
http://127.0.0.1:8000/predict?Glucose=1&Bp=1&Insulin=1&BMI=1
```

Server response

Code	Details
200	Response body

```
Prediction is [0]
```

Response headers

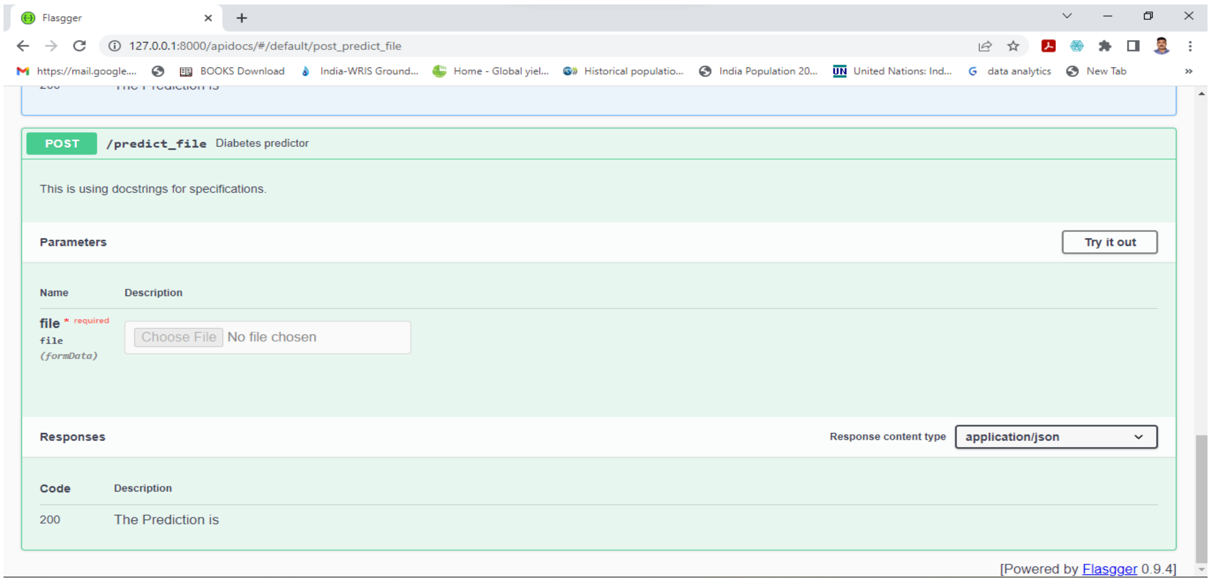
```
content-length: 17
content-type: text/html; charset=utf-8
date: Wed, 06 Jul 2022 06:56:38 GMT
server: Werkzeug/0.15.5 Python/3.6.1
```

Responses

Code	Description
200	The Prediction is

POST `/predict_file` Diabetes predictor

[Powered by Flasgger 0.9.4]


POST `/predict_file` Diabetes predictor

This is using docstrings for specifications.

Parameters

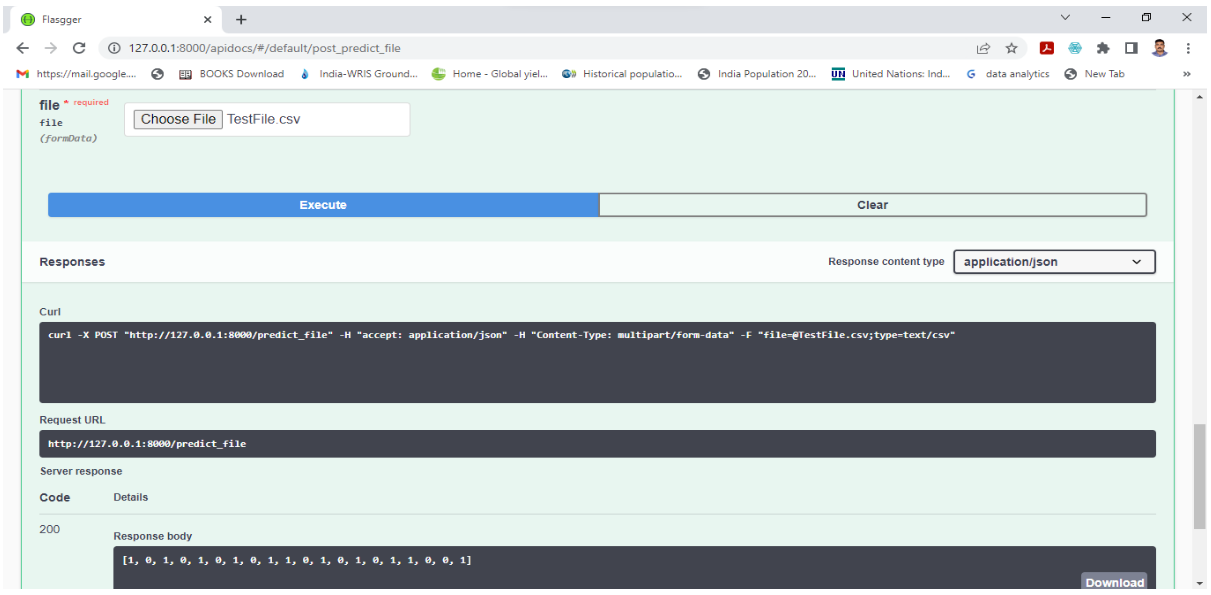
Name	Description
file * required file *(formData)*	Choose File No file chosen

Responses

Response content type: application/json

Code	Description
200	The Prediction is

[Powered by Flasgger 0.9.4]



file * required
file
(formData)

Choose File TestFile.csv

Execute | Clear

Responses

Response content type: application/json

Curl

```
curl -X POST "http://127.0.0.1:8000/predict_file" -H "accept: application/json" -H "Content-Type: multipart/form-data" -F "file=@TestFile.csv;type=text/csv"
```

Request URL

```
http://127.0.0.1:8000/predict_file
```

Server response

Code	Details
200	Response body

```
[1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1]
```

Download