

Deployment on Flask

Cancer Detection App

Name: Maviya Shaikh

Batch code: LISUM02

Submission Date: 15th August, 2021

Submitted to: Data Glacier

Table of Contents

1. Purpose	3
2. Dataset Summary.....	3
3. Exploratory Data Analysis	3
4. Deployment.....	4
Model Building.....	4
App.py file	5
Running the App	5
User Interface	6
HTTP verbs	7
5. Conclusion.....	7

1. Purpose

The purpose of this project is to develop an application that helps to detect whether the patient has Type I or Type II cancer of stage 1 or patient does not have cancer. The objective of this project is to provide a solution to Hospitals that can alert the patient at potential cancer risk.

2. Dataset Summary

The data has been originated from Princess Margret Hospital. It is the Clinical Research Unit for Cancer. So, it is assumed that the data source is reliable, and all the data provided is correct.

Below is a brief overview of the dataset:

- The dataset has 10 variables in which 9 variables are independent and represent the structure of the cell in numerical values, whereas 1 variable is dependent and represent a class with categorical value (i.e., 0 describe that patient is not detected with cancer, while 1 (Type I) and 2 (Type II) describe that patient is detected with stage 1 cancer)
- The total number of observations in the dataset is 1690 from which 10 observations (or 100 cells) data are missing
- It is evident that nine independent variables are highly correlated with each other as all the cells for particular observation came from the same body

3. Exploratory Data Analysis

- With the help of **pandas profiling**, it was found that the dataset has a total of 1690 observations with 100 missing cells and 433 duplicate rows. In addition, all the independent variables are numerical continuous values whereas the dependent variable is 0, 1, or 2. Also, it shows the warning that most of the variables are highly correlated and the class has unbalanced data for no cancer and cancer.
Note: The duplicate rows will not be removed because it seems to be duplicate records, but it is original data of different patients
- With the help of **boxplot**, it is determined that the dataset contain outliers and the distribution of the dataset is not normal
- With the help of **heatmap**, the correlation between variables was determined and it was found that except V3 and V6 all other independent variables are highly correlated with **more than 90% correlation**

4. Deployment

Model Building

The logistic Regression Model is developed to solve this classification problem of detecting whether a patient has cancer or not. The classification report was generated, and we get an accuracy of 97%.

After creating the model, the model is saved in network-supported form and ready for use in our application.

```
In [12]: ▶ #Train the model
model = LogisticRegression()
model.fit(x_train, y_train) #Training the model
#Test the model
predictions = model.predict(x_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

	precision	recall	f1-score	support
No Cancer	0.96	0.96	0.96	112
Type I Cancer	0.98	1.00	0.99	112
Type II Cancer	0.97	0.96	0.96	112
accuracy			0.97	336
macro avg	0.97	0.97	0.97	336
weighted avg	0.97	0.97	0.97	336

0.9702380952380952

Saving the model in pickle

```
In [13]: ▶ pickle.dump(model,open('model.pkl','wb'))
```

The logistic regression model is performing good giving 97% of accuracy as well as the weighted average of precision, recall and f1-score is also 97%.

App.py file

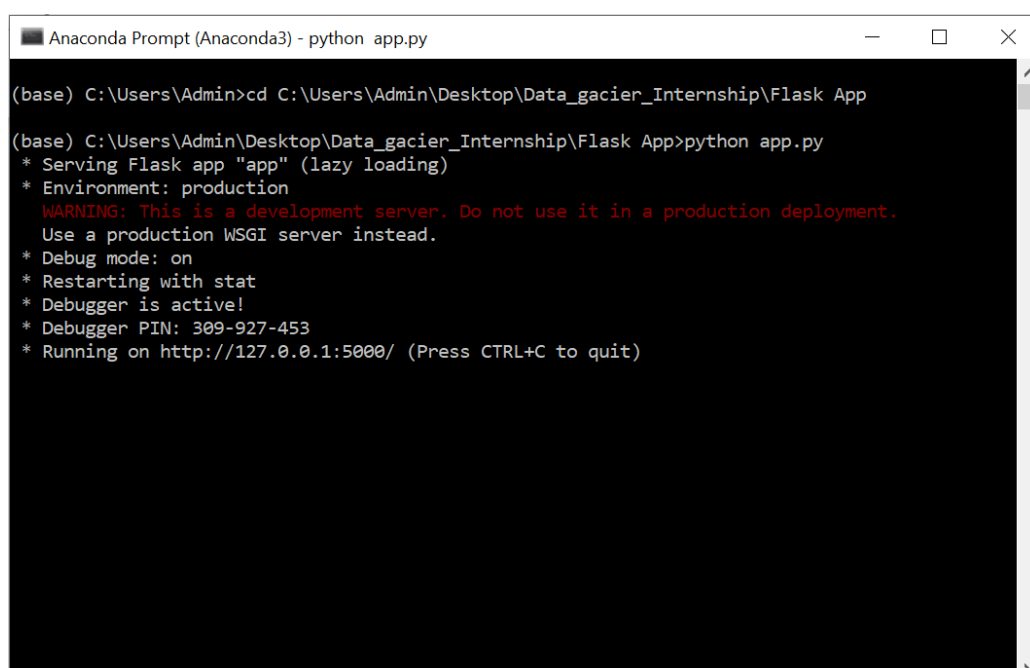
To develop the web application app.py file is created in which the model is loaded to convert it into python supported form. Also, an index.html file is created in which the user can enter cell measurements and this data will be passed to the model to detect cancer.

```
1  # -*- coding: utf-8 -*-
2
3  import numpy as np
4  from flask import Flask, request, jsonify, render_template
5  import pickle
6
7  app = Flask(__name__)
8  model = pickle.load(open('model.pkl', 'rb'))
9
10 @app.route('/')
11 def home():
12     return render_template('index.html')
13
14 @app.route('/predict', methods=['POST'])
15 def predict():
16     '''
17     For rendering results on HTML GUI
18     '''
19     int_features = [float(x) for x in request.form.values()]
20
21     final_features = [np.array(int_features)]
22     prediction = model.predict(final_features)
23
24     output = prediction[0]
25
26     return render_template('index.html', prediction_text='Type of Cancer: {}'.format(output))
27
28
29 if __name__ == "__main__":
30     app.run(debug=True)
```

Two methods are created; home will by default show the home page and the predict method will take the data from the user and return the text for the type of cancer.

Running the App

In the command prompt, the **python app.py** command is run and it is giving the location of the local host on which our application will run which is as shown below:



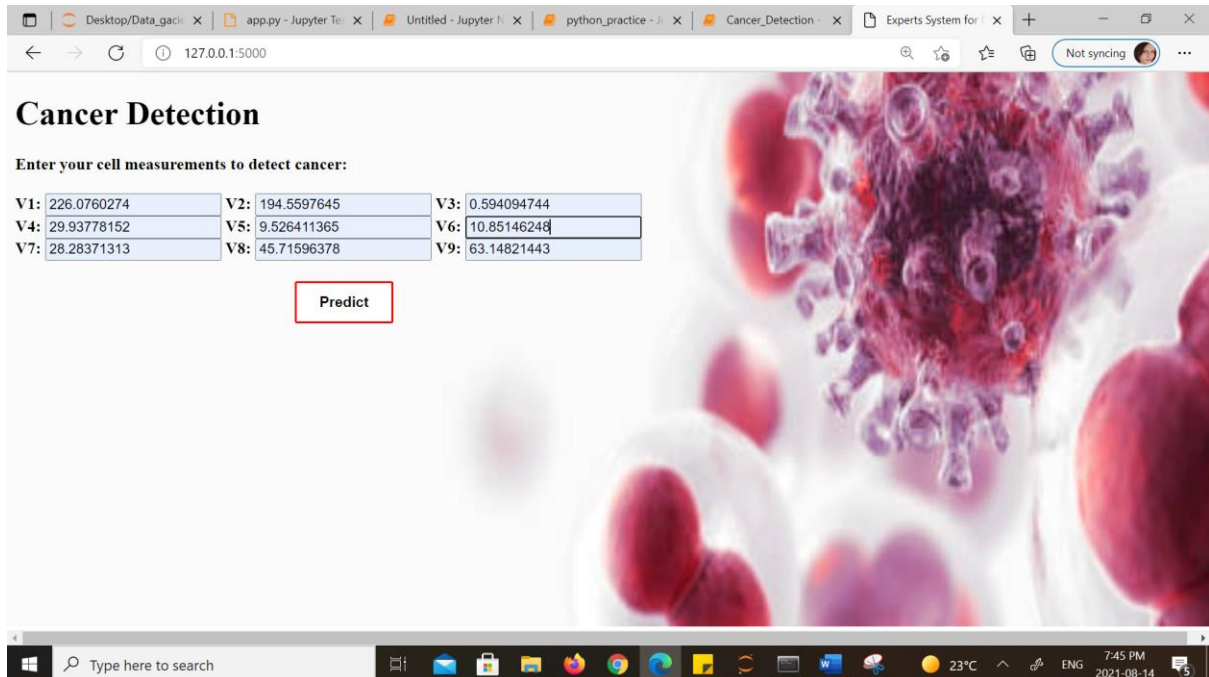
```
Anaconda Prompt (Anaconda3) - python app.py

(base) C:\Users\Admin>cd C:\Users\Admin\Desktop\Data_gacier_Internship\Flask App

(base) C:\Users\Admin\Desktop\Data_gacier_Internship\Flask App>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 stat
* Debugger is active!
* Debugger PIN: 309-927-453
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

User Interface

The app is running on the localhost, and I entered cell measurements for V1 to V9 based on the dataset. This page will first call the home method of the app.py file.

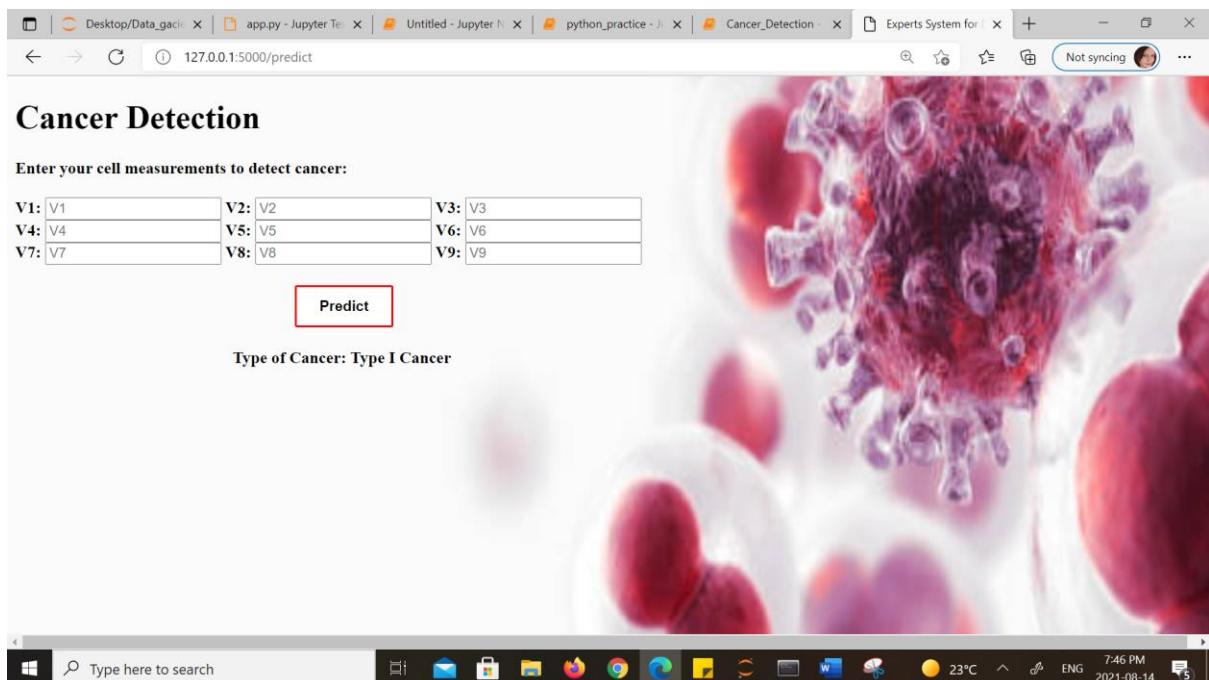


The screenshot shows a web browser window with the URL `127.0.0.1:5000`. The page is titled "Cancer Detection" and has a background image of a virus. Below the title, it says "Enter your cell measurements to detect cancer:". There are nine input fields arranged in a 3x3 grid, labeled V1 through V9. The values entered are: V1: 226.0760274, V2: 194.5597645, V3: 0.594094744, V4: 29.93778152, V5: 9.526411365, V6: 10.85146248, V7: 28.28371313, V8: 45.71596378, and V9: 63.14821443. Below the input fields is a red "Predict" button.

V1:	226.0760274	V2:	194.5597645	V3:	0.594094744
V4:	29.93778152	V5:	9.526411365	V6:	10.85146248
V7:	28.28371313	V8:	45.71596378	V9:	63.14821443

Predict

As soon as I hit the predict button, it will send the data to the predict method in which the model will be run and display the result of the prediction. In our case, it is Type I Cancer.



The screenshot shows the same web browser window, but the URL is now `127.0.0.1:5000/predict`. The page is still titled "Cancer Detection" and has the same background image. The input fields are now empty and labeled V1 through V9. Below the input fields is a red "Predict" button. Below the button, the text "Type of Cancer: Type I Cancer" is displayed.

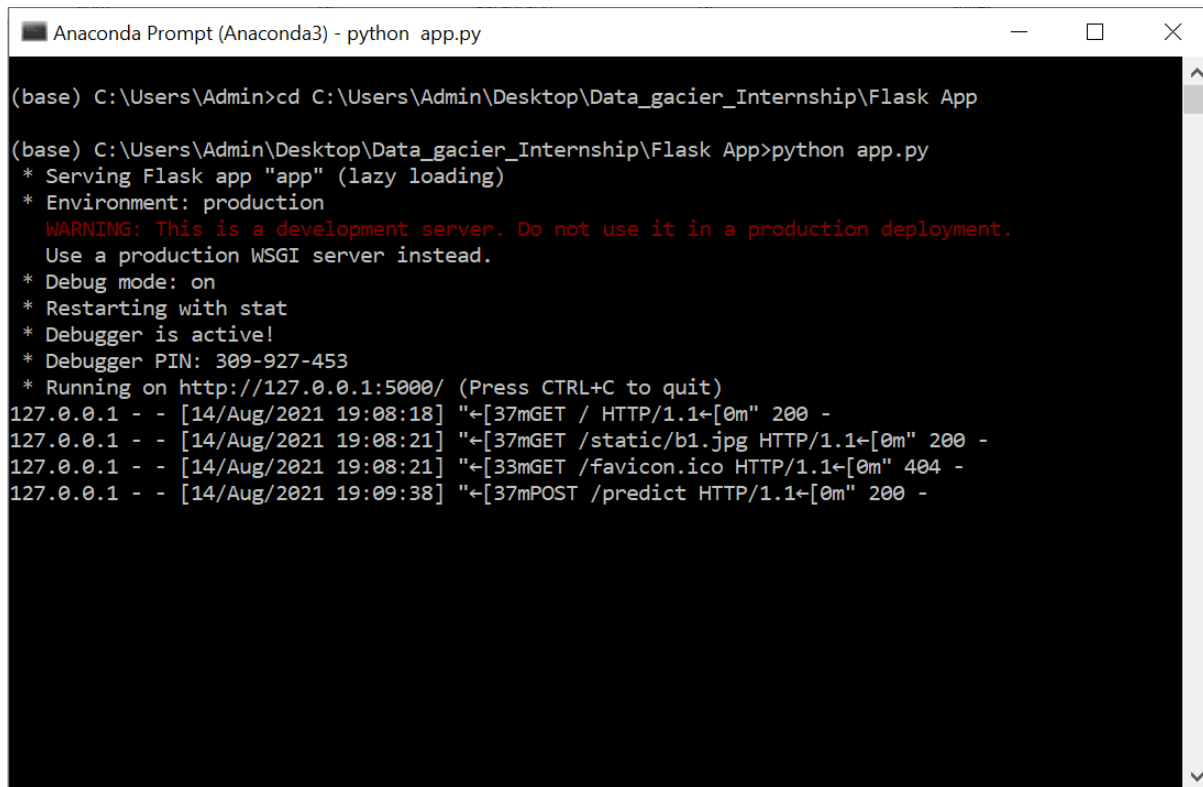
V1:	V1	V2:	V2	V3:	V3
V4:	V4	V5:	V5	V6:	V6
V7:	V7	V8:	V8	V9:	V9

Predict

Type of Cancer: Type I Cancer

HTTP verbs

In the command prompt, we can see all the requests; first, three is the GET request retrieving the data and the last one is the POST method sending the user data.



```
Anaconda Prompt (Anaconda3) - python app.py

(base) C:\Users\Admin>cd C:\Users\Admin\Desktop\Data_gacier_Internship\Flask App

(base) C:\Users\Admin\Desktop\Data_gacier_Internship\Flask App>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 stat
* Debugger is active!
* Debugger PIN: 309-927-453
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [14/Aug/2021 19:08:18] "[37mGET / HTTP/1.1[0m" 200 -
127.0.0.1 - - [14/Aug/2021 19:08:21] "[37mGET /static/b1.jpg HTTP/1.1[0m" 200 -
127.0.0.1 - - [14/Aug/2021 19:08:21] "[33mGET /favicon.ico HTTP/1.1[0m" 404 -
127.0.0.1 - - [14/Aug/2021 19:09:38] "[37mPOST /predict HTTP/1.1[0m" 200 -
```

5. Conclusion

The flask app is successfully developed with all of its necessary components, and it is correctly identifying no cancer and type of stage 1 cancer 97% of the time.