



Data Glacier

Your Deep Learning Partner

Data Science Internship

Week 4: Flask Model Deployment Report

By

Name: Sukurat Salam

Email: salamsukurat@gmail.com

Batch No: LISUM24

Date: 17th August 2023

Table of Content

1.0 Introduction	3
2.0 Data Description	3
3.0 Machine Learning Model Implementation	4
4.0 Model Deployment in Flask	5
4.1 Create a Web Page	6
4.2 Flask Application Setup	6
4.3 Running the Application	7
4.4 Access the Web Page	8

1.0 Introduction

This section will give the overview of the task and how this would be achieved. This project will build a model for predicting heart disease attack in patient using some medical features or characteristics that can easily be provided without going through laboratory testing or x-rays. A supervise learning model approach which is suitable for binary classification logistic regression would be used to train the model and will be deploy using flask.

This would be achieved with the following steps.

- Downloading the data
- Data pre-processing
- Training the model
- Deploying the model using Flask
- Building the web application

2.0 Data Description

For this project, 2015 dataset that was pre-processed by (Alex Teboul 2021) was downloaded from Kaggle. The dataset contains 253,680 survey responses from cleaned BRFSS 2015 to be used mainly for the binary classification of heart disease attack. The dataset contains 21 features with 1 binary target variable, but just 5 most important variable for the prediction would be consider for this project.

Source Link: <https://www.kaggle.com/alexteboul/heart-disease-health-indicators-dataset>.

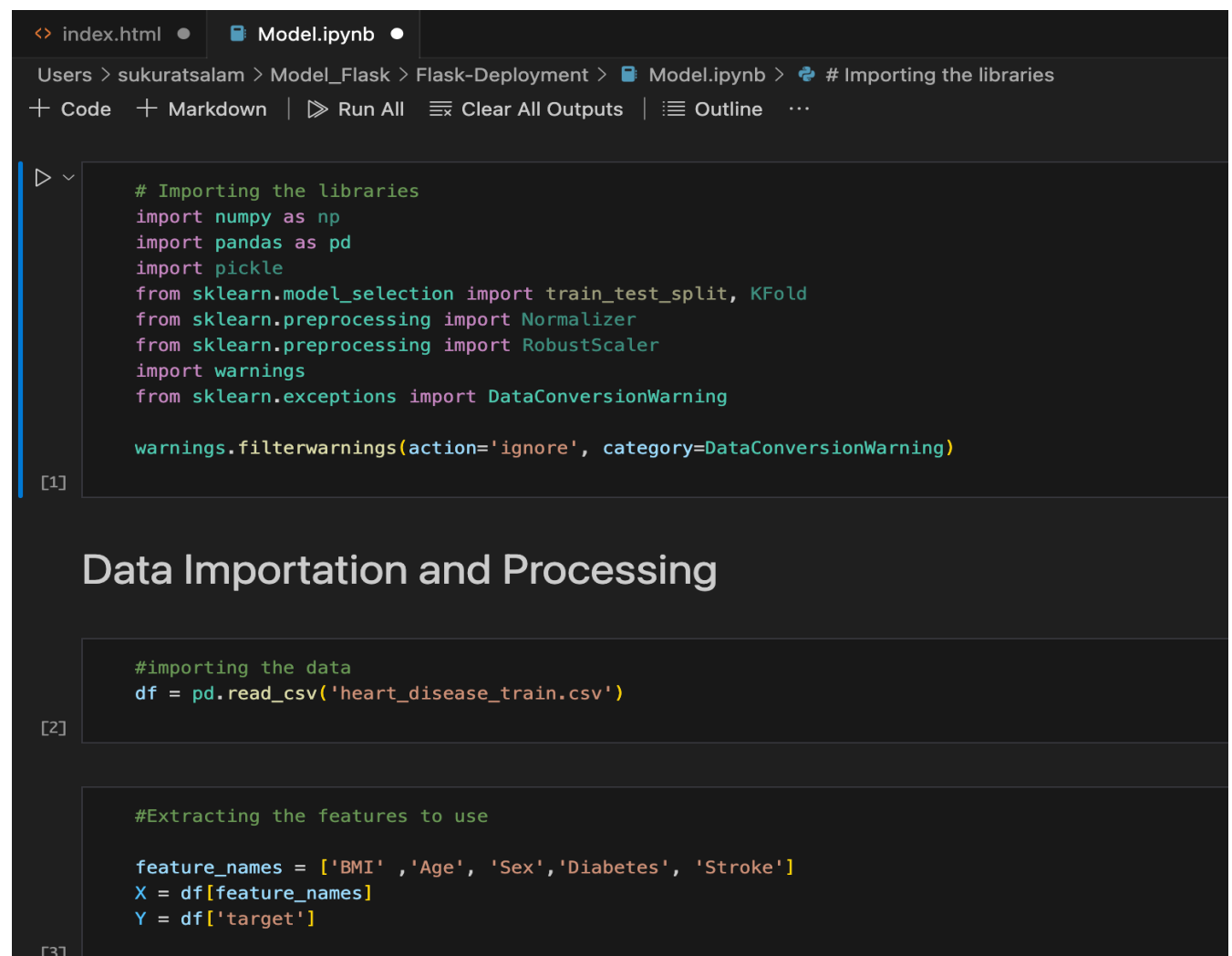
S/N	Variable	Description	Values
1	HeartDiseaseAttack	Respondents that have reported having coronary heart disease or Myocardial Infection	0 for No 1 for Yes
2	BMI	Body Mass Index	Numeric
3	Age	Patient Age Category "1= 18-24, 2= 25-30,3= 31-35, 4= 36-40,5= 41-45,6= 46-50,7= 51-55,8= 56-60,9= 61-65,10= 66-70,11= 71-75,12 =76-80 ,13 = 80 above"	
4	Gender	Patient Sex	0 for Female, 1 for Male
5	Diabetes	"0 for No Diabetes, 1 for Pre-Diabetes, 2 for Diabetes"	0 for No Diabetes, 1 for Pre-Diabetes, 2 for Diabetes
6	Stroke	Ever told you have stroke?	0 for No, 1 for Yes

3.0 Machine Learning Model Implementation

The first step to take when deploying a machine learning model in Flask is to train the model, and for the research, we will employ the use of logistic regression for binary classification of the heart disease attack (if a patient has had it in the past or not).

After training the model, it will be sterilized using a pickle library to save it in the file which will be loaded later when serving predictions. The training of the model and sterilization was done in Jupiter notebook and the screenshot of some parts of the Python code was given below.

3.1 Data Importation and Pre-processing



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar indicates the current file is 'Model.ipynb'. The notebook's breadcrumb trail shows the path: 'Users > sukuratsalam > Model_Flask > Flask-Deployment > Model.ipynb'. Below the breadcrumb, there are tabs for 'Code', 'Markdown', 'Run All', 'Clear All Outputs', and 'Outline'. The first code cell, labeled '[1]', contains the following Python code for importing libraries:

```
# Importing the libraries
import numpy as np
import pandas as pd
import pickle
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import RobustScaler
import warnings
from sklearn.exceptions import DataConversionWarning

warnings.filterwarnings(action='ignore', category=DataConversionWarning)
```

The second code cell, labeled '[2]', contains the following Python code for importing the data:

```
#importing the data
df = pd.read_csv('heart_disease_train.csv')
```

The third code cell, labeled '[3]', contains the following Python code for extracting the features to use:

```
#Extracting the features to use

feature_names = ['BMI', 'Age', 'Sex', 'Diabetes', 'Stroke']
X = df[feature_names]
Y = df['target']
```

3.2 Training the Logistics Regression Model

```
index.html • Model.ipynb •
Users > sukuratsalam > Model_Flask > Flask-Deployment > Model.ipynb > M4Data Importation and Processing > X.head()
+ Code + Markdown | ▶ Run All | ⌵ Clear All Outputs | ⌵ Outline ...
```

Building the Model

```
[16] #feature Scaling of the original dataset
#Splitting the dataset into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X,Y , test_size=0.2, random_state=99)

#Fitting Logistics Regression Algorithm
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver = 'liblinear', random_state=888)
model.fit(X_train, y_train)

#making predictions with the testing data
y_pred = model.predict(X_test)
print(y_pred)
```

3.3 Sterilization of the Model using Pickle

```
[22] ## # import pickle library
import pickle

# save the model (variable) model: LogisticRegression for writing
model1 = open("model.pkl", "wb")
pickle.dump(model1,model) # dumps an object to a file object
model1.close() # here we close the fileObject

with open("model.pkl", "wb") as file:
    pickle.dump(model, file)

[24] # Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))

result = model.predict([[29,13,0,0,0]])
if result == 1:
    print("The patient has Heart disease Attack")
else:
    print("The patient does not have Heart disease Attack")

[34]
```

4.0 Model Deployment in Flask

Deploying a machine learning model with Flask entails multiple phases, beginning with the creation of a web page and ending with the model being served via a Python application.

4.1 Create a Web Page

A basic HTML web page was created to interact with the deployed model, This includes the forms in which the user can input data for prediction, and display the result message after the prediction is done.

```
index.html
Users > sukuratsalam > Model_Flask > Flask-Deployment > templates > index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Heart Disease Attack</title>
6
7
8
9      <!-- <link rel="stylesheet" href="/Users/sukuratsalam/Flask-Deployment/static/css/style1.css"> -->
10     <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
11
12
13 </head>
14
15 <body>
16     <div class="login">
17         <h1>Predicting the Heart Disease Status of Patient</h1>
18
19         <!-- Main Input For Receiving Query to our ML -->
20         <form action="{{ url_for('predict') }}" method="post">
21             Enter the BMI of Patient:<input type="text" name="BMI" placeholder="Body Mass Index" required="required" />
22             <br>
23             Enter the Age Category:<input type="text" name="Age" placeholder="Click the next box for Category value" required="required" />
24             <select>
25                 <option value="1">1 for 18-24</option>
26                 <option value="2">2 for 25-30</option>
27                 <option value="3">3 for 31-35</option>
28                 <option value="4">4 for 36-40</option>
29                 <option value="5">5 for 41-45</option>
30                 <option value="6">6 for 46-50</option>
31                 <option value="7">7 for 51-55</option>
32                 <option value="8">8 for 56-60</option>
33                 <option value="9">9 for 61-65</option>
34                 <option value="10">10 for 66-70</option>
35                 <option value="11">11 for 71-75</option>
36                 <option value="12">12 for 76-80</option>
37                 <option value="13">13 for 80 above</option>
38             </select>
39             <br>
40             Enter the patient Gender:<input type="text" name="Sex" placeholder="0 for Female and 1 for Male" required="required" />
41             <br>
42             Enter Patient Diabetes Status:<input type="text" name="Diabetes" placeholder="0 for No Diabetes, 1 for Pre-Diabetes, 2 for Diabetes" required="required" />
43             <br>
44             Has the patient had stroke before?<input type="text" name="Stroke" placeholder="0 for No and 1 Yes" required="required" />
```

CSS/Style Page

```
Users > sukuratsalam > Model_Flask > Flask-Deployment > static > css > # style.css > body
1 body {
2     font-family: Arial, sans-serif;
3     background-color: #0e0e0e;
4     margin: 0;
5     padding: 0;
6     color: #f99b9b;
7 }
8
9 .login {
10     position: absolute;
11     top: 20%;
12     left: 30%;
13     margin: -150px 0 0 -150px;
14     width: 800px;
15     height: 800px;
16 }
17
18 .login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing: 1px; text-align: center; }
19
20 input {
21     width: 100%;
22     margin-bottom: 10px;
23     background: rgba(238, 200, 200, 0.3);
24     border: none;
25     outline: none;
26     padding: 10px;
27     font-size: 13px;
28     color: #e7c4c4;
29     text-shadow: 1px 1px 1px rgba(235, 221, 221, 0.3);
30     border: 1px solid rgba(215, 195, 195, 0.3);
31     border-radius: 4px;
32     box-shadow: inset 0 -5px 45px rgba(100, 100, 100, 0.2), 0 1px 1px rgba(23, 23, 23, 0.937);
33     transition: box-shadow 0.5s ease;
34 }
35
36 input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px rgba(246, 228, 228, 0.2); }
```

4.2 Flask Application Setup

Flask which is the micro web framework for Python was used to create an application that handles incoming HTTP requests and serves the prediction. This stage involves several steps which include,

- installation of flask
- app structure for organizing the project
- creating the flask app by importing flask to create app.py
- loading the serialized model (**model.pkl**) using pickle
- defining the route for the application, we have a route to render the HTML form, and another one was created to handle the predictions

```

Users > sukuratsalam > Model_Flask > Flask-Deployment > app.py > ...
1  import numpy as np
2  from flask import Flask, request, render_template
3  import pickle
4
5  app = Flask(__name__)
6  model = pickle.load(open('model.pkl', 'rb'))
7
8  @app.route('/')
9  def home():
10     return render_template('index.html')
11
12  @app.route('/predict', methods=['POST'])
13  def predict():
14     '''
15     For rendering results on HTML GUI
16     '''
17     int_features = [int(x) for x in request.form.values()]
18     final_features = [np.array(int_features)]
19     prediction = model.predict(final_features)
20     if prediction == 1:
21         answer = "Yes: The patient has had a heart disease attack in the past"
22     else:
23         answer = "No: The patient does not have a heart disease attack in the past"
24
25     return render_template('index.html', prediction_text= answer)
26
27
28  if __name__ == "__main__":
29     app.run(debug=True)

```

4.3 Running the Application

The application was run in the terminal by navigating to the Flask model- deployment folder and running the code `python app.py` to start the Flask app.

```

Flask-Deployment — python — python app.py — 80x24
(base) sukuratsalam@SUKURATS-MacBook-Pro ~ % cd Model_Flask
(base) sukuratsalam@SUKURATS-MacBook-Pro Model_Flask % cd Flask-Deployment
(base) sukuratsalam@SUKURATS-MacBook-Pro Flask-Deployment % 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
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with watchdog (fsevents)
* Debugger is active!
* Debugger PIN: 102-463-992
127.0.0.1 - - [10/Aug/2023 15:12:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Aug/2023 15:12:15] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [10/Aug/2023 15:12:15] "GET /static/images/heart.png HTTP/1.1" 200 -
127.0.0.1 - - [10/Aug/2023 15:12:16] "GET /favicon.ico HTTP/1.1" 404 -
/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
127.0.0.1 - - [10/Aug/2023 15:12:53] "POST /predict HTTP/1.1" 200 -

```


4.4 Access the Web Page

To visit the web page, open your web browser and navigate to <http://127.0.0.1:5000/>. You can now enter data, submit the form, and view the results.

Predicting the Heart Disease Status of Patient

Enter the BMI of Patient:

Enter the Age Category:

1 for 18-24

Enter the patient Gender:

Enter Patient Diabetes Status:

Has the patient had stroke before?

Predict

Input and Result Page

Predicting the Heart Disease Status of Patient

Enter the BMI of Patient:

Enter the Age Category:

1 for 18-24

Enter the patient Gender:

Enter Patient Diabetes Status:

Has the patient had stroke before?

Predict

Yes: The patient has had a heart disease attack in the past

