Name - Tanmay Umesh Potbhare
Batch Code - LISUM18
Submitted Date – 26th February 2023
Submitted to - GitHub | Data Glacier

ML Model Flask Deployment
Stroke Prediction Using Logistic Regression

Objective: "Deployment of any small- or large-scale ML Model on Flask."

Solution: I have created 4 files:

1. MLModel.py - This python file is the ML model where the dataset was read, exploratory data analysis was carried out and the model was trained using the logistic regression classification algorithm.
2. FlaskApp.py - This python file contains the actual flask web framework code, which includes the imported libraries, the URL route and port number(optional).
3. HomePage.html - This is the frontend web page which will be seen as soon as the local server starts running. This contains all the buttons or Graphical User Interface in a nutshell.
4. Stylesheet.css - This just styles the html page and makes it look good.

Generated file from MLModel.py file:

1. Pickle file - This file is generated from the ML Model file and used to run the and which converts the file into a byte stream sized file into the memory.

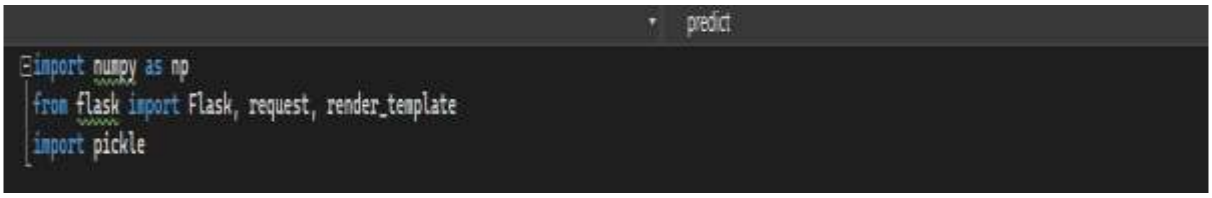
IDE Used - VSCode

Flask Deployment

Creating the Flask Deployment using Python

1. Importing Libraries

Starting with the flask micro web framework, from flask import Flask is the first library to import. The from flask is the module and from that I are importing the Flask class which will be the central registry for all the functions, URL rules, template configuration and so on.



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

Fig 1. Importing Libraries needed for flask deployment

2. Before continuing the flask deployment whatever IDE you use these are the files which are needed.

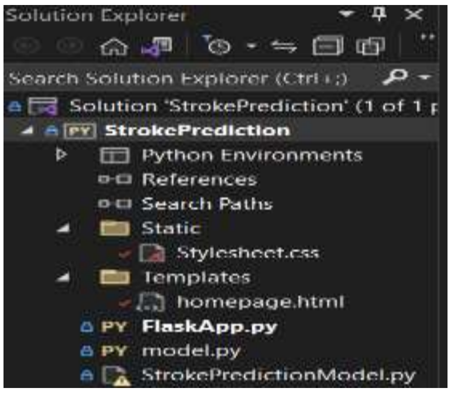


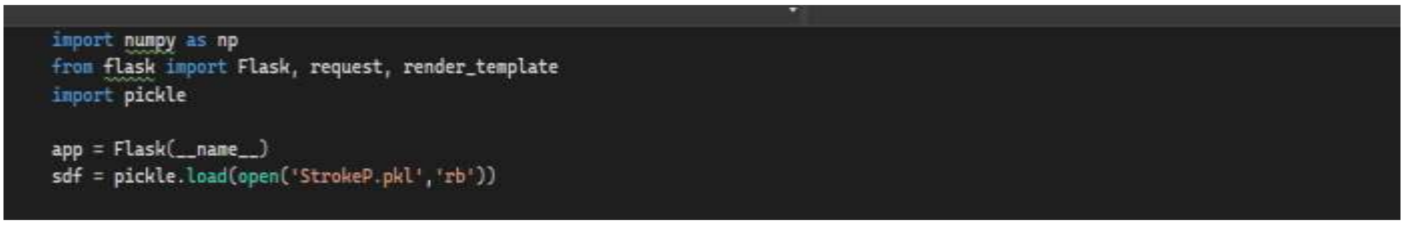
Fig 2 – Project Contents

Here in Fig 2, in my project, you will see static and templates and 3 python files.

So, Static is the folder where the files are added which will not change hence stylesheet. The templates will contain all the HTML files. The other python files are the ML model and Flask file.

3. Creating the Instance of Flask class

   Defining the flask class and then that package or instance created is used to resolve resources inside the package.
   Written as:
   App = Flask(__name__)

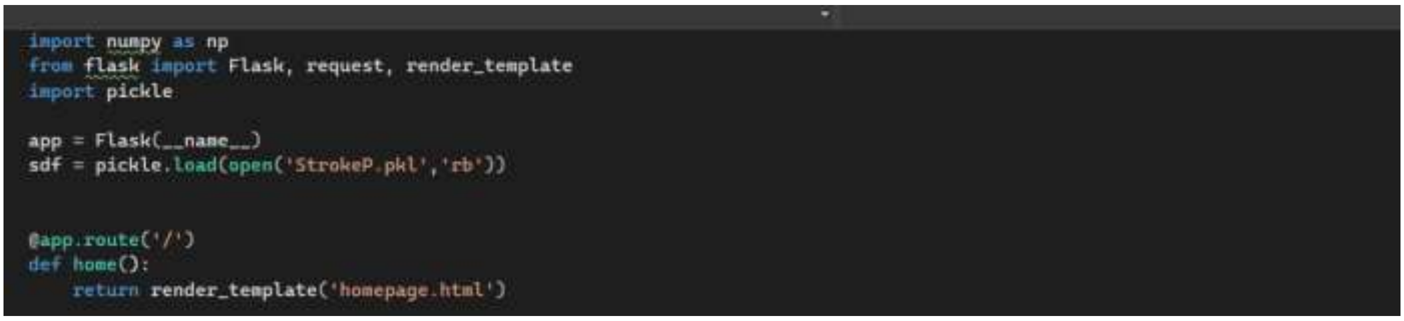


```
import numpy as np
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)
sdf = pickle.load(open('StrokeP.pkl','rb'))
```

Fig 3 – Creating Instance

4. Setting the URL Route

   After the instance is created called app that is then used to create the route for the URL showed in Fig 4. As @app.route('/') and whatever the we set inside the brackets will be the route for the html page.

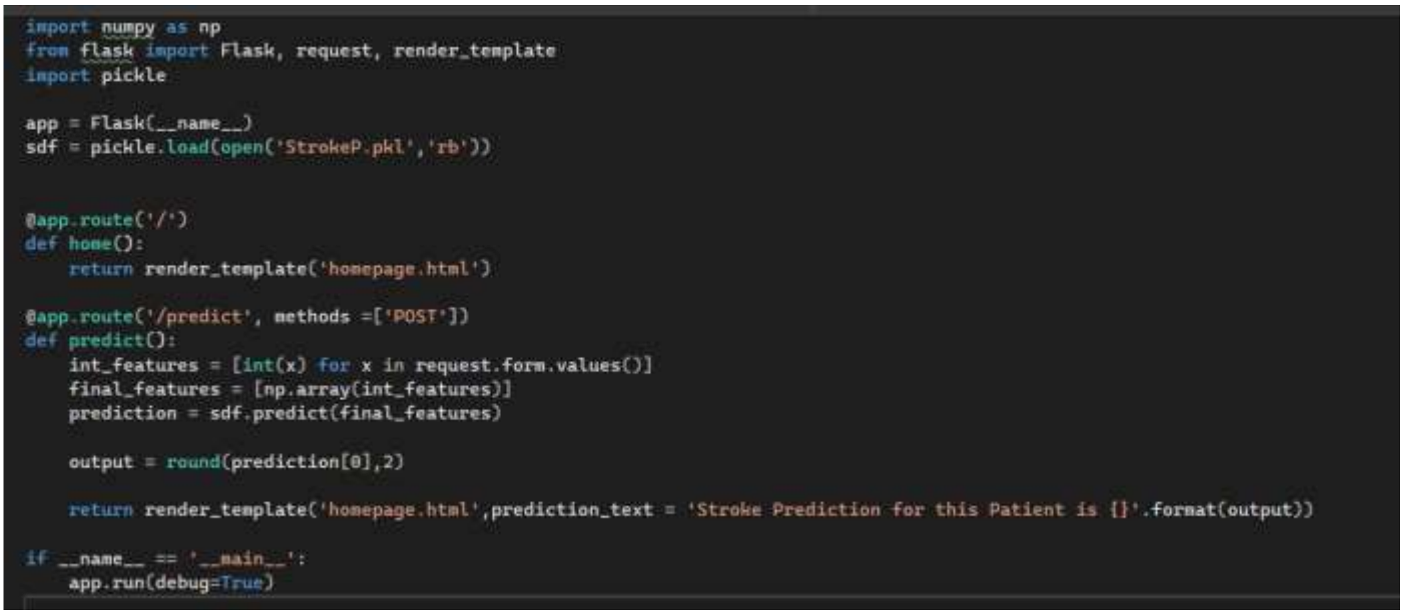


```
import numpy as np
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)
sdf = pickle.load(open('StrokeP.pkl','rb'))


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

Fig 4: Using the instance app to set the route of the html page.

   Now, in Fig 4, we have set the route as a slash and written a function of homepage and using the render_template imported from the flask module we assign that homepage.html to that route. So, whenever the server runs it will see ('/') this and assigned to it will be the homepage.html, as a result it will show the homepage.html as the webpage.

5. Creating ML model URL route after hitting Submit



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

app = Flask(__name__)
sdf = pickle.load(open('StrokeP.pkl','rb'))


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

@app.route('/predict', methods =['POST'])
def predict():
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = sdf.predict(final_features)

    output = round(prediction[0],2)

    return render_template('homepage.html',prediction_text = 'Stroke Prediction for this Patient is {}'.format(output))

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

Fig 5 – Setting the route for ML Model prediction on the same page

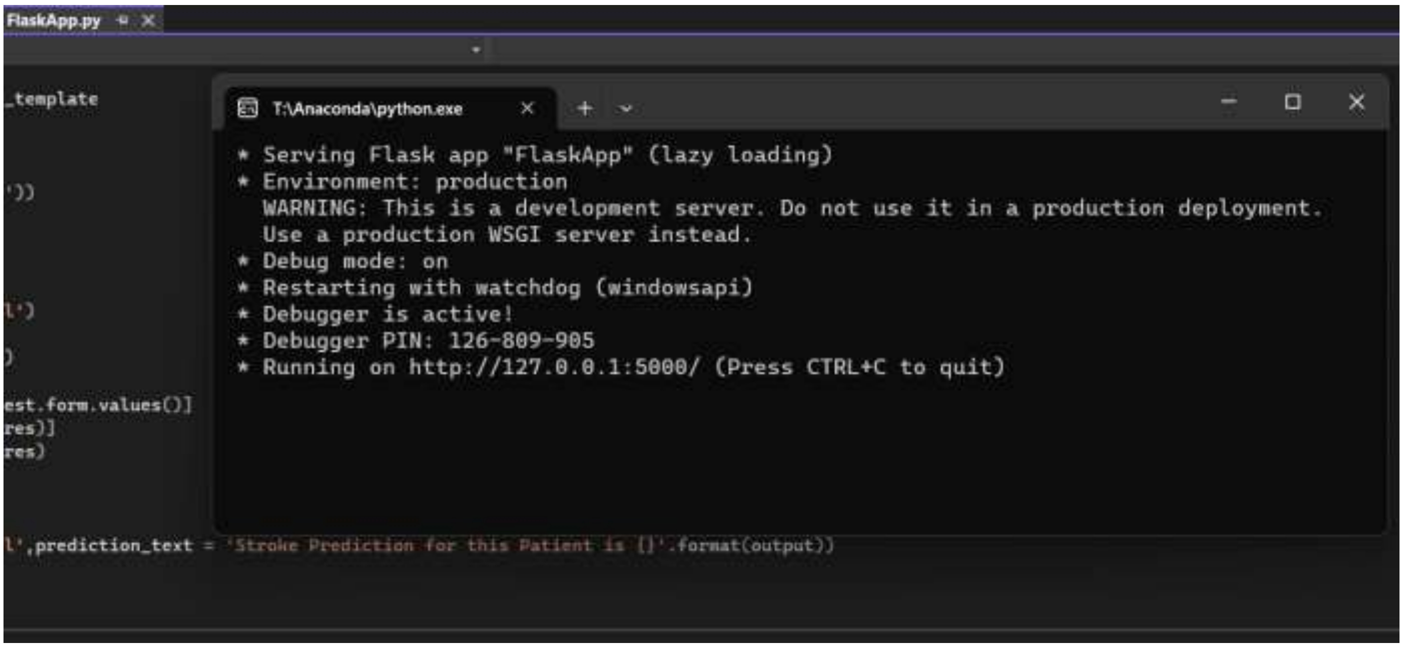
In Fig 5, as you see @app.route('/predict') is written and it means that as soon as predict button on the home page is clicked the function def predict() will run and route will get /predict running whatever is written in the function predict(). Also to add on, you will see methods = ['POST'] is written in the app route which tells the request method as POST which we imported from flask module as import request.

To add on, now I will speak about the pickle file which, is on the 5th line here, we are loading the .pkl (Pickle file) and assigning it to sdf instance which is used in the predict() function. What this does is instead of loading and opening the ML model python file, this pickle file is the byte stream sized file which takes less memory and this file is loaded which the same functionality and results as to ML model python file.

This is all that is needed to create the flask deployment using the flask module. The last lines which we see as if __name__ == main, if this is not initialized in the flask app python file then you need to create a .__init__ python file and initialize it there.

Running the Flask app created

1. Running the flask deployment project
   The flask code written have to be set as primary running file. This means that whenever the project will be executed the first point of interpretation will be the flask python file which can deploy all the things to the local server.



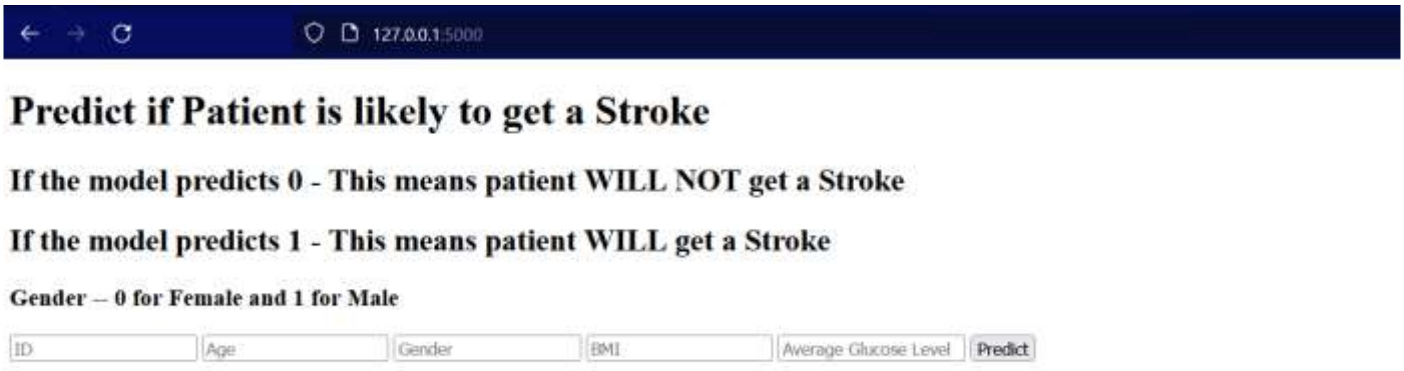
```
_template
'))
l')
)
est.form.values()]
res)]
res)
l',prediction_text = 'Stroke Prediction for this Patient is {}'.format(output))
```

```
* Serving Flask app "FlaskApp" (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 watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 126-809-905
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig 6 – Launching Flask on Web Server

In Fig 6, you will see command prompt where it says serving flask app. Also, on the last line it displays where this web application is running, in this case it is running on the local server with URL http://127.0.0.1.5000 .

2. Opening the Web page



Predict if Patient is likely to get a Stroke

If the model predicts 0 - This means patient WILL NOT get a Stroke

If the model predicts 1 - This means patient WILL get a Stroke

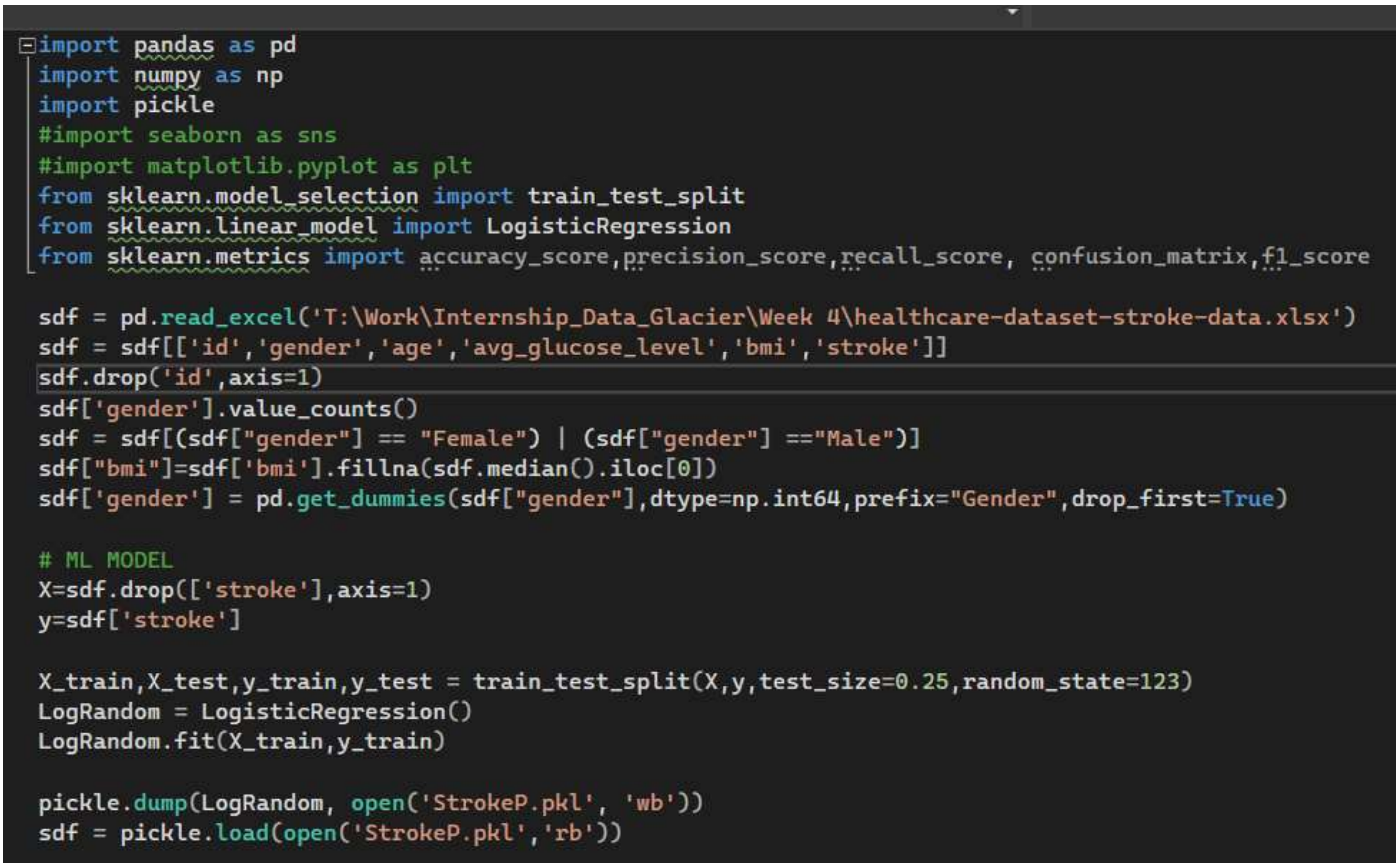
Gender – 0 for Female and 1 for Male

Fig 7 – The Stroke Prediction web App deployed using flask module.

As you see in Fig 7, the URL is the same when we run the flask app python file and this URL hosts the home page.

Linking the HTML and ML Model.

1. ML Model



```
import pandas as pd
import numpy as np
import pickle
#import seaborn as sns
#import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,precision_score,recall_score, confusion_matrix,f1_score

sdf = pd.read_excel('T:\Work\Internship_Data_Glacier\Week 4\healthcare-dataset-stroke-data.xlsx')
sdf = sdf[['id','gender','age','avg_glucose_level','bmi','stroke']]
sdf.drop('id',axis=1)
sdf['gender'].value_counts()
sdf = sdf[(sdf["gender"] == "Female") | (sdf["gender"] =="Male")]
sdf["bmi"]=sdf['bmi'].fillna(sdf.median().iloc[0])
sdf['gender'] = pd.get_dummies(sdf["gender"],dtype=np.int64,prefix="Gender",drop_first=True)

# ML MODEL
X=sdf.drop(['stroke'],axis=1)
y=sdf['stroke']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=123)
LogRandom = LogisticRegression()
LogRandom.fit(X_train,y_train)

pickle.dump(LogRandom, open('StrokeP.pkl', 'wb'))
sdf = pickle.load(open('StrokeP.pkl','rb'))
```

Fig 8 – ML Model

In Fig 8, I have created the ML model using Scikit Library and importing Logistic regression. The last two lines are important, as the pickle file is generated and loaded using the regressor or the instance of the Logistic Regression Class. This is then linked in the Flask app python file shown in Fig 4 and Fig 5, where the pickle file is loaded.

2. HTML Page
   This html homepage is linked to the flask app using the @app.route and writing the function homepage where the function will return the homepage.html using render_template.