# 

# **The Web Service For Classification By Neural Networks**

**Realised By:**

**ZAZA Zakaria**

**TAIBI Fadoua**

The ML Model

We will use an inbuilt Keras dataset of IMDB reviews to perform sentiment analysis on the movie reviews. We will create a simple MLP deep learning model. Details of the dataset and model are given in this [blog](https://machinelearningmastery.com/predict-sentiment-movie-reviews-using-deep-learning/).

Turning the Model into a Web Application

Having prepared the code for ***the model*** ready, we will develop a web application that consists of a simple web page with a form field that lets us enter a message. After submitting the message to the web application, it will render it on a new page, which gives us a result of spam or not spam.

First, we install the dependencies, i.e., ‘flask’ by using the command:

pip install Flask

Once we install Flask, we create a folder; in this example, we named it IMDB Classifier, and this is the directory tree inside the folder. We will explain each of the essential files as we go along.

app.py

imdb\_classifer.py

model\_architecture.json

model\_weights.h5

templates/

home.html

layout.html

Code.html

outputs/

Positive.html

Negative.html

includes/

\_form.html

\_navbar.html

static/

css/

style.css

pic1.jpg

pic/

Logow.png

App.py:

Now we need to create a base python file (app.py) that will contain the main code that will be executed by the Python interpreter to run the Flask web application:

app.py

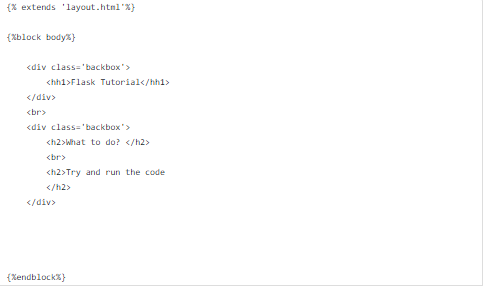
Note, we need to import the python file and call the run function that we need to run in the backend. This code takes the input in a form and then redirects the user to the respective output page.

Layout.html

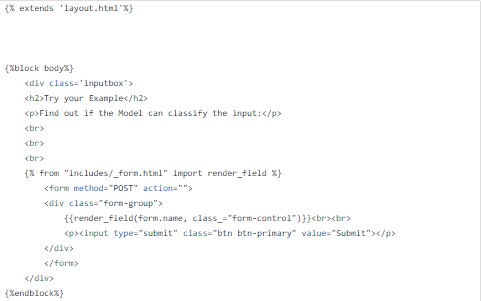
Once the base python file is ready we need to make the HTML web pages. We use a layout.html file so that we don’t have to repeat the same parts of the code again and again:

layout.html

Home.html

home.html

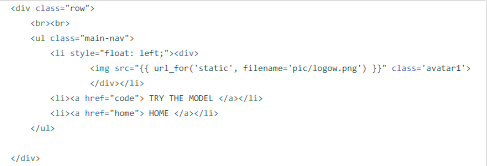
code.html

code.html

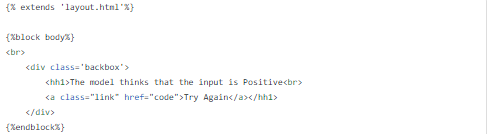
Style.css

In the header section of layout.html, we loaded style.css a file. CSS is to determine the look and feel of HTML documents. style.css has to be saved in a subdirectory called static, which is the default directory where Flask looks for static files such as CSS:

The ‘include’ files are used to add additional elements to the webpage like the navigation bar. They are partial codes that are called in other webpages:

\_navbar.html

The output files are the files that will be called on the basis of the output of the python code. Here we are using two output files one for ‘Positive’ and other ‘Negative’:

Positive.html

Negative.html

Well, that was a lot of code but we are almost there! we have created all the required files, now let's test it.

Once you have done all of the above, you can start running the API by either double click app.py , or executing the command from the Terminal:

cd ghazi-project

python app.py

Now you could open a web browser and navigate to <http://127.0.0.1:5000/>.

But to make it easier to use, we may host it in a temporary web server, using ngrok,

pip install flask-ngrok

And add this line to app.py to host the web app temporary in ngrok server:

app = Flask(\_\_name\_\_)

run\_with\_ngrok(app)

The output should be as follow;