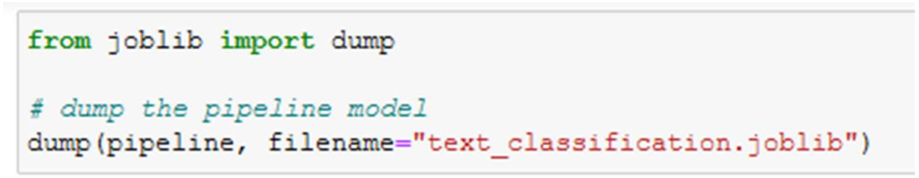
Deployment on Flask

We first stored the model on joblib.

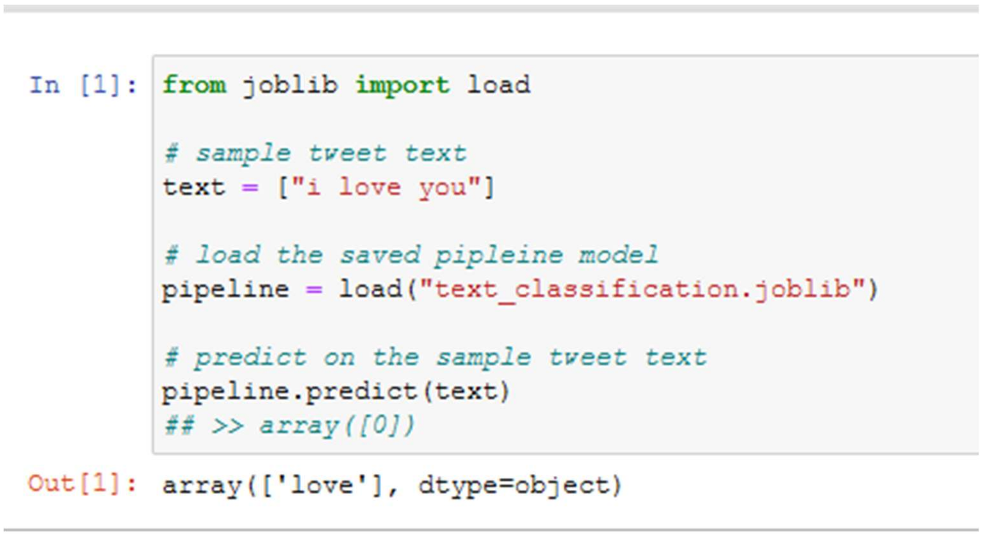
The library documentation can be read online on https://joblib.readthedocs.io/en/latest/

In the joblib library we used the dump() and load() the functions. The dump function is used to create a .joblib file



```
from joblib import dump

# dump the pipeline model
dump(pipeline, filename="text_classification.joblib")
```

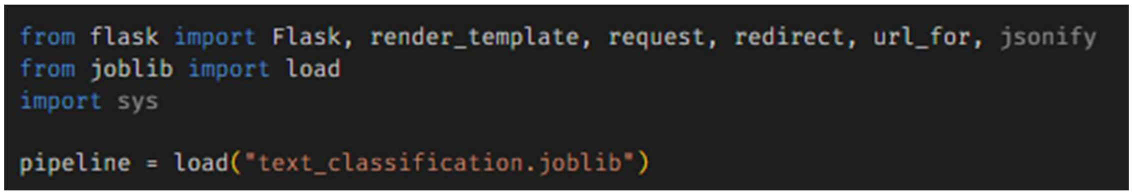
This .joblib file is the pushed into the load() function which then creates a model which is allowed to predict the sentiment used in a text.



```
In [1]: from joblib import load

        # sample tweet text
        text = ["i love you"]

        # load the saved pipleine model
        pipeline = load("text_classification.joblib")

        # predict on the sample tweet text
        pipeline.predict(text)
        ## >> array([0])

Out[1]: array(['love'], dtype=object)
```

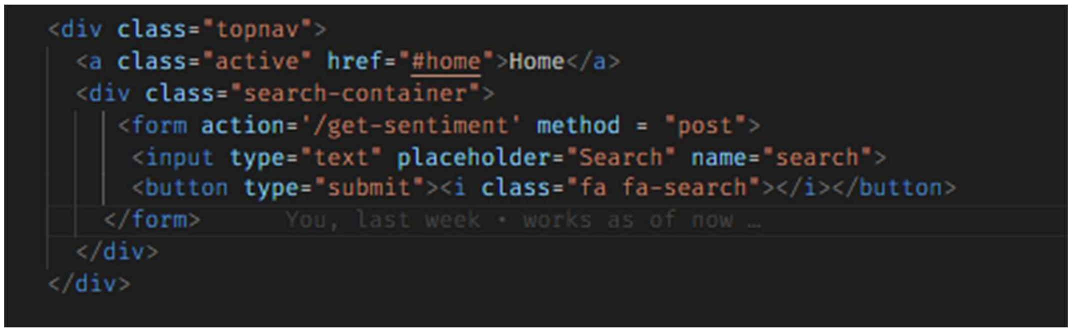
Moving on to the deployment on flask.

We import the load function in our app.py file. When used the load function returns the object stored in the .joblib file.



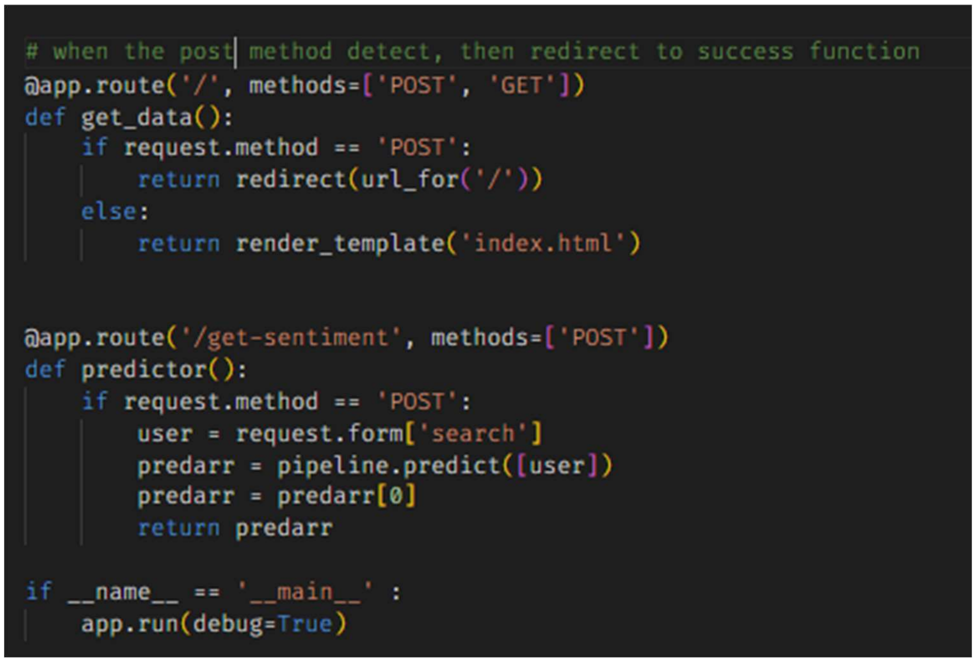
```
from flask import Flask, render_template, request, redirect, url_for, jsonify
from joblib import load
import sys

pipeline = load("text_classification.joblib")
```

We create an html file named index.html where we use forms to get text from our user.



```html
<div class="topnav">
  <a class="active" href="#home">Home</a>
  <div class="search-container">
    <form action='/get-sentiment' method = "post">
     <input type="text" placeholder="Search" name="search">
     <button type="submit"><i class="fa fa-search"></i></button>
   </form>      You, last week • works as of now …
  </div>
</div>
```

Once we have the required text, we pass this onto our app.py file



```python
# when the post method detect, then redirect to success function
@app.route('/', methods=['POST', 'GET'])
def get_data():
    if request.method == 'POST':
        return redirect(url_for('/'))
    else:
        return render_template('index.html')


@app.route('/get-sentiment', methods=['POST'])
def predictor():
    if request.method == 'POST':
        user = request.form['search']
        predarr = pipeline.predict([user])
        predarr = predarr[0]
        return predarr

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

In the image above app.route('/') represents the first page that will be shown to the user, which is where he will enter his text.

The second app.route('/get-sentiment') is where the page will be directed if the user decides he wants to know of the sentiment his tweet suggests. The html and the app.py files are connected using the action keyword in the form. This allows the user to be redirected to the /get-sentiment page. From there the words entered by the user in the form with id "search" are collected in the variable named user.

We then pass the text in an array to the predict() function which then returns another array which includes only one element which is the sentiment of the user.