Toimeenpano

Nimi: Petri Salminen
Opnum: 243840
Email: petri.salminen@tuni.fi

Tajusin koneoppimisen jälkeen, että eihän tästä kyllä mitään raporttia synny. Sen takia päädyin rakentamaan API:n, jonka kautta pystyy kokeilemaan, sopiiko jokin HTTP-pyyntö aiemmin rakennettuun malliin. Alla on aikalailla nähtävissä, miten tällainen API yksinkertaisuudessaan rakentuu.

In [ ]:
```
# Import flask stuff
from flask import request
from flask_api import FlaskAPI

# Import sklearn stuff
# Not sure if these are needed
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.neighbors import LocalOutlierFactor

# Import pickle
import pickle

# Import datetime
from datetime import datetime

# Import numpy
from numpy import array, concatenate

# Import json
import json

# Couple of helper-functions
def IP_to_int(ip):
    ip_a = ip.split('.')
    return int(ip_a[0])*256**3+int(ip_a[1])*256**2+int(ip_a[2])*256**1+int(ip_a
[3])

def time_to_timestamp(time_str):
    return int(datetime.timestamp(datetime.strptime(time_str, "%d/%b/%Y:%H:%M:%
S"))) # Example: 05/May/2019:07:53:22
```

Tämä seuraava steppi pitänee selittää kunnolla auki. Tein niin, että talletin localoutlierfactor-modelin, standardscalerin ja onehotencoderin picklellä levylle. Sitten API:n **käynnistyessä** (ei tapahdu joka pyynnöllä), ladataan kaikki kirjastot ja levyltä kyseiset muuttujat.

In [ ]:
```
# Reload the pickled model and other stuff
with open('detector.model', 'rb') as f:
    detector = pickle.load(f)
with open('scaler.pickle', 'rb') as f:
    scaler = pickle.load(f)
with open('encoder.pickle', 'rb') as f:
    enc = pickle.load(f)
    enc.set_params(handle_unknown='ignore')

app = FlaskAPI(__name__)

@app.route('/loganalyze/', methods=['GET'])
def loganalyze():
    # get the previously pickled variables
    global detector
    global scaler
    global enc

    # Convert the values on-the-fly
    ip = IP_to_int(request.args.get('ip'))
    http_version = request.args.get('http_version')
    timestamp = time_to_timestamp(request.args.get('time'))
    status = request.args.get('status')

    # Temporary array for the already calculated values
    tmp_a = array([ip, status, http_version, timestamp])

    # Then, the OH-encoding for url
    url = request.args.get('url')
    url_encoded = enc.transform([[url]])

    # Put the arrays together
    concated_a = array(concatenate((tmp_a, url_encoded[0]))).reshape(1,-1)

    # Scale them
    scaled_a = scaler.transform(concated_a)

    # Finally, predict and return
    return json.dumps({"detection": int(detector.predict(scaled_a)[0])})

app.run(debug=True)
```

Nyt kun ajaa `python api.py` , käynnistyy API nätisti ja sille pystyy lähettämään GET-pyyntöjä polkuun /loganalyze/. Tällä hetkellä virheenkäsittely on vielä olematonta, ja palvelin kaatuu, mikäli jokin parametreista puuttuu.

APIa voi testata esimerkiksi: https://salminen.dev/loganalyze/?ip=192.168.1.1&http_version=1.1&time=05/May/2019:07:47:51&status=200&url=/index.php (https://salminen.dev/loganalyze/?ip=192.168.1.1&http_version=1.1&time=05/May/2019:07:47:51&status=200&url=/index.php). Olen tyytyväinen kyseisen mallin nopeuteen, sillä yhden rivin ajamiseen ei kulu paljoakaan aikaa:

```
$ time curl -L 'https://salminen.dev/loganalyze/?ip=192.168.1.1&http_version=1.1&time
=05/May/2019:07:47:51&status=200&url=/index.php'
{"detection": -1}
real    0m0.076s
user    0m0.018s
sys 0m0.006s
```

Toki lähtödatan puitteissa kyseinen detector on suhteellisen hyödytön, koska se pitää suunnilleen kaikkea ei-sopivana malliin. Koen silti tämän harjoitustyön onnistuneeksi, sillä samaa mallia voisi hyödyntää myös monimutkaisemman datasetin kanssa, jolloin saataisiin oikeasti hyödyllinen työkalu.

Hyödylliset linkit

- Flask-Apin dokumentaatio (https://github.com/flask-api/flask-api)
- Apuja datetimen muuttamiseksi timestampiksi (https://www.programiz.com/python-programming/datetime/timestamp-datetime)

Helpot ja hankalat asiat

- Tämä oli yllättävän helppo homma. Flask-Api oli juurikin niin näppärä kuin uskalsin odottaa!
- Picklellä modelin yms. siirtäminen oli myös erittäin helppo homma
- Hankalin homma oli saada kyseinen testi-Flask-palvelin näkymään ulkomaailmalle nginx-konffien avulla. Siitäkin selvittiin kumminkin :)

In [ ]: