|  |
| --- |
| rom flask import Flask, render\_template, request |
|  | import numpy as np |
|  | import pandas as pd |
|  | from sklearn import metrics |
|  | import warnings |
|  | import pickle |
|  | warnings.filterwarnings('ignore') |
|  | from features import FeatureExtraction |
|  | from flask\_mysqldb import MySQL |
|  | import requests |
|  |  |
|  | # NOTE: you must manually set API\_KEY below using information retrieved from your IBM Cloud account. |
|  | API\_KEY = "<API\_KEY>" #for security reasons haven't displayed API key |
|  | token\_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": |
|  | API\_KEY, "grant\_type": 'urn:ibm:params:oauth:grant-type:apikey'}) |
|  | mltoken = token\_response.json()["access\_token"] |
|  |  |
|  | header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken} |
|  |  |
|  | app = Flask(\_\_name\_\_) |
|  |  |
|  | app.config['MYSQL\_HOST'] = 'localhost' |
|  | app.config['MYSQL\_USER'] = 'root' |
|  | app.config['MYSQL\_PASSWORD'] = '' |
|  | app.config['MYSQL\_DB'] = 'report' |
|  |  |
|  | mysql = MySQL(app) |
|  |  |
|  | xgb = pickle.load(open("XGBoostClassifier.pkl", "rb")) |
|  |  |
|  | @app.route("/", methods=["GET", "POST"]) |
|  | def home(): |
|  | if request.method == "POST": |
|  |  |
|  | url = request.form["url"] |
|  | obj = FeatureExtraction(url) |
|  |  |
|  | x = np.array(obj.getFeaturesList()).reshape(1,13) |
|  | print(x) |
|  | t=[obj.getFeaturesList()] |
|  | print("t") |
|  | print(t) |
|  | # NOTE: manually define and pass the array(s) of values to be scored in the next line |
|  | payload\_scoring = {"input\_data": [{"fields": [['f0','f1','f2','f3','f4','f5','f6','f7','f8','f9','f10','f11','f12']], "values": t}]} |
|  |  |
|  | response\_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/859ae568-d692-4958-9dbe-60431a8a0918/predictions?version=2022-11-11', json=payload\_scoring, headers={'Authorization': 'Bearer ' + mltoken}) |
|  | print("Scoring response") |
|  | print(response\_scoring.json()) |
|  |  |
|  | y\_pred =xgb.predict(x)[0] |
|  | print(y\_pred) |
|  | y\_pro\_phishing = xgb.predict\_proba(x)[0,0] |
|  | print(y\_pro\_phishing) |
|  | y\_pro\_non\_phishing = xgb.predict\_proba(x)[0,1] |
|  | print(y\_pro\_non\_phishing) |
|  |  |
|  | if(y\_pro\_phishing\*100<60): |
|  | msg="Trick or Treat?\n Look at this tweet.\n This site is elite!\n" |
|  | flag=1 |
|  | else: |
|  | msg="Trick or Treat?\n Don't get deceit.\n This site is creep!\n" |
|  | flag=-1 |
|  |  |
|  | return render\_template('result.html', msg=msg, url=url, val=flag) |
|  |  |
|  | return render\_template("index.html") |
|  |  |
|  | @app.route("/report", methods=["GET", "POST"]) |
|  | def report(): |
|  | if request.method == 'GET': |
|  | return render\_template("contact.html") |
|  |  |
|  | if request.method == 'POST': |
|  | name = request.form['name'] |
|  | email = request.form['email'] |
|  | query = request.form['query'] |
|  | cursor = mysql.connection.cursor() |
|  | cursor.execute(''' INSERT INTO responses VALUES(%s,%s,%s)''',(name,email,query)) |
|  | mysql.connection.commit() |
|  | cursor.close() |
|  | return render\_template("alert.html") |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | app.run(debug=True) |