INTEGRATE FLASK WITH SCORING END POINT

Team ID	**PNT2022TMID16122**
Project Name	**Car Resale value Prediction**

INTEGRATE FLASK WITH SCORING END POINT

```
import pandas as pd
import numpy as np
from flask import Flask,render_template,Response,request
import pickle
from sklearn.preprocessing import LabelEncoder
import pickle

import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "MGU1iT6RDkhiyFrQhD8KbdYD1kWSOWNmSZCUhCB_IGDg"
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__,template_folder='templates/')
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/resaleintro.html')
def p():
    return render_template('resaleintro.html')

@app.route('/predict')
def predict():
    return render_template('resalepredict.html')

@app.route('/y_predict',methods=['GET','POST'])
def y_predict():
    regyear = int(request.form['regyear'])
    powerps = float(request.form['powerps'])
    kms = float(request.form['kms'])
    regmonth = int(request.form.get('regmonth'))
    gearbox = request.form['gearbox']
    damage = request.form['dam']
    model  = request.form.get('model_type')
    brand = request.form.get('brand')
```

```
    fuelType = request.form.get('fuel')
    vehicletype= request.form.get('vehicletype')
    new_row =
{'yearOfRegistration':regyear,'powerPS':powerps,'kilometer':kms,'monthOfRegistration':regm
onth,'gearbox':gearbox,'notRepairedDamage':damage,'model':model,'brand':brand,'fuelType':f
uelType,'vehicleType':vehicletype}

    print(new_row)
    new_df =
pd.DataFrame(columns=['vehicleType','yearOfRegistration','gearbox','powerPS','model','kilo
meter','monthOfRegistration','fuelType','brand','notRepairedDamage'])
    new_df = new_df.append(new_row,ignore_index=True)
    labels = ['gearbox','notRepairedDamage','model','brand','fuelType','vehicleType']
    mapper = {}
    for i in labels:
        mapper[i] = LabelEncoder()
        mapper[i].classes_ = np.load(str('classes'+i+'.npy'),allow_pickle=True)
        tr = mapper[i].fit_transform(new_df[i])
        new_df.loc[:,i+'_Labels'] = pd.Series(tr,index=new_df.index)
    labeled = new_df[ ['yearOfRegistration','powerPS','kilometer','monthOfRegistration'] +
[x+"_Labels" for x in labels]]

    X = labeled.values
    print(X)

    payload_scoring = {"input_data": [{"field":
[['vehicleType','yearOfRegistration','gearbox','powerPS','model','kilometer','monthOfRegis
tration','fuelType','brand','notRepairedDamage']], "values": X.tolist()}]}

    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/392de23a-3e4e-40ce-85b0-
b1accd4fddda/predictions?version=2022-11-17', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    predictions = response_scoring.json()
    print(predictions['predictions'][0]['values'][0][0])
    return
render_template('resalepredict.html',ypred="{:.2f}".format(predictions['predictions'][0]['
values'][0][0]))

if __name__ == '__main__':
    app.run(host='Localhost',debug=True,threaded=False)
```

```
(base) PS C:\Users\jerry\Desktop\IBM-PROJECT> CD..
(base) PS C:\Users\jerry\Desktop> cd IBM_CLOUD
(base) PS C:\Users\jerry\Desktop\IBM_CLOUD> python CloudApp.py
 * Serving Flask app "CloudApp" (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: 119-200-533
 * Running on http://Localhost:5000/ (Press CTRL+C to quit)
```