INTEGRATEFLASKWITHSCORINGENDPOINT

TeamID	**PNT2022TMID30453**
ProjectName	**CarResalevaluePrediction**

INTEGRATEFLASKWITHSCORINGENDPOINT

```
import pandas as
pdimportnumpyasnp
fromflaskimportFlask,render_template,Response,requestimp
ortpickle
from sklearn.preprocessing import
LabelEncoderimportpickle

import
requestsimportj
son
#NOTE:youmustmanuallysetAPI_KEYbelowusinginformationretrievedfromyourIBMCloudaccount.
API_KEY="hEAn_mcoP3u_-ZjagjeqlxDayqUiETpYVYWdR1OLKAby"
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('
/')
defindex():
    returnrender_template('index.html')

@app.route('/resaleintro.html')de
fp():
    returnrender_template('resaleintro.html')

@app.route('/predict')
defpredict():
    returnrender_template('resalepredict.html')

@app.route('/y_predict',methods=['GET','POST'])d
efy_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=reques
```

```
t.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','kilome
ter','monthOfRegistration','fuelType','brand','notRepairedDamage'])
    new_df=new_df.append(new_row,ignore_index=True)
    labels =
    ['gearbox','notRepairedDamage','model','brand','fuelType','vehicleType']mapper=
    {}
    foriinlabels:
        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"forxin labels]]

    X =
    labeled.valuesprin
    t(X)
    #returnrender_template('resalepredict.html',ypred="{:.2f}".format(y_prediction[0]))

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

    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/816483ac-44ed-4be2-b780-
7f63d68fc7ce/predictions?version=2022-11-17',json=payload_scoring,
    headers={'Authorization': 'Bearer ' +
    mltoken})print("Scoringresponse")
    predictions =
    response_scoring.json()print(predictions['prediction
    s'][0]['values'][0][0])return
render_template('resalepredict.html',ypred="{:.2f}".format(predictions['predictions'][0]['va
lues'][0][0]))

ifname==
    'main':app.run(host='Localhost',debug=True,threaded=
    False)
```

```
(base) PS C:\Users\SUGARANJAN\Desktop\IBM>
(base) PS C:\Users\SUGARANJAN\Desktop\IBM> cd IBM_CLOUD
(base) PS C:\Users\SUGARANJAN\Desktop\IBM\IBM_CLOUD> python Appcloud.py
 * Serving Flask app "Appcloud" (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: 363-377-968
 * Running on http://Localhost:5000/ (Press CTRL+C to quit)
```