Development Phase
Integrate Flask with scoring End point

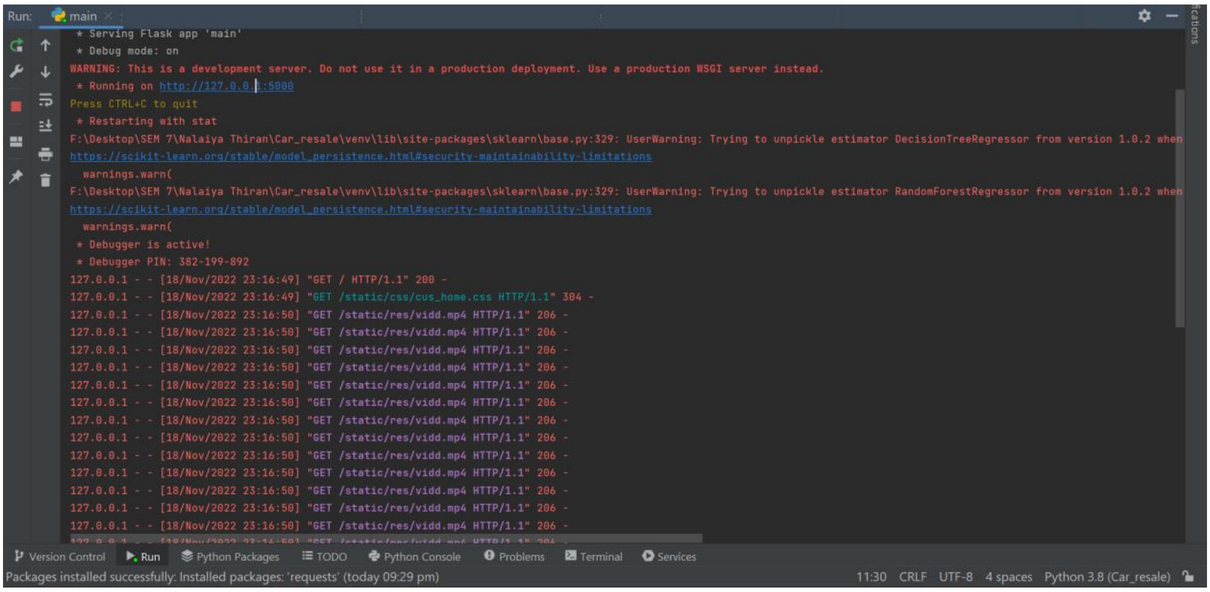
Date	18 November 2022
Team ID	PNT2022TMID21553
Project Name	Project – Car Resale Value Prediction

```
import pickle

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

API_KEY = "GxGc70sFN0c3WjkhCyutIq8zsCOhhQ0MrznbSeQ8aTw0"
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__)
cmodel = pickle.load(open('resale_model.pkl', 'rb'))
autos = pd.read_csv('car_resale_preprocessed.csv')


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

@app.route('/c_predict', methods=['POST'])
def c_predict():
   months = ["January", "February", "March", "April", "May", "June", "July", "August",
"September", "October", "November", "December"]
   regyear = int(request.form['reg_year'])
   powerps = float(request.form['car_power'])
   kms = float(request.form['kilo_driven'])
   regmonth = int(months.index(request.form.get('reg_month')))+1
   gearbox = request.form['gear_type']
   damage = request.form['car_condition']
   model = request.form.get('model')
   brand = request.form.get('brand')
   fuelType = request.form.get('fuel_type')
```

```
    vehicletype = request.form.get('veh_type')
    new_row = {'yearOfRegistration': regyear,
               'monthOfRegistration': regmonth,
               'gearbox': gearbox, 'notRepairedDamage': damage,
               'model': model, 'brand': brand, 'fuelType': fuelType,
               'vehicleType': vehicletype, 'powerPS': powerps, 'kilometer': kms}
    print(new_row)
    new_df = pd.DataFrame(columns=['vehicleType', 'yearOfRegistration', 'gearbox',
                                   'powerPS', 'model', 'kilometer', 'monthOfRegistration', 'fuel Type',
                                   '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)
        val = int(np.where(mapper[i].classes_ == new_row[i])[0][0])
        print(i, new_row[i], val)
        new_df.loc[:, i + '_labels'] = val
    labeled = new_df[['yearOfRegistration','powerPS'
           , 'kilometer'
           , 'monthOfRegistration']
         + [x + '_labels' for x in labels]]
    X = labeled.values
    print(X)

    payload_scoring = {"input_data": [{"field": [
        ["months", "regyear", "powerps", "kms", "regmonth", "gearbox", "damage", "model",
"brand", "fuelType",
         "vehicletype"]]}]}

    response_scoring = requests.post(
        'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/604f91b1-17c9-4661-ab72-
02e7b6d5bc4e/predictions?version=2022-11-18',
        json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    print(response_scoring.json())

    y_prediction = cmodel.predict(X)
    print(y_prediction)
    return 'The resale value predicted is {:.2f}$'.format(y_prediction[0])


@app.route('/car_price', methods=['GET', 'POST'])
def car_price():
    months = ["January", "February", "March", "April", "May", "June", "July", "August",
```

```
"September", "October", "November", "December"]
    fuel_types=autos['fuelType'].unique()
    brands = autos['brand'].unique()
    models = autos['model'].unique()
    vehicle_types = autos['vehicleType'].unique()
    return render_template('carPrice.html', fuel_types=fuel_types, months=months,
brands=brands, models=models, vehicle_types=vehicle_types)


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

The IBM cloud API key and the scoring end point of the deployed ML model is used to integrate the model with the Flask application.



```
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
F:\Desktop\SEM 7\Nalaiya Thiran\Car_resale\venv\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator DecisionTreeRegressor from version 1.0.2 when
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
F:\Desktop\SEM 7\Nalaiya Thiran\Car_resale\venv\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator RandomForestRegressor from version 1.0.2 when
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 382-199-892
127.0.0.1 - - [18/Nov/2022 23:16:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 23:16:49] "GET /static/css/cus_home.css HTTP/1.1" 304 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
127.0.0.1 - - [18/Nov/2022 23:16:50] "GET /static/res/vidd.mp4 HTTP/1.1" 206 -
```

Packages installed successfully: Installed packages: 'requests' (today 09:29 pm)