Development Phase
Sprint 4 delivery

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

Sprint 4:

The final sprint is to deploy the Machine Learning Model in IBM cloud.

An IBM cloud account is created and the ML model is deployed in the IBM Watson Studio.

Code:

Jupyter notebook:

```
import pandas as pd
import numpy as np
import matplotlib as plt
from sklearn.preprocessing import LabelEncoder
import pickle



import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your
credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='OcEMgCpub2nZF3LK07mkqLs1luADFC07vVBCeF5JGpVe',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'carresalevalueprediction-donotdelete-pr-blg0jnxocswfh1'
object_key = 'car_resale.csv'
```

```
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()

df.tail()

#different sellers
print(df.seller.value_counts())

#remove the seller 'gewerblich'
df[df.seller != 'gewerblich']

#all entries of column 'seller' are same
#drop the column 'seller'
df = df.drop('seller', 1)

#different offer types
print(df.offerType.value_counts())

#remove the offertype 'Gesuch'
df[df.offerType != 'Gesuch']

#column 'offerType' has same entires
#drop the column 'offerType'
df = df.drop('offerType', 1)

print(df.shape)

#remove cars having power less than 50p and greater than 900p
df = df[(df.powerPS > 50) & (df.powerPS < 900)]
print(df.shape)

#remove cars with year of registration before 1950 and after 2017
df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]
print(df.shape)

#remove columns that are not relevant
df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen', 'postalCode',
'dateCreated'], axis='columns', inplace=True)

#creating a copy of the dataframe and remove the duplicates in the columns
new_df = df.copy()
```

```
new_df = new_df.drop_duplicates(['price', 'vehicleType', 'yearOfRegistration', 'gearbox',
'powerPS', 'model', 'kilometer', 'monthOfRegistration', 'fuelType', 'notRepairedDamage'])

#clean the dataset of German words and replace with proper English words
new_df.gearbox.replace(('manuell', 'automatik'), ('manual', 'automatic'), inplace=True)
new_df.fuelType.replace(('benzin', 'andere', 'elektro'), ('petrol', 'others', 'electric'),
inplace=True)
new_df.vehicleType.replace(('kleinwagen', 'cabrio', 'kombi', 'andere'), ('small  car',
'convertible', 'combination', 'others'), inplace=True)
new_df.notRepairedDamage.replace(('ja', 'nein'), ('Yes', 'No'), inplace=True)

#Outlier Removal
new_df = new_df[(new_df.price >= 100) & (new_df.price <= 150000)]

#Fill the not declared values of the columns as NaN using fillna function
new_df['notRepairedDamage'].fillna(value='not-declared', inplace=True)
new_df['fuelType'].fillna(value='not-declared', inplace=True)
new_df['gearbox'].fillna(value='not-declared', inplace=True)
new_df['vehicleType'].fillna(value='not-declared', inplace=True)
new_df['model'].fillna(value='not-declared', inplace=True)

#save the dataframe as csv
new_df.to_csv('car_resale_preprocessed.csv')



#label encode the categorical data
labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']

mapping = {}
for i in labels:
  mapping[i] = LabelEncoder()
  mapping[i].fit(new_df[i])
  tr = mapping[i].transform(new_df[i])
  np.save(str('classes'+i+'.npy'), mapping[i].classes_)
  print(i, ":", mapping[i])
  new_df.loc[:, i+'_labels'] = pd.Series(tr, index=new_df.index)

#'labeled' dataframe contains the final data

labelled = new_df[ ['price', 'yearOfRegistration', 'powerPS', 'kilometer',
'monthOfRegistration'] + [x+"_labels" for x in labels]]
print(labelled.columns)



#split price and other data into Y and X respectively
Y = labelled.iloc[:, 0].values
```

```
X = labelled.iloc[:, 1:].values
Y = Y.reshape(-1, 1)

#split dataset into train and test dataset
from sklearn.model_selection import cross_val_score, train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=3)



from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

regressor = RandomForestRegressor(n_estimators=1000, max_depth=10, random_state=34)

regressor.fit(X_train, np.ravel(Y_train, order='C'))

pred_1 = regressor.predict(X_test)

print(r2_score(Y_test, pred_1))

from sklearn.tree import DecisionTreeClassifier

ds = DecisionTreeClassifier(max_depth=5000, max_features=0.9, max_leaf_nodes=5000,
random_state=2, splitter='best')

ds.fit(X_train,  np.ravel(Y_train, order='C'))

pred_3 =ds.predict(X_test)

print(r2_score(Y_test, pred_3))



file_name = 'resale_model.pkl'
pickle.dump(regressor, open(file_name, 'wb'))




!pip install ibm_watson_machine_learning
from ibm_watson_machine_learning import APIClient
wml_credentials = {
   "url": "https://us-south.ml.cloud.ibm.com",
   "apikey": "GxGc70sFN0c3WjkhCyutIq8zsCOhhQ0MrznbSeQ8aTw0"
}
client = APIClient(wml_credentials)
```

```
#create deployment space
def guid_from_space_name(client, space_name):
   space = client.spaces.get_details()
   return(next(item for item in space['resources'] if item['entity']["name"] ==
space_name)['metadata']['id'])


#create deployment space as 'new space'
space_uid = guid_from_space_name(client, 'models')
print(space_uid)

#make the created space as default space
client.set.default_space(space_uid)

#view client software specifications
client.software_specifications.list()

software_spec_uid = client.software_specifications.get_uid_by_name("runtime-22.1-
py3.9")
software_spec_uid

import sklearn
sklearn.__version__

#store the model in the deployment space
MODEL_NAME = 'CAR_RESALE_PREDICTION'
DEPLOYMENT_NAME = 'models'
DEMO_MODEL = regressor

model_props = {
   client.repository.ModelMetaNames.NAME: MODEL_NAME,
   client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
   client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}

import json

model_details = client.repository.store_model(
   model = DEMO_MODEL,
   meta_props = model_props,
   training_data = X_train,
   training_target = Y_train
)

model_details

model_id = client.repository.get_model_id(model_details)
```

```
model_id

deployment_props = {
   client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
   client.deployments.ConfigurationMetaNames.ONLINE: {}
}

deployment = client.deployments.create(
   artifact_uid=model_id,
   meta_props=deployment_props
)
```

Integrate Flask with scoring end point:

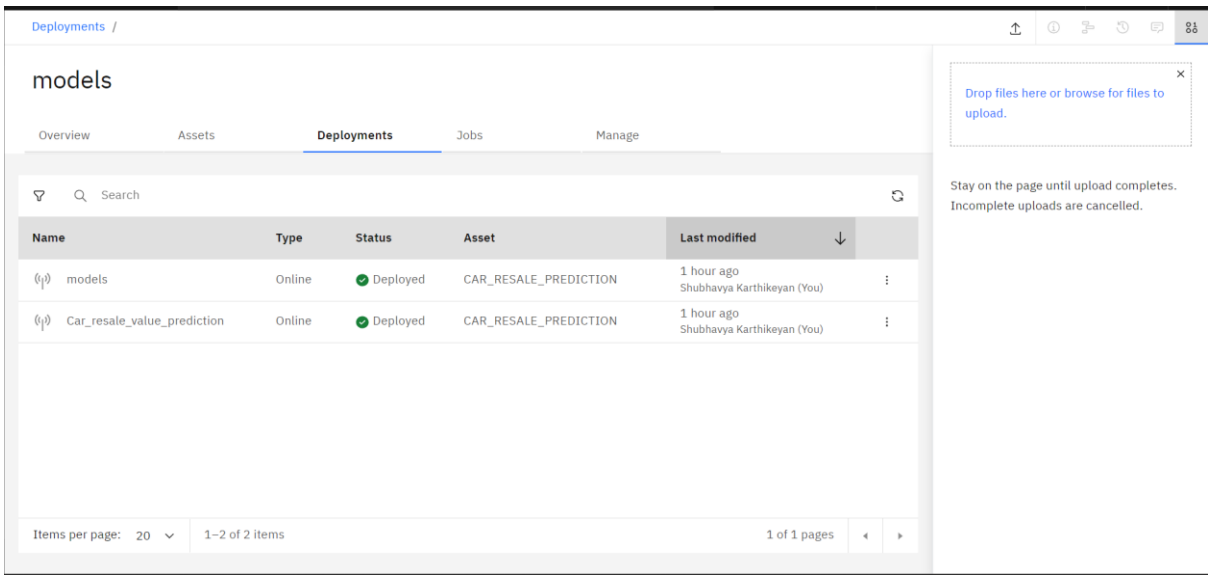
Scoring end point: https://us-south.ml.cloud.ibm.com/ml/v4/deployments/604f91b1-17c9-4661-ab72-02e7b6d5bc4e/predictions?version=2022-11-18

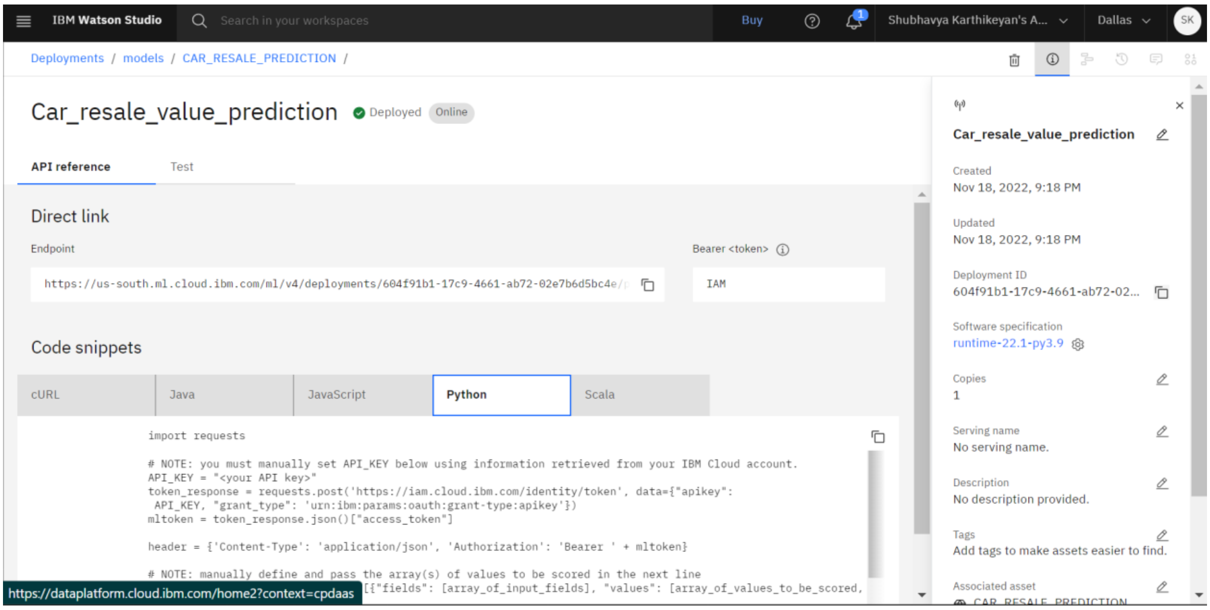
```
import pickle

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'])
```

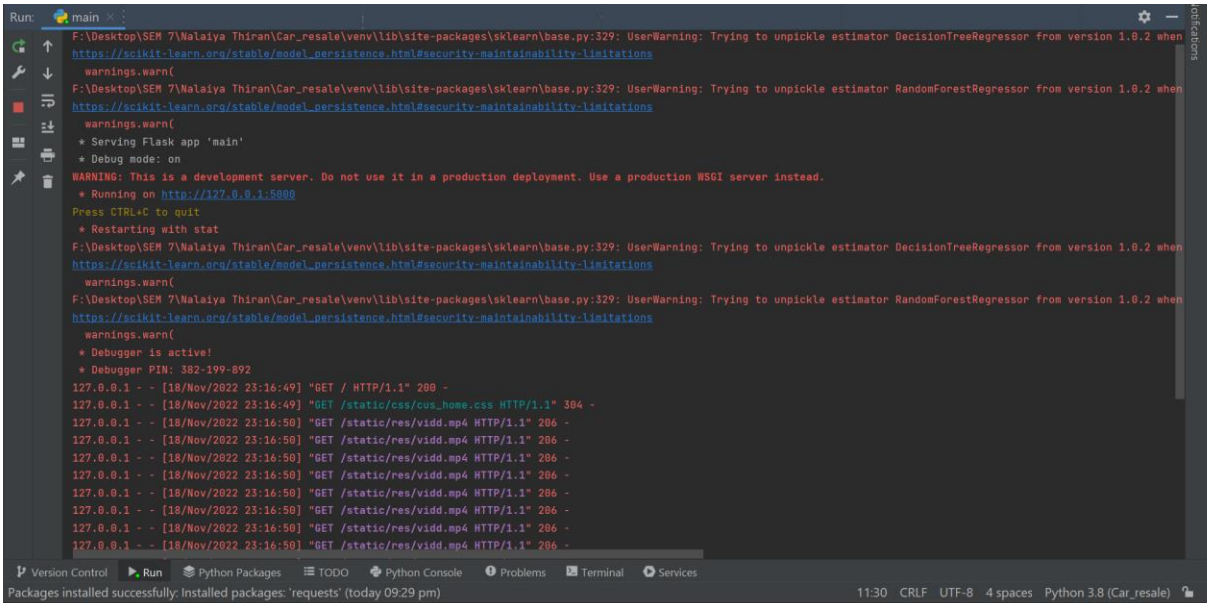
```
    powerps = float(request.form['car_power'])
    kms = float(request.form['kilo_driven'])
    regmonth = int(months.index(request.form.get('reg_month')))+1
    gearbox = request.form['gear_type']
    damage = request.form['car_condition']
    model = request.form.get('model')
    brand = request.form.get('brand')
    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())
```

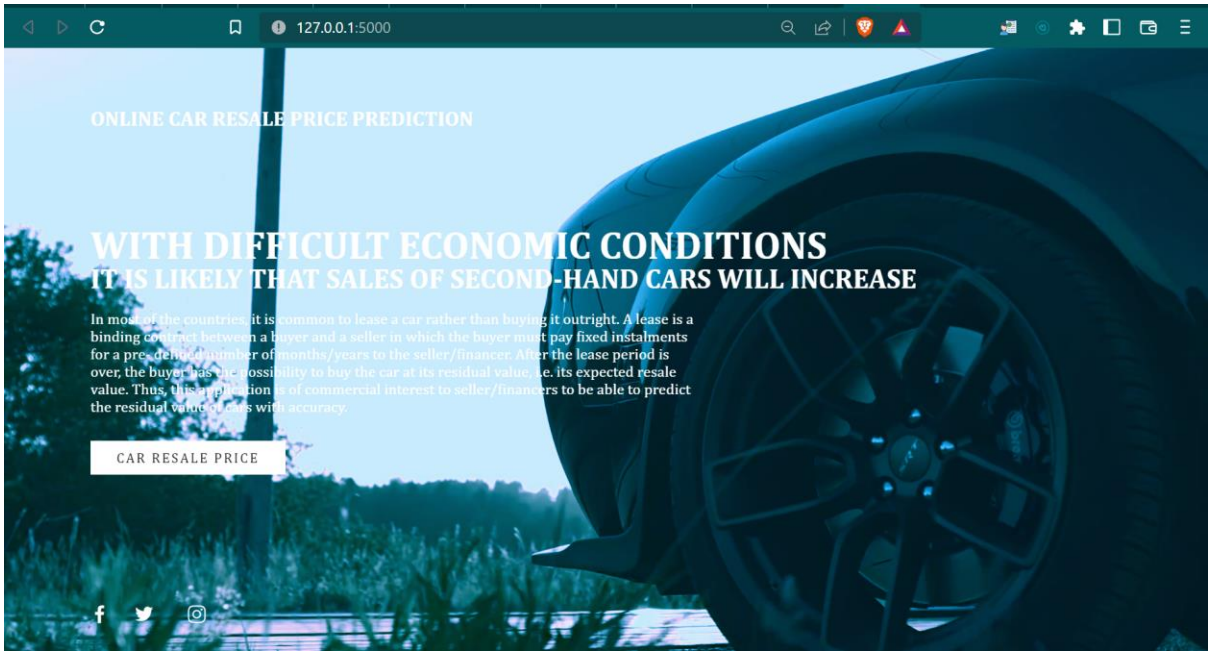
```
    y_prediction = cmodel.predict(X)
    print(y_prediction)
    return 'The resale value predicted is {:.2f}$'.format(y_prediction[0])


@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)
```

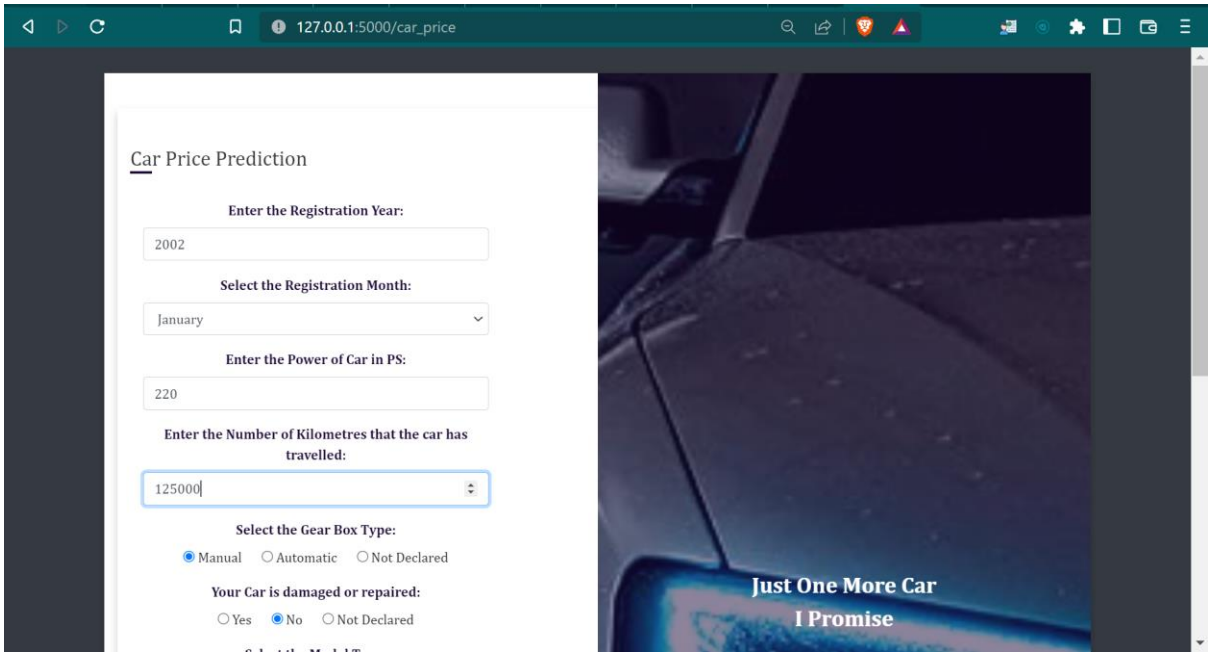
Test case:

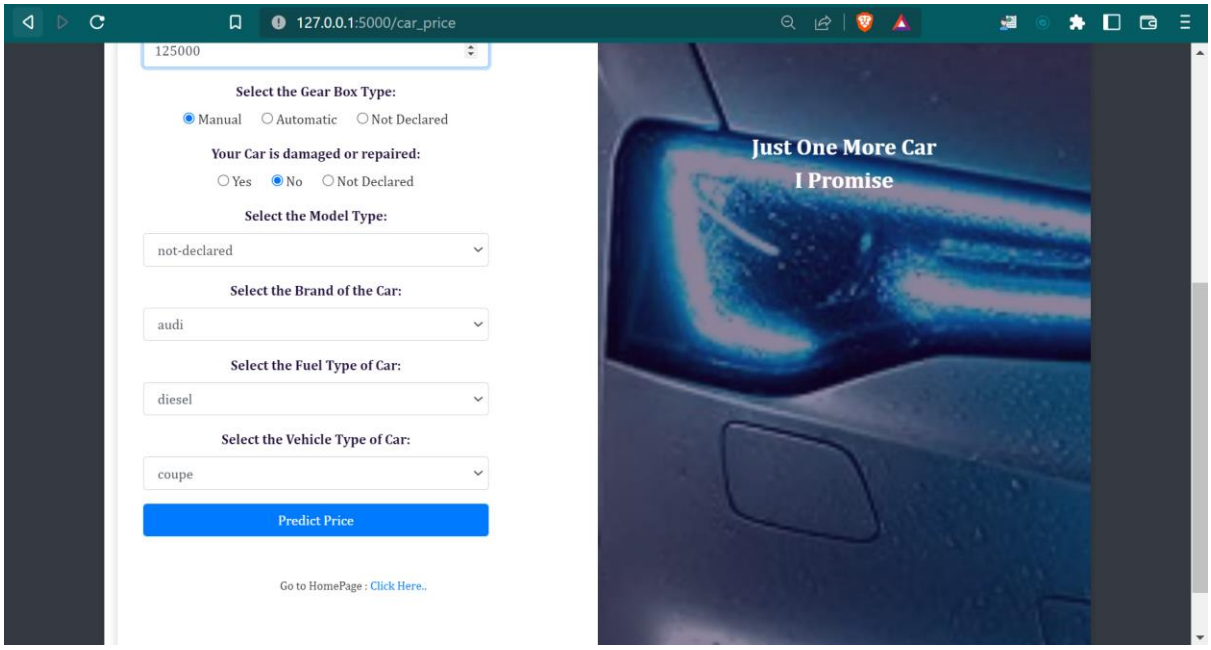


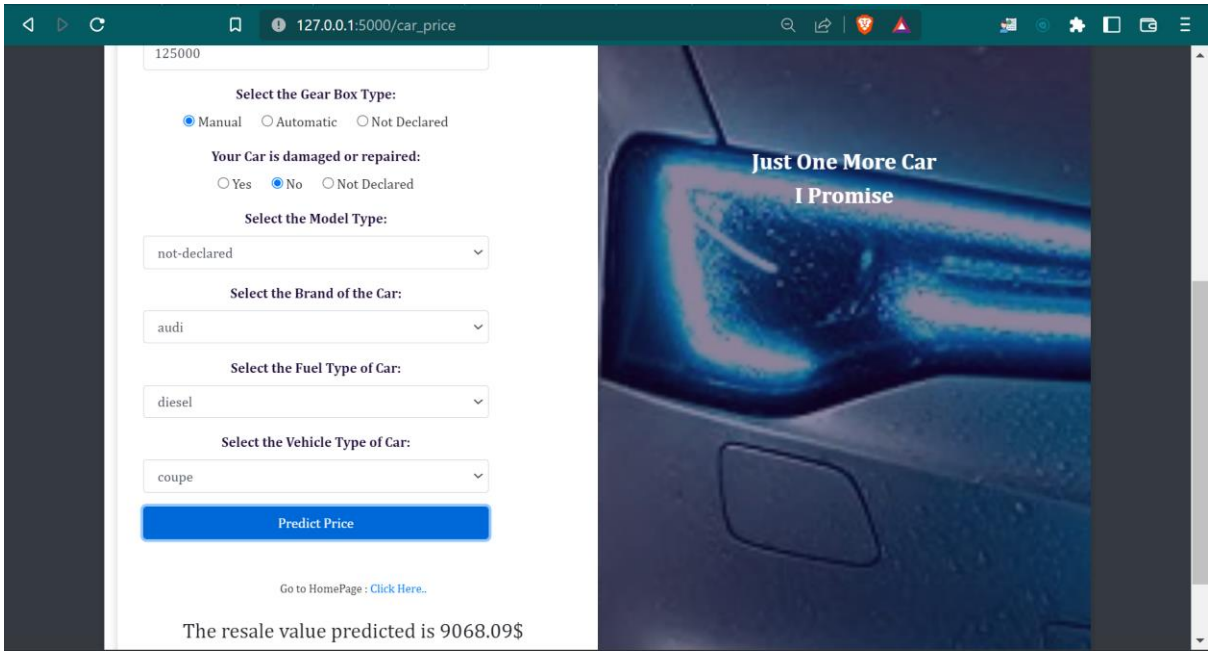

IBM Watson Studio
Search in your workspaces
Buy
Shubhavya Karthikeyan's A...
Dallas
SK
Deployments / models / CAR_RESALE_PREDICTION /
Car_resale_value_prediction
Deployed
Online
API reference
Test
Direct link
Endpoint
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/604f91b1-17c9-4661-ab72-02e7b6d5bc4e/
Bearer <token>
IAM
Code snippets
cURL
Java
JavaScript
Python
Scala
import requests
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "<your 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}
# NOTE: manually define and pass the array(s) of values to be scored in the next line
[{"fields": [array_of_input_fields], "values": [array_of_values_to_be_scored,
https://dataplatform.cloud.ibm.com/home2?context=cpdaas
Car_resale_value_prediction
Created
Nov 18, 2022, 9:18 PM
Updated
Nov 18, 2022, 9:18 PM
Deployment ID
604f91b1-17c9-4661-ab72-02...
Software specification
runtime-22.1-py3.9
Copies
1
Serving name
No serving name.
Description
No description provided.
Tags
Add tags to make assets easier to find.
Associated asset


Run: main
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(
* 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 -
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
Packages installed successfully: Installed packages: 'requests' (today 09:29 pm)
11:30 CRLF UTF-8 4 spaces Python 3.8 (Car_resale)


127.0.0.1:5000
ONLINE CAR RESALE PRICE PREDICTION
WITH DIFFICULT ECONOMIC CONDITIONS
IT IS LIKELY THAT SALES OF SECOND-HAND CARS WILL INCREASE
In most of the countries, it is common to lease a car rather than buying it outright. A lease is a binding contract between a buyer and a seller in which the buyer must pay fixed instalments for a pre-defined number of months/years to the seller/financer. After the lease period is over, the buyer has the possibility to buy the car at its residual value, i.e. its expected resale value. Thus, this application is of commercial interest to seller/financers to be able to predict the residual value of cars with accuracy.
CAR RESALE PRICE


127.0.0.1:5000/car_price
Car Price Prediction
Enter the Registration Year:
2002
Select the Registration Month:
January
Enter the Power of Car in PS:
220
Enter the Number of Kilometres that the car has travelled:
125000
Select the Gear Box Type:
Manual
Automatic
Not Declared
Your Car is damaged or repaired:
Yes
No
Not Declared
Just One More Car
I Promise


127.0.0.1:5000/car_price
125000
Select the Gear Box Type:
Manual Automatic Not Declared
Your Car is damaged or repaired:
Yes No Not Declared
Select the Model Type:
not-declared
Select the Brand of the Car:
audi
Select the Fuel Type of Car:
diesel
Select the Vehicle Type of Car:
coupe
Predict Price
Go to HomePage : Click Here..
Just One More Car
I Promise


127.0.0.1:5000/car_price
125000
Select the Gear Box Type:
Manual Automatic Not Declared
Your Car is damaged or repaired:
Yes No Not Declared
Select the Model Type:
not-declared
Select the Brand of the Car:
audi
Select the Fuel Type of Car:
diesel
Select the Vehicle Type of Car:
coupe
Predict Price
Go to HomePage : Click Here..
The resale value predicted is 9068.09$
Just One More Car
I Promise