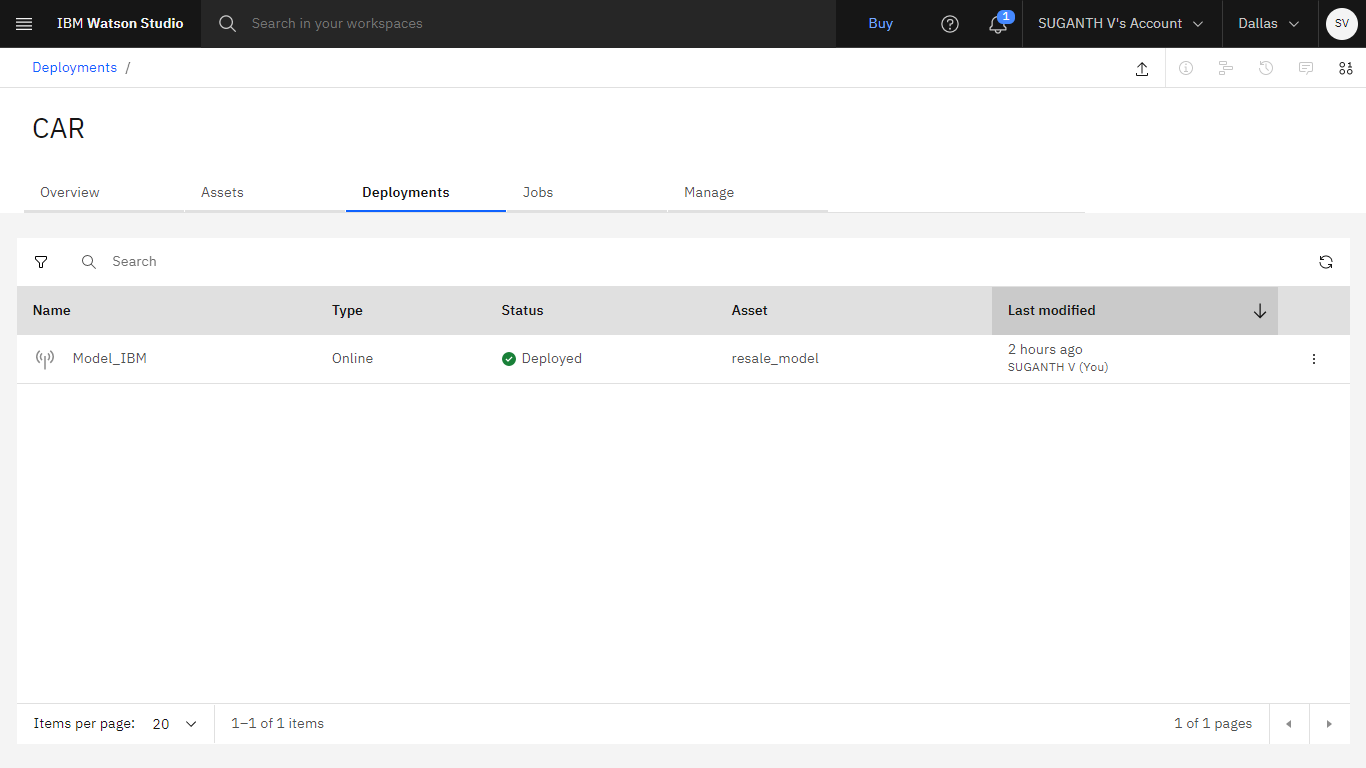
TRAIN THE ML MODEL ON IBM

|  |  |
| --- | --- |
| **Team ID** | **PNT2022TMID42057** |
| **Project Name** | **Car Resale value Prediction** |

**TRAIN THE ML MODEL ON IBM**



import pandas as pd import numpy as np import matplotlib as plt

from sklearn.preprocessing import LabelEncoder import pickle

print("IMPORTED REQUIRED LIBRARIES")

# df = pd.read\_csv("C:/Users/M.HEMNATH/Desktop/IBM/Data/autos.csv", header=0 , sep=','

,encoding='Latin1',low\_memory=False) # df.head()

import os, types import pandas as pd

from botocore.client import Config import ibm\_boto3

import io

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='DT15l-lL0017uhnUGwXyhG\_Eort5gohoW6XJTNoT3RKk', 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-yuhtmzidi0ka1p' object\_key = 'autos.csv'

body = cos\_client.get\_object(Bucket=bucket,Key=object\_key)

df = pd.read\_csv((io.BytesIO(body['Body'].read())) , header=0 , sep=',' ,encoding='Latin1',low\_memory=False) df.head()

# df = pd.read\_csv("C:/Users/M.HEMNATH/Desktop/IBM/Data/autos.csv", header=0 , sep=','

,encoding='Latin1',low\_memory=False) # df.head()

import os, types import pandas as pd

from botocore.client import Config import ibm\_boto3

import io

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='DT15l-lL0017uhnUGwXyhG\_Eort5gohoW6XJTNoT3RKk', 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-yuhtmzidi0ka1p' object\_key = 'autos.csv'

body = cos\_client.get\_object(Bucket=bucket,Key=object\_key)

df = pd.read\_csv((io.BytesIO(body['Body'].read())) , header=0 , sep=',' ,encoding='Latin1',low\_memory=False) df.head()

print(df.seller.value\_counts()) df[df.seller !='gewerblich'] df=df.drop('seller',axis=1)

print(df.offerType.value\_counts()) df[df.offerType !='Gesuch'] df=df.drop('offerType',axis=1) print(df.shape)

df=df[(df.powerPS>50) & (df.powerPS<900)] print(df.shape)

df=df[(df.yearOfRegistration>=1950)&(df.yearOfRegistration<2022)] print(df.shape)

df.drop(['name','abtest','dateCrawled','nrOfPictures','lastSeen','postalCode','dateCreated'], axis='columns',inplace=True) new\_df=df.copy() new\_df=new\_df.drop\_duplicates(['price','vehicleType','yearOfRegistration','gearbox','powerPS','model','kilometer','mo nthOfRegistration','fuelType','notRepairedDamage']) 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'),('samll car','convertible','combination','others'),inplace=True)

new\_df.notRepairedDamage.replace(('ja','nein'),('Yes','No'),inplace=True) new\_df=new\_df[(new\_df.price>=100)&(new\_df.price<=150000)]

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)

from ibm\_watson\_machine\_learning import APIClient wml\_credentials={

"url":"https://us-south.ml.cloud.ibm.com",

"apikey":"hEAn\_mcoP3u\_-ZjagjeqlxDayqUiETpYVYWdR1OLKAby"

}

client =APIClient(wml\_credentials)

def guide\_from\_space\_name(client, space\_name): space = client.spaces.get\_details()

# print(space)

return(next(item for item in space['resources'] if item['entity']["name"]==space\_name)['metadata']['id']) space\_uid=guide\_from\_space\_name(client,'CAR')

print("Space UID"+ space\_uid) client.set.default\_space(space\_uid) client.software\_specifications.list()

software\_spec\_uid = client.software\_specifications.get\_uid\_by\_name("runtime-22.1-py3.9") software\_spec\_uid

print(new\_df) labels=['gearbox','notRepairedDamage','model','brand','fuelType','vehicleType']

mapper={} for i in labels:

mapper[i]=LabelEncoder() mapper[i].fit(new\_df[i]) tr=mapper[i].transform(new\_df[i]) np.save(str('classes'+i+'.npy'),mapper[i].classes\_) print(i,":",mapper[i])

new\_df.loc[:, i+ '\_labels']=pd.Series(tr,index=new\_df.index)

labeled = new\_df[['price','yearOfRegistration','powerPS','kilometer','monthOfRegistration']+[x+"\_labels" for x in labels]]

print(labeled.columns) Y=labeled.iloc[:,0].values X=labeled.iloc[:,1:].values

Y=Y.reshape(-1,1)

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')) y\_pred = regressor.predict(X\_test) print(r2\_score(Y\_test,y\_pred)) filename='resale\_model.sav' pickle.dump(regressor,open(filename,'wb'))

model\_details = client.repository.store\_model(model=regressor,meta\_props={ client.repository.ModelMetaNames.NAME: "resale\_model", client.repository.ModelMetaNames.SOFTWARE\_SPEC\_UID: software\_spec\_uid, client.repository.ModelMetaNames.TYPE: "scikit-learn\_1.0"

})

model\_id = client.repository.get\_model\_id(model\_details) model\_id

X\_train[0]

regressor.predict([[2012.0, 179.0, '1500000', 12.0, 0, 0, 30, 1, 1, 4]])

