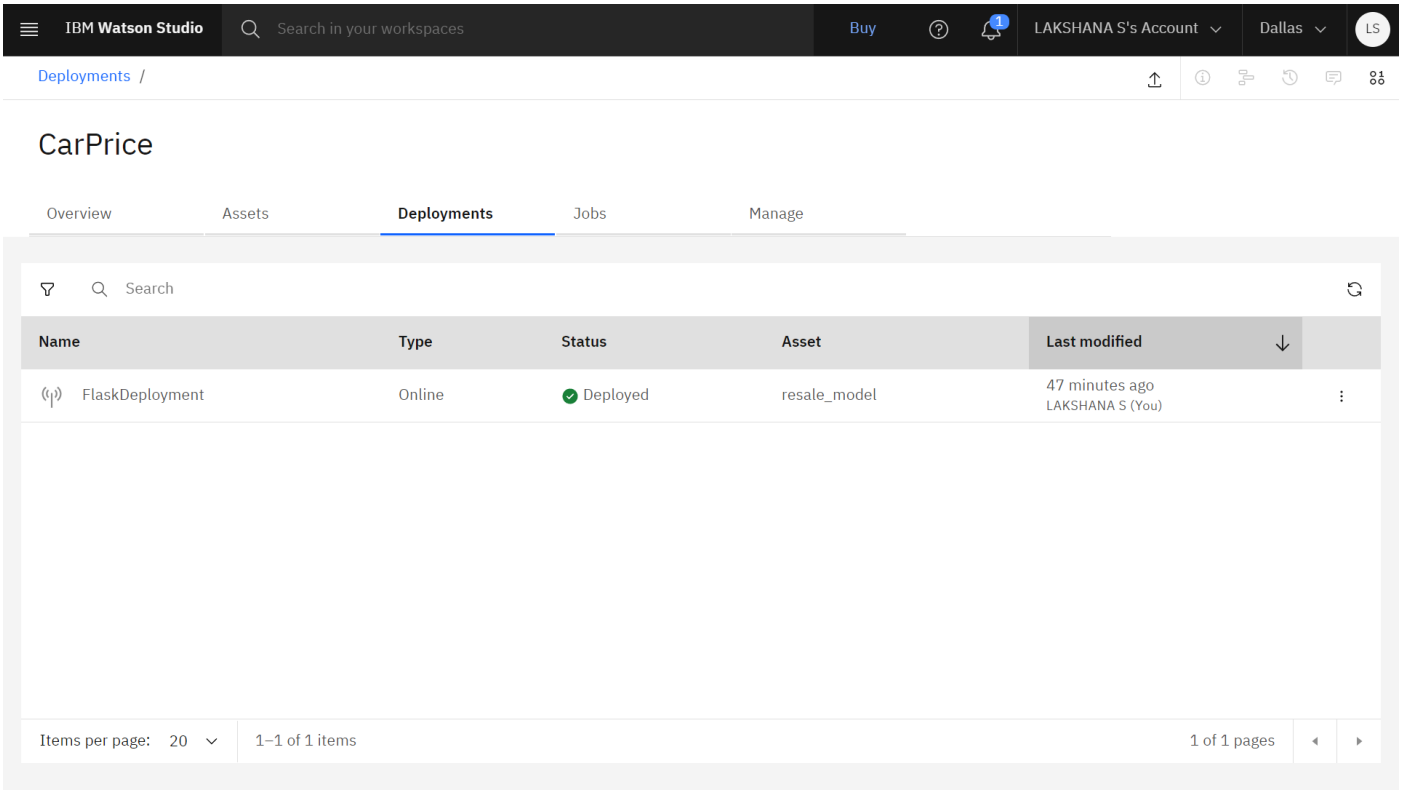
TRAIN THE ML MODEL ON IBM

Team ID	**PNT2022TMID16122**
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

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='B__tpAmla-ENJ92sDpmg_feytHFBlctw-IrdKxSRqx2w',
    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-u86776xsgyuhtj'
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','kilo
meter','monthOfRegistration','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":"MGU1iT6RDkhiyFrQhD8KbdYD1kWSOWNmSZCUhCB_IGDg"
}
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,'CarPrice')
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

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))

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([[2022.0, 179.0, '1500', 12.0, 0, 0, 30, 1, 1, 4]])
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
from sklearn.preprocessing import LabelEncoder
import pickle
```

In [2]:
```
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='B__tpAmla-ENJ92sDpmg_feytHFBlctw-IrdKxSRqx2w',
    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-u86776xsgyuhtj'
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()
```

Out[2]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelTyp
0	24-03-2016 11.52	Golf_3_1.6	privat	Angebot	480.0	test	NaN	1993.0	manuell	0.0	golf	150000	0.0	benzi

IBM Watson Studio | Search in your workspaces | Buy | LAKSHANA S's Account | Dallas | LS

Projects / CAR RESALE VALUE PREDICTION / Model Building

Out[2]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelTyp
0	24-03-2016 11.52	Golf_3_1.6	privat	Angebot	480.0	test	NaN	1993.0	manuell	0.0	golf	150000	0.0	benzi
1	24-03-2016 10.58	A5_Sportback_2.7_Tdi	privat	Angebot	18300.0	test	coupe	2011.0	manuell	190.0	NaN	125000	5.0	dies
2	14-03-2016 12.52	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800.0	test	suv	2004.0	automatik	163.0	grand	125000	8.0	dies
3	17-03-2016 16.54	GOLF_4_1_4__3TÜRER	privat	Angebot	1500.0	test	kleinwagen	2001.0	manuell	75.0	golf	150000	6.0	benzi
4	31-03-2016 17.25	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600.0	test	kleinwagen	2008.0	manuell	69.0	fabia	90000	7.0	dies

In [3]:
```
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)
```

```
privat        371534
gewerblich         3
golf               1
Name: seller, dtype: int64
Angebot    371525
Gesuch         12
150000          1
Name: offerType, dtype: int64
```

In [4]:
```
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)
```

```
(371539, 18)
(319717, 18)
(319649, 18)
```

In [5]:
```
df.drop(['name','abtest','dateCrawled','nrOfPictures','lastSeen','postalCode','dateCreated'], axis='columns',inplace=True)
```

In [6]:
```
new_df=df.copy()
new_df=new_df.drop_duplicates(['price','vehicleType','yearOfRegistration','gearbox','powerPS','model','kilometer','monthOfRegistration','fuelType','notRepai
```

In [7]:
```
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)
```

In [8]:
```
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)
```

In [9]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"MGU1iT6RDkhiyFrQhD8KbdYD1kWSOWNmSZCUhCB_IGDg"
}
client =APIClient(wml_credentials)
```

In [10]:
```
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'])
```

In [12]:
```
space_uid=guide_from_space_name(client,'CarPrice')
print("Space UID"+ space_uid)
```

```
Space UID94d5f98d-c0de-4301-b1b1-dcf187a14e30
```

In [15]:
```
client.set.default_space(space_uid)
```

Out[15]:
```
'SUCCESS'
```

In [16]:
```
client.software_specifications.list()
```

```
------------------------------  ------------------------------------  ----
NAME                            ASSET_ID                              TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6           0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                      0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base
autoai-kb_rt22.2-py3.10         125b6d9a-5b1f-5e8d-972a-b251688ccf40  base
runtime-22.1-py3.9              12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                    1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6          1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6            1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt   1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6            1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
```

In [17]:
```
software_spec_uid  = client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
software_spec_uid
```

Out[17]:
```
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

In [18]:
```
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)
```

```
gearbox : LabelEncoder()
notRepairedDamage : LabelEncoder()
model : LabelEncoder()
brand : LabelEncoder()
fuelType : LabelEncoder()
vehicleType : LabelEncoder()
Index(['price', 'yearOfRegistration', 'powerPS', 'kilometer',
       'monthOfRegistration', 'gearbox_labels', 'notRepairedDamage_labels',
       'model_labels', 'brand_labels', 'fuelType_labels',
       'vehicleType_labels'],
      dtype='object')
```

In [19]:
```
Y=labeled.iloc[:,0].values
X=labeled.iloc[:,1:].values

Y=Y.reshape(-1,1)
```

IBM Watson Studio | Search in your workspaces | Buy | LAKSHANA S's Account | Dallas | LS
Projects / CAR RESALE VALUE PREDICTION / Model Building


```
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'))
```

Out[44]:
```
RandomForestRegressor(max_depth=10, n_estimators=1000, random_state=34)
```

In [45]:
```
y_pred = regressor.predict(X_test)
print(r2_score(Y_test,y_pred))
```

```
0.8310350387286918
```

In [46]:
```
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)
```

In [47]:
```
model_id
```

Out[47]:
```
'828b7c6d-8c37-4786-88a4-4136cce344ee'
```

In [48]:
```
X_train[0]
```

Out[48]:
```
array([2005.0, 179.0, '150000', 12.0, 0, 0, 30, 1, 1, 4], dtype=object)
```

In [50]:
```
regressor.predict([[2022.0, 179.0, '1500', 12.0, 0, 0, 30, 1, 1, 4]])
```

Out[50]:
```
array([29692.51884068])
```