In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
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='Uq1L_fr0mGem56BJ158Ro0C5ks-jybAjXNehBzPseAZ1',
    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 = 'carresalevalue-donotdelete-pr-2lv9juqbtt5eqf'
object_key = 'autos.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()
```

Out[2]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen	2008	manuell	69

In [3]:
```
# df = pd.read_csv('autos.csv')
```

In [4]:
```
df.drop(columns= ['dateCrawled', 'name', 'seller', 'offerType', 'model', 'dateCreated',
                  'nrOfPictures', 'postalCode', 'lastSeen'], inplace = True)
```

In [5]:
```
df
```

Out[5]:

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	kilometer	monthOfRegistration	fuelType	brand
0	480	test	NaN	1993	manuell	0	150000	0	benzin	volkswagen
1	18300	test	coupe	2011	manuell	190	125000	5	diesel	audi
2	9800	test	suv	2004	automatik	163	125000	8	diesel	jeep
3	1500	test	kleinwagen	2001	manuell	75	150000	6	benzin	volkswagen
4	3600	test	kleinwagen	2008	manuell	69	90000	7	diesel	skoda
...	...	...	...	...	...	...	...	...	...	...
371523	2200	test	NaN	2005	NaN	0	20000	1	NaN	sonstige_autos
371524	1199	test	cabrio	2000	automatik	101	125000	3	benzin	smart
371525	9200	test	bus	1996	manuell	102	150000	3	diesel	volkswagen
371526	3400	test	kombi	2002	manuell	100	150000	6	diesel	volkswagen
371527	28990	control	limousine	2013	manuell	320	50000	8	benzin	bmw

371528 rows × 11 columns

In [6]:
```
# Removing Missing Values
df['vehicleType'].fillna(df['vehicleType'].mode()[0], inplace = True)
df['gearbox'].fillna(df['gearbox'].mode()[0], inplace = True)
df['fuelType'].fillna(df['fuelType'].mode()[0], inplace = True)
df['notRepairedDamage'].fillna(df['notRepairedDamage'].mode()[0], inplace = True)
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
price                0
abtest               0
vehicleType          0
yearOfRegistration   0
gearbox              0
powerPS              0
kilometer            0
monthOfRegistration  0
fuelType             0
brand                0
notRepairedDamage    0
dtype: int64
```

In [8]:
```
# Checking for Duplicates
df.duplicated().sum()
```

Out[8]: 24211

In [9]:
```
df = df.drop_duplicates()
```

In [10]:
```
df.duplicated().sum()
```

Out[10]: 0

In [11]:
```
df
```

Out[11]:

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	kilometer	monthOfRegistration	fuelType	brand
0	480	test	limousine	1993	manuell	0	150000	0	benzin	volkswagen
1	18300	test	coupe	2011	manuell	190	125000	5	diesel	audi
2	9800	test	suv	2004	automatik	163	125000	8	diesel	jeep
3	1500	test	kleinwagen	2001	manuell	75	150000	6	benzin	volkswagen
4	3600	test	kleinwagen	2008	manuell	69	90000	7	diesel	skoda
...	...	...	...	...	...	...	...	...	...	...
371523	2200	test	limousine	2005	manuell	0	20000	1	benzin	sonstige_autos
371524	1199	test	cabrio	2000	automatik	101	125000	3	benzin	smart
371525	9200	test	bus	1996	manuell	102	150000	3	diesel	volkswagen
371526	3400	test	kombi	2002	manuell	100	150000	6	diesel	volkswagen
371527	28990	control	limousine	2013	manuell	320	50000	8	benzin	bmw

347317 rows × 11 columns

In [12]:
```
# Dropping outliers in price
a = df[df['price'] > 50000].index
df.drop(a, inplace = True)
```

In [13]:
```
# Dropping outliers in yearOfRegistration
a = df[df['yearOfRegistration'] > 2019].index
df.drop(a, inplace = True)
a = df[df['yearOfRegistration'] < 1985].index
df.drop(a, inplace = True)
```

In [14]:
```
# Dropping outliers in powerPS
a = df[df['powerPS'] > 300].index
df.drop(a, inplace = True)
a = df[df['powerPS'] < 40].index
df.drop(a, inplace = True)
```

In [15]:
```
# Label Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['abtest'] = le.fit_transform(df['abtest'])
df['vehicleType'] = le.fit_transform(df['vehicleType'])
df['gearbox'] = le.fit_transform(df['gearbox'])
df['fuelType'] = le.fit_transform(df['fuelType'])
df['brand'] = le.fit_transform(df['brand'])
df['notRepairedDamage'] = le.fit_transform(df['notRepairedDamage'])
```

In [16]:
```
df
```

Out[16]:

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	kilometer	monthOfRegistration	fuelType	brand	notRepair
1	18300	1	3	2011	1	190	125000	5	3	1	
2	9800	1	7	2004	0	163	125000	8	3	14	
3	1500	1	4	2001	1	75	150000	6	1	38	
4	3600	1	4	2008	1	69	90000	7	3	31	
5	650	1	6	1995	1	102	150000	10	1	2	
...	...	...	...	...	...	...	...	...	...	...	...
371519	5250	0	6	2016	0	150	150000	12	1	0	
371520	3200	0	6	2004	1	225	150000	5	1	30	
371524	1199	1	2	2000	0	101	125000	3	1	32	
371525	9200	1	1	1996	1	102	150000	3	3	38	
371526	3400	1	5	2002	1	100	150000	6	3	38	

298133 rows × 11 columns

In [17]:
```
# Splitting x and y variables
x = df.drop(columns = 'price')
y = df['price']
```

In [18]:
```
# Splitting into test and train
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

Building Models

In [18]:
```
# Linear Regression
```

In [19]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[19]: LinearRegression()

In [20]:
```
# Lasso Regression
```

In [21]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=0.01, normalize=True)
lasso.fit(x_train, y_train)
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/linear_model/_base.py:141: FutureWarning: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing stage. To reproduce the previous behavior:
```

```
  warnings.warn(
```

Out[21]:
```
Lasso(alpha=0.01, normalize=True)
```

In [22]:
```
# Ridge Regression
```

In [23]:
```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=0.01, normalize=True)
ridge.fit(x_train, y_train)
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/linear_model/_base.py:141: FutureWarning: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing stage. To reproduce the previous behavior:

from sklearn.pipeline import make_pipeline

model = make_pipeline(StandardScaler(with_mean=False), Ridge())

If you wish to pass a sample_weight parameter, you need to pass it as a fit parameter to each step of the pipeline as follows:

kwargs = {s[0] + '__sample_weight': sample_weight for s in model.steps}
model.fit(X, y, **kwargs)

Set parameter alpha to: original_alpha * n_samples.
  warnings.warn(
```

Out[23]:
```
Ridge(alpha=0.01, normalize=True)
```

In [24]:
```
# Decision Tree
```

In [25]:
```
from sklearn.tree import DecisionTreeRegressor
DT = DecisionTreeRegressor()
DT.fit(x_train, y_train)
```

Out[25]:
```
DecisionTreeRegressor()
```

In [26]:
```
# KNN
```

In [27]:
```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor()
knn.fit(x_train, y_train)
```

Out[27]:
```
KNeighborsRegressor()
```

In [28]:
```
# Random Forest
```

In [19]:
```
from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor()
RF.fit(x_train, y_train)
```

Out[19]:
```
RandomForestRegressor()
```

In [30]:
```
# Linear Regression
acc_lr = lr.score(x_test, y_test)*100
```

In [33]:
```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test,lr.predict(x_test)))
```

Out[33]:
```
3956.5815636035836
```

In [34]:
```
# Lasso Regression
acc_lasso = lasso.score(x_test, y_test)*100
```

In [35]:
```
np.sqrt(mean_squared_error(y_test,lasso.predict(x_test)))
```

Out[35]:
```
3956.664250218968
```

In [36]:
```
# Ridge Regression
acc_ridge = ridge.score(x_test, y_test)*100
```

In [37]:
```
np.sqrt(mean_squared_error(y_test,ridge.predict(x_test)))
```

Out[37]:
```
3956.8355512022918
```

In [38]:
```
# K Nearest Neighbour
acc_knn = knn.score(x_test, y_test)*100
```

In [39]:
```
np.sqrt(mean_squared_error(y_test,knn.predict(x_test)))
```

Out[39]:
```
2675.1305595164295
```

In [40]:
```
# Decision Tree
acc_dt = DT.score(x_test, y_test)*100
```

In [41]:
```
np.sqrt(mean_squared_error(y_test,DT.predict(x_test)))
```

Out[41]:
```
2936.6216848726476
```

In [42]:
```
# Random Forest
acc_rf = RF.score(x_test, y_test)*100
```

In [43]:
```
np.sqrt(mean_squared_error(y_test,RF.predict(x_test)))
```

Out[43]:
```
2247.585702127866
```

In [44]:
```
models = pd.DataFrame(
    {'Model': ['Linear Regression', 'Lasso Regression', 'Ridge Regression', 'KNN',
               'Decision Tree', 'Random Forest'],
     'Score': [acc_lr, acc_lasso, acc_ridge, acc_knn, acc_dt, acc_rf]})
```

```
ite-packages (from packaging->ibm-watson-machine-learning) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

In [25]:
```
from ibm_watson_machine_learning import APIClient
import json
wml_credentials = {'apikey': 'g1tk1KaS5mB-mBztAhCCv8gMemWgixeSSsD4qW0Smqr9',
                   'url': 'https://us-south.ml.cloud.ibm.com'}
wml_client = APIClient(wml_credentials)
```

In [20]:
```
# def guid_from_space_name(client, space_name):
#     space = client.spaces.get_details()
#     return(next(iter for item in space['resources'] if item['entity']["name"] == space_name)['metao
```

In [21]:
```
# space_uid = guid_from_space_name(client, 'model1')
# print('space_uid = ' + space_uid)
```

In [26]:
```
wml_client.spaces.list()
```

```
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records excee
d 50
------------------------------------  ------  -------------------------
ID                                    NAME    CREATED
cb3b5444-ea5c-4352-914d-cf7569dd8c9f  model1  2022-11-12T06:55:56.978Z
------------------------------------  ------  -------------------------
```

In [27]:
```
space_id = 'cb3b5444-ea5c-4352-914d-cf7569dd8c9f'
```

In [29]:
```
wml_client.set.default_space(space_id)
```

Out[29]:
```
'SUCCESS'
```

In [42]:
```
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
spark-mllib_3.2	20047f72-0a98-58c7-9ff5-a77b012eb8f5	base
tensorflow_2.4-py3.8-horovod	217c16f6-178f-56bf-824a-b19f20564c49	base
runtime-22.1-py3.9-cuda	26215f05-08c3-5a41-a1b0-da66306ce658	base
do_py3.8	295addb5-9ef9-547e-9bf4-92ae3563e720	base
autoai-ts_3.8-py3.8	2aa0c932-798f-5ae9-abd6-15e0c2402fb5	base
tensorflow_1.15-py3.6	2b73a275-7cbf-420b-a912-eae7f436e0bc	base
kernel-spark3.3-py3.9	2b7961e2-e3b1-5a8c-a491-482c8368839a	base
pytorch_1.2-py3.6	2c8ef57d-2687-4b7d-acce-01f94976dac1	base
spark-mllib_2.3	2e51f700-bca0-4b0d-88dc-5c6791338875	base
pytorch-onnx_1.1-py3.6-edt	32983cea-3f32-4400-8965-dde874a8d67e	base
spark-mllib_3.0-py37	36507ebe-8770-55ba-ab2a-eafe787600e9	base
spark-mllib_2.4	390d21f8-e58b-4fac-9c55-d7ceda621326	base
autoai-ts_rt22.2-py3.10	396b2e83-0953-5b86-9a55-7ce1628a406f	base
xgboost_0.82-py3.6	39e31acd-5f30-41dc-ae44-60233c80306e	base
pytorch-onnx_1.2-py3.6-edt	40589d0e-7019-4e28-8daa-fb03b6f4fe12	base
pytorch-onnx_rt22.2-py3.10	40e73f55-783a-5535-b3fa-0c8b94291431	base
default_r36py38	41c247d3-45f8-5a71-b065-8580229facf0	base
autoai-ts_rt22.1-py3.9	4269d26e-07ba-5d40-8f66-2d495b0c71f7	base
autoai-obm_3.0	42b92e18-d9ab-567f-988a-4240ba1ed5f7	base
pmml-3.0_4.3	493bcb95-16f1-5bc5-bee8-81b8af80e9c7	base
spark-mllib_2.4-r_3.6	49403dff-92e9-4c87-a3d7-a42d0021c095	base
xgboost_0.90-py3.6	4ff8d6c2-1343-4c18-85e1-689c965304d3	base
pytorch-onnx_1.1-py3.6	50f95b2a-bc16-43bb-bc94-b0bed208c60b	base
autoai-ts_3.9-py3.8	52c57136-80fa-572e-8728-a5e7cbb42cde	base
spark-mllib_2.4-scala_2.11	55a70f99-7320-4be5-9fb9-9edb5a443af5	base
spark-mllib_3.0	5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9	base
autoai-obm_2.0	5c2e37fa-80b8-5e77-840f-d912469614ee	base
spss-modeler_18.1	5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b	base
cuda-py3.8	5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e	base
autoai-kb_3.1-py3.7	632d4b22-10aa-5180-88f0-f52dfb6444d7	base
pytorch-onnx_1.7-py3.8	634d3cdc-b562-5bf9-a2d4-ea90a478456b	base

```
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

In [56]:
```
import sklearn
sklearn.__version__
```

Out[56]:
```
'1.0.2'
```

In [30]:
```
model_name = 'rf'
deployment_name = 'rf'
demo_model = RF
```

In [31]:
```
software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
software_spec_uid
```

Out[31]:
```
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

In [32]:
```
model_props = {
    wml_client.repository.ModelMetaNames.NAME: model_name,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

In [33]:
```
model_details = wml_client.repository.store_model(
    model = demo_model,
    meta_props=model_props,
    training_data = x_train,
    training_target = y_train)
```

```
autoai-obm_2.0                5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1             5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                    5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
autoai-kb_3.1-py3.7           632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8        634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
------------------------------ ------------------------------------ ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

In [56]:
```
import sklearn
sklearn.__version__
```

Out[56]: '1.0.2'

In [30]:
```
model_name = 'rf'
deployment_name = 'rf'
demo_model = RF
```

In [31]:
```
software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
software_spec_uid
```

Out[31]: '12b83a17-24d8-5082-900f-0ab31fbfd3cb'

In [32]:
```
model_props = {
    wml_client.repository.ModelMetaNames.NAME: model_name,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

In [33]:
```
model_details = wml_client.repository.store_model(
    model = demo_model,
    meta_props=model_props,
    training_data = x_train,
    training_target = y_train)
```

In [34]:
```
model_details
```

Out[34]:
```
{'entity': {'hybrid_pipeline_software_specs': [],
  'label_column': 'price',
  'schemas': {'input': [{'fields': [{'name': 'abtest', 'type': 'int64'},
      {'name': 'vehicleType', 'type': 'int64'},
      {'name': 'yearOfRegistration', 'type': 'int64'},
      {'name': 'gearbox', 'type': 'int64'},
      {'name': 'powerPS', 'type': 'int64'},
      {'name': 'kilometer', 'type': 'int64'},
      {'name': 'monthOfRegistration', 'type': 'int64'},
      {'name': 'fuelType', 'type': 'int64'},
      {'name': 'brand', 'type': 'int64'},
      {'name': 'notRepairedDamage', 'type': 'int64'}],
     'id': '1',
     'type': 'struct'}],
   'output': []},
  'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
   'name': 'runtime-22.1-py3.9'},
  'type': 'scikit-learn_1.0'},
 'metadata': {'created_at': '2022-11-12T08:02:16.890Z',
  'id': 'dfaf6ba0-6752-4846-abb2-91f5c4ff27e0',
  'modified_at': '2022-11-12T08:08:20.432Z',
  'name': 'rf',
  'owner': 'IBMid-666002OF0N',
  'resource_key': '5d76cf1d-a018-474e-824d-11a308540d6c',
  'space_id': 'cb3b5444-ea5c-4352-914d-cf7569dd8c9f'},
 'system': {'warnings': []}}
```

In [35]:
```
model_id = wml_client.repository.get_model_id(model_details)
```

In [36]:
```
model_id
```

Out[36]: 'dfaf6ba0-6752-4846-abb2-91f5c4ff27e0'

In [37]:
```
deployment_props  = {
    wml_client.deployments.ConfigurationMetaNames.NAME:deployment_name,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

In [38]:
```
deployment = wml_client.deployments.create(
artifact_uid = model_id,
meta_props = deployment_props)
```

```
#######################################################################################

Synchronous deployment creation for uid: 'dfaf6ba0-6752-4846-abb2-91f5c4ff27e0' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.
....
ready


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='7a83cc3a-f415-4c17-9e47-da929f749bc5'
------------------------------------------------------------------------------------------------
```

In [ ]:
```

```

Give feedback