DATE : 11 November 2022

TEAM ID : PNT2022TMID21221

PROJECT NAME: Car Resale Value Prediction

PRE-PROCESS THE DATA

Import Required Libraries

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

Read the datasets

In [2]:
```
df = pd.read_csv("autos.csv")
df.head()
```

Out[2]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv
3	2016-03-17 16:54:04	GOLF_4_1_4__3T�RER	privat	Angebot	1500	test	kleinwagen
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen

In [3]:
```
df.tail()
```

Out[3]:

	dateCrawled	name	seller	offerType	price
371523	2016-03-14 17:48:27	Suche_t4___vito_ab_6_sitze	privat	Angebot	2200
371524	2016-03-05 19:56:21	Smart_smart_leistungssteigerung_100ps	privat	Angebot	1199
371525	2016-03-19 18:57:12	Volkswagen_Multivan_T4_TDI_7DC_UY2	privat	Angebot	9200
371526	2016-03-20 19:41:08	VW_Golf_Kombi_1_9l_TDI	privat	Angebot	3400
371527	2016-03-07 19:39:19	BMW_M135i_vollausgestattet_NP_52.720____Euro	privat	Angebot	28990

Cleaning the dataset

In [4]:
```
#printing different sellers
print(df.seller.value_counts())
```

```
privat        371525
gewerblich         3
Name: seller, dtype: int64
```

In [5]:
```
#removing the seller "gewerblich"
df[df.seller != 'gewerblich']
```

Out[5]:

	dateCrawled	name	seller	offerType	price	ab
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	
3	2016-03-17 16:54:04	GOLF_4_1_4__3T`  `RER	privat	Angebot	1500	
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	
...	...	...	...	...	...	...
371523	2016-03-14 17:48:27	Suche_t4___vito_ab_6_sitze	privat	Angebot	2200	
371524	2016-03-05 19:56:21	Smart_smart_leistungssteigerung_100ps	privat	Angebot	1199	
371525	2016-03-19 18:57:12	Volkswagen_Multivan_T4_TDI_7DC_UY2	privat	Angebot	9200	
371526	2016-03-20 19:41:08	VW_Golf_Kombi_1_9l_TDI	privat	Angebot	3400	
371527	2016-03-07 19:39:19	BMW_M135i_vollausgestattet_NP_52.720____Euro	privat	Angebot	28990	cor

71525 rows × 20 columns

In [6]:
```
#dropping the coloumn seller as all the entries are same
df = df.drop('seller',1)
```

In [7]:
```
#printing different offerType
print(df.offerType.value_counts())
```

```
Angebot    371516
Gesuch         12
Name: offerType, dtype: int64
```

In [8]:
```
#dropping the offerType 'Gesuch'
df[df.offerType != 'Gesuch']
```

Out[8]:

	dateCrawled	name	offerType	price	abtest
0	2016-03-24 11:52:17	Golf_3_1.6	Angebot	480	test
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	Angebot	18300	test
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	Angebot	9800	test
3	2016-03-17 16:54:04	GOLF_4_1_4__3T�RER	Angebot	1500	test
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	Angebot	3600	test
...	...	...	...	...	...
371523	2016-03-14 17:48:27	Suche_t4___vito_ab_6_sitze	Angebot	2200	test
371524	2016-03-05 19:56:21	Smart_smart_leistungssteigerung_100ps	Angebot	1199	test
371525	2016-03-19 18:57:12	Volkswagen_Multivan_T4_TDI_7DC_UY2	Angebot	9200	test
371526	2016-03-20 19:41:08	VW_Golf_Kombi_1_9l_TDI	Angebot	3400	test
371527	2016-03-07 19:39:19	BMW_M135i_vollausgestattet_NP_52.720____Euro	Angebot	28990	control

371516 rows × 19 columns

In [9]:
```
#dropping the coloumn offerType since it has the same entries
df = df.drop('offerType',1)
```

In [10]:
```
print(df.shape)
```

```
(371528, 18)
```

In [11]:
```
#removing cars having power less that 50p and greater than 900p
df = df[(df.powerPS > 50) & (df.powerPS < 900)]
print(df.shape)
```

```
(319709, 18)
```

In [12]:
```
#Keeping all the cars which is registered between 1950 and 2017 and removing
df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]
print(df.shape)
```

```
(309171, 18)
```

In [13]:
```
#removing irrelevant coloumns
df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen', 'postal
       inplace = True)
```

In [14]:
```
#dropping the duplicates in the dataframe and storing it in a new dataframe
newdf = df.copy()
newdf = newdf.drop_duplicates(['price', 'vehicleType', 'yearOfRegistration',
                               'monthOfRegistration', 'fuelType', 'notRepaired
```

In [15]:
```
#replacing the german words with proper english words
newdf.gearbox.replace(('manuell','automatik'), ('manual', 'automatic'), inpla
newdf.fuelType.replace(('benzin','andere','elektro'), ('petrol', 'others', 'e
newdf.vehicleType.replace(('kleinwagen','cabrio','kombi','andere'), ('small c
                          'others'), inplace = True)
newdf.notRepairedDamage.replace(('ja','nein'), ('yes', 'no'), inplace = True)
```

In [16]:
```
#Removing the outliers
newdf = newdf[(newdf.price >= 100) & (newdf.price < 15000)]
```

In [17]:
```
#filling NaN using fillna
newdf['notRepairedDamage'].fillna(value = 'not-declared', inplace = True)
newdf['fuelType'].fillna(value = 'not-declared', inplace = True)
newdf['gearbox'].fillna(value = 'not-declared', inplace = True)
newdf['vehicleType'].fillna(value = 'not-declared', inplace = True)
newdf['model'].fillna(value = 'not-declared', inplace = True)
```

In [18]:
```
#saving the cleaned dataset
newdf.to_csv("autos_preprocessed.csv")
```

In [19]:
```
#Label Encoding the categorical data
labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehi
mapping = {}
for i in labels:
    mapping[i] = LabelEncoder()
    mapping[i].fit(newdf[i])
    trans = mapping[i].transform(newdf[i])
    np.save(str('classes'+i+'.npy'),mapping[i].classes_)
    print(i,":",mapping[i])
    newdf.loc[:,i+'_labels'] = pd.Series(trans, index = newdf.index)

#final data is put inside a new dataframe called labeled
labeled = newdf[["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')
```

Splitting data into independent and dependent variables

In [20]:
```
#sorting price in Y and rest of the data in X
Y = labeled.iloc[:,0].values
X = labeled.iloc[:,1:].values
Y = Y.reshape(-1,1)
```

In [21]:
```
#splitting the dataset into testing and training set
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, ra
```

MODEL BUILDING

In [22]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
regressor = RandomForestRegressor(n_estimators=1000, max_depth = 10, random_s
#fitting the model
regressor.fit(X_train, np.ravel(Y_train,order='C'))
```

Out[22]:
```
RandomForestRegressor(max_depth=10, n_estimators=1000, random_state=34)
```

In [23]:
```
#predicting the values of test test
y_pred = regressor.predict(X_test)
#predicting the accuracy for test set
print(r2_score(Y_test, y_pred))
```

```
0.8118243572325188
```

In [24]:
```
#saving the model for future use
filename = 'resale_model.sav'
pickle.dump(regressor, open(filename,'wb'))
```