

CLEANING THE DATASET

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Profile

Files

- sample_data
- autos.csv
- autos_preprocessed.csv
- classesbrand.npy
- classesfuelType.npy
- classesgearbox.npy
- classesmodel.npy
- classesnotRepairedDamage.npy
- classesvehicleType.npy

Code

```
[72] df = pd.read_csv("autos.csv", header=0, sep=',', encoding='Latin1',)

# print all the different sellers
print(df.seller.value_counts())
# remove the seller type having only 3 cars
df[df.seller != 'gewerblich']
# now all the sellers are same so we can get rid of this column
df=df.drop('seller',1)
# print all the different sellers
print(df.offerType.value_counts())
# remove the Offer Type having only 12 listings
df[df.offerType != 'Gesuch']
# now all the offers are same so we can get rid of this column
df=df.drop('offerType',1)
```

privat 371525
gewerblich 3
Name: seller, dtype: int64
Angebot 371516
Gesuch 12

0s completed at 12:15 PM

Read the dataset.pdf Doc2.pdf Untitled3.ipynb image9.png

Activate Windows Go to Settings to activate Windows. Show all

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Profile

Files

- sample_data
- autos.csv
- autos_preprocessed.csv
- classesbrand.npy
- classesfuelType.npy
- classesgearbox.npy
- classesmodel.npy
- classesnotRepairedDamage.npy
- classesvehicleType.npy

Code

```
[74] #Cars having power less than 50ps and above 900ps seems a little suspicious,
#let's remove them and see what we've got now
print(df.shape)
df = df[(df.powerPS > 50) & (df.powerPS < 900)]
print(df.shape)
#around 50000 cars have been removed which could have introduced error to our data
#similarly, filtering out the cars having registration years not in the mentioned range
#print(df.shape)
df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]
print(df.shape)
# not much of a difference but still, 10000 rows have been reduced. it's better to get rid of faulty data

(371528, 18)
(319709, 18)
(309171, 18)

[75] #removing irrelevant columns which are either the same for all the cars in the dataset, or can introduce
df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen',
        'postalCode', 'dateCreated'], axis='columns', inplace=True)

[76] #is from the dataframe and storing it in a new df.
#same value in all the mentioned columns will be deleted and by default,
```

Activate Windows Go to Settings to activate Windows.

Files



sample_data
autos.csv
autos_preprocessed.csv
classesbrand.npy
classesfuelType.npy
classesgearbox.npy
classesmodel.npy
classesnotRepairedDamage.npy
classesvehicleType.npy

<>

≡

⌵

Disk 85.08 GB available

+ Code + Text

RAM Disk Editing

```
[76] is from the dataframe and stroing it in a new df.
same value in all the mentioned columns will be deleted and by default,
>f any such row is kept

duplicates(['price','vehicleType','gearbox','powerPS','model','kilometer','fuelType','notRepairedDamage'])

[77] #As the dataset contained some german words for many features, cahnging them to english
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)

#### Removing the outliers
new_df = new_df[(new_df.price >= 100) & (new_df.price <= 150000)]
#Filling NaN values for columns whose data might not be there with the information provider, #which might
#but we will still be able to give some estimate to the user
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)
```

Activate Windows
Go to Settings to activate Windows.

Files



sample_data
autos.csv
autos_preprocessed.csv
classesbrand.npy
classesfuelType.npy
classesgearbox.npy
classesmodel.npy
classesnotRepairedDamage.npy
classesvehicleType.npy

<>

≡

⌵

Disk 85.08 GB available

+ Code + Text

RAM Disk Editing

```
[79] #can save the csv for future purpose.
new_df.to_csv("autos_preprocessed.csv")

#Columns which contain categorical values, which we'll need to convert via label encoding
labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']
#looping over the labels to do the label encoding for all at once and #saving the LABEL ENCODING FILES
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)
#Final data to be put in a new dataframe called "LABLED",
labeled = new_df[ ['price','yearOfRegistration','powerPS','kilometer','monthOfRegistration']+["_labels"]
print(labeled.columns)

gearbox : LabelEncoder()
notRepairedDamage : LabelEncoder()
model : LabelEncoder()
brand : LabelEncoder()
fuelType : LabelEncoder()
vehicleType : LabelEncoder()
yearOfRegistration : LabelEncoder()
powerPS : LabelEncoder()
kilometer : LabelEncoder()
monthOfRegistration : LabelEncoder()
```

Activate Windows
Go to Settings to activate Windows.