Data Analysis for eBay Car Sales in Germany

February 26, 2019

1 Project: Data Analysis for eBay Car Sales in Germany

--By Lu Tang

1.1 Table of Contents

Introduction

> This project will analyze the vehicle market in Germany. The dataset used in the project was scraped and uploaded to Kaggle https://www.kaggle.com/orgesleka/used-cars-database/data, saved as 'auto_kaggle.csv'.

The data columns description as following: - `dateCrawled` - When this ad was first crawled. All field-values are taken from this date. - `name` - Name of the car. - `seller` - Whether the seller is private or a dealer. - `offerType` - The type of listing - `price` - The price on the ad to sell the car. - `abtest` - Whether the listing is included in an A/B test. - `vehicleType` - The vehicle Type. - `yearOfRegistration` - The year in which which year the car was first registered. - `gearbox` - The transmission type. - `powerPS` - The power of the car in PS. - `model` - The car model name. - `kilometer` - How many kilometers the car has driven. - `monthOfRegistration` - The month in which year the car was first registered. - `fuelType` - What type of fuel the car uses. - `brand` - The brand of the car. - `notRepairedDamage` - If the car has a damage which is not yet repaired. - `dateCreated` - The date on which the eBay listing was created. - `nrOfPictures` - The number of pictures in the ad. - `postalCode` - The postal code for the location of the vehicle. - `lastSeenOnline` - When the crawler saw this ad last online.

The project amis to answer the following questions: > - Question 1: What is the most common brands of cars in Germany and their listed average prices? > - Question 2: Among common brands, are there large differences on kilometer that can affect listing price? > - Question 3: What are the factors that can affect car prices?

Data Wrangling

1.1.1 Step1_1. Initial Data Exploring and drop irrelevant columns and duplicated rows

```
In [2]: # Import the libraries we will use
        import matplotlib.pyplot as plt
```

```python
import seaborn as sns
%matplotlib inline
import pandas as pd
import numpy as np

# Loading data and check information and first 3 rows
autos=pd.read_csv('autos_kaggle.csv', encoding='Latin-1')
autos.info()
autos.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 371528 entries, 0 to 371527
Data columns (total 20 columns):
dateCrawled            371528 non-null object
name                   371528 non-null object
seller                 371528 non-null object
offerType              371528 non-null object
price                  371528 non-null int64
abtest                 371528 non-null object
vehicleType            333659 non-null object
yearOfRegistration     371528 non-null int64
gearbox                351319 non-null object
powerPS                371528 non-null int64
model                  351044 non-null object
kilometer              371528 non-null int64
monthOfRegistration    371528 non-null int64
fuelType               338142 non-null object
brand                  371528 non-null object
notRepairedDamage      299468 non-null object
dateCreated            371528 non-null object
nrOfPictures           371528 non-null int64
postalCode             371528 non-null int64
lastSeen               371528 non-null object
dtypes: int64(7), object(13)
memory usage: 56.7+ MB
```

```
Out[2]:            dateCrawled                                name  seller offerType  \
        0  2016-03-24 11:52:17                          Golf_3_1.6  privat   Angebot
        1  2016-03-24 10:58:45                A5_Sportback_2.7_Tdi  privat   Angebot
        2  2016-03-14 12:52:21  Jeep_Grand_Cherokee_"Overland"  privat   Angebot
        3  2016-03-17 16:54:04                 GOLF_4_1_4__3TÜRER  privat   Angebot
        4  2016-03-31 17:25:20  Skoda_Fabia_1.4_TDI_PD_Classic  privat   Angebot

           price abtest vehicleType  yearOfRegistration    gearbox  powerPS  model  \
        0    480   test         NaN                1993    manuell        0   golf
        1  18300   test       coupe                2011    manuell      190    NaN
        2   9800   test         suv                2004  automatik      163  grand
```

```
3   1500   test  kleinwagen                2001     manuell       75   golf
4   3600   test  kleinwagen                2008     manuell       69  fabia

   kilometer  monthOfRegistration fuelType       brand notRepairedDamage  \
0     150000                    0   benzin  volkswagen               NaN
1     125000                    5   diesel        audi                ja
2     125000                    8   diesel        jeep               NaN
3     150000                    6   benzin  volkswagen              nein
4      90000                    7   diesel       skoda              nein

           dateCreated  nrOfPictures  postalCode            lastSeen
0  2016-03-24 00:00:00             0       70435  2016-04-07 03:16:57
1  2016-03-24 00:00:00             0       66954  2016-04-07 01:46:50
2  2016-03-14 00:00:00             0       90480  2016-04-05 12:47:46
3  2016-03-17 00:00:00             0       91074  2016-03-17 17:40:17
4  2016-03-31 00:00:00             0       60437  2016-04-06 10:17:21
```

Analysis

- Column names need to be changed to be more descriptive and easier to work with.
- There are some columns contain null-value data.
- Some columns may not useful for analysis.
- Some columns contain non-English words and need to change to English to undertand.

```
In [3]: autos.describe(include='all')
```

```
Out[3]:                 dateCrawled         name  seller offerType         price  \
        count                371528       371528  371528    371528  3.715280e+05
        unique               280500       233531       2         2           NaN
        top     2016-03-24 14:49:47  Ford_Fiesta  privat   Angebot           NaN
        freq                      7          657  371525    371516           NaN
        mean                    NaN          NaN     NaN       NaN  1.729514e+04
        std                     NaN          NaN     NaN       NaN  3.587954e+06
        min                     NaN          NaN     NaN       NaN  0.000000e+00
        25%                     NaN          NaN     NaN       NaN  1.150000e+03
        50%                     NaN          NaN     NaN       NaN  2.950000e+03
        75%                     NaN          NaN     NaN       NaN  7.200000e+03
        max                     NaN          NaN     NaN       NaN  2.147484e+09

                abtest vehicleType  yearOfRegistration  gearbox        powerPS  \
        count   371528      333659       371528.000000   351319  371528.000000
        unique       2           8                 NaN        2            NaN
        top       test   limousine                 NaN  manuell            NaN
        freq    192585       95894                 NaN   274214            NaN
        mean       NaN         NaN         2004.577997      NaN     115.549477
        std        NaN         NaN           92.866598      NaN     192.139578
        min        NaN         NaN         1000.000000      NaN       0.000000
```

```
25%          NaN          NaN          1999.000000       NaN      70.000000
50%          NaN          NaN          2003.000000       NaN     105.000000
75%          NaN          NaN          2008.000000       NaN     150.000000
max          NaN          NaN          9999.000000       NaN   20000.000000

         model     kilometer  monthOfRegistration fuelType       brand  \
count   351044 371528.000000        371528.000000   338142      371528
unique     251           NaN                  NaN        7          40
top       golf           NaN                  NaN   benzin  volkswagen
freq     30070           NaN                  NaN   223857       79640
mean       NaN 125618.688228             5.734445      NaN         NaN
std        NaN  40112.337051             3.712412      NaN         NaN
min        NaN   5000.000000             0.000000      NaN         NaN
25%        NaN 125000.000000             3.000000      NaN         NaN
50%        NaN 150000.000000             6.000000      NaN         NaN
75%        NaN 150000.000000             9.000000      NaN         NaN
max        NaN 150000.000000            12.000000      NaN         NaN

       notRepairedDamage          dateCreated  nrOfPictures   postalCode  \
count             299468               371528      371528.0 371528.00000
unique                 2                  114           NaN          NaN
top                 nein  2016-04-03 00:00:00           NaN          NaN
freq              263182                14450           NaN          NaN
mean                 NaN                  NaN           0.0  50820.66764
std                  NaN                  NaN           0.0  25799.08247
min                  NaN                  NaN           0.0   1067.00000
25%                  NaN                  NaN           0.0  30459.00000
50%                  NaN                  NaN           0.0  49610.00000
75%                  NaN                  NaN           0.0  71546.00000
max                  NaN                  NaN           0.0  99998.00000

                   lastSeen
count                371528
unique               182806
top     2016-04-06 13:45:54
freq                     17
mean                    NaN
std                     NaN
min                     NaN
25%                     NaN
50%                     NaN
75%                     NaN
max                     NaN
```

Analysis:

- `seller` and `offerType` only have 2 unique value, with more than 370000 frequency;

- The following columns have *odd max and min value*: `price` `yearOfRegistration` `powerPS` `nrOfPictures`

In [4]:
```
autos["seller"].value_counts()
```

Out[4]:
```
privat        371525
gewerblich         3
Name: seller, dtype: int64
```

In [5]:
```
autos["offerType"].value_counts()
```

Out[5]:
```
Angebot    371516
Gesuch         12
Name: offerType, dtype: int64
```

In [6]:
```
autos["nrOfPictures"].value_counts()
```

Out[6]:
```
0    371528
Name: nrOfPictures, dtype: int64
```

Analysis: >`seller` and `offerType` have most of the values the same; `nrOfPictures`column has 0 for every column; `dateCrawled`, `abtest`, `nrOfPictures`, `monthOfRegistration`, `postalCode` and `lastSeen` are irrelevant to our analysis for car price, so we can drop these columns.

In [7]:
```
#Drop unnecessary columns
drop_col=['seller', 'offerType', 'abtest', 'dateCrawled', 'nrOfPictures', 'monthOfRegi
autos = autos.drop(drop_col, axis=1)
autos.head(1)
```

Out[7]:
```
         name  price vehicleType  yearOfRegistration  gearbox  powerPS model  \
0  Golf_3_1.6    480         NaN                1993  manuell        0  golf

   kilometer fuelType       brand notRepairedDamage          dateCreated
0     150000   benzin  volkswagen               NaN  2016-03-24 00:00:00
```

In [8]:
```
# Find out how many rows are duplicated
sum(autos.duplicated())
```

Out[8]:
```
3934
```

In [9]:
```
# Drop duplicated rows
autos.drop_duplicates(inplace=True)
```

1.1.2 Step1_2. Clean Column name

In [10]:
```
autos.columns
```

Out[10]:
```
Index(['name', 'price', 'vehicleType', 'yearOfRegistration', 'gearbox',
       'powerPS', 'model', 'kilometer', 'fuelType', 'brand',
       'notRepairedDamage', 'dateCreated'],
      dtype='object')
```

Analysis: >- Change the columns from camelcase to snakecase. >- Change a few wordings to more accurately describe the columns.

```
In [11]: autos.columns = ['name', 'price', 'vehicle_type', 'registration_year', 'gearbox', 'pow
                'kilometer','fuel_type', 'brand','unrepaired_damage', 'ad_created']
         autos.head(1)
```

```
Out[11]:          name  price vehicle_type  registration_year  gearbox  power_ps model  \
         0  Golf_3_1.6    480          NaN               1993  manuell         0  golf

            kilometer fuel_type       brand unrepaired_damage           ad_created
         0     150000    benzin  volkswagen               NaN  2016-03-24 00:00:00
```

1.1.3 Step1_3 Investigate the columns (1.'price', 2.'registration_year', 3.'power_ps') that have abnormal values:

1. Investigate on "price" column

```
In [12]: # Find out the rows with extreme small value on price.
         autos["price"].value_counts().sort_index().head(10)
```

```
Out[12]: 0     10667
         1      1176
         2        12
         3         7
         4         1
         5        26
         7         3
         8         9
         9         8
         10       83
         Name: price, dtype: int64
```

```
In [13]: # Find out the rows with extreme large value on price
         autos["price"].value_counts().sort_index(ascending=False).head(10)
```

```
Out[13]: 2147483647     1
         99999999      15
         99000000       1
         74185296       1
         32545461       1
         27322222       1
         14000500       1
         12345678       9
         11111111      10
         10010011       1
         Name: price, dtype: int64
```

```
In [14]: # Find out how many car prices are under 100
         sum(autos["price"]<=100)
```

Out[14]: 14218

```
In [15]: # Find out how many car prices are over 200000
         sum(autos["price"]>200000)
```

Out[15]: 170

Analysis: > - As ebay is an auction site, it is possible to have listing with opening bid very low, based on common sense, we assume any price under 100 is too low. The amount of cars with price under 100 is less than 4%, so we will remove these rows. > - Although it is possible for luxury cars with very high price, we will limit the price within 200000 in our analysis

```
In [16]: # Remove the rows with price values under 100 and above 200000
         autos=autos[autos['price'].between(100,200000)]
```

2. Investigate on 'registration_year' column

```
In [17]: # Find out the extreme small value with percentage
         autos["registration_year"].value_counts(normalize=True).sort_index().head()
```

```
Out[17]: 1000    0.000065
         1001    0.000003
         1039    0.000003
         1111    0.000003
         1234    0.000011
         Name: registration_year, dtype: float64
```

```
In [18]: # Find out extreme large value with percentage
         autos["registration_year"].value_counts(normalize=True).sort_index(ascending=False).he
```

```
Out[18]: 9999    0.000037
         9450    0.000003
         9000    0.000011
         8888    0.000006
         8500    0.000003
         Name: registration_year, dtype: float64
```

Analysis: >- There are some listings with extremely small and large registration years, but the percentage is small. Based on common sense, we will cut the registration year by 1950. >- We will use the year of the 'ad_created' as the threshold year for the highes values for registration_year because the car can be be listed on sale before it's registered.

```
In [19]: # We can also use the following method to find out the sales year
         (autos["ad_created"]
                 .str[:4]
                 .value_counts(normalize=True, dropna=False)
                 .sort_index()
                 )
```

```
Out[19]: 2014    0.000003
         2015    0.000082
         2016    0.999915
         Name: ad_created, dtype: float64
```

Analysis: >- Most of the cars in this dataset are for sale in 2016

```
In [20]: # The percentage of our data that has unrealistic values in this column
         (~autos['registration_year'].between(1900,2016)).sum()/autos.shape[0]
```

```
Out[20]: 0.03893460555845696
```

```
In [21]: # As the number ablove is below 4%, we will remove rows with value below 1900 and abo
         autos=autos[autos['registration_year'].between(1900,2016)]
```

```
In [22]: # Since we have found out most of the list are in 2016, this is unrelated information
         # We can drop this columns.
         autos.drop('ad_created', axis=1,inplace=True)
```

3. Investigate on 'power_ps' column and do the same analysis and remove the rows with unrealistic values

```
In [23]: autos=autos[autos['power_ps'].between(10,500)]
```

1.1.4 Step1_ 4 Change the values in the columns (1. gearbox, 2. 'unrepaired_damage') which have only 2 unique values and are not in English

1.'gearbox'

```
In [24]: autos.gearbox.value_counts()
```

```
Out[24]: manuell      234851
         automatik     68018
         Name: gearbox, dtype: int64
```

```
In [25]: mapping_dict2={'manuell':'manual', 'automatik':'automatic'}
         autos['gearbox']=autos['gearbox'].map(mapping_dict2)
         autos['gearbox'].value_counts()
```

```
Out[25]: manual       234851
         automatic     68018
         Name: gearbox, dtype: int64
```

2.'unrepaired_damage'

```
In [26]: autos.unrepaired_damage.value_counts()
```

```
Out[26]: nein    236921
         ja       28544
         Name: unrepaired_damage, dtype: int64
```

In [27]:
```
mapping_dict4={'nein':'no', 'ja':'yes'}
autos['unrepaired_damage']=autos['unrepaired_damage'].map(mapping_dict4)
autos['unrepaired_damage'].value_counts()
```

Out[27]:
```
no     236921
yes     28544
Name: unrepaired_damage, dtype: int64
```

1.1.5 Step1_ 5 Investigate Null-values

In [28]:
```
autos.isnull().sum()
```

Out[28]:
```
name                     0
price                    0
vehicle_type         10868
registration_year        0
gearbox               5290
power_ps                 0
model                11424
kilometer                0
fuel_type            15431
brand                    0
unrepaired_damage    42694
dtype: int64
```

Analysis >- The columns with null-values are all text or boolean values, it is possible for not having complete informations in eBay, and as our focus is to analyze car 'price', we don't need to remove or fill these null values.

In [29]:
```
# Check our changes
autos.describe(include='all')
```

Out[29]:
```
           name          price vehicle_type  registration_year gearbox  \
count    308159  308159.000000       297291      308159.000000  302869
unique   189396            NaN            8                NaN       2
top     BMW_318i           NaN    limousine                NaN  manual
freq        616            NaN        86094                NaN  234851
mean        NaN    6239.105154          NaN        2003.148037     NaN
std         NaN    8244.307901          NaN           6.865561     NaN
min         NaN     100.000000          NaN        1910.000000     NaN
25%         NaN    1450.000000          NaN        1999.000000     NaN
50%         NaN    3500.000000          NaN        2003.000000     NaN
75%         NaN    7999.000000          NaN        2008.000000     NaN
max         NaN  200000.000000          NaN        2016.000000     NaN

             power_ps   model      kilometer fuel_type       brand  \
count   308159.000000  296735  308159.000000    292728      308159
unique            NaN     250            NaN         7          40
top               NaN    golf            NaN    benzin  volkswagen
```

```
freq           NaN   25147            NaN    192147       65698
mean    125.968004     NaN  125418.988250       NaN         NaN
std      60.088679     NaN   39283.109438       NaN         NaN
min      10.000000     NaN    5000.000000       NaN         NaN
25%      80.000000     NaN  100000.000000       NaN         NaN
50%     116.000000     NaN  150000.000000       NaN         NaN
75%     150.000000     NaN  150000.000000       NaN         NaN
max     500.000000     NaN  150000.000000       NaN         NaN

       unrepaired_damage
count             265465
unique                 2
top                   no
freq              236921
mean                 NaN
std                  NaN
min                  NaN
25%                  NaN
50%                  NaN
75%                  NaN
max                  NaN
```

Exploratory Data Analysis

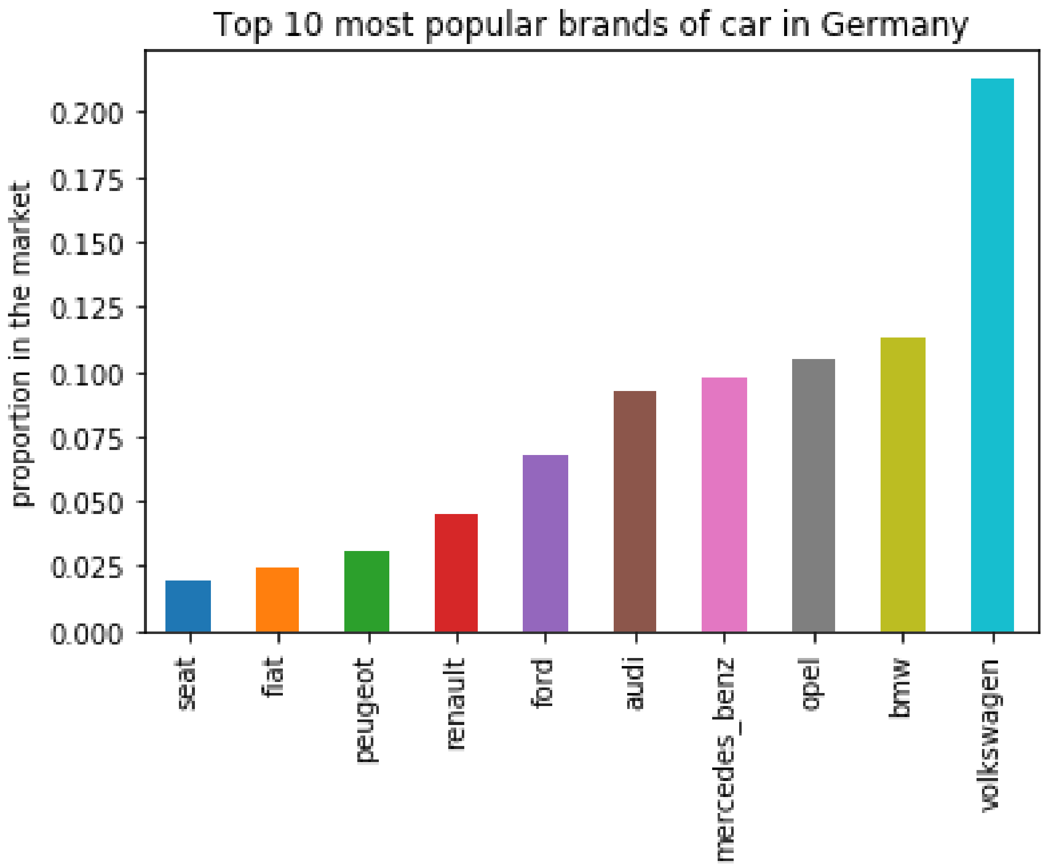
1.1.6 Question 1: What are the common brands of vehicls in Germany and their average price ?

```
In [30]: # List of top 10 most popular brands
         brand_counts=autos['brand'].value_counts(normalize=True)
         brand_counts.head(10)
```

```
Out[30]: volkswagen       0.213195
         bmw              0.113370
         opel             0.104488
         mercedes_benz    0.097239
         audi             0.092348
         ford             0.067423
         renault          0.044789
         peugeot          0.030254
         fiat             0.024698
         seat             0.019178
         Name: brand, dtype: float64
```

```
In [31]: brand_counts.head(10).sort_values().plot(kind='bar', title='Top 10 most popular brands
         plt.ylabel('proportion in the market')
```

```
Out[31]: Text(0, 0.5, 'proportion in the market')
```



Analysis >- Volkswagen is the most polular choice, counting more than 20% of the market >- BMW, Opel, mercedes_benz and audi are the next popular one, but far from volkswagen's popularity

```
In [32]: # Select the brands that are more than 5% of the market to analyze
         common_brands=brand_counts[brand_counts > .05].index
         common_brands

Out[32]: Index(['volkswagen', 'bmw', 'opel', 'mercedes_benz', 'audi', 'ford'], dtype='object')

In [33]: #Analyze the common brands and its average price
         brand_mean_prices = {}

         for brand in common_brands:
             brand_only = autos[autos["brand"] == brand]
             mean_price = brand_only["price"].mean()
             brand_mean_prices[brand] = int(mean_price)

         brand_mean_prices

Out[33]: {'volkswagen': 5688,
          'bmw': 8680,
```

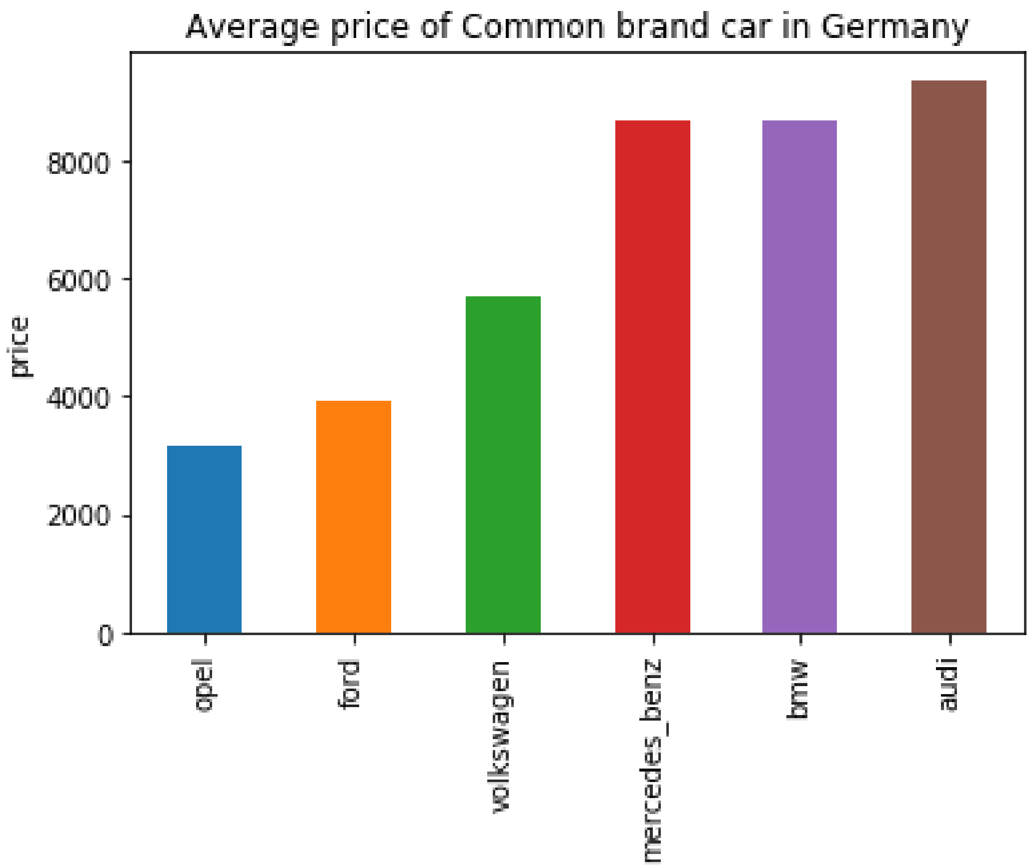
```
 'opel': 3176,
 'mercedes_benz': 8664,
 'audi': 9381,
 'ford': 3942}
```

In [34]:
```
# Convert the dictionary to a pandas series and sort its value
mean_prices=pd.Series(brand_mean_prices).sort_values(ascending=False)
mean_prices
```

Out[34]:
```
audi             9381
bmw              8680
mercedes_benz    8664
volkswagen       5688
ford             3942
opel             3176
dtype: int64
```

In [35]:
```
mean_prices.sort_values().plot(kind='bar', title='Average price of Common brand car in
plt.ylabel('price')
```

Out[35]:
```
Text(0, 0.5, 'price')
```



1.1.7 Answer 1:

- Volkswagen is the most popular brand, followed by Opel,BMW, Mercedes, Audi and Ford.
- Among these popular brands, Audi is the most expensive, average price is 9381 dollars, followed by 8680 for BMW and 8664 for Mercedes. Volkswagen is more affordable ofr most people, average price is 5688. Ford and Opel are least expensive with average price under 4000.

1.1.8 Question 2: Among common brands, are there large differences on kilometer that can affect listing price?

In [36]:
```
#Analyze the common brands and its average odometer_km
brand_mean_km = {}

for brand in common_brands:
    brand_only = autos[autos["brand"] == brand]
    mean_km = brand_only["kilometer"].mean()
    brand_mean_km[brand] = int(mean_km)

brand_mean_km
```

Out[36]:
```
{'volkswagen': 128117,
 'bmw': 132985,
 'opel': 128378,
 'mercedes_benz': 130802,
 'audi': 129142,
 'ford': 123575}
```

In [37]:
```
# Convert the dictionary to a pandas series
mean_km=pd.Series(brand_mean_km)
```

In [38]:
```
# Convert pandas series to a data frame
common_brand_info=pd.DataFrame(mean_prices, columns=['mean_price'])
```

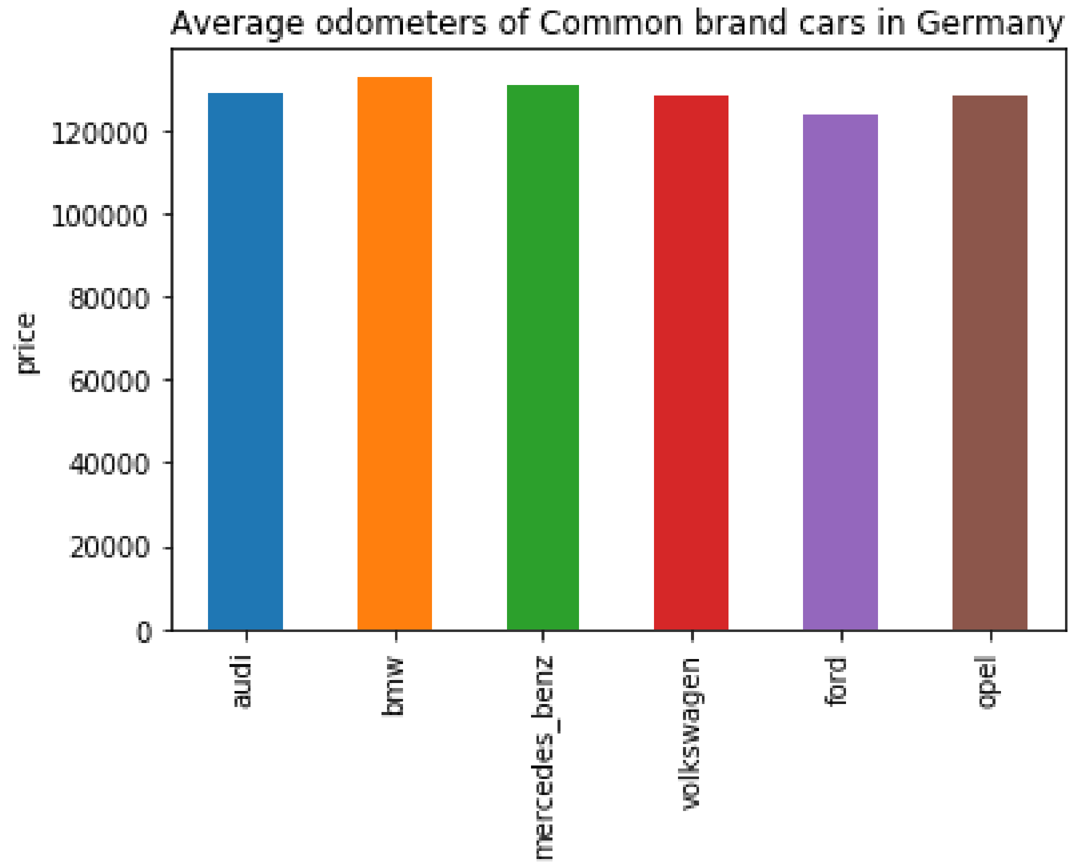
In [39]:
```
# Add the 'mean_km' to the data frame
common_brand_info['mean_km']=mean_km
common_brand_info
```

Out[39]:

	mean_price	mean_km
audi	9381	129142
bmw	8680	132985
mercedes_benz	8664	130802
volkswagen	5688	128117
ford	3942	123575
opel	3176	128378

In [40]:
```
common_brand_info['mean_km'].plot(kind='bar', title='Average odometers of Common brand
plt.ylabel('price')
```

Out[40]: Text(0, 0.5, 'price')



1.1.9 Answer 2:

- Among these common brands of cars on sale, the average of odometers are all ablove 100000km; The range of car mileages does not vary as much as the prices do by brand.

1.1.10 Alternative ways to answering Questions 1&2

In [41]:
```
brand_counts=autos['brand'].value_counts(normalize=True)
brand_counts.head(10)
```

Out[41]:
```
volkswagen       0.213195
bmw              0.113370
opel             0.104488
mercedes_benz    0.097239
audi             0.092348
ford             0.067423
renault          0.044789
```

```
peugeot          0.030254
fiat             0.024698
seat             0.019178
Name: brand, dtype: float64
```

In [42]:
```
common_brands=brand_counts[brand_counts > .05].index
common_brands
```

Out[42]:
```
Index(['volkswagen', 'bmw', 'opel', 'mercedes_benz', 'audi', 'ford'], dtype='object')
```

In [43]:
```
# using query methods to select commom brands
autos_common_brands=autos.query('brand in ["volkswagen", "bmw", "opel", "mercedes_ben
autos_common_brands.head(2)
```

Out[43]:
```
                   name  price vehicle_type  registration_year gearbox  \
1  A5_Sportback_2.7_Tdi  18300        coupe               2011  manual
3    GOLF_4_1_4__3TÜRER   1500  kleinwagen                2001  manual

   power_ps model  kilometer fuel_type       brand unrepaired_damage
1       190   NaN     125000    diesel        audi               yes
3        75  golf     150000    benzin  volkswagen                no
```

In [44]:
```
common_brands_info=autos_common_brands.groupby('brand').mean()[['price','kilometer']]
common_brands_info
```

Out[44]:
```
                     price      kilometer
brand
audi           9381.733959  129142.771804
bmw            8680.722063  132985.029769
ford           3942.425567  123575.588391
mercedes_benz  8664.469681  130802.603037
opel           3176.230069  128378.365788
volkswagen     5688.166885  128117.827027
```

1.1.11 Question 3: What are the factors that affect car price?

1.1.12 Q3_Step1: First we will analyze the columns with numerical value and see how it is correlated with the car price using correlation heatmap and scatter chart.

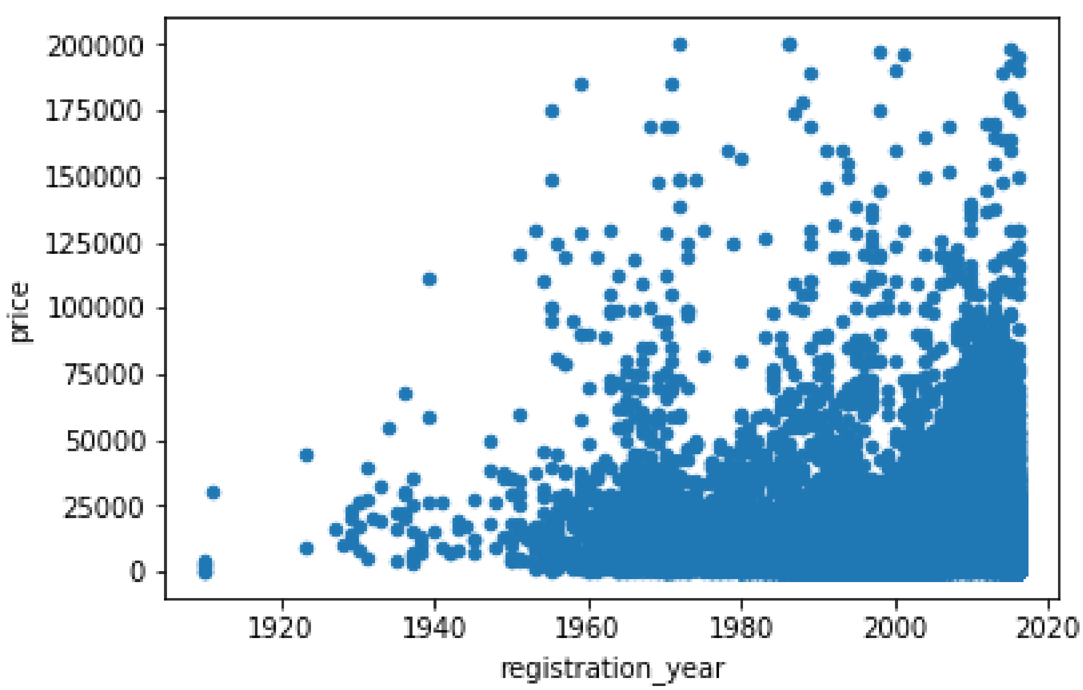
In [45]:
```
# plot correlation heatmap
sns.heatmap(autos.corr(),annot=True,cmap='coolwarm')
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1eaf5080>
```

Analysis >- Prices are positively correlated with power_ps and registration_year. Power_ps has stronger correlation. >- Prices are negatively correlated with kilometer.

In [46]:
```
# plot scatter chart to check the relation between 'registration_year' and 'price'
autos.plot(kind='scatter', x='registration_year', y='price')
```

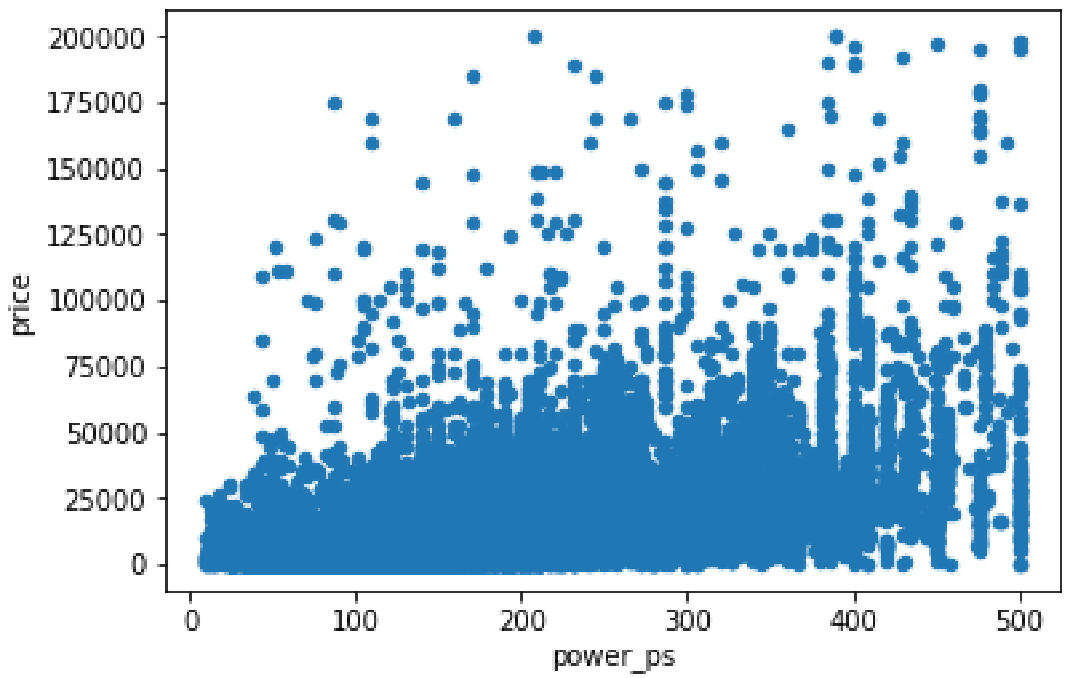
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1f67d6a0>



Analysis >- In general, the newer the cars, the higher the prices, but for a given registration year, there are still huge gap on prices

```
In [47]: # plot scatter chart to check the relation between 'power_ps' and 'price'
         autos.plot(kind='scatter', x='power_ps', y='price')
```

```
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1f6e4cf8>
```



Analysis >- Most cars have pow_ps under 400, and in general the higher the power_ps, the higher the price; but there are cars with extremely high power_ps, but prices range is still very large.

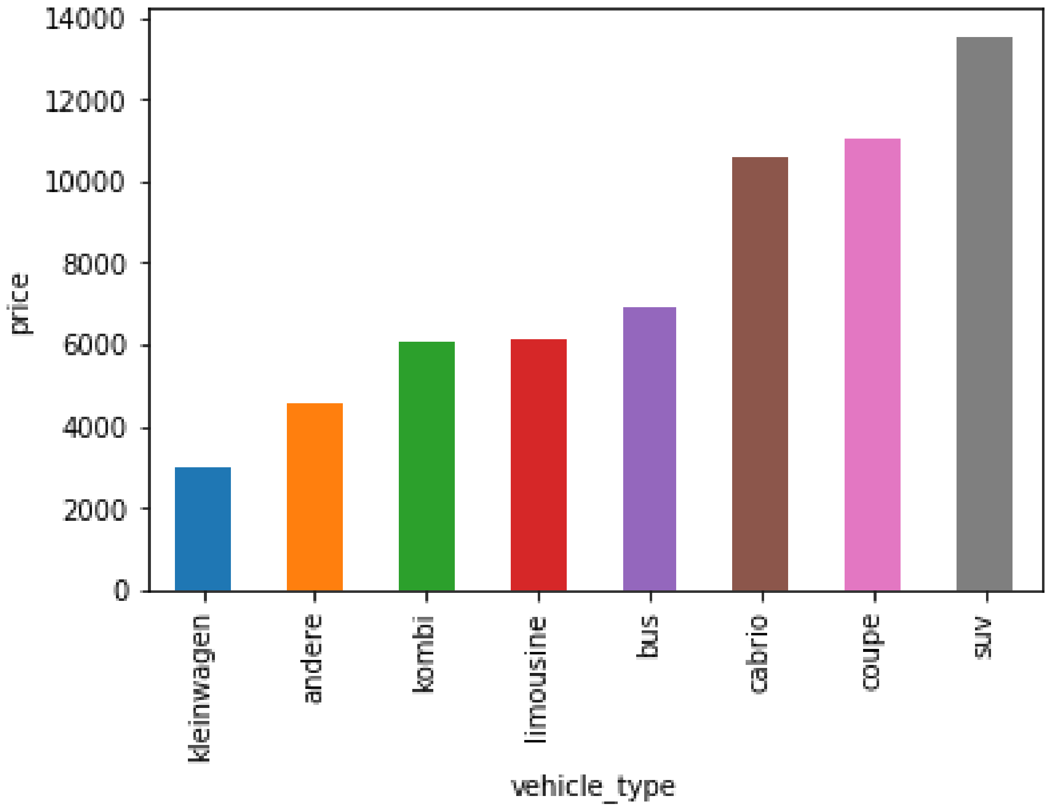
1.1.13 Q3_Step2: We will analyze the columns with catergorical string values using bar chart.

1.'vehicle_type'

```
In [48]: type_price=autos.groupby('vehicle_type').mean()['price']
         type_price.sort_values().plot(kind='bar')

         plt.ylabel('price')
```

```
Out[48]: Text(0, 0.5, 'price')
```

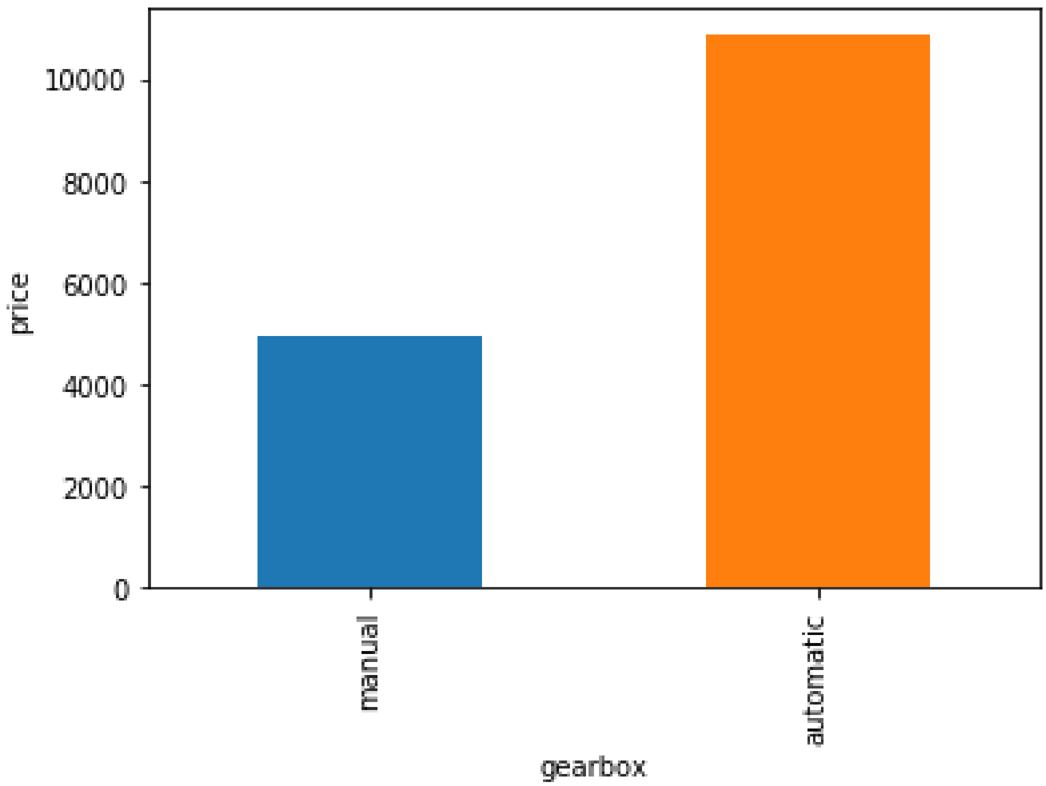


Analyze >- As we can see, suv is the most expensive ones, and kleinwagen the lease expensive.
2.'gearbox'

In [49]:
```
# Analyze whether the damege is repaired can positively influnce the price
gearbox_price=autos.groupby('gearbox')['price'].mean()
gearbox_price.sort_values().plot(kind='bar')

plt.ylabel('price')
```

Out[49]:
```
Text(0, 0.5, 'price')
```

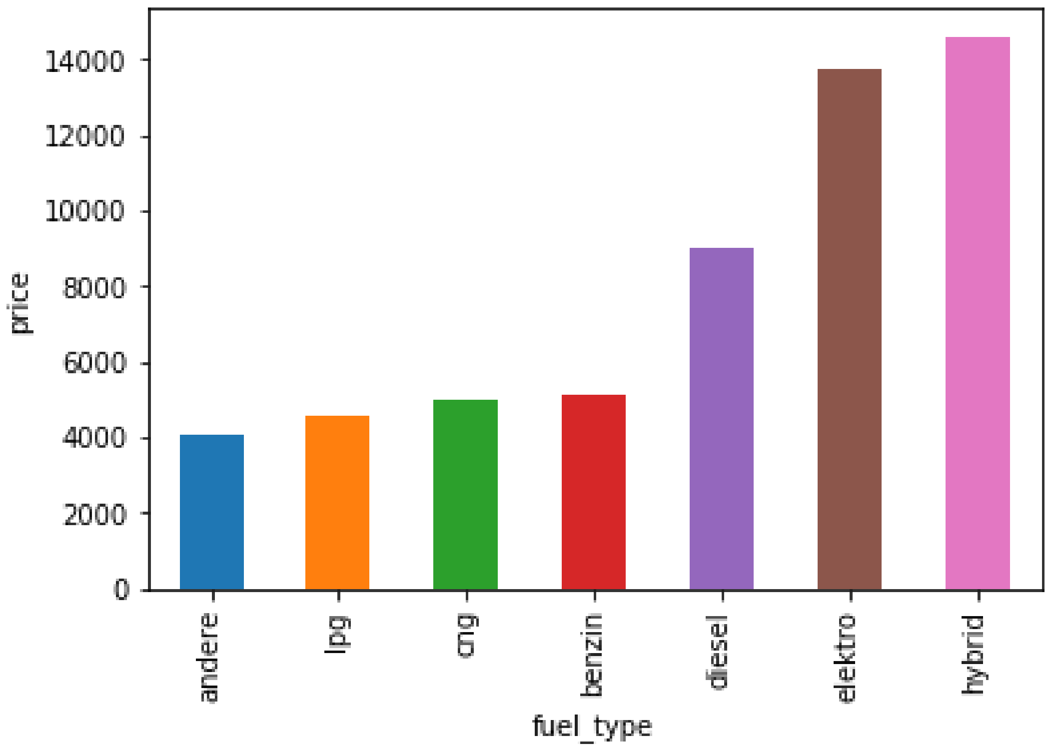


Analysis >- In general, automatic cars are more expensive
3.'fuel_type'

```
In [50]: fuel_price=autos.groupby('fuel_type').mean()['price']
         fuel_price.sort_values().plot(kind='bar')

         plt.ylabel('price')

Out[50]: Text(0, 0.5, 'price')
```

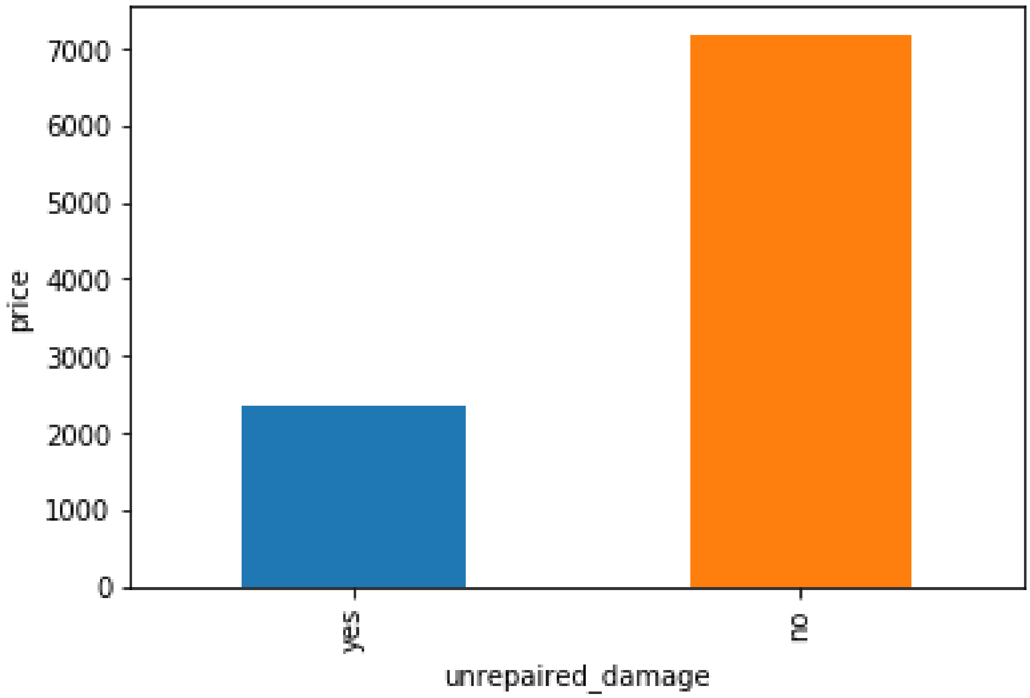


Analysis >- hybrid are most expensive one and tightlt followed by elektro, diesel and benzin. **4.'unrepaired_damage'**

```
In [51]: # Analyze whether the damege is repaired can positively influnce the price
         unrepaired_price=autos.groupby('unrepaired_damage')['price'].mean()
         unrepaired_price.sort_values().plot(kind='bar')

         plt.ylabel('price')

Out[51]: Text(0, 0.5, 'price')
```

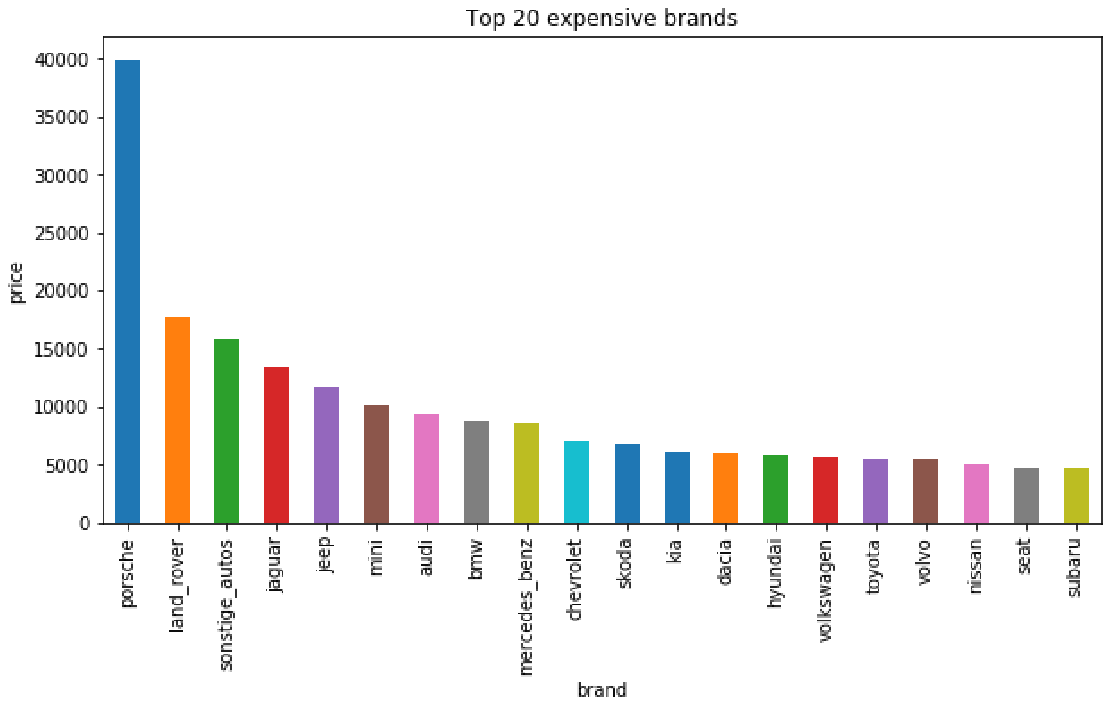


Analysis >- In general, when the demage is repaired, the car prices are higher
5.'brand'

```
In [52]: top_20_expensive=autos.groupby('brand').mean().sort_values('price',ascending=False).he
         top_20_expensive['price'].plot(kind='bar',title='Top 20 expensive brands', figsize=(1(

         plt.ylabel('price')

Out[52]: Text(0, 0.5, 'price')
```



Analysis >- Porsche is the most expensive brand, the average price double the following competitor 'land_rover'. >- The top 6 most expensive brands are all not the most common brands. The most popular brand volkswagen has very affordable average price.

1.1.14 Q3_Step4: Let's analyze the most popular brand 'volkswagen', and see how car model and car name length can affect prices

In [53]:
```
# Selet the rows that are volkswagen for analyze
volkswagen=autos[autos['brand']=='volkswagen']
volkswagen.describe(include='all')
```

Out[53]:

	name	price	vehicle_type	registration_year
count	65698	65698.000000	62552	65698.000000
unique	40985	NaN	8	NaN
top	Volkswagen_Golf_1.4	NaN	limousine	NaN
freq	573	NaN	18529	NaN
mean	NaN	5688.166885	NaN	2002.711148
std	NaN	6409.844585	NaN	7.110533
min	NaN	100.000000	NaN	1910.000000
25%	NaN	1400.000000	NaN	1998.000000
50%	NaN	3333.000000	NaN	2003.000000
75%	NaN	7800.000000	NaN	2008.000000
max	NaN	123456.000000	NaN	2016.000000

```
        gearbox      power_ps   model      kilometer fuel_type       brand  \
```

```
count     64564  65698.000000  63665   65698.000000     62156       65698
unique        2           NaN     22            NaN         7           1
top      manual           NaN   golf            NaN    benzin  volkswagen
freq      55398           NaN  25147            NaN     37641       65698
mean        NaN    106.265746    NaN  128117.827027       NaN         NaN
std         NaN     45.209319    NaN   38422.095700       NaN         NaN
min         NaN     10.000000    NaN    5000.000000       NaN         NaN
25%         NaN     75.000000    NaN  125000.000000       NaN         NaN
50%         NaN    101.000000    NaN  150000.000000       NaN         NaN
75%         NaN    131.000000    NaN  150000.000000       NaN         NaN
max         NaN    500.000000    NaN  150000.000000       NaN         NaN

       unrepaired_damage
count              55382
unique                 2
top                   no
freq               49797
mean                 NaN
std                  NaN
min                  NaN
25%                  NaN
50%                  NaN
75%                  NaN
max                  NaN
```

In [54]:
```
# Add a new columns for the name length
volkswagen['name_length']=volkswagen['name'].apply(len)
volkswagen[['price','name_length']].corr()
```

```
/Users/lutang/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: SettingWithCopyWa
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

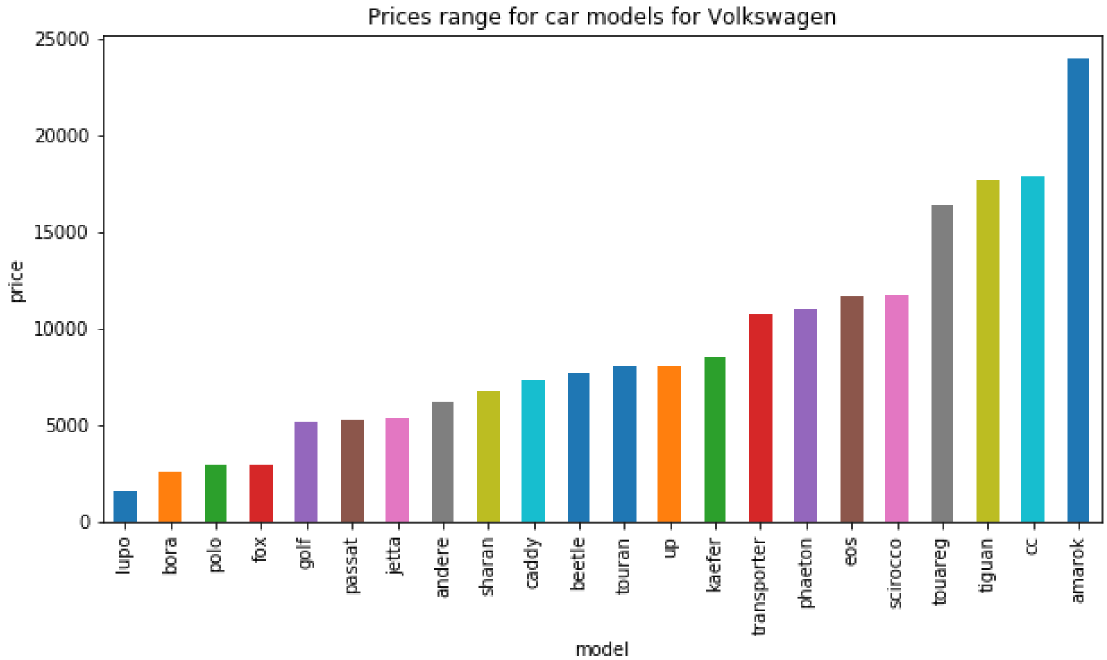
Out[54]:
```
               price  name_length
price        1.00000      0.29155
name_length  0.29155      1.00000
```

Analysis >- name_lenth is positively correlated with price, as the longer the name is, the more features are added, so the price is higher, but the correlation is not strong.

In [55]:
```
model_price=volkswagen.groupby('model')['price'].mean()
model_price.sort_values().plot(kind='bar',figsize=(10,5), title='Prices range for car

plt.ylabel('price')
```

Out[55]: Text(0, 0.5, 'price')



Analysis >- There are huge price gap on different car model. For volkswagen, amorok is the most expensive ones, and lupo the least expensive one. >- I have done the same analysis for other factors for volkswagen, and results are consistant compared with the whole dataset analysis.

1.1.15 Answer 3:

- From the correlation heatmap and scatter chart, we can conclude `price` are positively correlated with `power_ps` and `registration_year` and are negatively correlated with `kilometer` in general, and `power_ps` is has stronger influence.
- The other strong catagorical factors that affect the car price are the brand and whether the damage is repaired or not; Also automic are much mroe expensive than manual
- vehicle_type and fuel_type have strong effects too
- By analyzing data for volkswagen, the most common brand in Germany, we can see the above conclusions are consistent for specific car brand. And for the same brand, different models have high price ranges too.