Phân tích danh sách xe đã qua sử dụng trên eBay Kleinanzeigen

Chúng ta sẽ làm việc với tập dữ liệu về ô tô đã qua sử dụng từ eBay Kleinanzeigen, một mục rao vặt trên trang web eBay của Đức. Mục đích của dự án này là làm sạch dữ liệu và phân tích danh sách xe đã qua sử dụng.

Bộ dữ liệu ban đầu đư ợc thu thập và tải lên Kaggle. Phiên bản của bộ dữ liệu mà chúng tôi đang làm việc là một mẫu gồm 50.000 điểm dữ liệu do Dataquest chuẩn bị, bao gồm cả việc mô phỏng phiên bản dữ liệu ít đư ợc làm sạch hơ n.

Từ điển dữ liệu đư ợc cung cấp với dữ liệu như sau:

- dateCrawled - Khi quảng cáo này đư ợc thu thập lần đầu tiên. Tất cả các giá trị trư ờng đư ợc lấy từ ngày này.
- tên - Tên của chiếc xe.
- ngư ời bán - Cho dù ngư ời bán là tư nhân hay đại
- lý. offerType - Loại giá niêm
- yết - Giá trên rao bán xe. abtest - Liệu
- danh sách có đư ợc đư a vào thử nghiệm A/B hay không.
- Loại xe - Loại xe.
- yearOfRegistration - Năm mà chiếc xe đư ợc đăng ký lần đầu tiên. hộp số - Loại
- truyền động. powerPS - Sức mạnh
- của chiếc xe trong PS.
- model - Tên kiểu xe.
- km - Xe đã đi đư ợc bao nhiêu km. monthOfRegistration -
- Tháng trong đó chiếc xe đư ợc đăng ký lần đầu. FuelType - Loại nhiên liệu xe sử dụng.
- thư ơ ng hiệu - Thư ơ ng hiệu của chiếc xe.
-
- notRepairedDamage - Nếu xe có hư hỏng chư a đư ợc sửa chữa.

- dateCreated - Ngày tạo danh sách eBay.
- nrOfPictures - Số lượng ảnh trong quảng cáo.
- PostalCode - Mã bưu chính cho vị trí của chiếc xe.
- lastSeenOnline - Khi trình thu thập thông tin nhìn thấy quảng cáo này trực tuyến lần cuối.

Mục đích của dự án này là làm sạch dữ liệu và phân tích danh sách xe đã qua sử dụng. Vì vậy, hãy để chúng tôi bắt đầu bằng cách nhập và đọc dữ liệu.

Đọc dữ liệu

Trong [208]:
```
nhập gấu trúc dưới dạng pd
nhập numpy như ny

autos = pd.read_csv("autos.csv", mã hóa = "Latin_1")
```

Trong [209]:
```
ô tô
```

Hết[209]:

	ngày đã thu thập dữ liệu	tên người bán đề nghịType	giá
0	2016-03-26 17:47:46	Peugeot_807_160_NAVTECH_ON_BOARD Angebot tư nhân	$5.000
1	2016-04-04 13:38:56	BMW_740i_4_4_Liter_HAMANN_UMBAU_Mega_Optik Angebot riêng tư	$8.500
2	2016-03-26 18:57:24	Volkswagen_Golf_1.6_United Angebot tư nhân	$8,990
3	2016-03-12 16:58:10	Smart_smart_fortwo_coupe_softouch/F1/Klima/Pan... riêng Angebot	$4,350
4	2016-04-01 14:38:50	Ford_Focus_1_6_Benzin_TÜV_neu_ist_sehr_gepfleg... riêng Angebot	$1,350
...	...	... ...	...
49995	2016-03-27 14:38:19	Audi_Q5_3.0_TDI_qu._S_tr.__Navi__Panorama__Xenon tư nhân Angebot	$24,900
49996	2016-03-28 10:50:25	Opel_Astra_F_Cabrio_Bertone_Edition___TÜV_neu+... Angebot riêng	$1,980
49997	2016-04-02 14:44:48	Fiat_500_C_1.2_Dualogic_Lounge Angebot tư nhân	$13.200
49998	2016-03-08 19:25:42	Audi_A3_2.0_TDI_Sportback_Ambition tư nhân Angebot	$22,900
49999	2016-03-14 00:42:12	Opel_Vectra_1.6_16V Angebot tư nhân	$1,250

50000 hàng × 20 cột

Trong [210]:
```
autos.info()
```

```
<lớp 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 mục, 0 đến 49999
Các cột dữ liệu (tổng cộng 20 cột):
 # Cột Non-Null Count Dtype
--- ------ -------------- -----
 0 ngày Đã thu thập dữ liệu 50000 đối tượng không null
 1 tên 2 50000 đối tượng không null
 người bán 50000 đối tượng không null
 3 offerType 50000 đối tượng không null
 4 giá 50000 đối tượng không null
 5 abtest 50000 đối tượng không null
 6 xeType 44905 đối tượng không null
 7 yearOfRegistration 50000 non-null int64 8 hộp số 47320
 non-null object
 powerPS 10 50000 non-null int64 9
 model 47242 đối tượng non-null
 11 đo dặm 50000 đối tượng không rỗng
 12 thángOfRegistration 50000 non-null int64 13 FuelType
 45518 non-null object
 14 nhãn hiệu 50000 đối tượng không null
 15 notRepairedDamage 40171 đối tượng không null
 16 ngàyTạo 50000 đối tượng không null
 17 nrOfPictures 50000 không null int64 18 Mã bưu chính
 50000 không null int64 19 lastSeen 50000 đối tượng không
 null
dtypes: int64(5), đối tượng(15)
sử dụng bộ nhớ: 7,6+ MB
```

Trong [211]:
```
autos.head()
```

Hết[211]:

	ngày đã thu thập dữ liệu	tên người bán cung cấpType giá abte	
0	2016-03-26 17:47:46	Peugeot_807_160_NAVTECH_ON_BOARD Angebot tư nhân trị giá 5.000 đô la	
1	2016-04-04 13:38:56	BMW_740i_4_4_Liter_HAMANN_UMBAU_Mega_Optik riêng tư Angebot $8.500 contr	
2	2016-03-26 18:57:24	Volkswagen_Golf_1.6_United Angebot tư nhân $8,990	te
3	2016-03-12 16:58:10	Smart_smart_fortwo_coupe_softouch/F1/Klima/Pan... riêng Angebot $4,350 contr	
4	2016-04-01 14:38:50	Ford_Focus_1_6_Benzin_TÜV_neu_ist_sehr_gepfleg... riêng Angebot $1,350	te

Từ những điều trên, chúng ta có thể nhận thấy:

- Chúng tôi có 5 cột với các giá trị null - loại phương tiện, hộp số, kiểu xe, loại nhiên liệu, notRepairedDamage - tất cả những thứ này chứa ít hơn 20% giá trị null.

- Dữ liệu bằng tiếng Đức, vì vậy để giúp ngư ời dân không nói tiếng Đức dễ dàng hơ n, nó có thể là một ý tư ởng hay để thay thế/dịch một số cột sang tiếng Anh.
- Hầu hết dữ liệu ở dạng chuỗi.

tôi litdf P th' fd k \

Đổi tên cột

Trong [212]:
```
autos.columns
```

Hết[212]:
```
Chỉ mục(['dateCrawled', 'name', 'seller', 'offerType', 'price', 'abtest',
       'vehicleType', 'yearOfRegistration', 'hộp số', 'powerPS', 'model',
       'đồng hồ đo đư ờng', 'thángOfRegistration', 'loại nhiên liệu', 'nhãn hiệu',
       'notRepairedDamage', 'dateCreated', 'nrOfPictures', 'postalCode',
       'nhìn thấy lần cuối'],
      dtype='đối tư ợng')
```

Trong [213]:
```
autos.columns = ['date_crawled', 'name', 'seller', 'offer_type', 'price', 'abt
          'vehicle_type', 'registration_year', 'hộp số', 'power_ps', 'model',
          'đồng hồ đo đư ờng', 'tháng_đăng ký', 'loại_nhiên liệu', 'nhãn hiệu',
          'unrepaired_damage', 'ad_created', 'num_photos', 'postal_code',
          'nhìn thấy lần cuối']
ô tô.head()
```

Hết[213]:

	date_crawled	tên ngư ời bán	offer_type	giá	abt
0	2016-03-26 17:47:46	Peugeot_807_160_NAVTECH_ON_BOARD	tư nhân	Angebot $5,000	con
1	2016-04-04 13:38:56	BMW_740i_4_4_Liter_HAMANN_UMBAU_Mega_Optik	riêng tư	Angebot $8,500	con
2	2016-03-26 18:57:24	Volkswagen_Golf_1.6_United	tư nhân	Angebot $8,990	
3	2016-03-12 16:58:10	Smart_smart_fortwo_coupe_softouch/F1/Klima/Pan...	riêng tư	Angebot $4,350	con
4	2016-04-01 14:38:50	Ford_Focus_1_6_Benzin_TÜV_neu_ist_sehr_gepfleg...	riêng tư	Angebot $1,350	

Bây giờ, chúng tôi có các tên cột ở định dạng dễ hiểu hơ n và ở dạng rắn

Thăm dò và vệ sinh cột

Trong [214]:
```
autos.describe()
```

Hết[214]:

	đăng ký_năm	power_ps	đăng ký_tháng	num_ảnh_mã bư u chính	
đếm	50000.000000	50000.000000	50000.000000	50000.0	50000.000000
nghĩa là	2005.073280	116.355920	5.723360	0.0	50813.627300
tiêu chuẩn	105.712813	209.216627	3.711984	0.0	25779.747957
tối thiểu	1000.000000	0,000000	0,000000	0.0	1067.000000
25%	1999.000000	70.000000	3.000000	0.0	30451.000000
50%	2003.000000	105.000000	6.000000	0.0	49577.000000
75%	2008.000000	150.000000	9.000000	0.0	71540.000000
tối đa	9999.000000	17700.000000	12.000000	0.0	99998.000000

Trong [215]:
```
autos.describe(include='all')
```

Hết[215]:

	date_crawled	tên ngư ời bán ư u đãi_loại giá gần nhất	xe_loại đăng ký		
đếm	50000	50000 50000	50000 50000 50000	44905	50000.0
độc nhất	48213	38754 2	2 2357 2	số 8	
đứng đầu	2016-03-08 10:40:35	Ford_Fiesta riêng tư Angebot	$0 bài kiểm tra	xe limousine	
tần số	3	78 49999	49999 1421 25756	12859	
nghĩa là	NaN	NaN NaN	NaN NaN NaN	NaN	2005.0
tiêu chuẩn	NaN	NaN NaN	NaN NaN NaN	NaN	105,7
tối thiểu	NaN	NaN NaN	NaN NaN NaN	NaN	1000.0
25%	NaN	NaN NaN	NaN NaN NaN	NaN	1999.0
50%	NaN	NaN NaN	NaN NaN NaN	NaN	2003.0
75%	NaN	NaN NaN	NaN NaN NaN	NaN	2008.0

Trư ớc tiên, hãy thực hiện công việc treo thấp - chuyển đổi các cột giá và công tơ mét thành số và đổi tên các cột để bao gồm đơ n vị đo lư ờng:

Trong [216]:
```
ô tô["giá"].head()
```

Hết[216]:
```
0 1    5.000 USD
       $8,500
2      $8,990
       $4,350
3 4    $1,350
Tên: giá, dtype: đối tư ợng
```

Trong [217]:
```
ô tô["giá"] = (ô tô["giá"]
                                .str.replace("$","")
                                .str.replace(",","")
                                .astype(int)
                                )
autos.rename({"price": "price_usd"}, axis=1, inplace=True)
ô tô["price_usd"].head()
```

Hết[217]:
```
0    5000
1    8500
2    8990
3    4350
4 1350
Tên: price_usd, dtype: int64
```

Trong [218]:
```
autos["odometer"].head()
```

Hết[218]:
```
0 1    150.000km
       150.000km
2       70.000km
       70.000km
3 4    150.000km
Tên: máy đo đường, dtype: đối tượng
```

Trong [219]:
```
autos["odometer"] = (autos["odometer"]
                            .str.replace("km", "")
                            .str.replace(",", "")
                            .astype(int)
                        )
autos.rename({"odometer": "odometer_km"}, axis=1, inplace=True)
ô tô["odometer_km"].head()
```

Hết[219]:
```
0 1    150000
       150000
2       70000
       70000
3 4    150000
Tên: odometer_km, dtype: int64
```

Trong [220]:
```
autos.info()
```

```
<lớp 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 mục, 0 đến 49999
Các cột dữ liệu (tổng cộng 20 cột):
 # Cột                          Dtype đếm không null
--- ------                      ----------- -----

 0 date_crawled 50000 đối tượng không null
 1 tên 2                       50000 đối tượng không null
 người bán 50000 đối tượng không null
 3 đối tượng không null 3 offer_type 50000
 4 price_usd 50000 non-null int64 5 abtest 50000 non-null
 object
 6 vehicle_type 44905 đối tượng không null
 7 đăng ký_năm 50000 không null int64 8 hộp số 9 power_ps
 10 kiểu máy 11                47320 đối tượng không null
 odometer_km 12                50000 không null int64
 đăng                          47242 đối tượng không null
 ký_tháng 50000                50000 không null int64
 không null int64 13 fuel_type 45518 đối tượng không null

 14 nhãn hiệu 50000 đối tượng không null
 15 unrepaired_damage 40171 đối tượng không null
 16 ad_created 50000 đối tượng không null
 17 num_photos 50000 không null int64 18 post_code 50000
 không null int64 19 last_seen 50000 đối tượng không null

dtypes: int64(7), đối tượng(13)
sử dụng bộ nhớ: 7,6+ MB
```

Bây giờ quả treo thấp đã được chăm sóc, hãy cùng khám phá từng cột để

xác định những gì khác cần phải được thực hiện. Chúng tôi sẽ liệt kê tất cả các vấn đề mà chúng tôi quan sát được, rồi từng bước

bước chúng ta có thể bắt đầu làm sạch các cột.

Trong [221]:
```
autos.describe(bao gồm="tất cả")
```

Hết[221]:

	date_crawled	tên ngư ời bán	offer_type	price_usd	abtest	xe_type	đăng ký	
đếm	50000	50000	50000	50000	5.000000e+04	50000	44905	5000
độc nhất	48213	38754	2	2	NaN	2		
đứng đầu	2016-03-08 10:40:35	Ford_Fiesta	riêng tư	Angebot	kiểm tra NaN	xe limousine		
tần số	3	78	49999	49999	NaN	25756	12859	
nghĩa là	NaN	NaN	NaN	NaN	9.840044e+03	NaN	NaN	200
tiêu chuẩn	NaN	NaN	NaN	NaN	4.811044e+05	NaN	NaN	10
tối thiểu	NaN	NaN	NaN	NaN	0,000000e+00	NaN	NaN	100
25%	NaN	NaN	NaN	NaN	1.100000e+03	NaN	NaN	199
50%	NaN	NaN	NaN	NaN	2.950000e+03	NaN	NaN	200
75%	NaN	NaN	NaN	NaN	7.200000e+03	NaN	NaN	200
tối đa	NaN	NaN	NaN	NaN	1.000000e+08	NaN	NaN	999

Khám phá cột seller và offer_type

Trong [222]:
```
autos["seller"].value_counts()
```

Ra[222]:
```
tư nhân 49999
gewerblich 1
Tên: ngư ời bán, dtype: int64
```

Chúng ta có thể thấy rằng cột ngư ời bán chứa 49999 giá trị giống hệt nhau. Chúng ta có thể thả cột đó vì nó không có giá trị cho phân tích của chúng tôi.

Trong [223]:
```
autos["offer_type"].value_counts()
```

Ra[223]:
```
Angebot 49999
Gesuch 1
Tên: offer_type, dtype: int64
```

Tư ơ ng tự với cột offer_type. Hãy xóa các cột đó bằng phư ơ ng thức df.drop().

Trong [224]:
```
autos = autos.drop(nhãn = ["seller", "offer_type"], axis=1)
```

Chúng ta cũng có thể quan sát thấy rằng cột 'num_photos' trông kỳ lạ (tất cả = 0), vì vậy chúng ta sẽ khám phá nó:

Trong [225]:
```
autos["num_photos"].value_counts()
```

Ra[225]:
```
0 50000
Tên: num_photos, dtype: int64
```

Trong [226]:
```
autos["num_photos"].head()
```

Hết[226]:
```
0 1 0
0
2 0

3 4 0 0
Tên: num_photos, dtype: int64
```

Tất cả các giá trị trong cột num_photos là 0, vì vậy chúng tôi có thể loại bỏ cột đó vì nó không có giá trị đối với chúng tôi.

Trong [227]:
```
autos = autos.drop(nhãn = "num_photos", axis=1)
```

Trong [228]:
```
autos.columns
```

Hết[228]:
```
Chỉ mục(['date_crawled', 'name', 'price_usd', 'abtest', 'vehicle_type',
'registration_year', 'gearbox', 'power_ps', 'model', 'odometer_km', 'registration_month', '
Fuel_type', 'brand', 'unrepaired_damage', 'ad_created', 'postal_code', 'last_seen'],
dtype ='đối tượng')
```

Trong [229]:
```
autos.shape
```

Hết[229]:
```
(50000, 17)
```

Từ phần trên, chúng ta có thể thấy chúng ta đã loại bỏ thành công 3 cột không cần thiết - số lượng cột mới của chúng ta là 17

Khám phá các cột price_usd và odometer_km

Không đắt tiền, và cũng không đi lại nhiều

ĐƯỢC RỒI. Vậy 'giá cả' và 'công tơ mét' có gì thú vị?

Chắc chắn, tôi muốn:

1. Một chiếc ô tô giá rẻ. Tất nhiên rồi.
2. (Nhưng tôi cũng tò mò về giá của chiếc xe đắt nhất)
3. Xe ít chạy nhất. Bằng cách nào đó, tôi có ấn tượng về số dặm ít nhất = tốt hơn tình trạng. Vâng, thành thật mà nói thì tôi không biết nhiều về ô tô lắm haha...
4. (Nhưng tôi cũng tò mò với chiếc xe đi được nhiều km nhất.)

Và hãy xem liệu có bất kỳ mức giá hoặc quãng đường vô nghĩa nào trong bộ dữ liệu của chúng tôi không.

Trong [230]:
```
print(autos["odometer_km"].unique().shape)
print(ô tô["odometer_km"].describe())
autos["odometer_km"].value_counts().sort_index(ascending=False)
```

```
(13,)
đếm           50000.000000
nghĩa là      125732.700000
tiêu           40042.211706
chuẩn           5000.000000
tối           125000.000000
thiểu 25% 50% 150000.000000
75%           150000.000000
tối đa        150000.000000
Tên: odometer_km, dtype: float64
```

Ra[230]:
```
150000 125000    32424
                  5170
100000            2169
90000             1757
80000             1436
70000             1230
60000             1164
50000             1027
40000              819
30000              789
20000              784
10000              264
5000               967
Tên: odometer_km, dtype: int64
```

Khi khám phá odometer_km, chúng ta có thể thấy rằng các giá trị được làm tròn và bằng nhau tăng lên, nghĩa là người dùng phải chọn từ các giá trị được xác định trước. Chúng ta có thể quan sát khoảng - 65% phương tiện có quãng đường đi được cao (150.000 km) và khoảng - 80% có quãng đường đi được từ 100.000 km trở lên.

Trong [231]:
```
print(autos["price_usd"].unique().shape)
print(autos["price_usd"].describe())
print(autos["price_usd"].value_counts().sort_index(ascending=False).head(10))
ô tô["price_usd"].value_counts().sort_index().head(10)
```

```
(2357,)
đếm          5.000000e+04
nghĩa là     9.840044e+03
tiêu         4.811044e+05
chuẩn        0.000000e+00
tối thiểu 25%    1.100000e+03
50%          2.950000e+03
75%          7.200000e+03
tối đa       1.000000e+08
Tên: price_usd, dtype: float64
99999999    1
27322222    1
12345678    3
11111111    2
10000000    1
3890000     1
1300000     1
1234566     1
999999      2
999990      1
Tên: price_usd, dtype: int64
```

Hết[231]:
```
0 1          1421
              156
                3
2 3             1
5               2
                1
                1
9 10            7
11              2
12              3
Tên: price_usd, dtype: int64
```

Có thể thấy có một số xe giá cao bất thường - có 15 xe được niêm yết giá trên hoặc gần một triệu, chúng tôi có thể bỏ những hàng đó và giữ lại mọi thứ có giá 350.000 trở xuống, vì điều đó thực tế hơn nhiều.

Khi nhìn vào thứ tự giá tăng dần, chúng ta có thể thấy rằng có những giá trị tăng dần từ 1 USD. Vì eBay là một trang web đấu giá, nên có thể những người bán này đã bắt đầu cuộc đấu giá với giá trị thấp. Vì giả định đó là khả thi.

Trong [232]:
```
autos = autos[autos["price_usd"].between(1, 350100)]
ô tô["price_usd"].describe()
```

Hết[232]: tính
```
48565.000000
nghĩa là 5888.935591
tiêu chuẩn 9059.854754
tối thiểu 1.000000
25% 1200.000000
50% 3000.000000
75% 7490.000000
tối đa 350000.000000
Tên: price_usd, dtype: float64
```

Chúng tôi có thể thấy rằng tất cả các ngoại lệ đã được loại bỏ thành công và giá của chúng tôi hiện nằm trong khoảng từ 1 đến 350.000 USD

Khám phá và làm sạch dữ liệu đăng ký

Thời gian có quan trọng không?

Ngoài giá cả, thời gian có đóng một vai trò thú vị nào ở đây không?

Hãy khám phá nó và xem nếu chúng ta có thể tìm thấy một cái gì đó thú vị.

Trong [233]:
```
autos[['date_crawled', 'ad_created', 'last_seen']].head()
```

Hết[233]:

	date_crawled	ad_created	nhìn thấy lần cuối
0	2016-03-26 17:47:46	2016-03-26 00:00:00	2016-04-06 06:45:54
1	2016-04-04 13:38:56	2016-04-04 00:00:00	2016-04-06 14:45:08
2	2016-03-26 18:57:24	2016-03-26 00:00:00	2016-04-06 20:15:37
3	2016-03-12 16:58:10	2016-03-12 00:00:00	2016-03-15 03:16:28
4	2016-04-01 14:38:50	2016-04-01 00:00:00	2016-04-01 14:38:50

Để chọn 10 ký tự đầu tiên trong mỗi cột, chúng ta có thể sử dụng Series.str[:10]:

Trong [234]:
```
print(autos['date_crawled']
      .str[:10]
      .value_counts(normalize=True, dropna=False)
      .sort_index()
      )
```

```
2016-03-05      0,025327
2016-03-06      0,014043
2016-03-07      0,036014
2016-03-08      0,033296
2016-03-09      0,033090
2016-03-10      0,032184
2016-03-11      0,032575
2016-03-12      0,036920
2016-03-13      0,015670
2016-03-14      0,036549
2016-03-15      0,034284
2016-03-16      0,029610
2016-03-17      0,031628
2016-03-18      0,012911
2016-03-19      0,034778
2016-03-20      0,037887
2016-03-21      0,037373
2016-03-22      0,032987
2016-03-23      0,032225
2016-03-24      0,029342
2016-03-25      0,031607
2016-03-26      0,032204
2016-03-27      0,031092
28-03-2016      0,034860
29-03-2016      0,034099
2016-03-30      0,033687
2016-03-31      0,031834
2016-04-01      0,033687
2016-04-02      0,035478
2016-04-03      0,038608
2016-04-04      0,036487
2016-04-05      0,013096
2016-04-06      0,003171
2016-04-07      0,001400
Tên: date_crawled, dtype: float64
```

Tôi không thấy nhiều điều thú vị ở đây. Nó giống như có một cuộc thu thập thông tin hàng ngày như ng chỉ có thể, tỷ lệ phần trăm khá không nhất quán.

Trong [235]:
```
print(autos['ad_created']
      .str[:10]
      .value_counts(normalize=True, dropna=False)
      .sort_index()
      )
```

```
2015-06-11       0,000021
2015-08-10       0,000021
2015-09-09       0,000021
2015-11-10       0,000021
2015-12-05       0,000021
                    ...
2016-04-03       0,038855
2016-04-04       0,036858
2016-04-05       0,011819
2016-04-06       0,003253
2016-04-07 0,001256
Tên: ad_created, Độ dài: 76, dtype: float64
```

Ooookay, giống như có một khoảng cách phần trăm rất lớn giữa tháng 2 và tháng 3 năm 2016 trong một thời gian chỉ 2 tuần. Hấp dẫn!

Trong [236]:
```
print(autos['last_seen']
      .str[:10]
      .value_counts(normalize=True, dropna=False)
      .sort_index()
      )
```

```
2016-03-05       0,001071
2016-03-06       0,004324
2016-03-07       0,005395
2016-03-08       0,007413
2016-03-09       0,009595
2016-03-10       0,010666
2016-03-11       0,012375
2016-03-12       0,023783
2016-03-13       0,008895
2016-03-14       0,012602
2016-03-15       0,015876
2016-03-16       0,016452
2016-03-17       0,028086
2016-03-18       0,007351
2016-03-19       0,015834
2016-03-20       0,020653
2016-03-21       0,020632
22-03-2016       0,021373
23-03-2016       0,018532
2016 03 24       0 019767
```

Tỷ lệ phần trăm tăng đột biến trong ba ngày qua, như ng tôi không chắc tại sao.

Những chiếc xe này có tính năng du hành thời gian không?

Trong [237]:
```
autos["registration_year"].describe()
```

Ra[237]: đếm 48565.000000
nghĩa là 2004.755421
tiêu chuẩn 88.643887
tối 1000.000000
thiểu 1999.000000
25% 2004.000000
50% 75% 2008.000000
tối đa 9999.000000
Tên: register_year, dtype: float64

Năm đăng ký tối thiểu là 1000, điều này là không thể. Năm đăng ký tối đa là 9999. Tôi, chắc chắn rồi, tôi sẽ mua ô tô với bất kỳ tính năng du hành thời gian nào LOL

Trong [238]:
```
autos["registration_year"].value_counts().sort_index(ascending=False).head(20)
```

Ra[238]: 9999 9000 3
1
8888 1
6200 1
5911 1
5000 4
4800 1
4500 1
4100 1
2800 1
2019 2
2018 470
2017 1392
2016 1220
2015 392
2014 663
2013 803
2012 1310
2011 1623
2010 1589
Tên: register_year, dtype: int64

Trong [239]:
```
autos.describe(include='all')
```

Hết[239]:

	date_crawled	tên	giá_USD	tiết kiệm nhất_loại	xe_đăng ký_năm	hộp số	
đếm	48565	48565	48565.000000	48565	43979	48565.000000	46222
độc nhất	46882	37470	NaN	2	số 8	NaN	2
đứng đầu	2016-03-12 16:06:22	Ford Fiesta	NaN	Bài kiểm tra	xe limousine	phân NaN	
tần số	3	76	NaN	25019	12598	NaN	36102
nghĩa là	NaN	NaN	5888.935591	NaN	NaN	2004.755421	NaN
tiêu chuẩn	NaN	NaN	9059.854754	NaN	NaN	88.643887	NaN
tối thiểu	NaN	NaN	1.000000	NaN	NaN	1000.000000	NaN
25%	NaN	NaN	1200.000000	NaN	NaN	1999.000000	NaN
50%	NaN	NaN	3000.000000	NaN	NaN	2004.000000	NaN
75%	NaN	NaN	7490.000000	NaN	NaN	2008.000000	NaN
tối đa	NaN	NaN	350000.000000	NaN	NaN	9999.000000	NaN

Trong [240]:
```
autos["last_seen"].max()
```

Ra[240]: '2016-04-07 14:58:50'

Trong [241]:
```
autos["date_crawled"].min()
```

Hết[241]: '2016-03-05 14:06:30'

Hãy chọn một điểm cắt tốt của một năm thực tế và hợp lệ để đăng ký xe ô tô.

Vì giá trị tối đa của last_seen trong năm 2016 nên các năm đăng ký từ 2017 trở lên không hợp lệ. TRONG
Ngoài ra, chúng tôi cần kết hợp năm và tháng đăng ký để không vư ợt quá tháng 3 năm 2016, vì
đó là ngày quảng cáo đã đư ợc thu thập thông tin. Bất kỳ hàng nào có tháng/năm đăng ký mới hơ n
Tháng 2 năm 2016 nên đư ợc gỡ bỏ.

Trong [242]:
```
autos["registration_year"].value_counts().sort_index().head(20)
```

Ra[242]:
```
1000 1001      1
               1
1111           1
1800           2
1910           5
1927           1
1929           1
1931           1
1934           2
1937           4
1938           1
1939           1
1941           2
1943           1
1948           1
1950           3
1951           2
1952           1
1953           1
1954           2
Tên: register_year, dtype: int64
```

Đối với giới hạn dư ới của khoảng thời gian đăng ký, chúng tôi có thể thấy rằng có một vài hàng hiển thị năm đăng ký từ đầu thế kỷ 20. Có lẽ đó là thời xư a, vì vậy trư ớc khi chúng tôi xóa hoàn toàn các hàng này, bạn nên khám phá thêm. Thế còn... năm 1886 cho giá trị tối thiểu? Đã gần đến năm ra đời của xe hơ i Mercedes-Benz theo Wikipedia (https://vi.wikipedia.org/wiki/Xe).

Trong [243]:
```
autos[autos["registration_year"] < 1911]
```

Hết[243]:

	date_crawled	tên	giá_USD	abtest	xe_loại	lại
3679	2016-04-04 00:36:17	suche_Auto	1	Bài kiểm tra		NaN
10556	2016-04-01 06:02:10	UNFAL_Tự động	450	kiểm soát		NaN
22316	2016-03-29 16:56:41	VW_Kaefer.__Zwei_zum_Preis_von_einem.	kiểm	soát	1500	NaN
22659	2016-03-14 08:51:18	Opel_Corsa_B	500	Bài kiểm tra		NaN
24511	2016-03-17 19:45:11	Trabant__wartburg__Ostalgie	kiểm	soát	490	NaN
28693	2016-03-22 17:48:41	Renault_Twingo	kiểm	soát	599	kleinwagen
30781	25-03-2016 13:47:46	Opel_Calibra_V6_DTM_Bausatz_1:24	30	Bài kiểm tra		NaN
32585	2016-04-02 16:56:39	UNFAL_Tự động	450	kiểm soát		NaN
45157	2016-03-11 22:37:01	xe máy	15	kiểm soát		NaN
49283	2016-03-15 18:38:53	Citroen_HY	điều	khiển	7750	NaN

Chúng ta có thể thấy rằng các kết quả là hỗn hợp - có một số ngư ời già, như ng cũng có một số xe hơ i tồn tại rõ ràng trong khoảng thời gian đó, ví dụ như Opel Corsa (những năm 1950) hoặc Renault Twingo (những năm 1910). ĐẾN đảm bảo chất lư ợng dữ liệu tốt hơ n, chúng tôi có thể xóa tất cả các hàng có năm đăng ký trư ớc năm 1911.

Trong [244]:
```
autos[(autos["registration_year"] == 2016) & (autos["registration_month"] > 2)
```

Hết[244]:

	date_crawled	tên	giá_usd	abtest	xe
135	2016-03-12 11:00:10	Opel_Meriva_B_Panoramadach__Sitz__und_Lenkradh...	điều khiển	8500	
256	2016-04-03 20:50:38	Passat_1.9TDI_4Motion_Highline	4250	Bài kiểm tra	
295	2016-03-28 03:36:22	riêng tư _anbiter	kiểm soát	1000	
307	2016-03-15 22:50:48	Giessen_ford	2800	Bài kiểm tra	
437	25-03-2016 16:39:01	Mazda__klima_leder__Alufelgen	điều khiển	550	
...	...	...	...	...	...
49547	2016-03-30 16:49:46	Smart_Passion_mit_Panorama_Dach	3600	Bài kiểm tra	
49852	2016-04-01 04:02:25	TOP__Golf_3_1.8l	điều khiển	1450	
49876	2016-03-22 17:57:24	Audi_a5_3.0_tdi_s_line	kiểm soát	14700	
49919	2016-03-10 09:49:43	Fiat_Punto	180	Bài kiểm tra	
49938	2016-03-28 18:45:06	Mercedes_Benz_A_160_Avantgarde	điều khiển	2300	

795 hàng × 17 cột

Hãy xem tỷ lệ của những ngoại lệ này là bao nhiêu:

Trong [245]:
```
((autos["registration_year"] == 2016) & (autos["registration_month"] > 2)).sum
```

Hết[245]: 0,016369813651806855

Trong [246]:
```
(autos["registration_year"] < 1911).sum() / autos.shape[0]
```

Hết[246]: 0,00020590960568310512

Trong [247]:
```
autos = autos[autos["registration_year"].between(1911,2016)]
```

Trong [248]:
```
autos = autos[~((autos["registration_year"] == 2016) & (autos["registration_mo
```

Trong [249]:
```
autos["registration_year"].value_counts(normalize = True).head(15)
```

Hết[249]:
```
2000    0,068787
2005    0,063992
1999    0,063142
2004    0,058913
2003    0,058826
2006    0,058194
2001    0,057453
2002    0,054184
1998    0,051503
2007    0,049628
2008    0,048277
2009    0,045444
1997    0,042523
2011    0,035374
2010    0,034633
Tên: register_year, dtype: float64
```

Trong [250]:
```
autos.shape
```

Hết[250]: (45881, 17)

Chúng tôi đã giảm tập dữ liệu của mình xuống còn 45881 hàng dữ liệu. Phân phối năm đăng ký có vẻ tốt, với phần lớn dữ liệu rơ i vào phạm vi năm 1997+.

Khám phá giá theo thư ơ ng hiệu

Đầu tiên, chúng ta sẽ xem xét tất cả các thư ơ ng hiệu trong tập dữ liệu và chọn ra những thư ơ ng hiệu hàng đầu theo phần trăm.

Trong [251]:
```
autos["brand"].unique().shape
```

Hết[251]: (40,)

Chúng ta có thể thấy có 40 thư ơ ng hiệu duy nhất trong bộ dữ liệu. Hãy xem đó là những cái nào:

Trong [252]:
```
autos["brand"].value_counts(normalize=True)
```

Ra[252]:
```
volkswagen bmw                0,210719
                              0,110743
opel                          0,106951
mercedes_benz audi            0,096946
                              0,086942
ford                          0,069767
renault                       0,047013
peugeot                       0,029685
fiat                          0,025544
ghế                           0,018134
skoda                         0,016521
nissan                        0,015213
mazda                         0,015148
thông                         0,014080
minh                          0,013971
citroen                       0,012750
toyota                        0,010026
hyundai                       0,009808
                              0,009220
sonstige_autos volvo ii       0 008762
```

Đúng như dự đoán, phần lớn các thư ơ ng hiệu đư ợc cung cấp là của Châu Âu (hơ n 75%), có vẻ như Các thư ơ ng hiệu Đức thống trị các thư ơ ng hiệu hàng đầu.

Ngoài ra, đây là lần đầu tiên tôi nghe nói về những thư ơ ng hiệu xe hơ i này: 'skoda', 'sonstige_autos', 'dacia', 'saab', 'trabant', 'lancia' và 'lada'. Vì vậy, tôi quyết định tìm kiếm chúng trên google và... tôi hoàn toàn bỏ lỡ. Những chiếc xe này là siêu mát mẻ! Hãy xem chiếc Skoda Superb màu xanh nham thạch xinh xắn này!!

Chà, chúng tôi sẽ chọn những thư ơ ng hiệu hàng đầu (chiếm hơ n 1%) để phân tích giá của chúng tôi:\

- volkswagen
- xe BMW
- ô tô

- mercedes_benz
- audi
- ford
- renault
- peugeot
- fiat
- ghế
- skoda
- nissan
- mada
- thông minh
- xi măng
- toyota
- huyndai

Tạo một từ điển trống để chứa dữ liệu giá:

Trong [253]:
```
brand_mean_prices = {}
```

Chúng tôi sẽ chỉ định số lượng giá trị được chuẩn hóa của mình cho một biến mới, sau đó sử dụng thuộc tính .index để truy cập các thương hiệu có thị phần hàng đầu:

Trong [254]:
```
brands_counts = autos["brand"].value_counts(normalize=True)
thương hiệu = thương hiệu_counts[brands_counts > .01].index #brands_counts[:15].sum() thương hiệu
```

Ra[254]:
```
Index(['volkswagen', 'bmw', 'opel', 'mercedes_benz', 'audi', 'ford', 'renaul
t',
       'peugeot', 'fiat', 'ghế', 'skoda', 'nissan', 'mazda', 'smart', 'citroen', 'toyota',
       'hyundai'], dtype='object')
```

Bây giờ chúng ta sẽ lặp lại từng thương hiệu và tính giá trung bình. Sau đó, chúng tôi sẽ chỉ định thương hiệu làm khóa cho từ điển và giá trung bình được tính cho từng thương hiệu dưới dạng giá trị cho khóa (dưới dạng số nguyên để dễ đọc hơn)

Trong [255]:
```
đối với b trong nhãn
    hiệu: select_rows = autos[autos["brand"] == b]
    mean_price = select_rows["price_usd"].mean()
    brand_mean_prices[b] = int(mean_price)
```

Trong [256]:
```
brand_mean_prices
```

Ra[256]:
```
{'volkswagen': 5453, 'bmw':
  8375, 'opel':
  2997,
  'mercedes_benz': 8682,
  'audi': 9357,
  'ford': 3797,
  'renault': 2493,
  'peugeot ': 3111,
  'fiat': 2834,
  'chỗ ngồi':
  4441, 'skoda':
  6375, 'nissan':
  4789, 'mazda':
  4164, 'thông
  minh': 3603,
  'citroen': 3824,
  'toyota': 5200, 'hyundai': 5437}
```

Chúng ta có thể thấy rằng thương hiệu số 1 Volkswagen có mức giá trung bình rất hấp dẫn - rẻ hơn nhiều so với BMW, Mercedes hay Audi, trong khi đắt hơn so với Opel, Renault hay Peugeot ở mức trung bình. Giá hấp dẫn và nguồn gốc từ Đức rất có thể khiến nó trở nên phổ biến.
Mặt khác, BMW, Mercedes và Audi đắt nhất nhưng vẫn nằm trong top 5.

Opel, Ford, Peugeot và Renault rẻ hơn tất cả các thương hiệu kể trên, vì vậy thực tế là họ sẽ chiếm một phần lớn thị phần.
Các thương hiệu châu Á như Renault, Mazda, Toyota, Huyndai với mức giá tầm trung đứng cuối bảng.

Tính quãng đường trung bình

Sử dụng nguyên tắc tương tự như trên, chúng tôi sẽ tính số dặm trung bình cho mỗi nhãn hiệu:

Trong [257]:
```
brand_mean_mileage = {}
```

Trong [258]:
```
cho b trong nhãn hiệu:
    select_rows = autos[autos["brand"] == b]
    mean_mileage = select_rows["odometer_km"].mean()
    brand_mean_mileage[b] = int(mean_mileage)

brand_mean_mileage
```

Ra[258]:
```
{'volkswagen': 128526,
 'bmw': 132498,
 'opel': 129242,
 'mercedes_benz': 130683,
 'audi': 129251,
 'ford': 124039,
 'renault': 127950,
 'peugeot': 127063,
 'sắc lệnh': 116970,
 'ghế': 120907,
 'skoda': 110916,
 'nissan': 118524,
 'mazda': 124079,
 'thông minh': 98769,
 'xe máy': 119329,
 'toyota': 115777,
 'hyundai': 105847}
```

Tạo một khung dữ liệu mới để so sánh giá và số dặm

Đầu tiên chúng ta sẽ sử dụng pandas series constructor để chuyển đổi cả brand_mean_prices và từ điển brand_mean_mileage cho các đối tư ợng sê-ri:

Trong [259]:
```
mean_prices_series = pd.Series(brand_mean_prices).sort_values(ascending=False)
mean_mileage_series = pd.Series(brand_mean_mileage).sort_values(ascending=Fals
```

Trong [260]:
```
mean_prices_series #dict thành danh sách
```

Ra[260]:
```
audi                9357
mercedes_benz bmw   8682
                    8375
skoda               6375
volkswagen          5453
hyundai             5437
toyota              5200
nissan              4789
ghế                 4441
mazda               4164
citroen             3824
ford                3797
thông minh          3603
peugeot             3111
opel                2997
fiat                2834
renault             2493
dtype: int64
```

Trong [261]:
```
mean_price_mileage_df = pd.DataFrame(mean_prices_series, cột = ["mean_pric
mean_price_mileage_df
```

Hết[261]:

	mean_prices_series
audi	9357
mercedes_benz	8682
xe BMW	8375
skoda	6375
volkswagen	5453
huyndai	5437
toyota	5200
nissan	4789
ghế	4441
mada	4164
xi măng	3824
ford	3797
thông minh	3603
peugeot	3111
ô tô	2997
sắc lệnh	2834
renault	2493

Trong [262]:
```
#mean_mileage_df = pd.DataFrame(mean_mileage_series, cột = ["mean_mileage_
#mean_mileage_df
mean_price_mileage_df['mean_mileage_series'] = mean_mileage_series
mean_price_mileage_df
```

Hết[262]:

	mean_prices_series	mean_mileage_series
audi	9357	129251
mercedes_benz	8682	130683
xe BMW	8375	132498
skoda	6375	110916
volkswagen	5453	128526
huyndai	5437	105847
toyota	5200	115777
nissan	4789	118524
ghế	4441	120907
mada	4164	124079
xi măng	3824	119329
ford	3797	124039

Trong [263]:
```
brand_info = mean_price_mileage_df
brand_info
```

Hết[263]:

	mean_prices_series	mean_mileage_series
audi	9357	129251
mercedes_benz	8682	130683
xe BMW	8375	132498
skoda	6375	110916
volkswagen	5453	128526
huyndai	5437	105847
toyota	5200	115777
nissan	4789	118524
ghế	4441	120907
mada	4164	124079
xi măng	3824	119329
ford	3797	124039
thông minh	3603	98769
peugeot	3111	127063
ô tô	2997	129242
sắc lệnh	2834	116970
renault	2493	127950

Chúng tôi đã hợp nhất cả hai chuỗi thành một khung dữ liệu có tên brand_info với các giá trị được sắp xếp theo thứ tự Thứ tự giảm dần. Bây giờ chúng ta có thể dễ dàng so sánh giá cả và số dặm.

Chúng tôi không thể quan sát thấy khoảng cách lớn về quãng đường đi được, mà là xu hướng mà các thương hiệu đắt tiền hơn có xu hướng để có số dặm cao hơn một chút so với các thương hiệu rẻ tiền hơn. Ngoại lệ là Skoda, có số dặm khá thấp cho giá trung bình.

Vì Mercedes, BMW và Audi chủ yếu sản xuất xe limousine nên có thể đó là lý do tại sao những thương hiệu có số dặm trung bình cao hơn - xe limousine chủ yếu được sử dụng cho các chuyến đi dài, trong khi các phương tiện rẻ hơn sẽ chủ yếu được sử dụng trong giới hạn thành phố, để đi lại.

Dịch tiếng Đức sang tiếng Anh

Vì nhiều người chỉ nói được một thứ tiếng và đôi khi tò mò với dữ liệu thô, chúng ta hãy dịch các từ tiếng Đức trong dữ liệu này sang tiếng Anh để đảm bảo an toàn.

Trong [264]:
```
autos.head()
```

Hết[264]:

	date_crawled	tên	price_usd	abtest	vehicle_typ
0	2016-03-26 17:47:46	Peugeot_807_160_NAVTECH_ON_BOARD	kiểm soát 5000		bu
1	2016-04-04 13:38:56	BMW_740i_4_4_Liter_HAMANN_UMBAU_Mega_Optik	điều khiển 8500		xe limousine
2	2016-03-26 18:57:24	Volkswagen_Golf_1.6_United	8990	Bài kiểm tra	xe limousine
3	2016-03-12 16:58:10	Smart_smart_fortwo_coupe_softouch/F1/Klima/Pan...	điều khiển 4350		lư ơ ng klein
4	2016-04-01 14:38:50	Ford_Focus_1_6_Benzin_TÜV_neu_ist_sehr_gepfleg...	1350	Bài kiểm tra	kom

Trong [265]:
```
autos["vehicle_type"].unique()
```

Ra[265]: mảng(['bus', 'limousine', 'kleinwagen', 'kombi', nan, 'coupe', 'suv',
'cabrio', 'andere'], dtype=đối tư ợng)

Trong [266]:
```
dịch = ({'bus': 'buss',
                    'xe limousine': 'xe limousine',
                    'kleinwagen': 'small_car',
                    'kombi': 'van',
                    'coupe': 'coupe',
                    'suv': 'suv',
                    'cabrio': 'mui trần',
                    'andere':'khác'})
```

Trong [267]:
```
autos["vehicle_type"] = autos["vehicle_type"].map(bản dịch)
autos["vehicle_type"].value_counts()
```

Ra[267]:
```
limousine             12591
small_car             10573
xe tải                 8925
hôn                    4031
có thể chuyển đổi      3014
xe hai bánh            2460
suv                    1962
khác                    390
Tên: xe_type, dtype: int64
```

Trong [268]:
```
ô tô["fuel_type"].unique()
```

Ra[268]: mảng(['lpg', 'benzin', 'diesel', nan, 'cng', 'hybrid', 'elektro',
'andere'], dtype=đối tư ợng)

Trong [269]:
```
dịch = ({'lpg': 'lpg',
                     'benzin': 'xăng',
                     'diesel': 'diesel',
                     'cng': 'cng',
                     'lai': 'lai',
                     'elektro': 'điện',
                     'andere': 'khác'})
```

Trong [270]:
```
autos["fuel_type"] = autos["fuel_type"].map(bản dịch)
ô tô[" Fuel_type"].value_counts()
```

Ra[270]:
```
xăng diesel        28172
lpg                13932
                     643
cng                   70
lai                   36
điện khác             17
                      14
Tên: Fuel_type, dtype: int64
```

Trong [271]:
```
ô tô["hộp số"].unique()
```

Ra[271]: mảng(['manuell', 'automatik', nan], dtype=object)

Trong [272]:
```
autos['gearbox'] = autos['gearbox'].str.replace('manuell', 'manual').str.repla
ô tô["hộp số"].value_counts()
```

Ra[272]:
```
tự động thủ        34081
công                9760
Tên: hộp số, dtype: int64
```

Trong [273]:
```
autos["unrepaired_damage"].unique()
```

Ra[273]: mảng(['nein', nan, 'ja'], dtype=object)

Trong [274]:
```
autos['unrepaired_damage'] = autos['unrepaired_damage'].str.replace('ja', 'yes
autos["unrepaired_damage"].value_counts()
```

Hết[274]:
```
số          33446
đúng         4443
Tên: unrepaired_damage, dtype: int64
```

Bây giờ tôi biết một số từ trong tiếng Đức!

#TeamCommon hay #TeamUnique?

Tôi cũng có thể thấy sự kết hợp giữa tên thư ơ ng hiệu và kiểu máy đư ợc phân tách bằng dấu gạch dư ới trong cột tên.

Hãy xem các combo phổ biến cho những chiếc xe này là gì.

Trong [275]:
```
brand_model_combo = autos.groupby(["brand","model"]).count() #df.groupby(['a
thư ơ ng hiệu_model_combo
```

Hết[275]:

		ngày_tên đã thu thập	giá_USD	tối thiểu	loại xe_loại	đăng ký_năm	
mô hình thư ơ ng hiệu							
	145	4	4	4	4	2	4
	147	78	78	78	78	73	78
alfa_romeo	156	87	87	87	87	84	87
	159	32	32	32	32	31	32
	Andere	59	59	59	59	56	59
...	...	...	...	...	...	...	...
	v40	86	86	86	86	84	86
	v50	28	28	28	28	28	28
volvo	v60	3	3	3	3	3	3
	v70	90	90	90	90	87	90

Trong [276]:
```
common_combo = brand_model_combo["name"].sort_values(ascending=False) #cột nà
common_combo
```

Ra[276]: thư ơ ng hiệu mẫu volkswagen
golf bmw 3er 3622
volkswagen polo 2586
opel volkswagen passat 1566
áo choàng 1556
1333
...
lancia kappa 2
xe thám hiểm khám phá 1
kiểm lâm 1
xe 200 1
ford b_max 1
Tên: tên, Độ dài: 289, dtype: int64

Nếu bạn muốn tham gia #TeamUnique, tốt hơ n hết bạn nên tránh chiếc Volkswagen chơ i gôn hoặc 3er BMW và đư ợc mua một chiếc Audi 200 hoặc b_max Ford

Thêm dữ liệu số, xin vui lòng

Để giải trí, hãy cũng chuyển đổi ngày thành dữ liệu số thống nhất, vì vậy "2016-03-21" trở thành số nguyên 20160321 .

Trong [277]:
```
autos["date_crawled"] = autos["date_crawled"].str[:10].str.replace("-","").ast
```

Trong [278]:
```
autos["ad_created"] = autos["ad_created"].str[:10].str.replace("-","").astype(
```

Trong [279]:
```
autos["last_seen"] = autos["last_seen"].str[:10].str.replace("-","").astype(in
```

Trong [280]:
```
autos.head()
```

Hết[280]:

	date_crawled	tên	price_usd	abtest	vehicle_typ
0	20160326	Peugeot_807_160_NAVTECH_ON_BOARD	kiểm soát 5000		xe buýt
1	20160404	BMW_740i_4_4_Liter_HAMANN_UMBAU_Mega_Optik	điều khiển 8500		xe limousine
2	20160326	Volkswagen_Golf_1.6_United	8990	Bài kiểm tra	xe limousine
3	20160312	Smart_smart_fortwo_coupe_softouch/F1/Klima/Pan...	điều khiển 4350		nhỏ_c
4	20160401	Ford_Focus_1_6_Benzin_TÜV_neu_ist_sehr_gepfleg...	1350	Bài kiểm tra	va

Nhóm số dặm

Vì chúng tôi đã kết luận số dặm đã đư ợc làm tròn nên chúng có thể dễ dàng minh họa cho các mục đích khác Phân tích. Chúng tôi cho rằng những phư ơ ng tiện có quãng đư ờng đi đư ợc ít hơ n sẽ có giá trung bình thấp hơ n, vì vậy, hãy xem liệu đúng rồi.

Chúng tôi sẽ bắt đầu bằng cách xem lại số lư ợng giá trị của công tơ mét:

Trong [281]:
```
autos["odometer_km"].value_counts().sort_index() #odometer_group = autos.group
```

Ra[281]:
```
5000      742
10000     239
20000     738
30000     756
40000     791
50000     988
60000    1117
70000    1175
80000    1359
90000    1655
100000   2032
125000   4803
150000  29486
Tên: odometer_km, dtype: int64
```

Chúng ta có thể thấy rằng có 13 loại số dặm, chúng tôi sẽ thu hẹp chúng thành 3 nhóm - thấp, trung bình và cao:

Trong [282]:
```
odometer_price = autos.groupby("odometer_km")
odometer_price["price_usd"].mean().sort_values(ascending=False)
#odometer_price["price_usd"].mean().sort_values(ascending=False)
```

Ra[282]:
```
odometer_km
10000        20574.305439
20000        18483.537940
30000        16644.611111
40000        15540.653603
50000        13844.735830
60000        12442.254252
70000        10987.248511
80000         9752.215600
90000         8515.038066
100000        8205.128937
5000          7267.716981
125000        6231.157402
150000        3806.961405
Tên: price_usd, dtype: float64
```

Từ những điều trên, chúng ta có thể thấy rằng giả định của chúng ta là đúng - nghĩa là giá giảm đáng kể với số dặm.

Khám phá ảnh hưởng thiệt hại đến giá cả

Xe hư rẻ hơn xe không hư . Đó là tiêu chuẩn. Nhưng, bằng bao nhiêu?

Trong [283]:
```
damage_group = autos.groupby("unrepaired_damage").count()
nhóm thiệt hại
```

Hết[283]:

	ngày_tên đã thu thập	giá_USD	tối thiểu	loại xe_loại	đăng ký_năm	thiết bị	
unrepaired_damage							
không	33446	33446	33446	33446	33046	33446	32
Đúng	4443	4443	4443	4443	4244	4443	4

Trong [284]:
```
no_damage = autos[autos["unrepaired_damage"] == "no"] #autos.groupby("unrepaired
thiệt hại = autos[autos["unrepaired_damage"] == "yes"]
```

Trong [285]:
```
print(no_damage["price_usd"].mean())
in(thiệt hại["price_usd"].mean())
damage_difference = no_damage["price_usd"].mean() - damage["price_usd"].mean()

print("Trung bình, ô tô bị hư hỏng có giá USD {:.2f}".format(damage_difference) " rẻ hơn so với
      + ô tô không bị hư hại.")
```

```
7165.327034622975
2266.5109160477155
```

Tính trung bình, ô tô có hư hỏng rẻ hơn 4898,82 USD so với ô tô không hư hỏng.
đồng nghiệp.

Chúng ta có thể thấy rằng, một chiếc ô tô dự kiến đã sửa chữa các hư hỏng có giá cao hơn nhiều ($4898,82) so với những ô tô chưa sửa chữa.

Tuy nhiên, hãy cùng xem những thương hiệu nào ít nhiều bị ảnh hưởng bởi hư hỏng chưa được sửa chữa:

Trong [286]:
```
brands_unrepaired_vc = damage["brand"].value_counts(normalize=True).sort_value brands_unrepaired
= brands_unrepaired_vc.index brands_unrepaired
```

Ra[286]:
```
Index(['volkswagen', 'opel', 'ford', 'bmw', 'mercedes_benz', 'audi', 'renaul
t',
         'peugeot', 'fiat', 'nissan'],
       dtype='đối tượng')
```

Trong [287]:
```
unrepaired_brand_price = {}

đối với b trong brands_unrepaired:
     select_rows = damage[damage["brand"] == b] mean_price
     = select_rows["price_usd"].mean() unrepaired_brand_price[b]
     = int(mean_price)
```

Trong [288]:
```
unrepaired_brand_price
```

Ra[288]:
```
{'volkswagen': 2196, 'opel':
  1369, 'ford':
  1391, 'bmw':
  3554,
  'mercedes_benz': 4000,
  'audi': 3350,
  'renault': 1167,
  'peugeot ': 1366,
  'fiat': 1166,
  'nissan': 1962}
```

Trong [289]:
```
ubp_series = pd.Series(unrepaired_brand_price).sort_values(ascending=False)
```

Trong [290]:
```
brands_repaired_vc = no_damage["brand"].value_counts(normalize=True).sort_valu
Brand_repaired = Brand_repaired_vc.index
Brand_repaired
```

Ra[290]:
```
Index(['volkswagen', 'bmw', 'mercedes_benz', 'opel', 'audi', 'ford', 'renaul
t',
         'peugeot', 'fiat', 'ghế'],
       dtype='đối tượng')
```

Trong [291]:
```
repair_brand_price = {}

cho b trong brands_repaired:
    select_rows = no_damage[no_damage["brand"] == b]
    mean_price = select_rows["price_usd"].mean()
    repair_brand_price[b] = int(mean_price)
```

Trong [292]:
```
repair_brand_price
```

Ra[292]:
```
{'volkswagen': 6505,
 'bmw': 9467,
 'mercedes_benz': 9834,
 'opel': 3673,
 'âm thanh': 10902,
 'ford': 4695,
 'renault': 3110,
 'peugeot': 3691,
 'sắc lệnh': 3452,
 'ghế': 5220}
```

Trong [293]:
```
rbp_series = pd.Series(repaired_brand_price).sort_values(ascending=False)
```

Trong [294]:
```
damage_price_info = pd.DataFrame(ubp_series, các cột = ["unrepaired_price"])
```

Trong [295]:
```
damage_price_info["repaired_price"] = rbp_series
```

Trong [301]:
```
damage_price_info["diff"] = (damage_price_info["unrepaired_price"] - damage_pr
damage_price_info["diff_%"] = (((damage_price_info["unrepaired_price"] - thiệt hại
```

Trong [302]:
```
damage_price_info.sort_values(bởi =["diff_%"])
```

Hết[302]:

	chư a sửa chữa_giá	đã sửa chữa_giá	khác biệt_%
ford	1391	4695.0	-3304.0 -70.0
audi	3350	10902.0	-7552.0 -69.0
volkswagen	2196	6505.0	-4309.0 -66.0
sắc lệnh	1166	3452,0	-2286,0 -66,0
ô tô	1369	3673.0	-2304.0 -63.0
peugeot	1366	3691.0	-2325.0 -63.0
xe BMW	3554	9467,0	-5913,0 -62,0
renault	1167	3110.0	-1943.0 -62.0
mercedes_benz	4000	9834,0	-5834,0 -59,0
nissan	1962	NaN	NaN NaN

Trong [303]:
```
damage_price_info["diff_%"].mean().round(2)
```

Hết[303]: -64,44

Trung bình, xe có hư hỏng rẻ hơn 59%-70% so với xe không hư hỏng

hầu hết các thương hiệu.

Phần kết luận

Để tôi kể cho bạn nghe một điều thú vị từ phân tích mà chúng ta vừa thực hiện. Giống như , hoàn toàn thú vị. Một số thứ thông thường như Audi là quá đắt đối với một chiếc xe vừa mới thành #TeamUnique; BMW, Mercedes-Benz và VW nằm trong số những thương hiệu xe hơi hàng đầu châu Âu, những chiếc xe bị hư hỏng rẻ hơn những chiếc xe không bị hư hỏng, và quãng đường đi được nhiều hơn có thể khiến giá xe rẻ hơn như thế nào. Những thứ đó có thể dễ dàng suy ra ngay cả khi chúng tôi không sử dụng dữ liệu này.

Mẫu xe Audi RS6 2023