2020春季学期

北京大学《AI引论》课程小组项目书面报告

二手车数据分析

尤俊浩(1900094810),汪蔚涝(1900094813), 钟山(1700011623)

Email: 1900094810@pku.edu.cn, 1900094813@pku.edu.cn, shanzhong@pku.edu.cn

院系：信息科学技术学院，信息科学技术学院，物理学院

指导老师：童云海老师

2020年5月20日

摘 要

通过对Kaggle上得到的二手车数据集，进行数据分析，以找出影响价格的变量。同时进行其他的变量相关性分析。最后通过构建回归模型，试图通过测试集预测价格。

关键词：二手车，相关性分析，回归模型，价格预测，机器学习，数据智能

目 录

1 项目背景

二手车市场上的待售车呈现出**售价**、**品牌**、**损耗程度**、**型号**等多个方面上的丰富性。出于多方面的要求，如：对于一类型二手车的价格和售卖情况，人们有一些先验猜测需要检验；卖方需要在挂牌前进行预测评估；规范和协调二手车市场需要对这些变量间联系的更深入的理解等等，有必要对二手车的销售情况和价格损失规律进行分析。

对从Kaggle上有关二手车的数据集进行数据分析。

- 进行数据预处理和数据清洗
- 进行数据分析，找出影响价格因素的变量，分析因子间的相关性
- 构建有关价格的回归模型
- 将项目成果进行展示

2 相关工作

a) 数据预处理
   - 去除无用、重复、异常的变量
   - 填补空值
   - 建新的变量
   - 将德文翻译成英文并建立中文词典

b) 数据概览
   - 观察变量值的分布

c) 数据分析
   - 关于汽车公里数
   - 关于品牌
   - 关于车龄
   - 相关性分析

d) 数据可视化
   - 画图表（柱状图、线型图、热图、相关性强度图）

e) 构建回归模型
   - 对价格进行预测

f) 整合

g) 报告

h) PPT

i) 海报

j) 视频录制

k) 视频剪辑
   - 页面与语音的连接
   - 字幕

3 分析方法与步骤

3.1 数据的认知

本数据集是从Ebay-Kleinanzeigen（德国二手市场网络平台）爬取下来的，数据提供者说明了“nrOfPictures ”一栏是无效的。

1. 引入数据

```
df = pd.read_csv("autos.csv",encoding = 'cp1252')
```

```
In [3]: df.columns

Out[3]: Index(['dateCrawled', 'name', 'seller', 'offerType', 'price', 'abtest',
               'vehicleType', 'yearOfRegistration', 'gearbox', 'powerPS', 'model',
               'kilometer', 'monthOfRegistration', 'fuelType', 'brand',
               'notRepairedDamage', 'dateCreated', 'nrOfPictures', 'postalCode',
               'lastSeen'],
              dtype='object')
```

2. 对数据进行概览

```
In [2]: df.head()
```

Out[2]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	month
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0	golf	150000	
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190	NaN	125000	
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163	grand	125000	
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75	golf	150000	
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen	2008	manuell	69	fabia	90000	

可以得到几个信息： 内容是德语的、至少有几个columns是对我们的分析没有意义的

```
In [4]: for i in c:
            print(df.groupby(i).count()['name'],'\n')

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

nrOfPictures
0    371528
Name: name, dtype: int64

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

‘seller’、‘offerType’占比过于悬殊，对分析没有意义

3.2 数据的预处理

把已知对分析无用的columns去除

```
In [5]: df.drop(['seller','offerType','postalCode','dateCreated','lastSeen','dateCrawled','nrOfPictures','abtest']
                ,axis = 1, inplace = True)
```

把重复值去除

```
In [7]: df = df.drop_duplicates(['name','price','vehicleType','yearOfRegistration'
                                 ,'gearbox','powerPS','model','kilometer','monthOfRegistration','fuelType'
                                 ,'notRepairedDamage'])
```

再来看看数据的基本情况

```
In [6]: df.describe()
```

Out[6]:

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration
count	3.715280e+05	371528.000000	371528.000000	371528.000000	371528.000000
mean	1.729514e+04	2004.577997	115.549477	125618.688228	5.734445
std	3.587954e+06	92.866598	192.139578	40112.337051	3.712412
min	0.000000e+00	1000.000000	0.000000	5000.000000	0.000000
25%	1.150000e+03	1999.000000	70.000000	125000.000000	3.000000
50%	2.950000e+03	2003.000000	105.000000	150000.000000	6.000000
75%	7.200000e+03	2008.000000	150.000000	150000.000000	9.000000
max	2.147484e+09	9999.000000	20000.000000	150000.000000	12.000000

注意到有几个地方是异常点，把他们排除

```
In [11]: df= df[(df['price']<=200000) & (df['price']>100)
                & (df['yearOfRegistration']>1960) & (df['yearOfRegistration']<=2016)
                &(df['powerPS'] <= 1000) & (df['powerPS'] >= 10)]
```

创建新的columns，以便更加方便地分析与讨论数据

```
In [13]: df['car_age'] = (2016) - (df['yearOfRegistration'])
```

```
In [14]: df['horse_power'] = (df['powerPS'] * 0.986)
```

将德文的内容都翻译成英文

```
In [17]: trans_eng = {'automatik':'automatic','manuell':'manual','kombi':'combi','andere':'others'
                      ,'kleinwagen':'supermini','cabrio':'convertible','benzin':'gasoline'
                      ,'elektro':'electro','nein':'no','ja':'yes','sonstige_autos':'miscellaneous_car'}
for i in trans_eng:
    df.replace(i,trans_eng[i],inplace=True)
```

同时，创立中文词典，方便日后的数据可视化

```
In [18]: trans_man = {'name':'Name','price':'价格','vehicleType':'种类','monthOfRegistration':'注册月份','powerPS':'公制马力'
                      ,'yearOfRegistration':'注册年份','gearbox':'变速箱','horse_power':'马力',
                      'model':'型号','kilometer':'里程','fuelType':'燃料','brand':'品牌','notRepairedDamage':'受损情况','car_age':'车龄',
                     'not-declared':'未知','coupe':'双座四轮轿车','suv':'运动型多功能车','supermini':'迷你车',
                      'limousine':'加长轿车','convertible':'敞篷车','bus':'巴士','combi':'旅行车',
                      'others':'其他','gasoline':'汽油','diesel':'柴油','nan':'未知','lpg':'液化石油气',
                      'hybrid':'油电混合','cng':'压缩天然气', 'electro':'电动','manual':'手动档','automatic':'自动档',
                      'volkswagen':'大众汽车','audi':'奥迪','jeep':'吉普车','skoda':'斯柯达','bmw':'宝马','peugeot':'标致',
                      'ford':'福特','mazda':'长安马自达','nissan':'日产','renault':'雷诺','mercedes_benz':'奔驰',
                      'seat':'西雅特','honda':'本田','fiat':'菲亚特','opel':'欧宝','mini':'宝马迷你','smart':'精灵汽车',
                      'hyundai':'现代汽车','alfa_romeo':'阿尔法·罗密欧','subaru':'斯巴鲁','volvo':'沃尔沃','mitsubishi':'三菱',
                      'kia':'起亚','suzuki':'铃木','lancia':'蓝旗亚','porsche':'保时捷','citroen':'雪铁龙',
                      'toyota':'丰田','chevrolet':'雪佛兰','dacia':'达西亚','daihatsu':'大发汽车','trabant':'卫星轿车',
                      'chrysler':'克莱斯勒','jaguar':'捷豹汽车','daewoo':'大宇汽车','rover':'罗孚','saab':'萨博','land_rover':'路虎',
                      'lada':'拉达汽车','no':'无','yes':'有','golf':'高尔夫','miscellaneous_car':'其他'}
```

看看多少个遗失数据

```
In [20]: df.isnull().sum()
```

```
Out[20]: name                    0
         price                   0
         vehicleType         10714
         yearOfRegistration      0
         gearbox              5231
         powerPS                 0
         model               11262
         kilometer               0
         monthOfRegistration     0
         fuelType            15260
         brand                   0
         notRepairedDamage   42610
         car_age                 0
         horse_power             0
         dtype: int64
```

大概占总数据的5%左右

我们采用Front Fill来填充数据

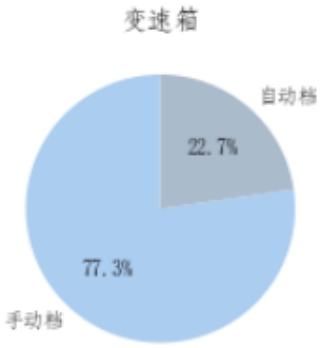
```
In [28]: df_fillna = df.copy()
         for i in na_set:
             df_fillna[i].fillna(method='ffill',inplace=True)
         for i in na_set:
             df_fillna[i].fillna(method='bfill',inplace=True)
```

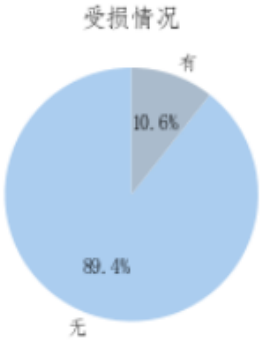
先进行可视化，大致了解一下数据的主导成员,我们选择观察columns的数量占比最多的前五项

In [29]:
```
for i, c in enumerate(categories):
    g = df_fillna.groupby(by=c)[c].count().sort_values(ascending=False)
    labels = g.index
    if(c != 'model'):
        translated  = []
        for i in labels:
            translated.append(trans_man[i])
        labels = translated
    l = len(labels)
    if(l > 5):
        print(g.head())
        r = range(min(l,5))
        plt.figure(figsize=(5,3))
        plt.bar(r, g.head(),color = '#aabbcc')
        plt.xticks(r, labels)
        plt.ylabel("数量")
        plt.title(trans_man[c],fontdict = {'fontweight':'bold','fontsize':18})
        plt.show()

    else:
        print(g)
        plt.figure(figsize=(5,3))
        colors = ['#abcdef','#aabbcc','#C2CFDC']
        plt.pie(g,labels=labels,autopct='%1.1f%%',shadow=False, startangle=90,colors = colors)
        plt.title(trans_man[c], fontdict = {'fontweight':'bold','fontsize':13})
        plt.show()
```

```
gearbox
manual        239411
automatic      70235
Name: gearbox, dtype: int64
```

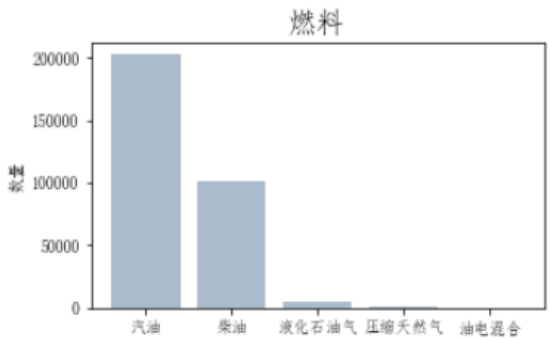
```
notRepairedDamage
no      276812
yes      32834
Name: notRepairedDamage, dtype: int64
```

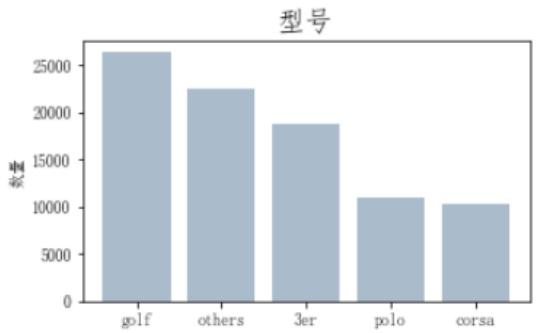





```
fuelType
gasoline    202355
diesel      101423
lpg           4990
cng            516
hybrid         250
Name: fuelType, dtype: int64
```

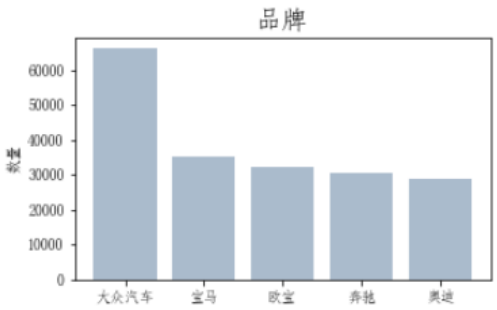
```
model
golf      26327
others    22570
3er       18785
polo      10955
corsa     10307
Name: model, dtype: int64
```

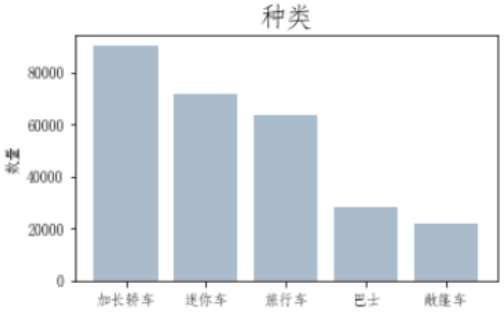




```
brand
volkswagen       66013
bmw              35375
opel             32343
mercedes_benz    30356
audi             28894
Name: brand, dtype: int64
```

```
vehicleType
limousine      89895
supermini      71444
combi          63513
bus            28518
convertible    21949
Name: vehicleType, dtype: int64
```





把已经填充的Data Frame复制给原df

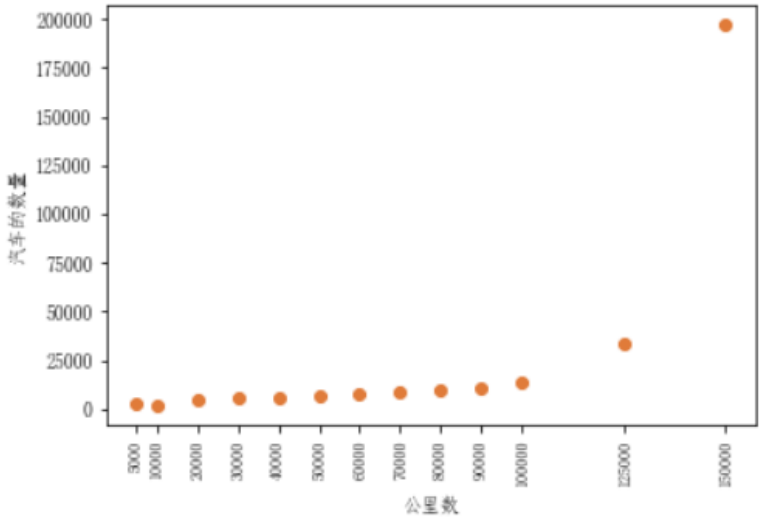
```
In [32]: df = df_fillna.copy()
```

3.3 数据分析

i. 关于汽车公里数

首先先简单的来看关于公里数与汽车数量的分布

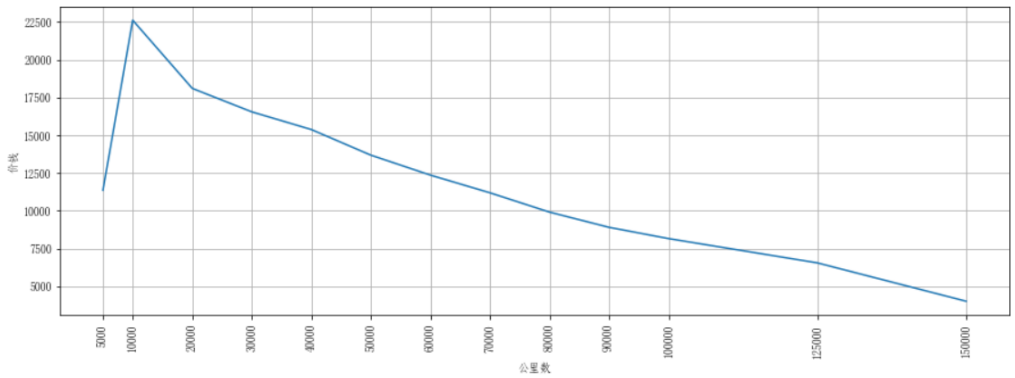
```
In [25]: km = df.groupby(['kilometer'])
         num = km.count()['name']

         km = [km for km, df in km]

         plt.scatter(km, num, color = '#e07b39')
         plt.xticks(km, rotation = 'vertical', size = 8)
         plt.xlabel('公里数')
         plt.ylabel('汽车的数量')
         plt.show()
```



从图表中可以看到，大多数的车子行驶的公里数是介于100000km至150000km之间，其中更有大约20万辆车的公里数达到了150000km

再来看汽车公里数和其售卖价钱

```
In [26]: plt.figure(figsize = (15,5))
prices = df.groupby(['kilometer']).mean()['price']
plt.plot(km,prices)
plt.grid()
plt.xticks(km, rotation = 'vertical', size = 10)
plt.xlabel('公里数')
plt.ylabel('价钱')
plt.show()
```



从图表中我们可以看出，峰值是在10000km，过后价钱呈现出下降的趋势。我们可以推断出，公里数较低的车让买家愿意相信车况较为良好，而公里数较高的车会让买家不禁怀疑车的受损情况与零件情况。

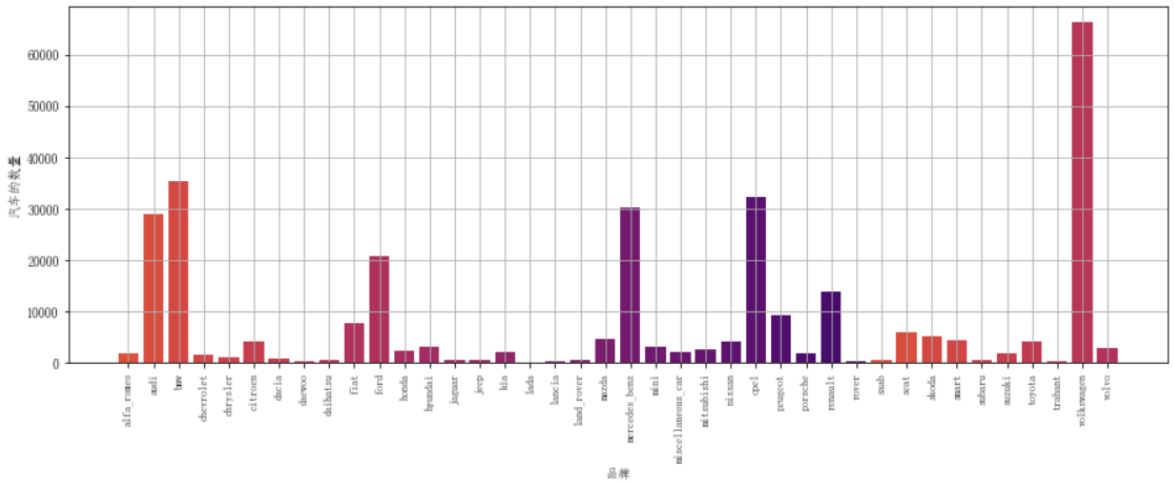
ii. 关于品牌

首先先import过后会在图标上用到的关于颜色的库，并把颜色设置好

```
In [27]: from matplotlib import cm
color = cm.inferno_r(np.linspace(.4, .8, 30))
##color
```

先简单的来看关于各品牌与汽车数量的分布

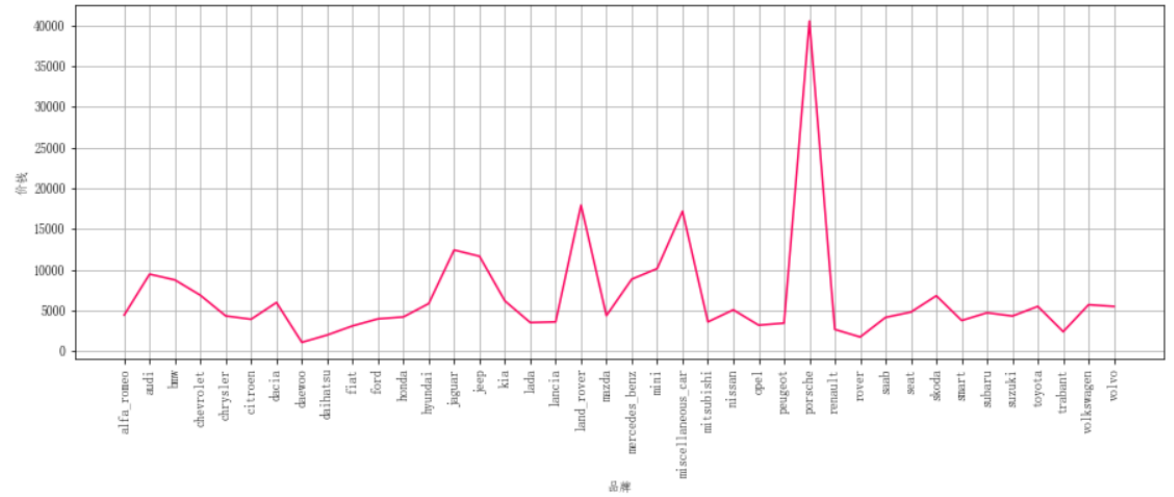
```
In [28]: ## Method 1
## Show out the relationship respectively in two graphs

## Find out the relationship between brand and number of cars sold
plt.figure(figsize = (15,5))
brand = df.groupby('brand')
num = brand.count()['car_age']

brands = [brand for brand, df in brand]
plt.grid()

plt.bar(brands,num, color = color)
plt.xticks(brands, rotation = 'vertical', size = 8)
plt.xlabel('品牌')
plt.ylabel('汽车的数量')
plt.show()

## Volkswagen is the most, follow by bmw and opel
```



可以看到大众汽车的数量最多、其次是宝马和欧宝。从品牌与汽车数量的图表，我们也能在一定程度上估算各品牌在当地的市场占有率。

再来看品牌与价钱之间的关系

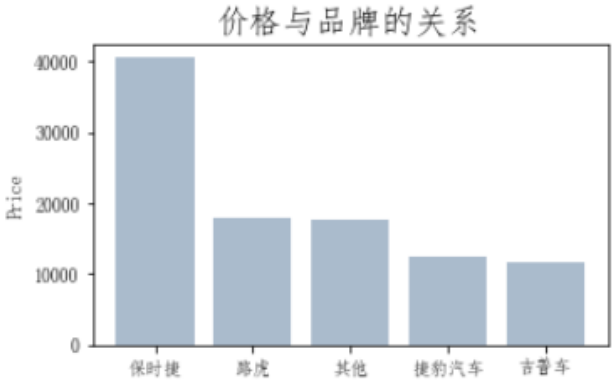
In [29]:
```
## Find out which brand sold with the highest price
plt.figure(figsize = (15,5))
prices = df.groupby(['brand']).mean()['price']
plt.plot(brands,prices,color = '#fc035e')
plt.grid()

plt.xticks(brands, rotation = 'vertical', size = 10)
plt.xlabel('品牌')
plt.ylabel('价钱')
plt.show()
## Porsche sold with a more highest price, follow by land_rover and sonstige_autos
```



很显然可以看到，保时捷售卖的价钱是最高的，相信这是因为大多数都是性能及配置都非常好的跑车。第二高的则是路虎。从图表中不难看到，二手车的价钱是与品牌自身的价位有很大的关系。

更进一步地观察价格与品牌的关系

In [33]:
```
plt.figure(figsize=(5,3))
r = range(5)
prices = df.groupby(['brand']).mean()['price'].sort_values(ascending=False)
plt.bar(r,prices.head(),color = '#aabbcc')
labels = prices.index
translated  = []
for i in labels:
    translated.append(trans_man[i])
labels = translated
plt.xticks(r,labels,size = 10)
plt.ylabel('Price')
plt.title('价格与品牌的关系',fontdict = {'fontweight':'bold','fontsize':18})
plt.show()
```



可以看到，显然，保时捷是远远超于其他品牌的。
除了保时捷、路虎外，捷豹汽车和捷普车的平均价格也很高。

iii. 关于车龄

我们直接在一张图表里展示车龄与其数量和价格的关系。

In [34]:
```
car_age = [car_age for car_age, d in df.groupby('car_age')]
price = df.groupby('car_age').mean()['price']

fig,ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.bar(car_age,price,color = 'y')
ax2.plot(car_age, df.groupby(['car_age']).count(),'b')
ax1.grid()
ax2.set_title('车龄与其数量和价格的关系')
ax2.set_xlabel('车龄')
ax2.set_ylabel('数量',color = 'b')
ax1.set_ylabel('价格',color = 'y')
plt.show()
```

可以看到，车价随着车龄的函数是呈现一个谷形的。新车显然价格是高的，而车龄达到一定程度后价格为什么不降反升呢？我们推断那些车由于其稀有性反而成为了古董车，具有收藏的价值，所以价格才会攀升。

另外，我们可以清楚看到车主大多选择在购入汽车10年后选择转卖。

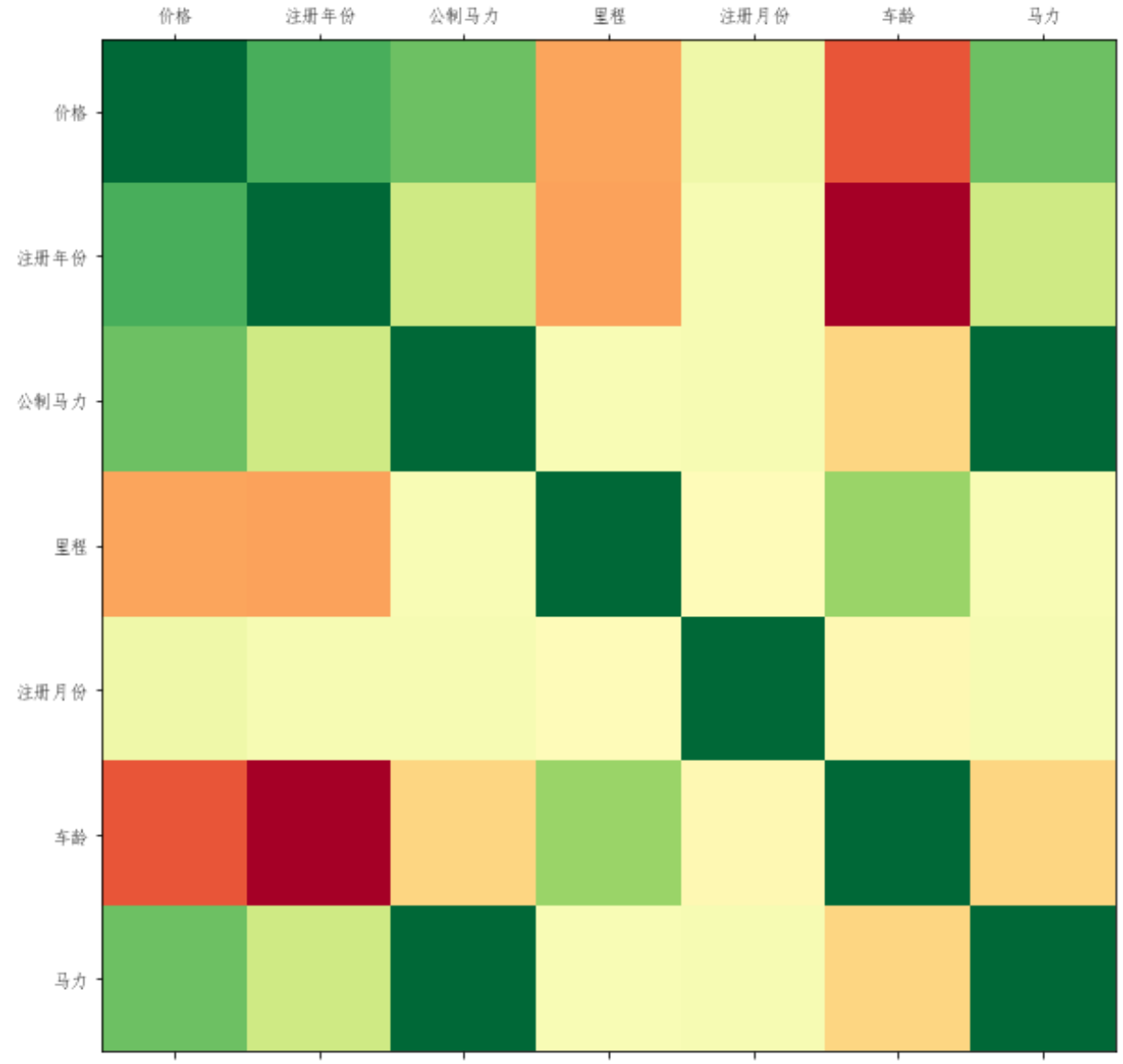
同时，也能看到价格和数量的趋势恰恰是相反的，这说明了市场的供给越少，车价则会越高。

iv. 相关性分析

接着对各个变量的相关性进行概览，对于Pearson、Kendall和Spearman法出来的结果相近，我们采用Spearman法进行说明

In [35]:
```
import numpy as np
```

In [36]:
```
df_corr = df.corr(method = 'pearson')
df_corr.columns
labels = [trans_man[c] for c in df_corr.columns]
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111)
ax.matshow(df_corr,cmap=plt.cm.RdYlGn)

ax.set_xticks(np.arange(len(labels)))
ax.set_yticks(np.arange(len(labels)))
ax.set_xticklabels(labels)
ax.set_yticklabels(labels)
plt.show()
```



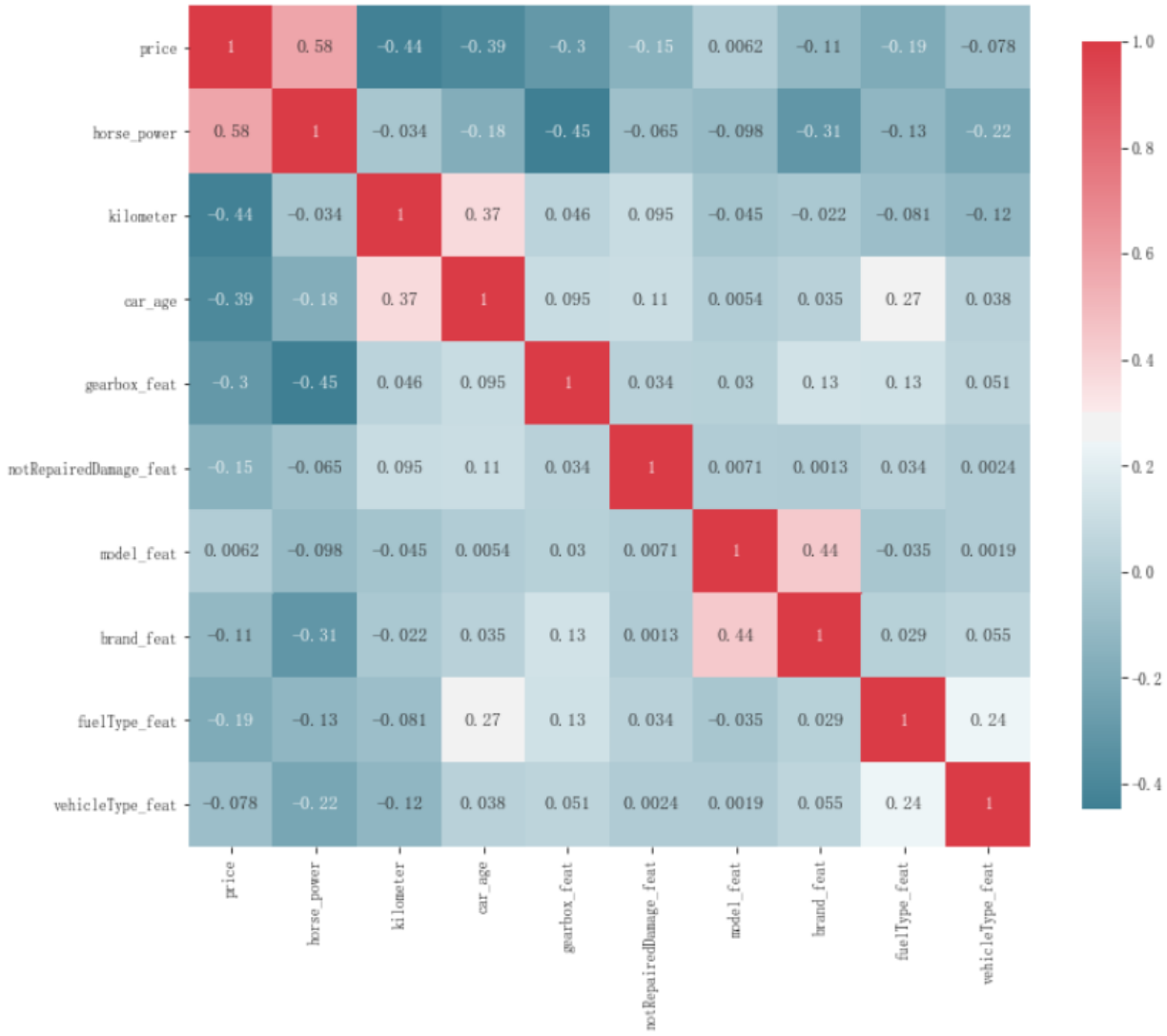
可以看到，显然车龄和注册年份是强负相关的；车龄与里程是正相关的。马力与车龄也是负相关的，这可能是因为造车的工艺不断进步所致。

我们重点想要观察的是价格与其他变量的相关性，可以看到，价格与注册年份，马力是正相关的；而与里程和车龄是负相关的。

进一步的用热图给出更具体与数值化的表示

In [40]:
```
from sklearn import datasets, linear_model, preprocessing, svm
from sklearn.preprocessing import StandardScaler, Normalizer

labels = ['name', 'gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']
les = {}

df2 = df.copy()
for l in labels:
    les[l] = preprocessing.LabelEncoder()
    les[l].fit(df2[l])
    tr = les[l].transform(df2[l])
    df2.loc[:, l + '_feat'] = pd.Series(tr, index=df2.index)

labeled = df2[ ['price','horse_power','kilometer','car_age']+ [x+"_feat" for x in labels]].copy()
len(labeled['name_feat'].unique()) / len(labeled['name_feat'])
labeled.drop(['name_feat'], axis='columns', inplace=True)
#print (labeled)
plot_correlation_map(labeled)
labeled.corr()
```

结果如下



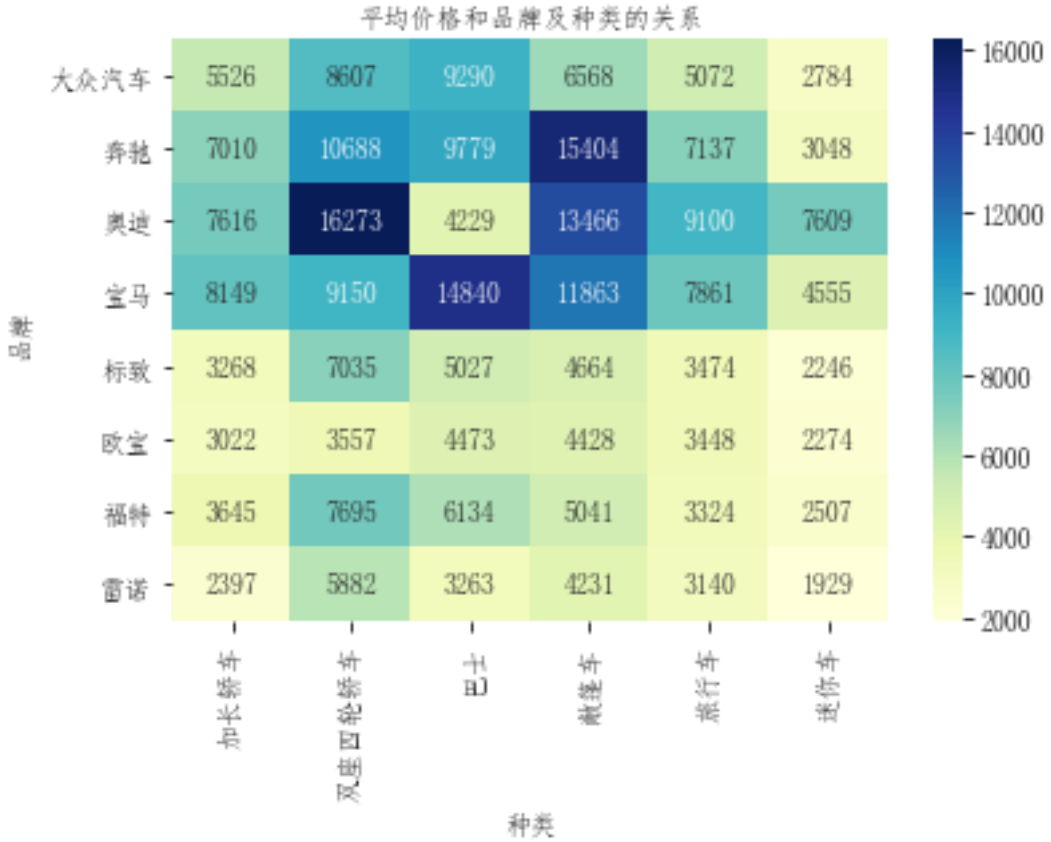
Out[40]:

	price	horse_power	kilometer	car_age	gearbox_feat	notRepairedDamage_feat	model_feat	brand_feat	fuelType_feat	vehicleTy
price	1.000000	0.576979	-0.436963	-0.389878	-0.296488	-0.151530	0.006214	-0.110271	-0.190292	-0
horse_power	0.576979	1.000000	-0.034015	-0.177302	-0.449143	-0.064688	-0.098227	-0.312178	-0.125766	-0
kilometer	-0.436963	-0.034015	1.000000	0.369881	0.045858	0.095014	-0.044510	-0.022334	-0.080525	-0
car_age	-0.389878	-0.177302	0.369881	1.000000	0.095378	0.105163	0.005424	0.035195	0.269586	0
gearbox_feat	-0.296488	-0.449143	0.045858	0.095378	1.000000	0.033527	0.030007	0.129020	0.127936	0
notRepairedDamage_feat	-0.151530	-0.064688	0.095014	0.105163	0.033527	1.000000	0.007133	0.001329	0.034128	0
model_feat	0.006214	-0.098227	-0.044510	0.005424	0.030007	0.007133	1.000000	0.438471	-0.034600	0
brand_feat	-0.110271	-0.312178	-0.022334	0.035195	0.129020	0.001329	0.438471	1.000000	0.029384	0
fuelType_feat	-0.190292	-0.125766	-0.080525	0.269586	0.127936	0.034128	-0.034600	0.029384	1.000000	0
vehicleType_feat	-0.078040	-0.221332	-0.121825	0.037540	0.051178	0.002389	0.001890	0.055430	0.244663	1

与之前分析的结果一致，进一步发现了变速器种类与燃料种类也与价格有相关性。
再看看汽车的品牌和种类与其价格的关系

In [41]:
```
# #### correlation analysis of descriptive features and price
brand_counts=df['brand'].value_counts(normalize=True)
common_brands=brand_counts[brand_counts>0.03].index

#selecting common brands
brand_avg_prices = {}#dictionary

for brand in common_brands:
    brand_only = df[df['brand'] == brand]
    avg_price = brand_only['price'].mean()
    brand_avg_prices[brand] = int(avg_price)

bap_series = pd.Series(brand_avg_prices).sort_values(ascending = False)
print(pd.DataFrame(bap_series, columns = ["avg_price"]) )
#simply use tables to show

type_counts=df['vehicleType'].value_counts(normalize=True)
common_types=type_counts[type_counts>0.05].index
#selecting common vehicle types

tri=pd.DataFrame()
for b in list(common_brands):
    for v in list(common_types):
        z=df[(df['brand']==b)&(df['vehicleType']==v)]['price'].mean()
        tri=tri.append(pd.DataFrame({'brand':b,'vehicleType':v,'avgPrice':z},index=[0]))
tri=tri.reset_index()
del tri['index']
tri["avgPrice"] = tri["avgPrice"].astype(int)

tri = tri.pivot("brand","vehicleType", "avgPrice")
fig, ax = plt.subplots()
sns.heatmap(tri,annot=True,cmap="YlGnBu",fmt="d")
ax.set_title("Average price of vehicles by vehicle type and brand",fontdict={'size':10})
plt.show()
#use 2D heatmaps to show how two factors influence the prices
```

同样的用热图展示



可以看到奥迪的双轮轿车平均价格最高，而以雷诺为首的迷你车系列平均价格最低。

3.4 价格的预测——建立回归模型

首先进行一份复制，并且对于所有种类划分的columns进行数字编码

```
In [42]: df_sci = df.copy()
```

```
In [43]: c = ['vehicleType','gearbox','model','fuelType','notRepairedDamage','brand','name']
         for i in c:
             df_sci[i] = df_sci[i].astype('category').cat.codes
```

把Data Frame打乱，避免以后的预测出现侥幸

```
In [44]: import sklearn
         from sklearn import svm
```

```
In [45]: df_sci = sklearn.utils.shuffle(df_sci)
```

把弱相关性的columns去掉，再对df进行压缩

```
In [46]: X = df_sci.drop(columns = ['price','name','model','vehicleType'],axis = 1).values
         X = preprocessing.scale(X)
         y = df_sci['price'].values
```

分离测试数据集与训练测试数据集，我们把测试数据量定在200份

```
In [47]: test_size = 200

         X_train = X[:-test_size]
         y_train = y[:-test_size]

         X_test = X[-test_size:]
         y_test = y[-test_size:]

         clf = sklearn.svm.LinearSVR()
         clf.fit(X_train,y_train)
```

```
Out[47]: LinearSVR(C=1.0, dual=True, epsilon=0.0, fit_intercept=True,
                   intercept_scaling=1.0, loss='epsilon_insensitive', max_iter=1000,
                   random_state=None, tol=0.0001, verbose=0)
```

采用Linear SVR模型的得分为0.590

```
In [46]: clf.score(X_test,y_test)
```

```
Out[46]: 0.590191569806229
```

换SGD Regressor试试

```
In [49]: clf = sklearn.linear_model.SGDRegressor()
         clf.fit(X_train,y_train)

Out[49]: SGDRegressor(alpha=0.0001, average=False, early_stopping=False, epsilon=0.1,
                      eta0=0.01, fit_intercept=True, l1_ratio=0.15,
                      learning_rate='invscaling', loss='squared_loss', max_iter=1000,
                      n_iter_no_change=5, penalty='l2', power_t=0.25, random_state=None,
                      shuffle=True, tol=0.001, validation_fraction=0.1, verbose=0,
                      warm_start=False)
```

得到的评分为 0.602

```
In [49]: clf.score(X_test,y_test)

Out[49]: 0.6017239976753991
```

再试试Linear Regression，得到的评分为0.590。

```
         lr_clf.score(X_test,y_test)

Out[51]: 0.5900371569388725
```

随机挑选几个测试数据来看看。

```
Model:5808.119357248426,Actual:7500
Model:11230.70710482097,Actual:14100
Model:8791.246608970541,Actual:4999
Model:7448.488210871268,Actual:7499
Model:-540.2733130797615,Actual:650
```

```
Model:8604.468895610962,Actual:7500
Model:13497.765294745182,Actual:14100
Model:11016.176723373303,Actual:4999
Model:9446.2741390656,Actual:7499
Model:-1392.4622832390005,Actual:650
```

```
Model:9444.65325304477,Actual:7500
Model:13836.65325304477,Actual:14100
Model:11593.40325304477,Actual:4999
Model:9960.65325304477,Actual:7499
Model:-1731.3467469552297,Actual:650
```

对于三个模型，可以看到对于部分数据来说预测结果还是相当准确的。然而，对于第五个数据，三个模型都给出了奇怪的预测结果。它们均做出了卖家应该付钱让买家领走他们的车的结论，这显然是不能接受的。

再来看看我们的最后一个模型Random Forest —随机森林

```
In [53]: from sklearn.ensemble import RandomForestRegressor
         from sklearn.model_selection import GridSearchCV

         rf = RandomForestRegressor()

         param_grid = { "criterion" : ["mse"]
                      , "min_samples_leaf" : [3]
                      , "min_samples_split" : [3]
                      , "max_depth": [10]
                      , "n_estimators": [500]}

         gs = GridSearchCV(estimator=rf, param_grid=param_grid, cv=2, n_jobs=-1, verbose=1)
         gs = gs.fit(X_train, y_train)

         Fitting 2 folds for each of 1 candidates, totalling 2 fits

         [Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
         [Parallel(n_jobs=-1)]: Done   2 out of   2 | elapsed:  3.4min finished
```

参数设置如下：criterion:MSE、min_sample_leaf:3、min_sample_split:3、max_depth:10、n_estimators:500

最终的得分如下

```
print('Score: %.2f' % forest.score(X_test, y_test))
```

```
Score: 0.87
```

得分为0.87，是所有模型里面得分最高的！

```
Model:7012.623417806082,Actual:7500
Model:14820.82477936605,Actual:14100
Model:4905.004534592366,Actual:4999
Model:6677.049310705619,Actual:7499
Model:971.0552846533263,Actual:650
```

更令人庆幸的是，之前奇怪的预测结果没有再出现了！

4 实验

各个回归模型的尝试，最后发现Random forest的效果最好；对于Back fill, Front fill, Drop na 和 直接填充not-declared 差别不大，最后决定采用Back fill来填充空值。对于Pearson、Spearman和Kendall相关性分析结果的差异也不是很大，我们最终使用Spearman来进行展示。

5 结论

a) 二手车的价值与马力是正相关的，而与车龄和里程数是负相关的。

b) 车龄和里程数在一定的程度上反映二手车的车况。

c) 二手车的其他指标如燃料种类与变速箱种类也会影响其价格。

d) 大多数车主选择在购入车子的十年后陆续转卖车辆。

e) 二手车的价格虽然与车龄是负相关的，但稀有的古董车却具有收藏价值，所以价格反而更高。

f) 二手车的价值与其在市场的供给有关，越稀有的二手车价值越高。

g) 二手车市场的分布能够在一定程度上反应各品牌在当地的市场占有率。

h) 二手车的品牌在某种程度上决定了二手车在市场的基本定价。

Python作为数据分析的主流工具集合了庞大的资源与库，使用起来非常方便，但想要用好各种工具需要很多的实践！分析过程中，及时进行可视化有助于检查操作的正确性和增加对当前数据的整体把握。

小组分工

任务	负责人
数据预处理	尤俊浩
数据概览	尤俊浩
数据相关性分析	钟山、汪蔚涝
数据可视化	尤俊浩、汪蔚涝、钟山
建立回归模型	尤俊浩
整合	尤俊浩
报告	尤俊浩
PPT	尤俊浩、汪蔚涝
海报设计	钟山
视频录制	汪蔚涝、钟山
视频剪辑	尤俊浩、汪蔚涝

致谢

感谢老师的教导，组员们的合作，以及各个数据分析社群提供的教程、资源以及帮助。我们意识到我们对于数据的处理还不够成熟，但是通过此次数据小班大作业，我们学到了很多，也使得我们对数据科学有了进一步的认识。

参考资料

[1] Dan—Trying to predict used car value (1kaggle.com/ddmngml/trying-to-predict-used-car-value)

[2] Sentdex – Data Analysis with Python and Pandas（https://www.youtube.com/watch?v=BpPJxtOk8uw&list=PLQVvvaa0QuDfSfqQuee6K8opKtZsh7sA9&index=6）

[3] Learning code with Chinese teacher — 5小时学会Python数据分析与展示 (https://www.youtube.com/watch?v=Obbc3pC5_g0&t=13416s)