注：

*没有把取对数的那些变量列放在里面考虑，即只用了“原始数据”的数据列。原来想加 date 进去的，觉得也许可以搞天气，但是好像要换格式而且会议论文里没提，所以我想着后面再说吧。

*因变量 drivingscore

```
# 数据导入
df = pd.read_excel('eng317.xlsx',
                   usecols=['ranking', 'alert', 'age', 'gender', 'accident', 'drivingexp',
'gcons',
'CO2emission',
'date',
'mileage',
'totalm',
'time',
'Speed_KMH',
'rapid_acc',
'rapid_deacc',
'sharp_turn',
'fatigdriving',
'nightdriving',
'drivingscore',
'app_usage'])
```

*图都有清晰版本的，这个文档里可能看不清，就随便意思下

*ipynb 文件是 python 代码；然后 STATA 的 log 文件我就不发了，就几行

***因为是初步结果，随便看看那种，所以那个 linearRegression 我没有用 python 看系数和 p-value 啥的，而是直接丢进了 STATA 后面确定了一些东西之后可能还是要用同一个软件？（还是说就这样也行？）**

1. 数据处理

eng371.xlsx 这个数据集是由最全的那个数据集经过如下处理：

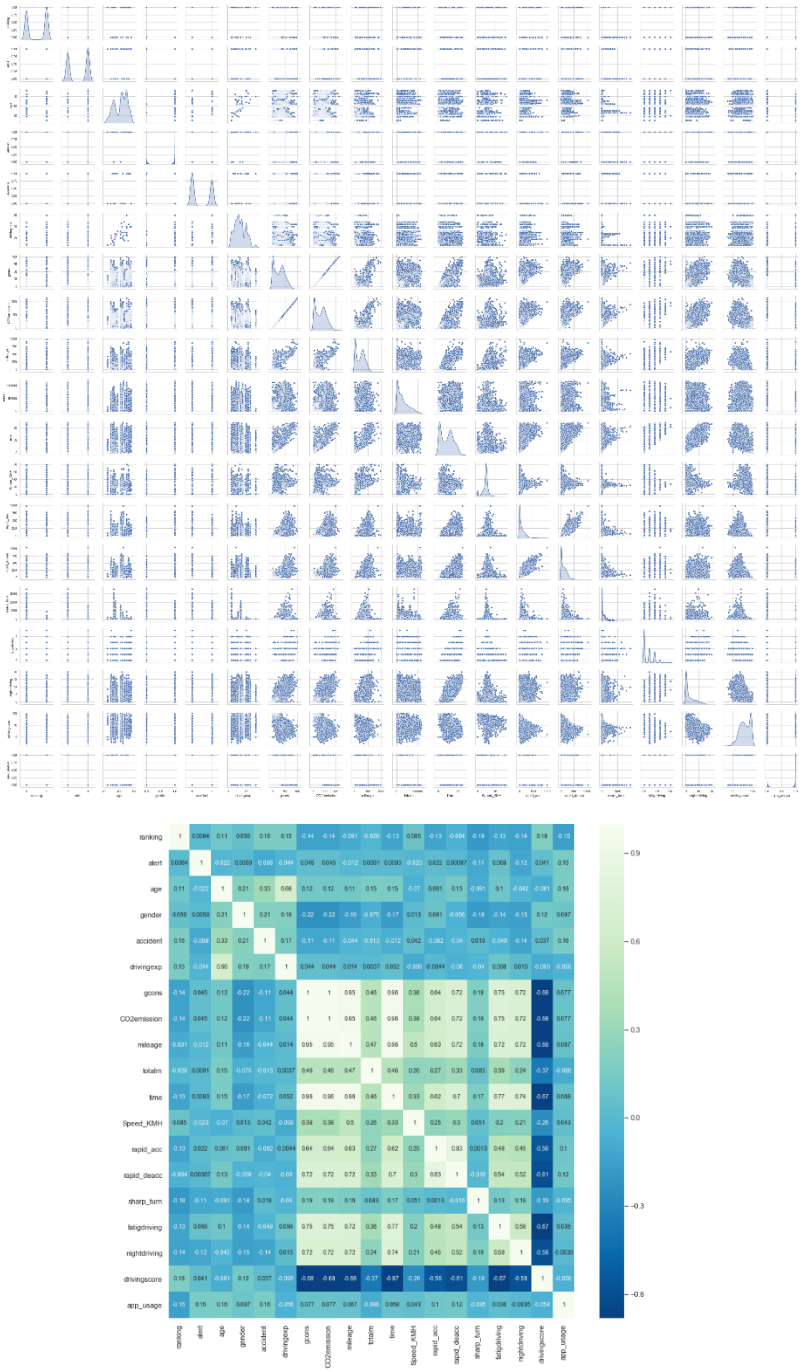
- 删除所有 CO2emission 为 0 的值——没出行
- 删除所有“闽 xxxx”和“1.22E”乱码车牌——那些跟 2019-2020 那年的实验不是一批
- 所有变量名为英文

最终得到 11187 条数据

2. 特征选择（Python，最优特征子集选择）

1) 观察 df 中所有变量相关性

发现几个相关度太高的：CO2 和 gasconsumption、mileage、time 强相关，这里仅选择 gascons



于是只剩下 15 个自变量和一个因变量。

2) 利用 RFECV 方法进行特征选择（特征子集与交叉验证）

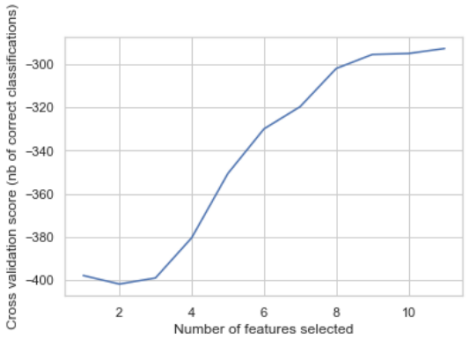
这个方法不能容忍缺失值，所以这里我分了两种情况处理缺失值。

a) 直接删除有缺失值的变量

能够直接删除的原因：只有 age, gender, accident, drivingexp 存在缺失值（这玩意儿全都是那个问卷上录下来的）

删除后特征选择结果：11 个特征都保留

```
RFECV特征选择结果————————————————————————————-
Optimal number of features : 11
Ranking of features : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```



b) 删除所有含缺失值的行

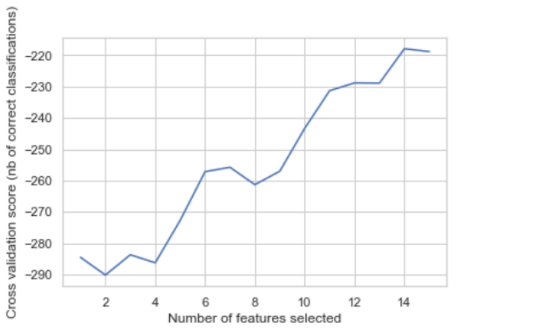
能够直接删除的原因：只有 age, gender, accident, drivingexp 存在缺失值，且缺失值占比不大

```
# 每列缺失值占比
print(feature.isnull().sum()/feature.shape[0])
```

```
ranking          0.000000
alert            0.000000
age              0.350496
gender           0.348351
accident         0.323947
drivingexp       0.323947
gcons            0.000000
totalm           0.000000
Speed_KMH        0.000000
rapid_acc        0.000000
rapid_deacc      0.000000
sharp_turn       0.000000
fatigdriving     0.000000
nightdriving     0.000000
app_usage        0.000000
dtype: float64
```

删除后特征选择结果：14 个特征保留，删除 totalm

```
RFECV特征选择结果————————————————————————————————
Optimal number of features : 14
Ranking of features : [1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1]
```



3. 跑模型与评估

（1） Python

***我看了五种回归模型，这个结果最好的是 GBR-梯度提升线性回归，其次就是 LinearR。但 GBR 这玩意儿系数、p-value 啥的我还在寻找中，暂时不知道怎么显示**

删除有缺失值的列跑模型，即用 11 个 features:

```
In [92]: # 各种评估结果
         df_met
```

Out[92]:

	ev	mae	mse	r2
BayesianRidge	0.593162	11.393480	275.534381	0.593162
LinearRegression	0.593165	11.394226	275.532348	0.593165
ElasticNet	0.567244	11.929585	293.087262	0.567244
SVR	0.080373	20.958478	622.867108	0.080310
GBR	0.868094	6.741127	89.334588	0.868094

删除有缺失值的行后跑模型，即用 14 个 features, 6993 个 obsevations:

```
In [96]: # 各种评估结果
         df_met
```

Out[96]:

	ev	mae	mse	r2
BayesianRidge	0.552181	10.559876	201.265388	0.552181
LinearRegression	0.552193	10.559920	201.260138	0.552193
ElasticNet	0.523547	10.987193	214.134391	0.523547
SVR	0.102873	15.892452	403.208416	0.102855
GBR	0.773510	7.610738	101.792393	0.773510

（2） STATA

我好像不大会用 STATA，老师你看下是不是有问题 QAQ

```
. //删除了age等变量
. regress drivingscore ranking alert gcons totalm Speed_KMH rapid_acc rapid_deacc sharp_turn fatigdriving nightdr
> iving app_usage , robust

Linear regression                               Number of obs     =     11,187
                                                F(11, 11175)      =    1632.81
                                                Prob > F          =     0.0000
                                                R-squared         =     0.5932
                                                Root MSE          =     16.608

------------------------------------------------------------------------------
             |               Robust
drivingscore |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     ranking |    3.325944   .3397733     9.79   0.000     2.659928    3.991959
       alert |    2.257169    .336778     6.70   0.000     1.597025    2.917314
       gcons |   -.016531   .0165344    -1.00   0.317    -.0489414    .0158794
      totalm |  -.0000839   6.34e-06   -13.24   0.000    -.0000964   -.0000715
   Speed_KMH |   -.125756   .0310783    -4.05   0.000    -.1866749   -.0648371
   rapid_acc |  -.0300621   .0025197   -11.93   0.000    -.0350012   -.0251231
 rapid_deacc |  -.0366178   .0024718   -14.81   0.000    -.0414631   -.0317726
  sharp_turn |  -.0137586   .0013368   -10.29   0.000    -.0163789   -.0111382
fatigdriving |  -9.841109   .2058287   -47.81   0.000    -10.24457   -9.437649
nightdriving |  -1.125353   .0600852   -18.73   0.000    -1.243131   -1.007575
   app_usage |  -1.883212   .6444087    -2.92   0.003    -3.146367   -.6200575
       _cons |   82.09536   .9605125    85.47   0.000     80.21259    83.97813
------------------------------------------------------------------------------
```

Linear regression

Number of obs	6,993
F(15, 6977)	552.98
Prob > F	0.0000
R-squared	0.5539
Root MSE	14.176

drivingscore	Coef.	Robust Std. Err.	t	P>|t|	[95% Conf.	Interval]
ranking	.3056983	.4176007	0.73	0.464	-.512926	1.124323
alert	.1521831	.3710628	0.41	0.682	-.5752128	.879579
age	-.0343025	.0325354	-1.05	0.292	-.0980819	.0294769
gender	2.386824	.6376597	3.74	0.000	1.136817	3.636831
accident	-1.297761	.3903241	-3.32	0.001	-2.062915	-.5326076
drivingexp	-.085139	.048653	-1.75	0.080	-.1805136	.0102356
gcons	.0799874	.0195902	4.08	0.000	.0415847	.1183902
totalm	-.0000364	7.04e-06	-5.16	0.000	-.0000502	-.0000226
Speed_KMH	.0982591	.0544377	1.80	0.071	-.0084554	.2049736
rapid_acc	-.026556	.0024016	-11.06	0.000	-.0312639	-.021848
rapid_deacc	-.0374228	.0024873	-15.05	0.000	-.0422987	-.0325468
sharp_turn	-.0146239	.0013633	-10.73	0.000	-.0172963	-.0119515
fatigdriving	-8.816613	.2161982	-40.78	0.000	-9.240428	-8.392799
nightdriving	-1.16681	.0633085	-18.43	0.000	-1.290914	-1.042706
app_usage	.3558633	.6255467	0.57	0.569	-.8703984	1.582125
_cons	72.59031	2.133729	34.02	0.000	68.40755	76.77307