Data loading

In [4]:
```
1 import pandas as pd
2 age_train=pd.read_csv("age_train.csv",names=["uId", "age_group"])
3 age_train.tail(10)
```

Out[4]:

	uId	age_group
2009990	3512491	3
2009991	3512492	2
2009992	3512493	3
2009993	3512494	2
2009994	3512495	5
2009995	3512496	1
2009996	3512497	6
2009997	3512498	2
2009998	3512499	2
2009999	3512500	3

In [5]:
```
1 age_test=pd.read_csv("age_test.csv",names=["uId"])
2 age_test.tail()
```

Out[5]:

	uId
502495	3180767
502496	3181007
502497	3181086
502498	3181099
502499	3181144

In [6]:
```
1 user_basic_info=pd.read_csv("user_basic_info.csv",names=["uId", "gender", "city", "prodName","ramCa
2 user_basic_info.tail(10)
```

Out[6]:

	uId	gender	city	prodName	ramCapacity	ramLeftRation	romCapacity	romLeftRation	color	fontSize	ct	c
2512490	3512472	0	c00145	p00147	4.0	0.32	64.0	0.55	幻夜黑	1.00	wifi	China_Te
2512491	3512473	0	c0066	p009	3.0	0.36	64.0	0.66	荣耀金	1.15	4g	China_Te
2512492	3512474	1	c00145	p00169	3.0	0.52	32.0	0.12	皓月银	1.15	3g#wifi	China_U
2512493	3512477	0	c00106	p0044	2.0	0.27	16.0	0.24	太空银	1.00	4g#wifi	China_M
2512494	3512478	0	c00281	p00155	6.0	0.32	64.0	0.46	备件颜色	1.00	4g#wifi	China_Te
2512495	3512489	0	c00192	p00102	6.0	0.36	128.0	0.74	翡冷翠	1.00	4g#wifi	China_U
2512496	3512491	0	c007	p00223	3.0	0.35	32.0	0.16	晨曦金	NaN	NaN	China_Te
2512497	3512492	0	c0038	p0016	6.0	0.33	64.0	0.20	琥珀金	1.00	4g#wifi	China_U
2512498	3512494	0	c0043	p0016	4.0	0.34	64.0	0.41	海鸥灰	NaN	4g#wifi	China_M
2512499	3512495	0	c00145	p00141	6.0	0.26	68.0	0.54	银钻灰	NaN	wifi	China_Te

In [7]:
```
1 user_behavior_info=pd.read_csv("user_behavior_info.csv",names=["uId", "bootTimes", "AFuncTimes", "B
2 user_behavior_info.tail(10)
3
```

Out[7]:

	uId	bootTimes	AFuncTimes	BFuncTimes	CFuncTimes	DFuncTimes	EFuncTimes	FFuncSum	GFuncSum
2512490	3512491	5	0.0	0.00	0.00	0.00	0.00	0.0	0
2512491	3512492	24	0.0	0.00	0.07	0.43	0.07	2.0	154683
2512492	3512493	5	0.0	0.00	0.47	0.23	0.00	0.0	3193
2512493	3512494	4	0.0	0.07	1.47	0.00	0.00	0.0	1306
2512494	3512495	0	0.0	0.00	0.00	0.00	0.00	0.0	0
2512495	3512496	37	0.0	0.00	0.17	0.03	0.03	0.0	19431
2512496	3512497	40	0.0	0.00	0.00	0.67	0.07	0.0	947
2512497	3512498	2	0.0	0.00	0.00	0.00	0.00	0.0	0
2512498	3512499	2	0.0	0.17	1.17	6.83	7.83	0.0	512
2512499	3512500	11	1.0	0.00	1.37	3.90	2.87	0.0	0

In [8]:
```
1 user_app_actived=pd.read_csv("user_app_actived.csv",names=["uId", "appId"])
2 user_app_actived.tail(10)
```

Out[8]:

	uId	appId
2512490	3512461	a00102782#a00104275#a00109386#a00109824#a00118...
2512491	3512464	a00102720#a00109386#a00145168#a0015685#a001704...
2512492	3512466	a00102939#a00109386#a00134746#a00135785#a00150...
2512493	3512475	a00109386#a00144187#a00157201#a0021880#a002447...
2512494	3512476	a00102448#a00109386#a00144187#a00157058#a00157...
2512495	3512477	a00109386#a00121683#a00158614#a00187480#a00225...
2512496	3512479	a00109386#a0011894#a00145168#a00149248#a001586...
2512497	3512483	a00109386#a00135785#a0017008#a00170432#a001874...
2512498	3512484	a00102895#a00109386#a00158614#a00170432#a00212...
2512499	3512491	a00109386#a0012409#a00142198#a0014473#a0020705...

In [9]:
```
app_info=pd.read_csv("app_info.csv",names=["appId", "category"])
app_info.tail(10)
```

Out[9]:

	appId	category
188854	a00488097	实用工具
188855	a00488109	便捷生活
188856	a00488109	实用工具
188857	a00488110	便捷生活
188858	a00488111	休闲益智
188859	a00488113	经营策略
188860	a00488114	便捷生活
188861	a00488115	实用工具
188862	a00488119	便捷生活
188863	a00488122	实用工具

In [10]:
```
merged_inner = pd.merge(left=age_train,right=user_basic_info, left_on='uId', right_on='uId')
merged_inner.tail()
```

Out[10]:

	uId	age_group	gender	city	prodName	ramCapacity	ramLeftRation	romCapacity	romLeftRation	color	fontSize	
2009995	3512496	1	0	c00245	p00210	4.0	0.55	64.0	0.33	银色	1.0	4g#w
2009996	3512497	6	0	c00136	p00208	3.0	0.41	32.0	0.76	金色	1.3	Na
2009997	3512498	2	0	c00176	p0045	2.0	0.29	16.0	0.28	银色	NaN	Na
2009998	3512499	2	0	c00126	p0022	8.0	0.17	128.0	0.84	渐变蓝	1.0	4g#w
2009999	3512500	3	0	c00229	p00110	4.0	0.29	64.0	0.06	琥珀金	1.0	w

In [11]:
```
1 user_behavior_info.head()
```

Out[11]:

	uId	bootTimes	AFuncTimes	BFuncTimes	CFuncTimes	DFuncTimes	EFuncTimes	FFuncSum	GFuncSum
0	1000001	108	0.0	0.00	1.00	0.07	0.0	0.0	3319
1	1000002	14	0.0	0.17	4.93	1.23	3.9	1.0	245
2	1000003	13	1.0	0.00	7.73	3.00	1.7	0.0	5987
3	1000004	57	0.0	0.03	1.37	0.63	0.0	0.0	7460
4	1000005	0	0.0	0.00	0.00	0.00	0.0	0.0	0

In [12]:
```
1 to_predict=pd.merge(left=merged_inner,right=user_behavior_info, left_on='uId', right_on='uId')
2 to_predict.tail()
```

Out[12]:

	uId	age_group	gender	city	prodName	ramCapacity	ramLeftRation	romCapacity	romLeftRation	color	...	carr
2009995	3512496	1	0	c00245	p00210	4.0	0.55	64.0	0.33	银色	...	China_Telec
2009996	3512497	6	0	c00136	p00208	3.0	0.41	32.0	0.76	金色	...	China_Unic
2009997	3512498	2	0	c00176	p0045	2.0	0.29	16.0	0.28	银色	...	China_Mo
2009998	3512499	2	0	c00126	p0022	8.0	0.17	128.0	0.84	渐变蓝	...	China_Unic
2009999	3512500	3	0	c00229	p00110	4.0	0.29	64.0	0.06	琥珀金	...	China_Mo

5 rows × 22 columns

In [13]:
```
1 to_predict['carrier'].unique()
```

Out[13]:
```
array(['China_Mobile', 'China_Telecom', 'China_Unicom', 'othercp'],
      dtype=object)
```

In [14]:
```
1 to_predict.shape
```

Out[14]:
```
(2010000, 22)
```

In [15]:
```
to_predict=to_predict.dropna()
to_predict.reset_index()
to_predict.shape
```

Out[15]: (1370640, 22)

Feature engineering

In [16]:
```
X_labels=["gender", "city","prodName","ramCapacity","ramLeftRation","romCapacity","romLeftRation","
y_label='age_group'

from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
to_predict["carrier"] = lb_make.fit_transform(to_predict["carrier"])
to_predict["color"] = lb_make.fit_transform(to_predict["color"])
to_predict["ct"] = lb_make.fit_transform(to_predict["ct"])
to_predict["prodName"] = lb_make.fit_transform(to_predict["prodName"])
to_predict["city"] = lb_make.fit_transform(to_predict["city"])
to_predict.head(11)

```

Out[16]:

	uId	age_group	gender	city	prodName	ramCapacity	ramLeftRation	romCapacity	romLeftRation	color	...	carrier	os	bootTi
0	1000001	4	0	48	50	3.0	0.43	32.0	0.46	70	...	0	8.0	
2	1000015	5	1	223	50	3.0	0.34	32.0	0.06	70	...	1	8.0	
4	1000023	2	1	288	105	2.0	0.34	16.0	0.06	103	...	1	7.0	
5	1000025	4	0	311	52	4.0	0.31	64.0	0.20	65	...	1	8.0	
6	1000029	4	0	346	41	6.0	0.20	68.0	0.27	31	...	2	9.0	
7	1000035	2	0	234	41	6.0	0.26	68.0	0.64	12	...	1	9.0	
9	1000038	3	0	329	10	4.0	0.29	64.0	0.16	42	...	0	7.0	
10	1000040	3	0	247	32	6.0	0.20	128.0	0.52	86	...	1	9.0	
11	1000044	4	0	323	43	6.0	0.22	137.0	0.30	2	...	2	9.0	
12	1000046	3	0	131	106	4.0	0.49	64.0	0.01	7	...	0	7.0	
13	1000051	3	1	28	45	6.0	0.27	128.0	0.17	115	...	0	8.0	

11 rows × 22 columns

In [17]:
```
to_predict.shape
```

Out[17]:
```
(1370640, 22)
```

In [18]:
```
to_predict['age_group'].value_counts()
```

Out[18]:
```
3    406461
4    345040
2    258685
5    214086
6    109579
1     36789
Name: age_group, dtype: int64
```

In [66]:
```
y_data=to_predict[y_label]
X_data=to_predict[X_labels]
# del to_predict['age_group']
# to_predict.tail()
print(X_data.shape)
y_data.shape
```

```
(1370640, 20)
```

Out[66]:
```
(1370640,)
```

In [73]:
```
from sklearn import preprocessing
X = X_data.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
X_scaled = min_max_scaler.fit_transform(X)
X_scaled
# df = pandas.DataFrame(x_scaled)
```

Out[73]:
```
array([[0.00000000e+00, 1.36363636e-01, 3.96825397e-01, ...,
        5.00000000e-01, 0.00000000e+00, 7.98260115e-05],
       [1.00000000e+00, 6.33522727e-01, 3.96825397e-01, ...,
        5.00000000e-01, 0.00000000e+00, 5.26264826e-04],
       [1.00000000e+00, 8.18181818e-01, 8.33333333e-01, ...,
        5.00000000e-01, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 4.43181818e-01, 5.39682540e-01, ...,
        5.00015000e-01, 0.00000000e+00, 4.67339328e-04],
       [0.00000000e+00, 7.95454545e-02, 5.95238095e-01, ...,
        5.03915000e-01, 0.00000000e+00, 1.23142265e-05],
       [0.00000000e+00, 3.94886364e-01, 8.73015873e-02, ...,
        5.01435000e-01, 0.00000000e+00, 0.00000000e+00]])
```

In [67]:
```
y=y_data.values
y
```

Out[67]:
```
array([4, 5, 2, ..., 1, 2, 3])
```

Model Training

In [75]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=11)

# for experimental purpose, we only use 10% of data to train within a perceivable period of time
# X_train, X_test, y_train, y_test = train_test_split(X_test, y_test, test_size=0.2, random_state=1
# X_train, X_test, y_train, y_test = train_test_split(X_test, y_test, test_size=0.2, random_state=1

print(X_train.shape)
print(y_train.shape)
print("------------")
print(X_test.shape)
print(y_test.shape)
```

```
(1096512, 20)
(1096512,)
------------
(274128, 20)
(274128,)
```

In [77]:
```
from sklearn.naive_bayes import GaussianNB
ml_m = GaussianNB()
%time ml_m.fit(X_train, y_train)
print('Accuracy of GaussianNB classifier on Test Set: {:.4f}'
     .format(ml_m.score(X_test, y_test)))

predicted_test = ml_m.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, predicted_test))
```

```
CPU times: user 566 ms, sys: 262 ms, total: 829 ms
Wall time: 891 ms
Accuracy of GaussianNB classifier on Test Set: 0.2412
              precision    recall  f1-score   support

           1       0.04      0.41      0.08      7164
           2       0.28      0.53      0.37     51697
           3       0.35      0.16      0.22     81683
           4       0.38      0.12      0.19     69078
           5       0.29      0.03      0.05     42510
           6       0.28      0.59      0.38     21996

   micro avg       0.24      0.24      0.24    274128
   macro avg       0.27      0.31      0.21    274128
weighted avg       0.32      0.24      0.22    274128
```

In [82]:
```
from sklearn.svm import LinearSVC
lsvc_pri_model = LinearSVC(C=0.1)

%time lsvc_pri_model.fit(X_train, y_train)
print('Accuracy of LinearSVC classifier on Test Set: {:.4f}'
     .format(lsvc_pri_model.score(X_test, y_test)))

predicted_test = lsvc_pri_model.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, predicted_test))
```

```
CPU times: user 1min 27s, sys: 1.52 s, total: 1min 29s
Wall time: 1min 37s
Accuracy of LinearSVC classifier on Test Set: 0.3390
              precision    recall  f1-score   support

           1       0.00      0.00      0.00      7164
           2       0.36      0.05      0.08     51697
           3       0.34      0.80      0.48     81683
           4       0.34      0.16      0.22     69078
           5       0.31      0.30      0.30     42510
           6       0.42      0.06      0.10     21996

   micro avg       0.34      0.34      0.34    274128
   macro avg       0.30      0.23      0.20    274128
weighted avg       0.34      0.34      0.27    274128


/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sklearn/metrics/classifi
cation.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Prediction for test set and submission

In [89]:

```
merged_inner_test = pd.merge(left=age_test,right=user_basic_info, left_on='uId', right_on='uId')
to_predict_ontest=pd.merge(left=merged_inner_test,right=user_behavior_info, left_on='uId', right_on=
# to_predict_ontest.tail()
print(to_predict_ontest.shape)
to_predict_ontest=to_predict_ontest.dropna()
print(to_predict_ontest.shape)

from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
to_predict_ontest["carrier"] = lb_make.fit_transform(to_predict_ontest["carrier"])
to_predict_ontest["color"] = lb_make.fit_transform(to_predict_ontest["color"])
to_predict_ontest["ct"] = lb_make.fit_transform(to_predict_ontest["ct"])
to_predict_ontest["prodName"] = lb_make.fit_transform(to_predict_ontest["prodName"])
to_predict_ontest["city"] = lb_make.fit_transform(to_predict_ontest["city"])
to_predict_ontest.head(11)
```

```
(502500, 21)
(342926, 21)
```

Out[89]:

	uId	gender	city	prodName	ramCapacity	ramLeftRation	romCapacity	romLeftRation	color	fontSize	...	carrier	os	bootTime
1	1000003	0	219	4	8.0	0.35	128.0	0.60	78	1.00000	...	0	9.0	1
2	1000004	1	195	22	3.0	0.36	32.0	0.09	18	1.00000	...	0	8.0	5
5	1000007	1	310	120	6.0	0.20	137.0	0.50	12	1.00000	...	1	9.0	1
6	1000008	1	294	39	6.0	0.36	128.0	0.47	84	1.15000	...	1	8.0	
7	1000009	0	332	39	6.0	0.39	64.0	0.23	97	1.15000	...	0	8.0	2
9	1000012	0	198	48	4.0	0.31	64.0	0.39	65	1.30001	...	0	8.0	9
10	1000013	0	131	38	4.0	0.49	64.0	0.48	97	1.00000	...	0	8.0	2
11	1000014	1	310	2	6.0	0.49	128.0	0.58	31	1.00000	...	0	9.0	11
12	1000016	0	343	89	3.0	0.41	32.0	0.46	99	1.00000	...	2	7.0	1
13	1000017	1	293	11	4.0	0.29	64.0	0.04	65	1.15000	...	0	8.0	2
15	1000020	0	223	80	4.0	0.38	64.0	0.09	28	1.00000	...	2	8.0	1

11 rows × 21 columns

In [90]:
```
test_labels=["gender", "city","prodName","ramCapacity","ramLeftRation","romCapacity","romLeftRation

test_data=to_predict_ontest[test_labels]

from sklearn import preprocessing
test = test_data.values #returns a numpy array

min_max_scaler = preprocessing.MinMaxScaler()
test_scaled = min_max_scaler.fit_transform(test)
test_scaled

# test_scaled=test
```

Out[90]:
```
array([[0.00000000e+00, 6.23931624e-01, 3.27868852e-02, ...,
        3.79937888e-01, 0.00000000e+00, 3.65952114e-04],
       [1.00000000e+00, 5.55555556e-01, 1.80327869e-01, ...,
        3.78881988e-01, 0.00000000e+00, 4.55988436e-04],
       [1.00000000e+00, 8.83190883e-01, 9.83606557e-01, ...,
        3.79670807e-01, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 9.37321937e-01, 6.06557377e-01, ...,
        3.79254658e-01, 5.00000000e-02, 5.80437827e-04],
       [1.00000000e+00, 4.13105413e-01, 1.88524590e-01, ...,
        3.78881988e-01, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 3.10541311e-01, 1.80327869e-01, ...,
        3.80726708e-01, 0.00000000e+00, 8.04397831e-05]])
```

In [91]:
```
predicted = lsvc_pri_model.predict(test_scaled)
predicted
```

Out[91]:
```
array([5, 2, 5, ..., 5, 5, 2])
```

In [92]:
```
print(to_predict_ontest['uId'].shape)
print(len(predicted))
```

```
(342926,)
342926
```

In [93]:
```
sub_df=pd.DataFrame(to_predict_ontest['uId']).reset_index(drop=True)
sub_df['label']=predicted
```

In [94]:
```
sub_df.columns=['id','label']
sub_df.to_csv("submission.csv",index=False)
print("submission.csv is saved sucessfully!")
```

```
submission.csv is saved sucessfully!
```

In [ ]:
```

```