选取指标

短期渠道评分体系

关键行为分：根据留存魔法数字确定关键行为

变现能力分：使用时长、广告展现次数、启动次数、PV

用户构成：老用户占比

归因得分：模糊归因占比、归因失败占比

数量分：按照数量多少打分

关键行为确定

根据留存魔法数字确定关键行为。留存魔法数字：找到影响用户留存的最关键变量，之后改善这个变量的数值，从而达到改善留存的目的。比如我们发现：一周内点击分享5次及以上的用户，留存率明显高出30%，所以我们就定5次为魔法数字，然后想办法让用户多分享。

具体操作分4步：

列出可采集数据的用户行为

分析行为与留存率关系，找出高度相关行为

得出合适做改善行为的魔法数字

In [1]:

```
# 导入库
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

# 中文乱码的处理
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']# 设置微软雅黑字体
# 忽略警告
warnings.filterwarnings("ignore")
```

In [2]:

```
# 读取数据
key_action_data = pd.read_excel('/home/mw/input/channel_quality3966/key_action.
key_action_data.head()
```

Out[2]:

	key_action	key_action_cnt	retention_rate
0	收藏	1	0.002
1	收藏	2	0.025
2	收藏	3	0.027
3	收藏	4	0.029
4	收藏	5	0.150

In [3]:

```
# 查看关键行为值
key_action_data['key_action'].unique()
```

Out[3]:

```
array(['收藏', '分享', '更换模板', '消息互动'], dtype=object)
```

In [4]:

```
# 收藏数
collection_cnt = key_action_data[key_action_data['key_action'] == '收藏']
# 分享数
share_cnt = key_action_data[key_action_data['key_action'] == '分享']
# 更换模板数
model_change_cnt = key_action_data[key_action_data['key_action'] == '更换模板']
# 消息互动数
message_cnt = key_action_data[key_action_data['key_action'] == '消息互动']
```

In [5]:

```
# 收藏数与留存的相关性
corr1 = collection_cnt.corr()
corr1
```

Out[5]:

	key_action_cnt	retention_rate
key_action_cnt	1.000000	0.814472
retention_rate	0.814472	1.000000

In [6]:

```
# 分享数和留存率的相关性
corr2 = share_cnt.corr()
corr2
```

Out[6]:

	key_action_cnt	retention_rate
key_action_cnt	1.000000	0.926569
retention_rate	0.926569	1.000000

In [7]:

```
# 更换模板数和留存率的相关性
corr3 = model_change_cnt.corr()
corr3
```

Out[7]:

	key_action_cnt	retention_rate
key_action_cnt	1.00000	0.85967
retention_rate	0.85967	1.00000

In [8]:

```
# 消息互动数和留存率的相关性
corr4 = message_cnt.corr()
corr4
```

Out[8]:

	key_action_cnt	retention_rate
key_action_cnt	1.000000	0.539625
retention_rate	0.539625	1.000000

相关系数r的绝对值一般在0.8以上，认为A和B有强的相关性。0.3到0.8之间，可以认为有弱的相关性。0.3以下，认为没有相关性。

所以这四个变量中取前与留存率强相关的三个指标:收藏数、分享数、更换模板数

In [9]:

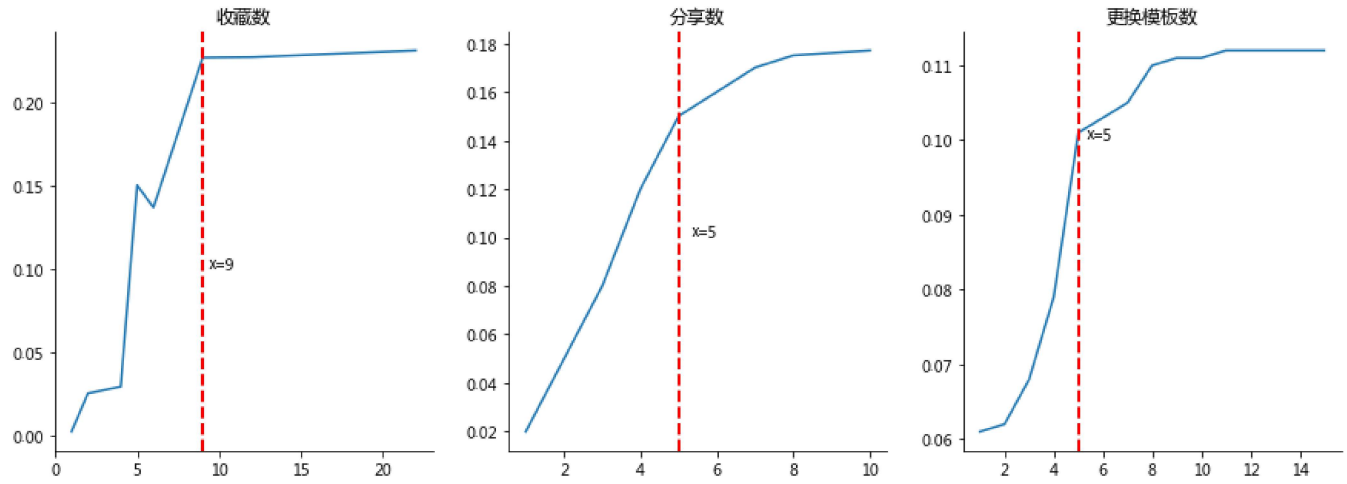
```
# 绘图
fig,ax = plt.subplots(1,3,figsize=(15,5))
ax1,ax2,ax3 = ax.flatten()

#收藏数
#设置数据
x1 = collection_cnt['key_action_cnt']
y1 = collection_cnt['retention_rate']
#绘制折线图
ax1.plot(x1,y1)
#绘制辅助线
ax1.axvline(x=9,c="r",ls="--",lw=2)
ax1.text(9.3,0.10,'x=9')
#消除左右边框
ax1.spines['right'].set_color('none')
ax1.spines['top'].set_color('none')
ax1.set_title('收藏数')

#分享数
#设置数据
x1 = share_cnt['key_action_cnt']
y1 = share_cnt['retention_rate']
#绘制折线图
ax2.plot(x1,y1)
#绘制辅助线
ax2.axvline(x=5,c="r",ls="--",lw=2)
ax2.text(5.3,0.10,'x=5')
#消除左右边框
ax2.spines['right'].set_color('none')
ax2.spines['top'].set_color('none')
ax2.set_title('分享数')

#更换模板数
#设置数据
x1 = model_change_cnt['key_action_cnt']
y1 = model_change_cnt['retention_rate']
#绘制折线图
ax3.plot(x1,y1)
#绘制辅助线
ax3.axvline(x=5,c="r",ls="--",lw=2)
ax3.text(5.3,0.10,'x=5')
#消除左右边框
ax3.spines['right'].set_color('none')
ax3.spines['top'].set_color('none')
ax3.set_title('更换模板数')

plt.show()
```



所以确定关键行为标准：收藏数9，分享数5，更换模板数5为关键行为。

指标筛选

目前我们对于变现能力分的评价有4个指标，我们想要对指标进行筛选，用更少的指标计算变现能力分

In [10]:

```
# 读取数据
revenue_ability_data = pd.read_csv('/home/mw/input/channel_quality3966/revenue_
revenue_ability_data
```

Out[10]:

	user_id	launch_cnt	PV	ad_show_cnt	duration
0	0	2	0	1	3056
1	1	2	0	4	0
2	2	2	0	3	0
3	3	5	15	5	312980
4	4	11	1	18	127880
5	5	2	0	5	0
6	6	2	0	1	20186
7	7	3	2	4	154288
8	8	5	1	16	156531
9	9	4	4	2	72475
10	10	6	0	7	45215
11	11	53	2	48	1044501
12	12	14	1	9	112784
13	13	2	0	14	0
14	14	2	0	9	0
15	15	57	4	6	417527
16	16	3	0	3	15086
17	17	2	0	1	0
18	18	2	0	1	0
19	19	6	1	6	158593
20	20	54	0	35	0
21	21	2	0	1	0
22	22	14	9	12	164186
23	23	3	0	6	21531
24	24	40	3	13	485988
25	25	2	1	4	7683
26	26	35	3	29	727777
27	27	2	0	8	0

	user_id	launch_cnt	PV	ad_show_cnt	duration
28	28	3	4	1	17133
29	29	12	1	2	191220
...	...	...	...	...	...
62311	62311	2	0	4	0
62312	62312	4	6	13	164944
62313	62313	2	0	9	0
62314	62314	2	0	14	0
62315	62315	3	0	9	164110
62316	62316	2	0	1	33857
62317	62317	2	0	1	0
62318	62318	3	0	2	0
62319	62319	7	1	4	255747
62320	62320	7	0	11	228123
62321	62321	4	0	3	24476
62322	62322	3	8	1	45767
62323	62323	12	0	10	1353759
62324	62324	30	2	21	777281
62325	62325	2	0	5	0
62326	62326	2	0	6	0
62327	62327	76	2	46	1645741
62328	62328	3	13	1	81667
62329	62329	3	12	1	319785
62330	62330	4	0	14	273784
62331	62331	2	0	30	1098442
62332	62332	4	0	5	11034
62333	62333	2	14	1	127668
62334	62334	21	2	15	325433
62335	62335	2	0	5	0
62336	62336	23	4	8	573997

	user_id	launch_cnt	PV	ad_show_cnt	duration
62337	62337	2	0	6	0
62338	62338	2	0	9	0
62339	62339	2	0	6	0
62340	62340	2	0	3	14800

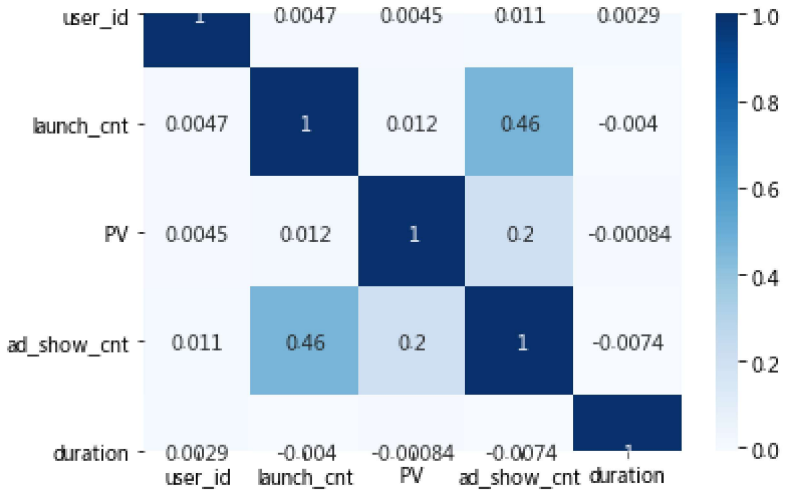
62341 rows × 5 columns

In [11]:

```
# 绘制变现能力指标的相关性矩阵
corr_revenue_ability_data = revenue_ability_data.corr()
sns.heatmap(corr_revenue_ability_data, cmap='Blues', annot=True)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe108ca780>
```



从上面的矩阵我们可以看出来的是：启动数与广告展示数呈现弱相关，其他指标之间没有相关性,所以我们需要对这四个指标进行权重打分。

权重打分

In [12]:

```
# 读取数据
index_weight_raw = pd.read_excel('/home/mw/input/channel_quality3966/权重打分表.›
index_weight_raw
```

Out[12]:

	launch_cnt	PV	ad_show_cnt	duration
Z				
launch_cnt	1	0.5	0.3333	0.2000
PV	2	1.0	0.5000	0.2500
ad_show_cnt	3	2.0	1.0000	0.3333
duration	5	4.0	3.0000	1.0000

In [13]:

```
# 列向量归一化
index_weight = index_weight_raw.apply(lambda x : x/x.sum())
index_weight
```

Out[13]:

	launch_cnt	PV	ad_show_cnt	duration
Z				
launch_cnt	0.090909	0.066667	0.068959	0.112152
PV	0.181818	0.133333	0.103449	0.140190
ad_show_cnt	0.272727	0.266667	0.206898	0.186901
duration	0.454545	0.533333	0.620694	0.560758

In [14]:

```
# 行求和
index_weight['row_sum'] = index_weight.apply(lambda x : x.sum(),axis=1)
index_weight
```

Out[14]:

	launch_cnt	PV	ad_show_cnt	duration	row_sum
Z					
launch_cnt	0.090909	0.066667	0.068959	0.112152	0.338686
PV	0.181818	0.133333	0.103449	0.140190	0.558790
ad_show_cnt	0.272727	0.266667	0.206898	0.186901	0.933193
duration	0.454545	0.533333	0.620694	0.560758	2.169331

In [15]:

```
# 求和项归一
index_weight['normalization'] = index_weight[['row_sum']].apply(lambda x :x/x.s
index_weight
```

Out[15]:

	launch_cnt	PV	ad_show_cnt	duration	row_sum	normalization
Z						
launch_cnt	0.090909	0.066667	0.068959	0.112152	0.338686	0.084672
PV	0.181818	0.133333	0.103449	0.140190	0.558790	0.139698
ad_show_cnt	0.272727	0.266667	0.206898	0.186901	0.933193	0.233298
duration	0.454545	0.533333	0.620694	0.560758	2.169331	0.542333

使用时长权重：0.542333

广告展现次数权重：0.233298

启动次数权重：0.084672

PV权重：0.139698

一致性检验

·一般认为一致性比率CR<0.1时，认为A的不一致程度在容许范围之内，有满意的一致性，通过一致性检验。

·CR = CI/RI

·CI = (λ-n)/(n-1)

·n是特征数量，n=4

·λ = 1/n(AW1/W1+AW2/W2+...+AWi/Wi)

·行向量求和、经归一化（使向量中各元素之和为1）后记为W

·A是原始的打分表数据

In [16]:

```
# 一致性检验计算-计算AW
AW1 = index_weight_raw.iloc[0,0]*index_weight.iloc[0,5]+index_weight_raw.iloc[0
AW2 = index_weight_raw.iloc[1,0]*index_weight.iloc[0,5]+index_weight_raw.iloc[1
AW3 = index_weight_raw.iloc[2,0]*index_weight.iloc[0,5]+index_weight_raw.iloc[2
AW4 = index_weight_raw.iloc[3,0]*index_weight.iloc[0,5]+index_weight_raw.iloc[3
```

In [17]:

```
print(f'AW1={AW1}')
print(f'AW2={AW2}')
print(f'AW3={AW3}')
print(f'AW4={AW4}')
```

```
AW1=0.34074519327536157
AW2=0.5612730068619691
AW3=0.947467529058454
AW4=2.2243753172453293
```

In [18]:

```
# 计算λ
lambda_ = 1/4*(AW1/index_weight.iloc[0,5]+AW2/index_weight.iloc[1,5]+AW3/index_
lambda_
```

Out[18]:

```
4.051192777069623
```

In [19]:

```
# 计算CI
CI = (lambda_-4)/(4-1)
CI
```

Out[19]:

```
0.01706425902320774
```

AHP方法中平均随机一致性指标RI取值参考

阶数	RI
1	0
2	0
3	0.52
4	0.89
5	1.12
6	1.26
7	1.36
8	1.41
9	1.46
10	1.49
11	1.52
12	1.54
13	1.56
14	1.58
15	1.59
16	1.5943
17	1.6064
18	1.6133
19	1.6207
20	1.6292
21	1.6385
22	1.6403
23	1.6462
24	1.6497
25	1.6556
26	1.6587
27	1.6631
28	1.667
29	1.6693
30	1.6724

In [20]:

```
# 计算CR
RI = 0.90
CR = CI/RI
CR <0.1
```

Out[20]:

```
True
```

所以判断矩阵的不一致程度在容许范围之内，有满意的一致性，通过一致性检验。

结果呈现

评判维度

1.关键行为分：根据关键行为占比的大小，

2.变现能力分：使用时长、广告展现次数、启动次数、PV的权重评分

3.用户构成：老用户占比评分

4.归因得分：根据模糊归因占比、归因失败占比评分（权重0.5：0.5）

5.数量分：按照数量多少打分

In [21]:

```
# 读取数据
score_data = pd.read_excel('/home/mw/input/channel_quality3966/score_data.xlsx'
score_data
```

Out[21]:

	channel	new_user_rate	key_action_rate	success_attribute_rate	accurate_attribute_rate	total_u
0	huawei	0.80	0.5	0.95	0.75	15000
1	xiaomi	0.80	0.3	0.92	0.58	12300
2	oppo	0.93	0.2	0.94	0.64	14580
3	vivo	0.60	0.1	0.91	0.75	8525
4	meizu	0.70	0.2	0.85	0.80	1202

In [22]:

```
# 构造打分公式
def score(column):
    score = (column-column.min())/(column.max()-column.min())*100
    return score

# 对每列进行打分
score_data[['new_user_rate','key_action_rate','success_attribute_rate','accurat
score_data
```

Out[22]:

	channel	new_user_rate	key_action_rate	success_attribute_rate	accurate_attribute_rate	total_u
0	huawei	60	100	100	77	100
1	xiaomi	60	49	70	0	80
2	oppo	100	25	89	27	96
3	vivo	0	0	60	77	53
4	meizu	30	25	0	100	0

In [23]:

```
score_data_final = pd.DataFrame()

# 关键行为分
score_data_final[['channel','key_action_score']] = score_data[['channel','key_a

# 变现能力分
score_data_final['revenue_ability_score'] = score_data['duration_avg']*0.54+sco

# 用户构成分
score_data_final['user_compose_score'] = score_data['new_user_rate']

# 归因得分
score_data_final['contribute_score'] = score_data['success_attribute_rate']*0.5

# 数量分
score_data_final['user_count_score'] = score_data['total_user']

# 得分表
score_data_final
```

Out[23]:

	channel	key_action_score	revenue_ability_score	user_compose_score	contribute_score	user_
0	huawei	100	16.99	60	88.5	100
1	xiaomi	49	19.47	60	35.0	80
2	oppo	25	35.90	100	58.0	96
3	vivo	0	96.32	0	68.5	53
4	meizu	25	2.97	30	50.0	0

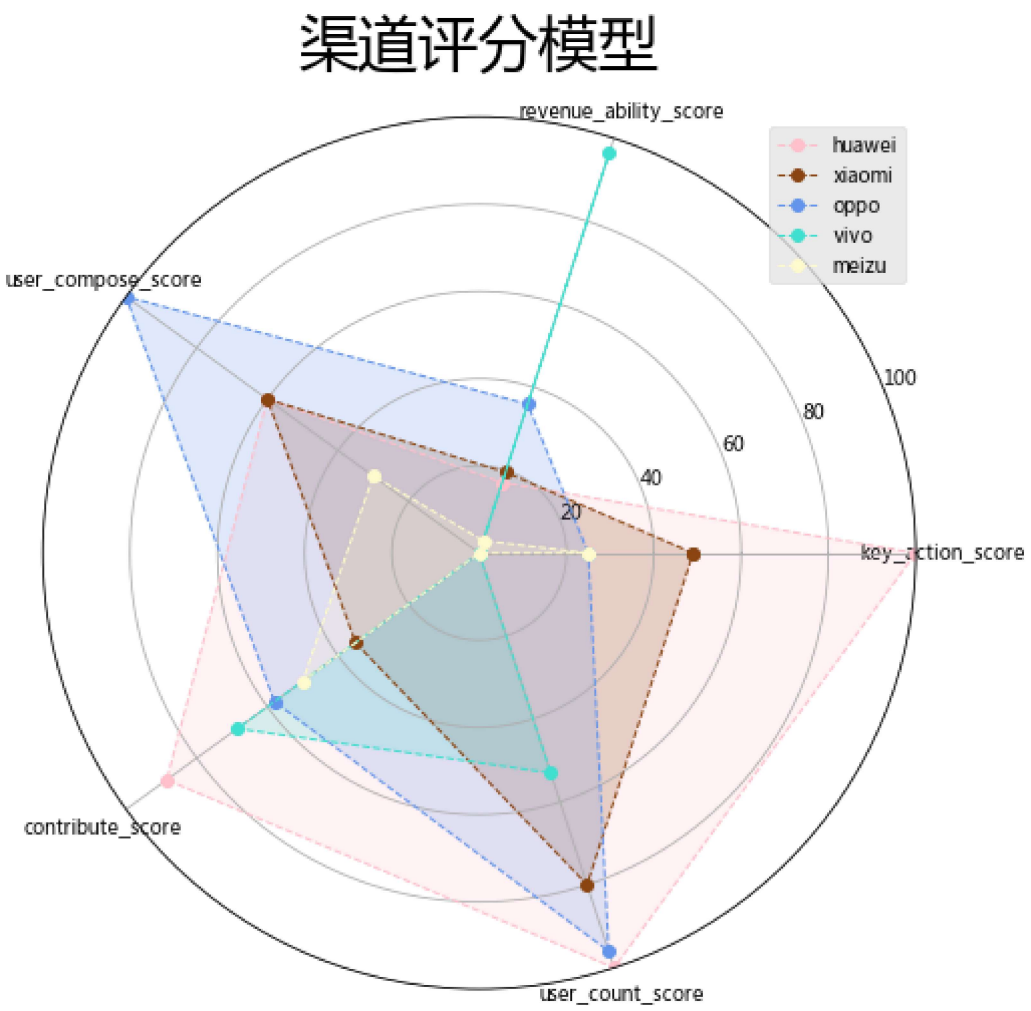
In [24]:

```
data = [100,16.99,60,88.5,100]
```

In [25]:

```
# 画布大小
plt.figure(figsize=(8,8))
# 设置颜色
colors = ['#FFC0CB','#8B4513','#6495ED','#40E0D0','#FFFACD']
# 标签
labels = np.array(['key_action_score','revenue_ability_score','user_compose_sco
# 分割圆
angles = np.linspace(0,2*np.pi,len(labels),endpoint=False)
# 闭合
angles = np.concatenate((angles,[angles[0]]))
# 设置循环绘图
for i in range(len(score_data_final['channel'])):
    # 闭合
    data = np.concatenate((score_data_final.loc[i,['key_action_score','revenue_
    # 绘图
    plt.polar(angles,data,'o--',color=colors[i],linewidth=1)
    # 填充
    plt.fill(angles, data, facecolor=colors[i], alpha=0.2)
# 做标签
plt.thetagrids(angles * 180/np.pi, labels,size=10)
# 设置刻度
plt.ylim(0,100)
# 设置背景
plt.style.use('ggplot')
# 标题
plt.title('渠道评分模型',size=30)
# 图例
plt.legend(score_data_final['channel'])

plt.show()
```


渠道评分模型
revenue_ability_score
user_compose_score
key_action_score
contribute_score
user_count_score
huawei
xiaomi
oppo
vivo
meizu
20
40
60
80
100

In [ ]: