基于BKVision图表平台实现用户画像与行为分析

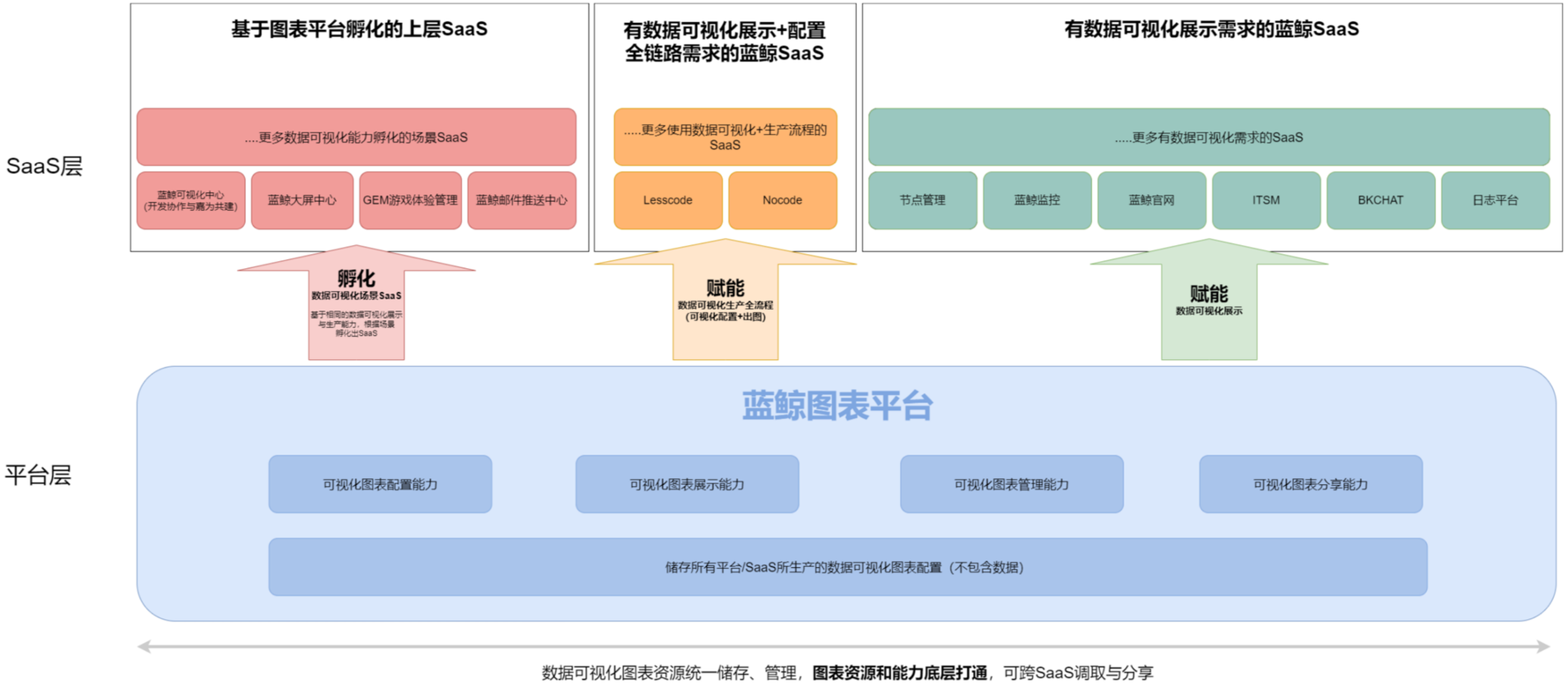
蓝鲸智云 Tencent 腾讯

目录

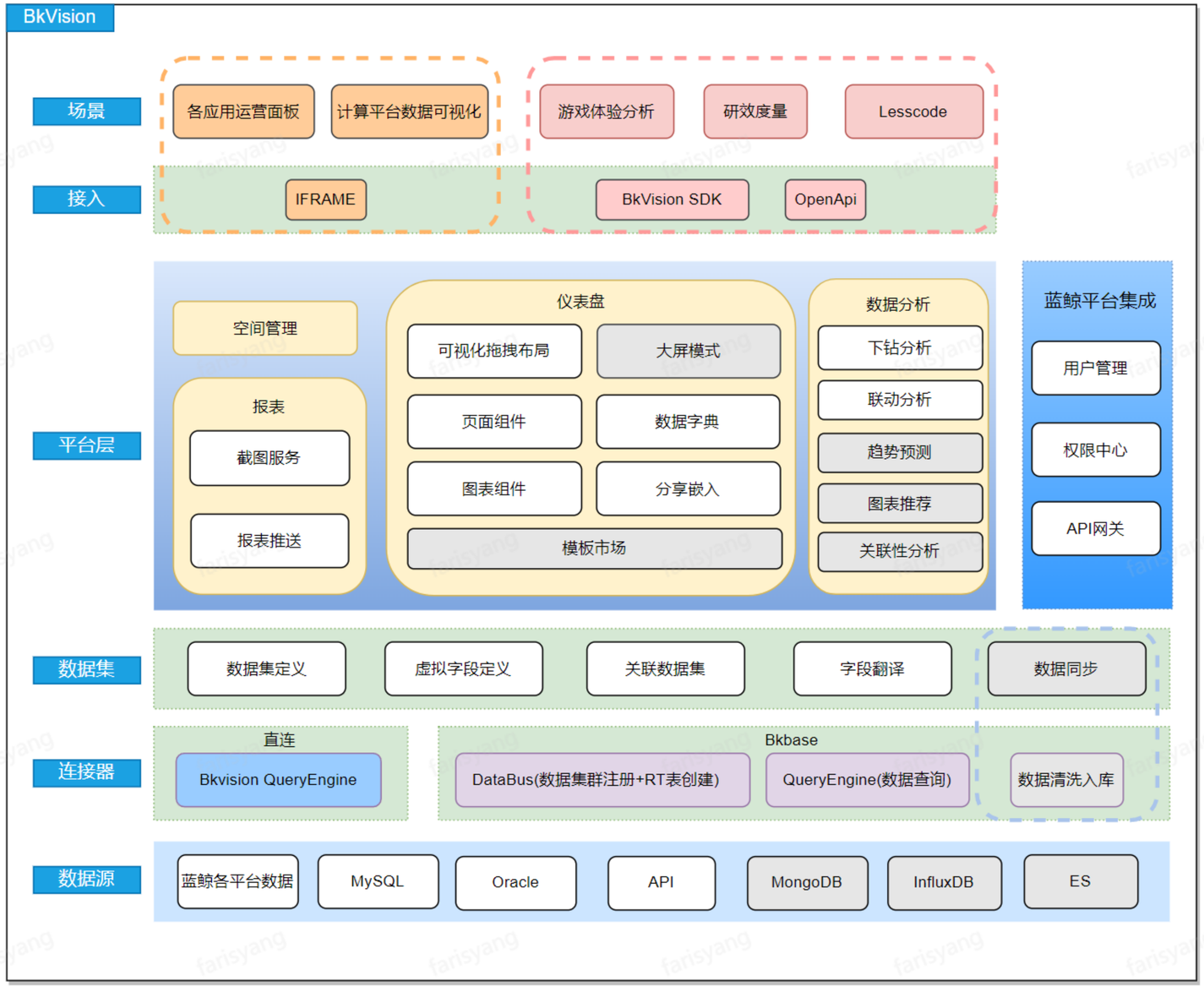
蓝鲸图表平台BKVision是什么

BKVision是蓝鲸推出的一款通用的图表可视化平台。

采用插件机制，以及强大的嵌入SDK，满足BI（ Business Intelligence ）、大屏监控等各种领域的数据分析展示。



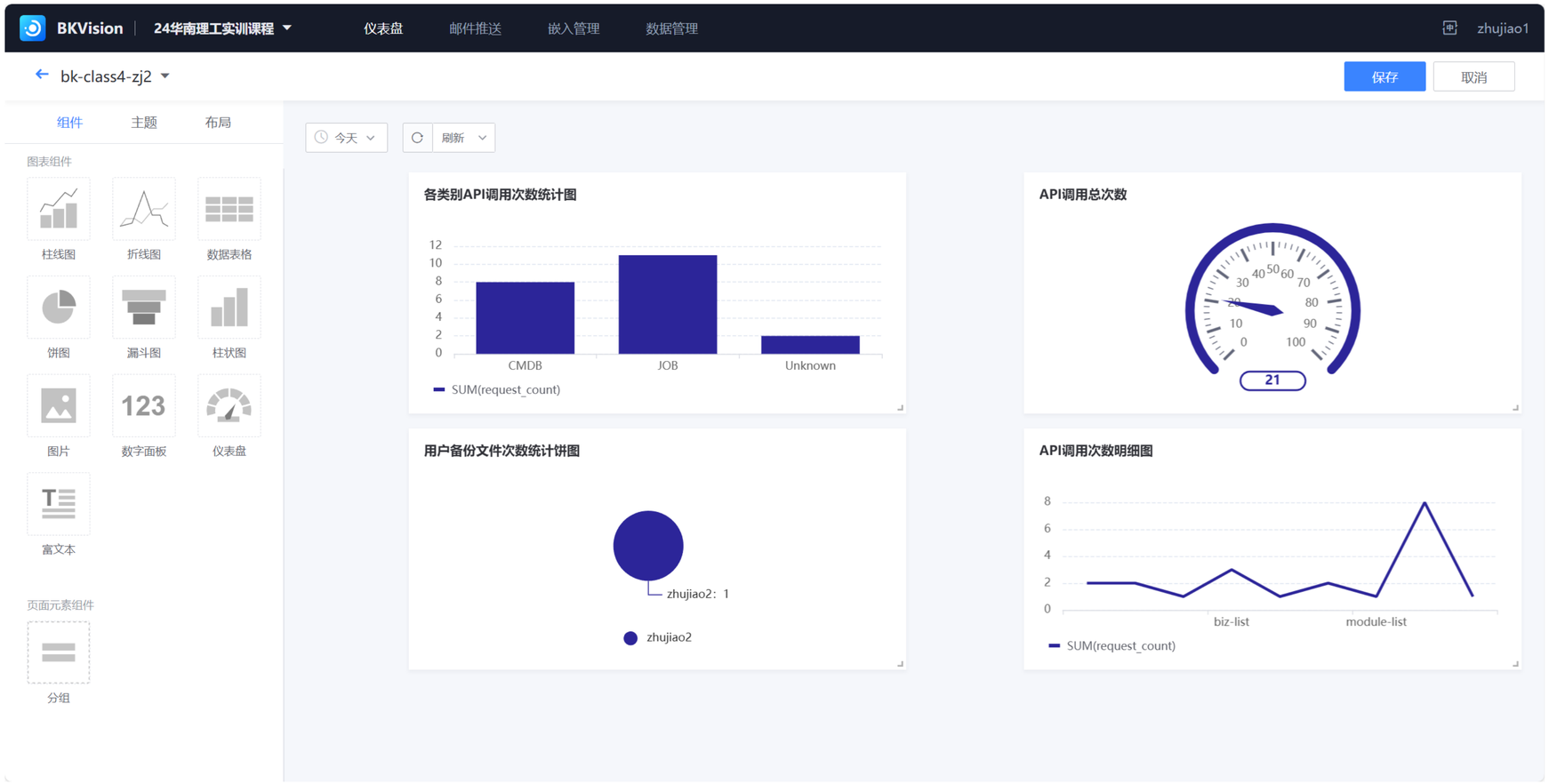
BKVision功能模块



BKVision功能模块

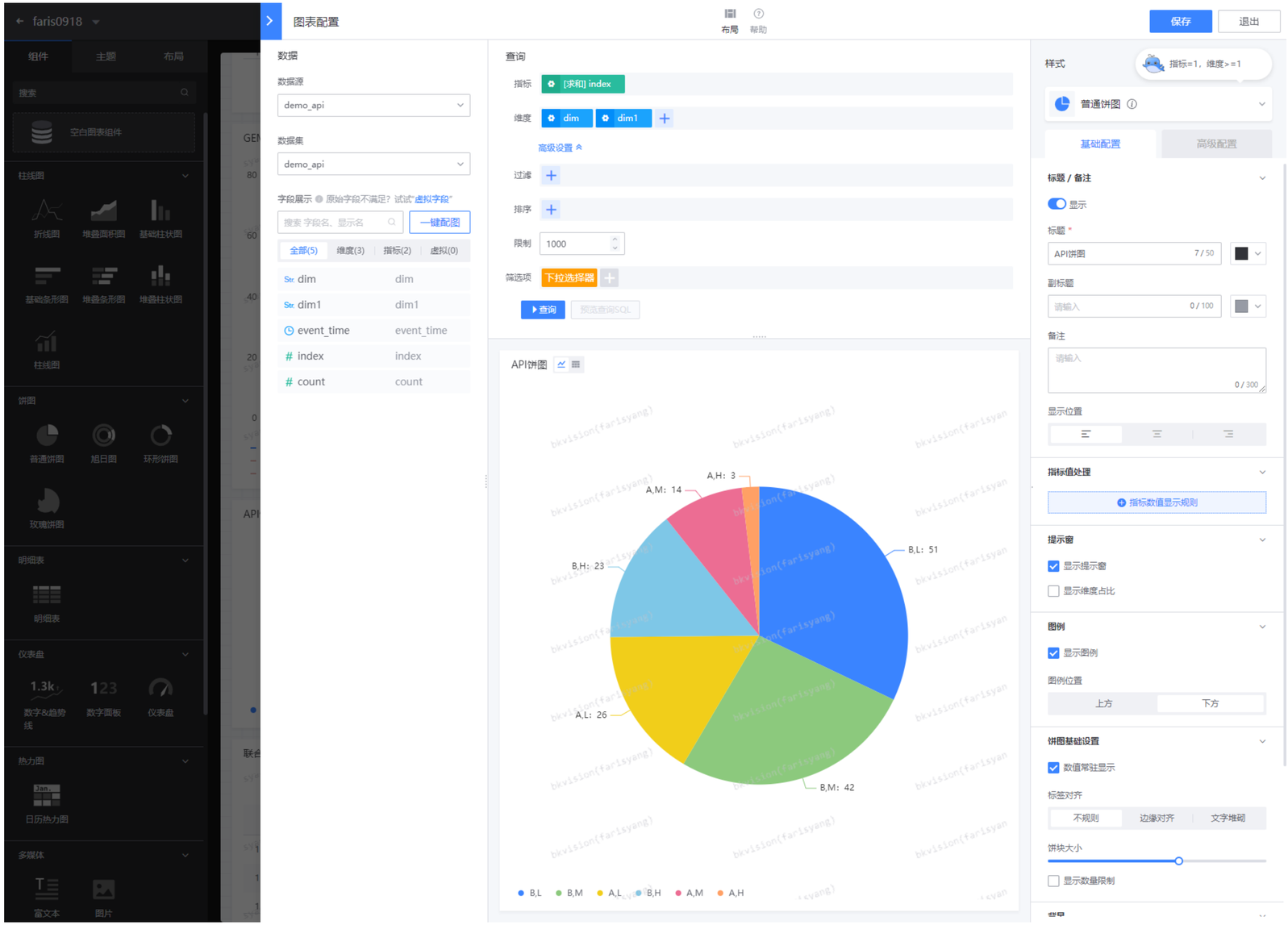
导航栏:

仪表盘工作台:



BKVision功能模块

图表配置:



什么是指标和维度?

指标

用来衡量某个事物的数量或质量的标准，通常是一个数值或一个比率。

例：衡量某个业务或产品的表现，指标有销售额、用户数量、转化率等。

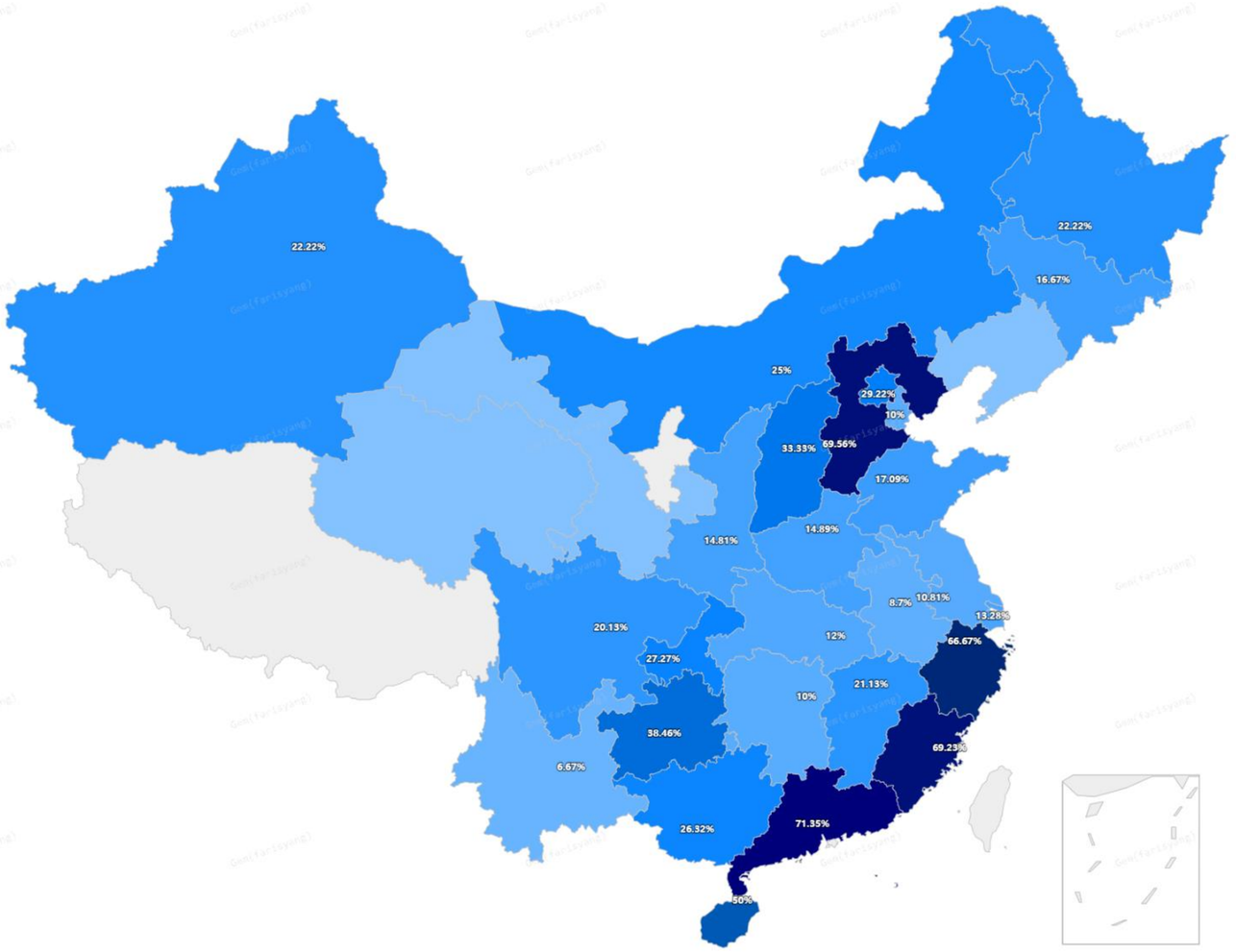
维度

用来描述指标的特征或属性的，通常是一个分类变量，对指标进行分类或分组，以便更好地理解和分析数据

例：对于销售额这个指标，可以按照时间、地区、产品类型等维度进行分类或分组

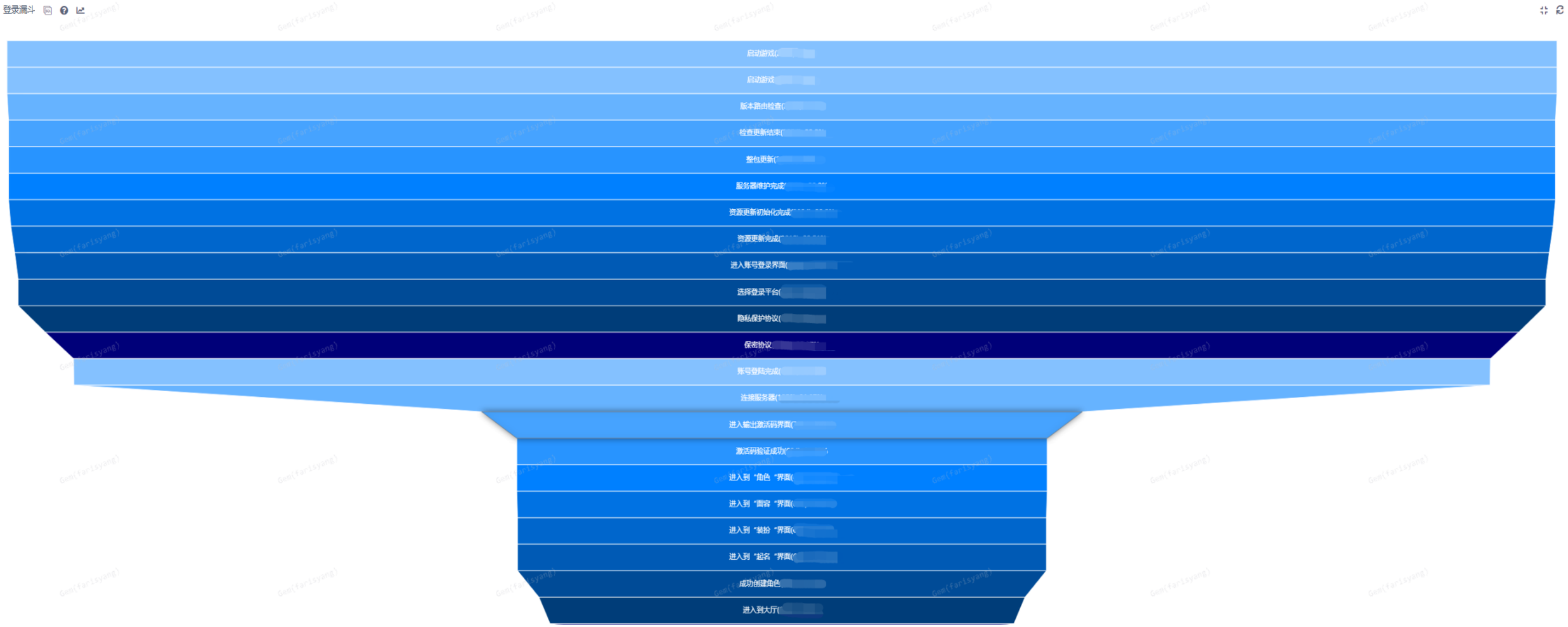
业务图表

登录成功率分布:

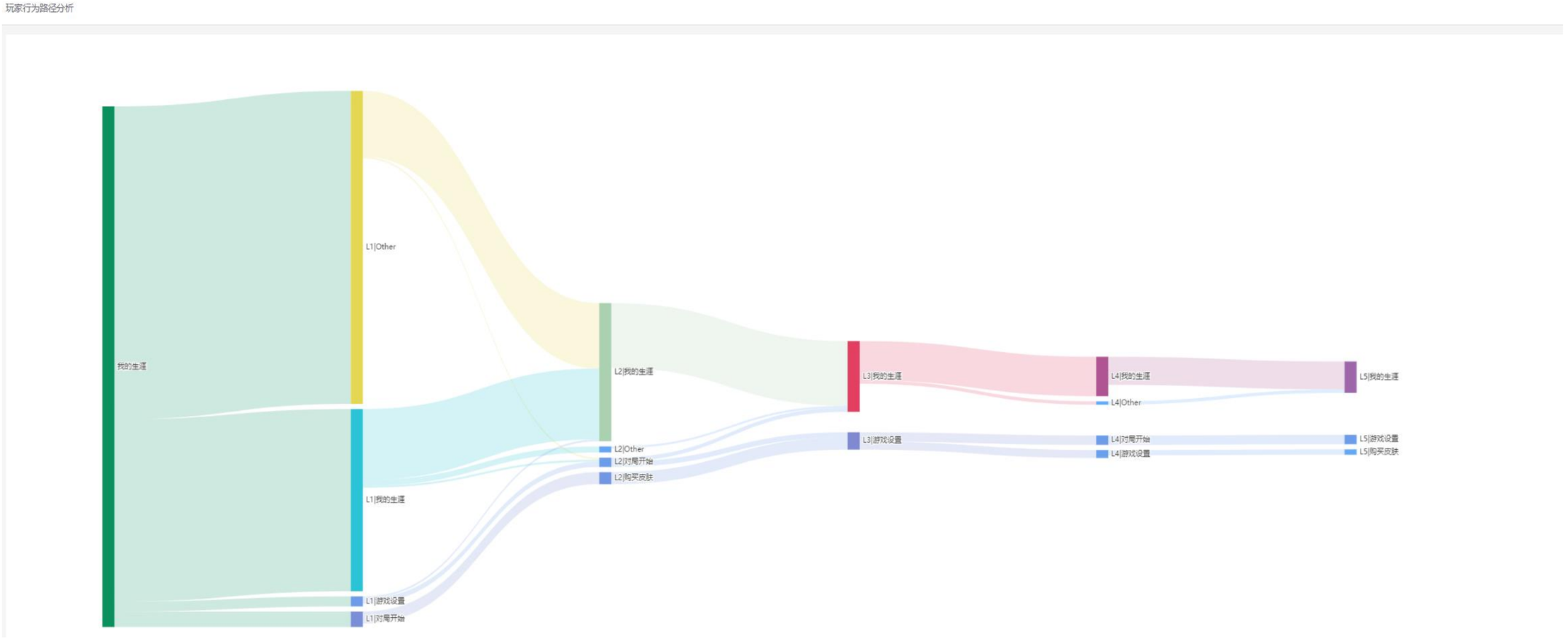


业务图表

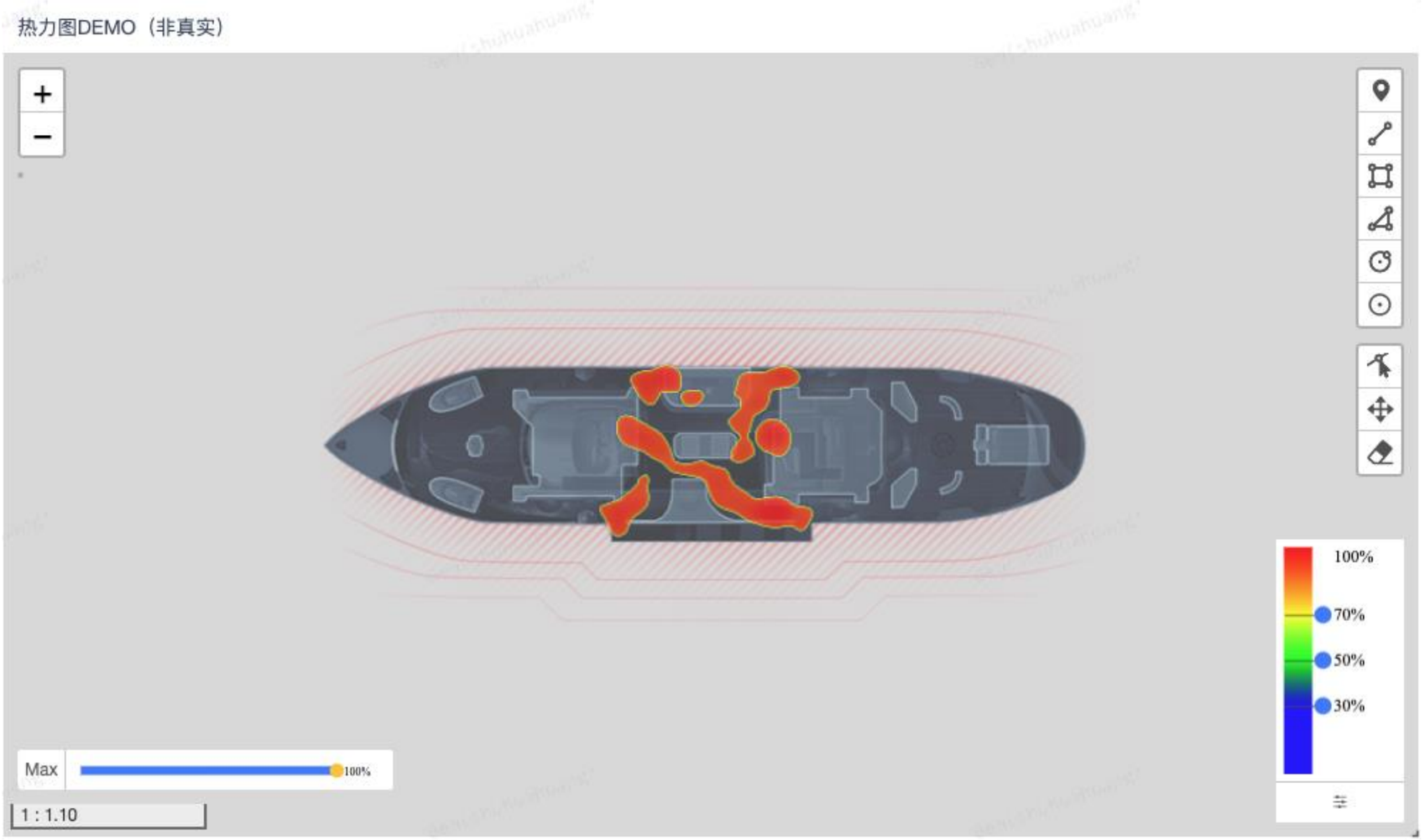
登录步骤转化率



玩家行为路径图


玩家行为路径分析
我的生涯
L1|Other
L1|我的生涯
L1|游戏设置
L1|对局开始
L2|我的生涯
L2|Other
L2|对局开始
L2|购买皮肤
L3|我的生涯
L3|游戏设置
L4|我的生涯
L4|Other
L4|对局开始
L4|游戏设置
L5|我的生涯
L5|游戏设置
L5|购买皮肤

游戏地图玩家热力图



目录

游戏数据分析步骤

1. **数据采集**：收集游戏中的各种数据，如玩家行为数据、游戏事件数据、游戏性能数据等。
2. **数据清洗**：对采集到的数据进行清洗和处理，去除无用的数据和错误的数据，以便更好地进行分析。
3. **数据整合**：将清洗后的数据整合到一个数据集中，以便更好地进行分析。
4. **数据分析**：对整合后的数据进行分析，探索数据中的规律和趋势，如玩家行为、游戏流程、游戏难度等。
5. **数据建模**：根据分析结果，建立相应的数据模型，以便更好地预测和解释数据。
6. **数据可视化**：将分析结果可视化，以便更好地展示数据和分析结果，如图表、地图、仪表盘等。

游戏数据分析基础能力：SQL

SQL (Structured Query Language结构化查询语言)：

用于在关系数据库中存储和处理信息。关系数据库以表格形式存储信息，行和列分别表示不同的数据属性和数据值之间的各种关系。您可以使用 SQL 语句从数据库中存储、更新、删除、搜索和检索信息。您还可以使用 SQL 来维护和优化数据库性能。

这类数据库包括：MySQL、SQL Server、Access、Oracle、Sybase、DB2 等等。

SQL可以做什么：

- 面向数据库执行查询
- 可从数据库取回数据
- 可在数据库中插入新的记录
- 可更新数据库中的数据
- 可从数据库删除记录
- 可创建新数据库
- 可在数据库中创建新表
- 可在数据库中创建存储过程
- 可在数据库中创建视图
- 可以设置表、存储过程和视图的权限

Declarative：Say **what** you want, not **how** to get it
声明式的语句

关系型数据库概念

- **Database**: Set of **Relations**
- **Relation** (**Table**):
  - **Schema** (description)
  - **Instance** (data satisfying the schema)
- **Attribute** (**Column**)
- **Tuple** (**Record**, **Row**)

- **Schema is fixed:**
  - **attribute names, atomic types**

  Eg. students(name text, gpa float, dept text)
- **Instance can change**
  - **a multiset of “rows”**
  - **{(‘Bob Snob’, 3.3, 'CS'),**
    (‘Bob Snob’, 3.3, 'CS'),
    (‘Mary Contrary’, 3.8, 'CS')}

SQL的DDL

DDL(Data Definition Language)

```
CREATE TABLE table_name
( { column_name data_type [ DEFAULT default_expr ]
[ column_constraint[, ... ] ] | table_constraint } [, ... ] )
```

Data Types (MySQL) include:

character(n) – fixed-length character string
character varying(n) – variable-length character string
binary(n), text(n), blob, mediumblob, mediumtext,
smallint, integer, bigint, numeric, real, double precision
date, time, timestamp, …

https://dev.mysql.com/doc/refman/5.7/en/data-types.html

创建数据表示例：

```
CREATE TABLE Persons
(
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

SQL的DML

DML(Data Manipulation Language)
INSERT 插入语句

- **INSERT** INTO <relation> values (.., .., ...)
- **INSERT** INTO <relation>(att1, .., attn) values(..., ..., ...)
- **INSERT** INTO <relation> <query expression>

插入数据示例：

INSERT INTO Persons VALUES (‘Gates’ , ‘Bill’ , ‘Xuanwumen 10’ , ‘Beijing’);
INSERT INTO Persons VALUES (‘Carter’ , ‘Thomas’ , ‘Changan Street’ , ‘Beijing’);
INSERT INTO Persons (LastName, Address) VALUES ('Wilson', 'Champs-Elysees') ;

LastName	FirstName	Address	City
Carter	Thomas	Changan Street	Beijing
Gates	Bill	Xuanwumen 10	Beijing
Wilson		Champs-Elysees	

SQL的DML

DML(Data Manipulation Language)
UPDATE 更新语句

UPDATE <relation>
SET <attribute> = <expression>
WHERE <predicate>

更新数据示例：

UPDATE Person SET FirstName = 'Fred' WHERE LastName = 'Wilson' ;

LastName	FirstName	Address	City
Carter	Thomas	Changan Street	Beijing
Gates	Bill	Xuanwumen 10	Beijing
Wilson	Fred	Champs-Elysees	

SQL的DML

DML(Data Manipulation Language)
DELETE 删除语句

DELETE FROM <relation>
[WHERE <predicate>]

删除数据示例：

DELETE FROM Person WHERE LastName = 'Wilson' ;

LastName	FirstName	Address	City
Carter	Thomas	Changan Street	Beijing
Gates	Bill	Xuanwumen 10	Beijing
~~Wilson~~		~~Champs-Elysees~~	

SQL的DML

DML(Data Manipulation Language)
SELECT 查询语句

SELECT [DISTINCT] <column expression list>
FROM <single table>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]
[**ORDER BY** <column list>] ;

查询数据示例： **SELECT LastName,FirstName FROM Persons**

LastName	FirstName
Adams	John
Bush	George
Carter	Thomas

SELECT查询：DISTINCT

DML(Data Manipulation Language)
SELECT 查询语句

SELECT [DISTINCT] <column expression list>
FROM <single table>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]
[**ORDER BY** <column list>] ;

查询去重数据示例： **SELECT DISTINCT Company FROM Orders;**

原始Orders数据表：

Company	OrderNumber
IBM	3532
W3School	2356
Apple	4698
W3School	6953

查询结果：

Company
IBM
W3School
Apple

SELECT查询：WHERE

DML(Data Manipulation Language)
SELECT 查询语句

SELECT [DISTINCT] <column expression list>
FROM <single table>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]
[**ORDER BY** <column list>] ;

查询过滤数据示例：

选取居住在城市Beijing中的人
SELECT * FROM Persons WHERE City='Beijing'

原始Persons数据表：

LastName	FirstName	Address	City	Year
Adams	John	Oxford Street	London	1970
Bush	George	Fifth Avenue	New York	1975
Carter	Thomas	Changan Street	Beijing	1980
Gates	Bill	Xuanwumen 10	Beijing	1985

查询结果：

LastName	FirstName	Address	City	Year
Carter	Thomas	Changan Street	Beijing	1980
Gates	Bill	Xuanwumen 10	Beijing	1985

SELECT查询：WHERE

WHERE 子句中的操作符：

操作符	描述
=	等于
<>	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于
BETWEEN	在某个范围内
LIKE	搜索某种模式

SELECT查询：WHERE

Persons原始数据

LastName	FirstName	Address	City
Adams	John	Oxford Street	London
Bush	George	Fifth Avenue	New York
Carter	Thomas	Changan Street	Beijing
Carter	William	Xuanwumen 10	Beijing

多条件过滤查询： AND 和 OR 可在 WHERE 子语句中把两个或多个条件结合起来。

- 显示所有姓为 "Carter" 并且名为 "Thomas" 的人
- SELECT * FROM Persons WHERE FirstName='Thomas' **AND** LastName='Carter'

LastName	FirstName	Address	City
Carter	Thomas	Changan Street	Beijing

- 显示所有姓为 "Carter" 或者名为 "Thomas" 的人
- SELECT * FROM Persons WHERE firstname='Thomas' **OR** lastname='Carter'

LastName	FirstName	Address	City
Carter	Thomas	Changan Street	Beijing
Carter	William	Xuanwumen 10	Beijing

SELECT查询：ORDER BY

DML(Data Manipulation Language)
SELECT 查询语句

SELECT [DISTINCT] <column expression list>
FROM <single table>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]
[**ORDER BY** <column list>] ;

查询时排序数据示例：

升序ASC(默认)，降序DESC
查询按LastName升序的人员
SELECT * FROM Persons ORDER BY LastName;

LastName	FirstName	Address	City	Year
Adams	John	Oxford Street	London	1970
Bush	George	Fifth Avenue	New York	1975
Carter	Thomas	Changan Street	Beijing	1980
Gates	Bill	Xuanwumen 10	Beijing	1985

SELECT查询：GROUP BY

DML(Data Manipulation Language)
SELECT 查询语句

SELECT [DISTINCT] <column expression list>
FROM <single table>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]
[**ORDER BY** <column list>] ;

GROUP BY用于根据一个或多个列对结果集进行分组，并对每组进行聚合函数的计算。

常用的聚合函数：

- **COUNT**：用于计算行数，可以用于计算某个列或整个表中的行数。
- **SUM**：用于计算某个列或整个表中数值型数据的总和。
- **AVG**：用于计算某个列或整个表中数值型数据的平均值。
- **MAX**：用于计算某个列或整个表中数值型数据的最大值。
- **MIN**：用于计算某个列或整个表中数值型数据的最小值。

SELECT查询：GROUP BY

Orders数据表：

O_Id	OrderDate	OrderPrice	Customer
1	2008/12/29	1000	Bush
2	2008/11/23	1600	Carter
3	2008/10/05	700	Bush
4	2008/09/28	300	Bush
5	2008/08/06	2000	Adams
6	2008/07/21	100	Carter

查找订单总金额少于 2000 的客户：

```
SELECT Customer, SUM(OrderPrice)
FROM Orders
GROUP BY Customer
HAVING SUM(OrderPrice)<2000
```

Customer	SUM(OrderPrice)
Carter	1700

查找客户 “Bush” 或 “Adams” 拥有超过 1500 的订单总金额：

```
SELECT Customer, SUM(OrderPrice)
FROM Orders
WHERE Customer='Bush' OR Customer='Adams'
GROUP BY Customer
HAVING SUM(OrderPrice)>1500
```

Customer	SUM(OrderPrice)
Bush	2000
Adams	2000

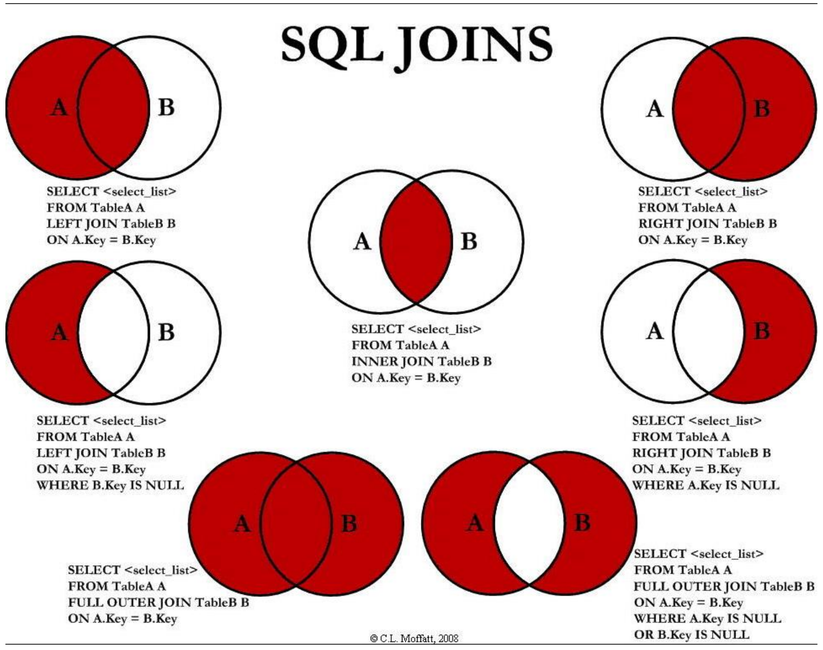
联表查询：JOIN

JOIN语句：用于根据两个或多个表中的列之间的关系，从这些表中查询数据。

语法：

```
SELECT (column_list)
FROM  table_name1
  [INNER | NATURAL | {LEFT | RIGHT | FULL} | {OUTER}]  JOIN table_name2
     ON qualification_list
[WHERE <predicate>]
[GROUP BY <column list> [HAVING <predicate>]
[ORDER BY <column list>] ;
```

JOIN类型：



- **INNER JOIN**：如果表中有至少一个匹配，则返回行
- **LEFT JOIN**：即使右表中没有匹配，也从左表返回所有的行
- **RIGHT JOIN**：即使左表中没有匹配，也从右表返回所有的行
- **FULL JOIN**：只要其中一个表中存在匹配，则返回行
- **OUTER JOIN**：不匹配连接条件的记录也将返回行，即两个表的并集

联表查询：JOIN

Persons数据表：

Id_P	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

Orders数据表：

Id_O	OrderNo	Id_P
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	65

列出所有人的订购详情，并根据LastName排序：

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.Id_P = Orders.Id_P
ORDER BY Persons.LastName
```

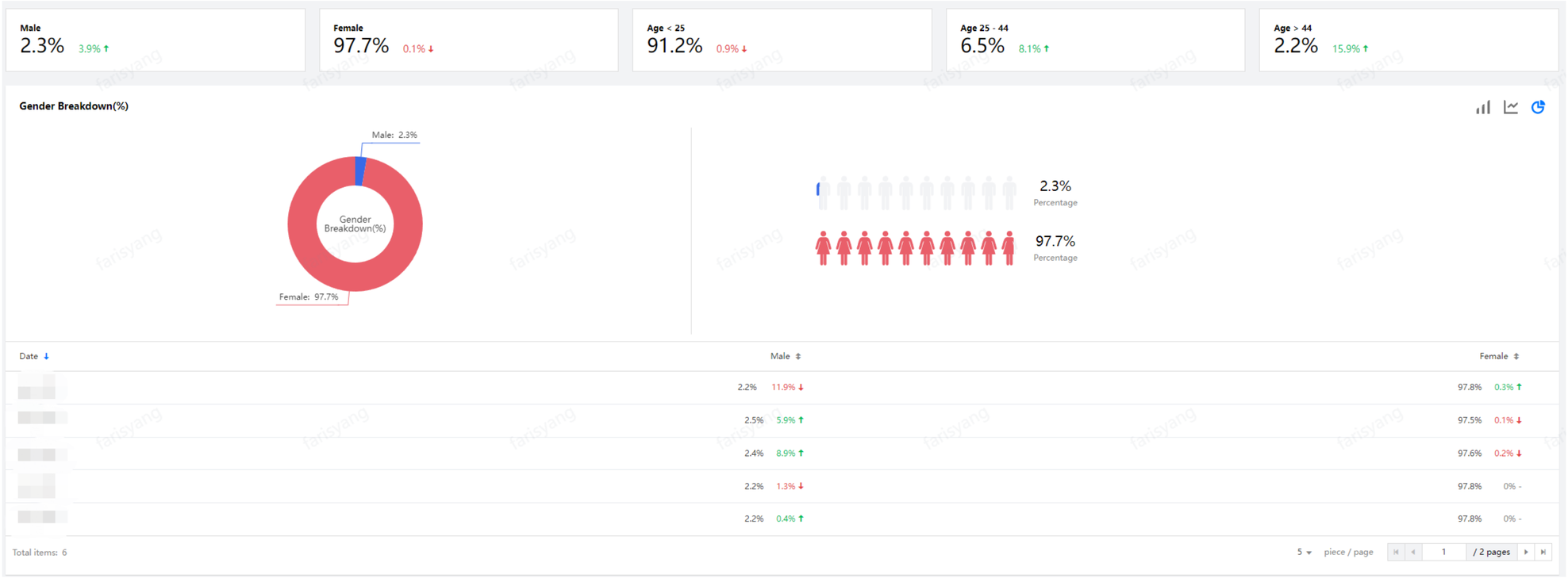
LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678

目录

游戏用户画像是什么?

游戏用户画像是指对游戏玩家进行分析和描述，以便更好地了解他们的特征和行为，从而为游戏的优化和改进提供支持和指导。

常见用户画像图表：



游戏用户画像

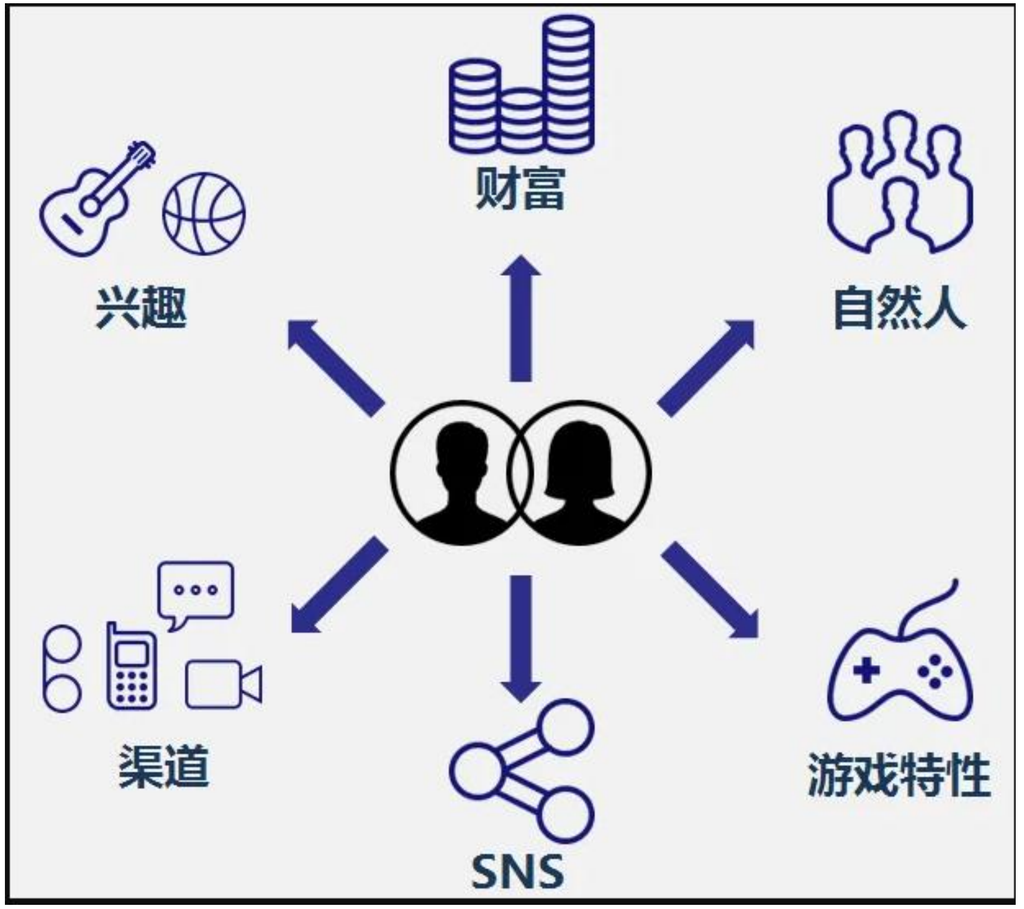
游戏用户画像由三大逻辑部分组成：**账号体系**，**画像数据**，**标签体系**。

一、 账号体系： 账号体系用于将不同系统下的数据关联到同一个ID

游戏用户画像

游戏用户画像由三大逻辑部分组成：**账号体系**，**画像数据**，**标签体系**。

二、画像数据： 有**6**大类画像数据



画像数据

1. 自然人属性:

- 性别
- 所处的地理位置（国家，省份，城市）
- 使用终端的配置（移动终端屏幕大小，操作系统，CPU频率，核数等；台式机CPU频数，核数，内存等）

2. 财富:

- 充值（次数，额度，渠道，时段等）
- 付费（从永久道具，消耗道具两个纬度看购买的次数，额度，时段，趋势等）
- 存量（已购买的道具分布及趋势）
- 已经有的会员分布及其趋势

3. 渠道:

- 公司外主流的**游戏门户**（小米/360/google play /应用商店等），公司内各**应用**渠道
- 每个渠道的发送次数，频率，到达时间，点击次数，点击时间
  对这些数据的趋势进行分析统计，最后排序，得到玩家的**渠道偏好**

画像数据

4. 兴趣：

- **游戏兴趣**：统计玩家在手/端/页 游内的付费，时长，等级，好友数等，进而去分析玩家是沉浸游戏中，兴趣消退，还是维持状态。
- **道具兴趣**：统计玩家倾向购买的道具价格，色调，功能特征，统计分布得到玩家对道具的偏好。
- **营销活动兴趣**：统计玩家参与不同类型营销活动的频度，并分析趋势与偏好。

5. SNS(Social Networking Services)：

- **个体方面**: 分析每个玩家与其它玩家的交互，包括邀请，被邀请，聊天，一起游戏等。
- **群体方面**: 分析群体的拓扑结构，节点数，边数等等。

例如：LOL / 王者荣耀里的大神玩家局内**5杀**，再加上酷炫英雄皮肤。会影响其他玩家的行为偏好。

画像数据

6. 游戏行为:

由于游戏多样性，会分为通用，特性和大盘三大部分

通用部分:

- 玩家的经验及其变化趋势，好友数
- 当前道具统计（付费道具，游戏币道具，时限道具及其到期天数）
- 任务执行状态（接受任务数，完成任务数，当前正在执行的任务）
- 登录的时长，次数，还有时段。

结合算法，就可以提炼玩家在游戏中的**通用行为模式**，例如是有钱有闲的土豪玩家，还是没钱有闲的屌丝玩家。

特性部分:

游戏**特有的玩法或特性**，譬如酷跑的最远距离， LOL的承受伤害，输出伤害，出装分布等。

大盘部分:

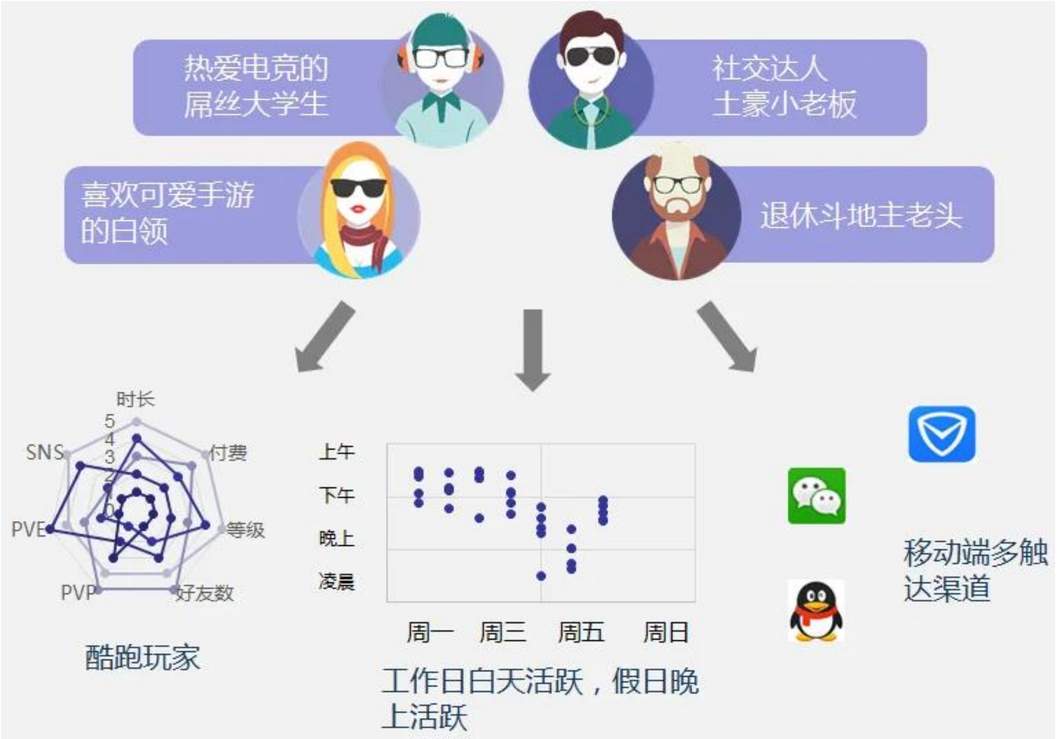
玩家在**所有手/端/页游中**的 游戏时长，付费，等级与经验。这些属性可以应用到**拉新模型的预测**中去。

游戏用户画像

游戏用户画像由三大逻辑部分组成：**账号体系**，**画像数据**，**标签体系**。

三、 标签体系：

用户画像会结合**业务逻辑**对数据的**含义**进行提炼，将数据转化成人可以理解的标签。
基于画像数据，再结合业务逻辑，我们可以为玩家打上各种各样的标签。
如：土豪小老板，屌丝大学生，文艺白领等。



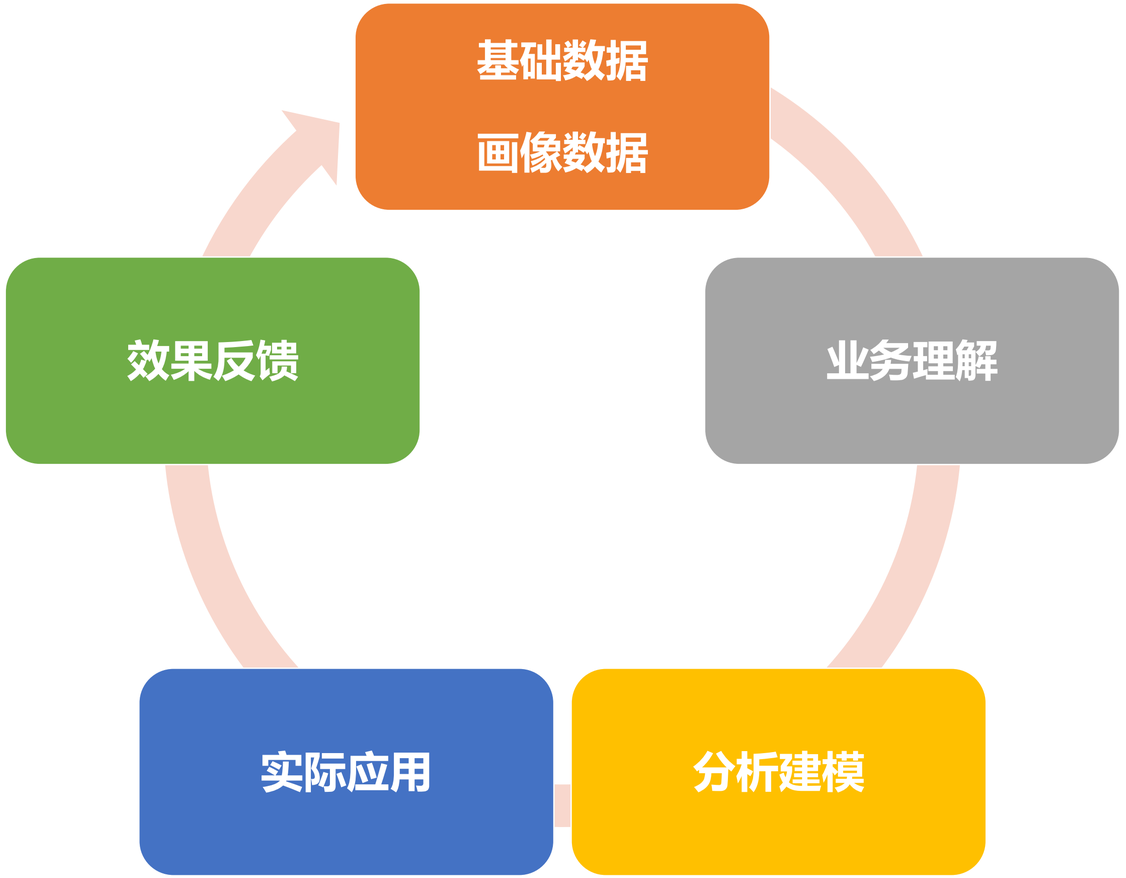
精细化的游戏运营：

- 千人千面的游戏商城
- 萝卜白菜各有所爱的阿卡丽神秘商店
- 还有感受温情的生日营销。

游戏用户画像迭代

形成一个动态的**闭环**，画像系统不断把**优质的数据**添加到画像中去

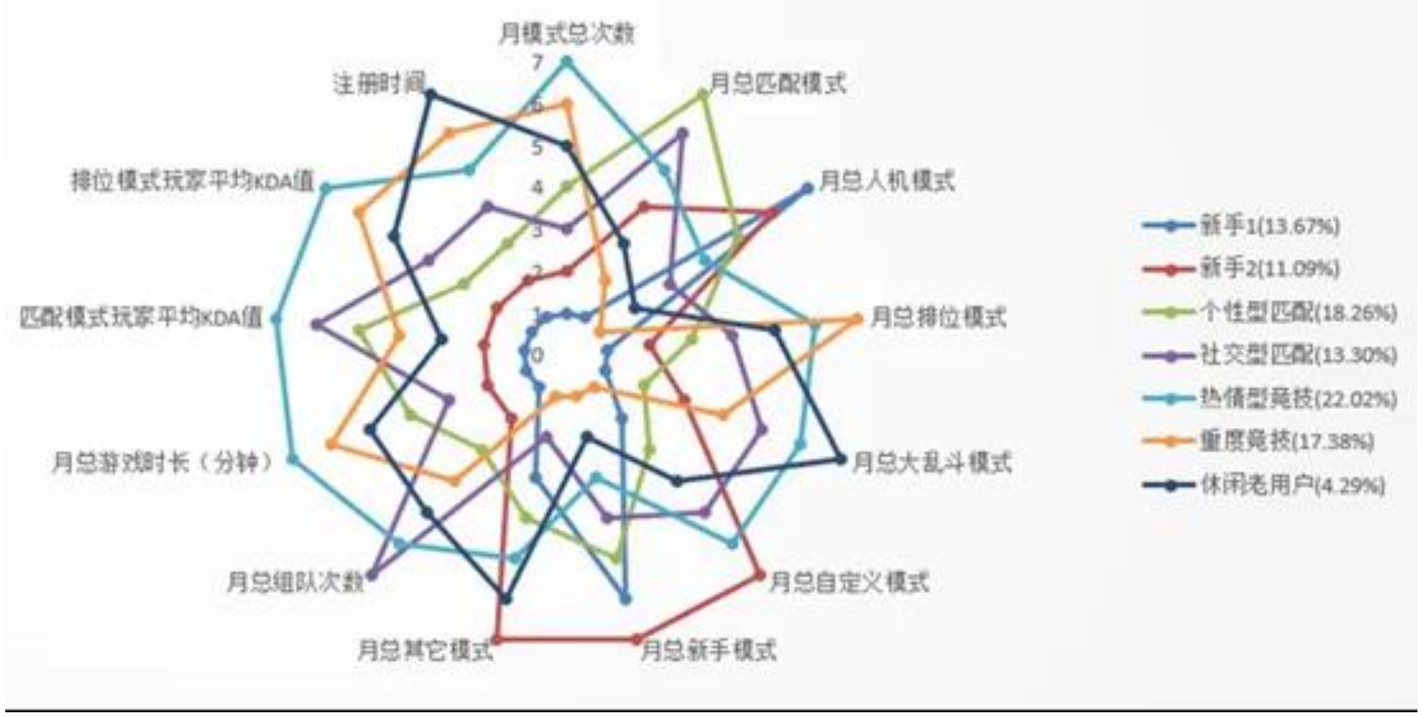
画像纬度将会越来越细、画像的使用方法也能集众家之长



游戏用户画像算法简介

算法从应用的角度可以分成两大类，一类是用于**结构化数据**的，另一类则是用于**非结构化数据**的。

结构化数据：占大部分场景，主要用到的是kmeans聚类算法，标签传播（LPA）算法



游戏用户画像算法简介

非结构化数据：主要用于文字和图片特征的提取

1. 道具文字描述相似度：

游戏道具有很多**描述**，根据contented-base的思想，玩家偏向购买相似的道具。
把道具的描述信息提取出来，通过**词频矩阵**，**计算相似度**。

2. 道具/皮肤颜色相似度：

把图片转换到HSV空间（比RGB空间更符合人眼对色彩的辨识），再**计算相似度**。

3. 道具形状相似度：

通过SIFT算法提取图片中的形状相关的**特征矢量**（角点、边缘点、暗区的亮点及亮区的暗点等）
再计算图片间形状矢量的**相似度**。

游戏用户画像算法简介

相似度高的非结构化数据：

项羽 海滩派对

盲僧 李青 泳池派对

魔法少女金克斯&蔷薇恋人孙尚香

夺命前锋卢锡安&激情绿茵马可波罗

蓝鲸智云

官方微信公众号

蓝鲸高校培训群

https://bk.tencent.com