淘宝双11数据分析与预测

笔记本： 实战项目
创建时间： 2019/12/19 18:41 **更新时间：** 2019/12/19 20:10
作者： 1137658346@qq.com
URL： http://dblab.xmu.edu.cn/post/8116/#banbenlishi

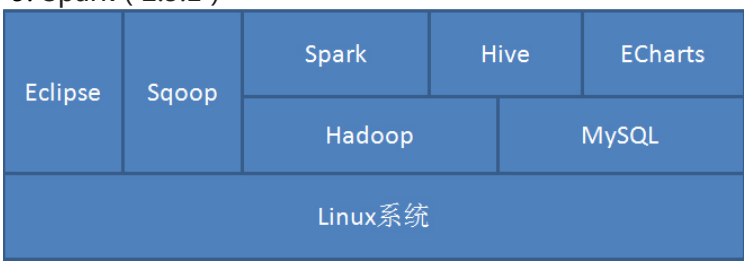
淘宝双11数据分析与预测

准备工作：

软件工具

本案例所涉及的系统及软件：

1. Linux系统（CENTOS 7）
2. MySQL
3. Tomcat（7.0.9）
4. Hadoop（3.2.0）
5. Hive（2.3.5）
6. Sqoop（1.4.6）
7. ECharts（4.5.0）
8. Idea（2019.1.3）
9. Spark（2.3.1）



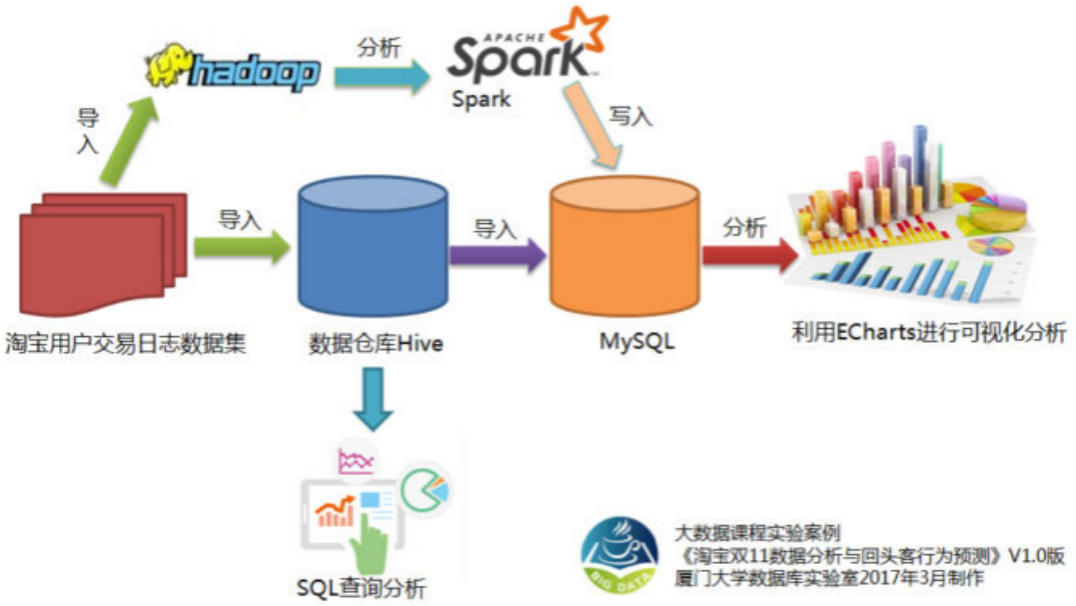
数据集

淘宝购物行为数据集 (5000万条记录，数据有偏移，不是真实的淘宝购物交易数据，但是不影响学习)

案例任务

1. 安装Linux操作系统
2. 安装关系型数据库MySQL
3. 安装大数据处理框架Hadoop
4. 安装数据仓库Hive
5. 安装Sqoop
6. Windows下安装IDEA
7. 安装 Spark

8. 对文本文件形式的原始数据集进行预处理
9. 把文本文件的数据集导入到数据仓库Hive中
10. 对数据仓库Hive中的数据进行查询分析
11. 使用Sqoop将数据从Hive导入MySQL
12. 利用Windows下IDEA搭建动态Web应用
13. 利用ECharts进行前端可视化分析
14. 利用Spark MLlib进行回头客行为预测



步骤一：本地数据集上传到数据仓库Hive

任务清单

1. **安装Linux系统**
2. **数据集下载与查看**
3. **数据集预处理（ 点击这里下载data_format.zip数据集 ）**
4. **把数据集导入分布式文件系统HDFS中**
5. **在数据仓库Hive上创建数据库**

1.安装Linux系统（略）

2.数据集下载与查看

用户行为日志user_log.csv，日志中的字段定义如下：
1. user_id | 买家id
2. item_id | 商品id
3. cat_id | 商品类别id
4. merchant_id | 卖家id
5. brand_id | 品牌id
6. month | 交易时间:月
7. day | 交易事件:日
8. action | 行为,取值范围{0,1,2,3},0表示点击，1表示加入购物车，2表示购买，3表示关注商品

9. age_range | 买家年龄分段：1表示年龄<18,2表示年龄在[18,24]，3表示年龄在[25,29]，4表示年龄在[30,34]，5表示年龄在[35,39]，6表示年龄在[40,49]，7和8表示年龄>=50,0和NULL则表示未知
10. gender | 性别:0表示女性，1表示男性，2和NULL表示未知

11. province| 收获地址省份

回头客训练集train.csv和回头客测试集test.csv，训练集和测试集拥有相同的字段，字段定义如下：

1. user_id | 买家id
2. age_range | 买家年龄分段：1表示年龄<18,2表示年龄在[18,24]，3表示年龄在[25,29]，4表示年龄在[30,34]，5表示年龄在[35,39]，6表示年龄在[40,49]，7和8表示年龄>=50,0和NULL则表示未知
3. gender | 性别:0表示女性，1表示男性，2和NULL表示未知
4. merchant_id | 商家id
5. label | 是否是回头客，0值表示不是回头客，1值表示回头客，-1值表示该用户已经超出我们所需要考虑的预测范围。NULL值只存在测试集，在测试集中表示需要预测的值。

3.**数据集预处理**
进入Linux下：

```
cd /usr/local
ls
sudo mkdir dbtaobao
//这里会提示你输入当前用户（本教程是hadoop用户名）的密码
//下面给hadoop用户赋予针对dbtaobao目录的各种操作权限
sudo chown -R hadoop:hadoop ./dbtaobao
cd dbtaobao
//下面创建一个dataset目录，用于保存数据集
mkdir dataset
//下面就可以解压缩data_format.zip文件
cd ~  //表示进入hadoop用户的目录
cd 下载
ls
unzip data_format.zip -d /usr/local/dbtaobao/dataset
cd /usr/local/dbtaobao/dataset
ls
```

现在你就可以看到在dataset目录下有三个文件：test.csv、train.csv、user_log.csv
我们执行下面命令取出user_log.csv前面5条记录看一下
执行如下命令:

```
1. head -5 user_log.csv
```

删除文件第一行记录，即字段名称
user_log.csv的第一行都是字段名称，我们在文件中的数据导入到数据仓库Hive中时，不需要第一行字段名称，因此，这里在做数据预处理时，删除第一行

```
cd /usr/local/dbtaobao/dataset
//下面删除user_log.csv中的第1行
sed -i '1d' user_log.csv //1d表示删除第1行，同理，3d表示删除第3行，nd表示删除第n行
//下面再用head命令去查看文件的前5行记录，就看不到字段名称这一行了
head -5 user_log.csv
```

获取数据集中双11的前100000条数据
由于数据集中交易数据太大，这里只截取数据集中在双11的前10000条交易数据作为小数据集small_user_log.csv
下面我们建立一个脚本文件完成上面截取任务，请把这个脚本文件放在dataset目录下：

```
1. cd /usr/local/dbtaobao/dataset
```

```
2. vim predeal.sh
```
predeal.sh内容：
```
#!/bin/bash
#下面设置输入文件，把用户执行predeal.sh命令时提供的第一个参数作为输入文件名称
infile=$1
#下面设置输出文件，把用户执行predeal.sh命令时提供的第二个参数作为输出文件名称
outfile=$2
#注意！！最后的$infile > $outfile必须跟在}'这两个字符的后面
awk -F "," 'BEGIN{
      id=0;
    }
    {
        if($6==11 && $7==11){
            id=id+1;
            print $1","$2","$3","$4","$5","$6","$7","$8","$9","$10","$11
            if(id==10000){
                exit
            }
        }
    }' $infile > $outfile
```
下面就可以执行predeal.sh脚本文件，截取数据集中在双11的前10000条交易数据作为小数据集small_user_log.csv，命令如下：
```
chmod +x ./predeal.sh
./predeal.sh ./user_log.csv ./small_user_log.csv
```
导入数据库：下面要把small_user_log.csv中的数据最终导入到数据仓库Hive中。为了完成这个操作，我们会首先把这个文件上传到分布式文件系统HDFS中，然后，在Hive中创建两个个外部表，完成数据的导入。

启动HDFS：
```
1. cd /usr/local/hadoop
2. ./sbin/start-dfs.sh
```
把user_log.csv上传到HDFS中

现在，我们要把Linux本地文件系统中的user_log.csv上传到分布式文件系统HDFS中，存放在HDFS中的“/dbtaobao/dataset”目录下。

首先，请执行下面命令，在HDFS的根目录下面创建一个新的目录dbtaobao，并在这个目录下创建一个子目录dataset，如下：
```
cd /usr/local/hadoop
./bin/hdfs dfs -mkdir -p /dbtaobao/dataset/user_log
```
然后，把Linux本地文件系统中的small_user_log.csv上传到分布式文件系统HDFS的“/dbtaobao/dataset”目录下，命令如下：
```
cd /usr/local/hadoop
./bin/hdfs dfs -put /usr/local/dbtaobao/dataset/small_user_log.csv
/dbtaobao/dataset/user_log
```
下面可以查看一下HDFS中的small_user_log.csv的前10条记录，命令如下：
```
cd /usr/local/hadoop
./bin/hdfs dfs -cat /dbtaobao/dataset/user_log/small_user_log.csv | head -10
```
既然现在数据在HDFS上了，那么就可以在Hive上创建数据库了。 由于Hive是基于Hadoop的数据仓库，使用HiveQL语言撰写的查询语句，最终都会被Hive自动解析成MapReduce任务由Hadoop去具体执行，因此，需要启动Hadoop，然后再启动Hive。由于前面我们已经启动了Hadoop，所以，这里不需要再次启动Hadoop。下面，在这个新的终端中执行下面命令进入Hive：
```
1. cd /usr/local/hive
2. ./bin/hive # 启动Hive
```
启动成功以后，就进入了“hive>”命令提示符状态，可以输入类似SQL语句的HiveQL语句。

下面，我们要在Hive中创建一个数据库dbtaobao，命令如下：

```
hive>  create database dbtaobao;
hive>  use dbtaobao;
```

创建外部表（外部数据库就是不移动数据文件到hive目录下，只是创建了一个表的元数据而已，删除表也不会删除数据）

```
hive>  CREATE EXTERNAL TABLE dbtaobao.user_log(user_id INT,item_id INT,cat_id
INT,merchant_id INT,brand_id INT,month STRING,day STRING,action INT,age_range
INT,gender INT,province STRING) COMMENT 'Welcome to xmu dblab,Now create
dbtaobao.user_log!' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE
LOCATION '/dbtaobao/dataset/user_log';
```

查询数据：　上面已经成功把HDFS中的“/dbtaobao/dataset/user_log”目录下的small_user_log.csv数据加载到了数据仓库Hive中，我们现在可以使用下面命令查询一下：

```
select * from user_log limit 10;
```

步骤二:Hive数据分析

```
hive> select brand_id from user_log limit 10; -- 查看日志前10个交易日志的商品品牌
hive> select month,day,cat_id from user_log limit 20; --查询前20个交易日志中购买商品时的
时间和商品的种类
hive> select ul.at, ul.ci  from (select action as at, cat_id as ci from user_log) as
ul limit 20;  --有时我们在表中查询可以利用嵌套语句，如果列名太复杂可以设置该列的别名，以简化
我们操作的难度
hive> select count(*) from user_log; -- 用聚合函数count()计算出表内有多少条行数据 --用聚
合函数count()计算出表内有多少条行数据
hive> select count(*) from (select
user_id,item_id,cat_id,merchant_id,brand_id,month,day,action from user_log group by
user_id,item_id,cat_id,merchant_id,brand_id,month,day,action having count(*)=1)a;  --
查询不重复的数据有多少条(为了排除客户刷单情况)
hive> create table scan(brand_id INT,scan INT) COMMENT 'This is the search of
bigdatataobao' ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE; --
创建新的数据表进行存储
hive> insert overwrite table scan select brand_id,count(action) from user_log where
action='2' group by brand_id; --导入数据
hive> select * from scan; -- 显示结果
```

步骤三:将数据从Hive导入到MySQL

创建临时表inner_user_log和inner_user_info， 这个命令执行完以后，Hive会自动在HDFS文件系统中创建对应的数据文件“/user/hive/warehouse/dbtaobao.db/inner_user_log”。

```
hive> create table dbtaobao.inner_user_log(user_id INT,item_id INT,cat_id
INT,merchant_id INT,brand_id INT,month STRING,day STRING,action INT,age_range
INT,gender INT,province STRING) COMMENT 'Welcome to XMU dblab! Now create inner table
inner_user_log ' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
```

将user_log表中的数据插入到inner_user_log,在[步骤一:本地数据集上传到数据仓库Hive]中，我们已经在Hive中的dbtaobao数据库中创建了一个外部表user_log。下面把dbtaobao.user_log数据插入到dbtaobao.inner_user_log表中，命令如下：

```
hive> INSERT OVERWRITE TABLE dbtaobao.inner_user_log select * from dbtaobao.user_log;
```

使用Sqoop将数据从Hive导入MySQL

创建数据库

```
mysql> show databases; #显示所有数据库
mysql> create database dbtaobao; #创建dbtaobao数据库
```

```
mysql> use dbtaobao; #使用数据库
mysql> show variables like "char%";
```

会显示类似下面的结果：

```
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | latin1                     |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

请确认当前编码为utf8，否则无法导入中文，修改方法：

Unbantu：编辑配置文件。`sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf，`添加一行`character_set_server=utf8`

CentOs：编辑 /etc/my.conf 添加一行`character_set_server=utf8`

修改完成之后：

```
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

创建表：

```
mysql> CREATE TABLE `dbtaobao`.`user_log` (`user_id` varchar(20),`item_id`
varchar(20),`cat_id` varchar(20),`merchant_id` varchar(20),`brand_id` varchar(20),
`month` varchar(6),`day` varchar(6),`action` varchar(6),`age_range`
varchar(6),`gender` varchar(6),`province` varchar(10)) ENGINE=InnoDB DEFAULT
CHARSET=utf8;
```

回到Sqoop：

```
cd /usr/local/sqoop
bin/sqoop export --connect jdbc:mysql://localhost:3306/dbtaobao --username root --
password root --table user_log --export-dir
'/user/hive/warehouse/dbtaobao.db/inner_user_log' --fields-terminated-by ',';
```

字段解释：

./bin/sqoop export ##表示数据从 hive 复制到 mysql 中
–connect jdbc:mysql://localhost:3306/dbtaobao
–username root #mysql登陆用户名
–password root #登录密码
–table user_log #mysql 中的表，即将被导入的表名称
–export-dir ‘/user/hive/warehouse/dbtaobao.db/user_log ‘ #hive 中被导出的文件
–fields-terminated-by ‘,’ #Hive 中被导出的文件字段的分隔符

返回Mysql查看数据：

```
mysql> use dbtaobao;
mysql> select * from user_log limit 10;
```

步骤四:利用Spark预测回头客行为

这里列出test.csv和train.csv中字段的描述，字段定义如下：

1. user_id | 买家id
2. age_range | 买家年龄分段：1表示年龄<18,2表示年龄在[18,24]，3表示年龄在[25,29]，4表示年龄在[30,34]，5表示年龄在[35,39]，6表示年龄在[40,49]，7和8表示年龄>=50,0和NULL则表示未知
3. gender | 性别:0表示女性，1表示男性，2和NULL表示未知
4. merchant_id | 商家id
5. label | 是否是回头客，0值表示不是回头客，1值表示回头客，-1值表示该用户已经超出我们所需要考虑的预测范围。NULL值只存在测试集，在测试集中表示需要预测的值。

这里需要预先处理test.csv数据集，把这test.csv数据集里label字段表示-1值剔除掉,保留需要预测的数据.并假设需要预测的数据中label字段均为1.

```
cd /usr/local/dbtaobao/dataset
vim predeal_test.sh
predeal_test.sh内容：
#!/bin/bash
#下面设置输入文件，把用户执行predeal_test.sh命令时提供的第一个参数作为输入文件名称
infile=$1
#下面设置输出文件，把用户执行predeal_test.sh命令时提供的第二个参数作为输出文件名称
outfile=$2
#注意！！最后的$infile > $outfile必须跟在}’这两个字符的后面
awk -F "," 'BEGIN{
      id=0;
    }
    {
        if($1 && $2 && $3 && $4 && !$5){
            id=id+1;
            print $1","$2","$3","$4","1
            if(id==10000){
                exit
            }
        }
    }' $infile > $outfile
```

然后执行：

```
chmod +x ./predeal_test.sh
./predeal_test.sh ./test.csv ./test_after.csv
```

train.csv的第一行都是字段名称，不需要第一行字段名称,这里在对train.csv做数据预处理时，删除第一行

```
sed -i '1d' train.csv
```

再用脚本处理train.csv

```
cd /usr/local/dbtaobao/dataset
vim predeal_train.sh
predeal_train.sh内容：
#!/bin/bash
#下面设置输入文件，把用户执行predeal_train.sh命令时提供的第一个参数作为输入文件名称
infile=$1
#下面设置输出文件，把用户执行predeal_train.sh命令时提供的第二个参数作为输出文件名称
outfile=$2
#注意！！最后的$infile > $outfile必须跟在}’这两个字符的后面
awk -F "," 'BEGIN{
         id=0;
    }
    {
        if($1 && $2 && $3 && $4 && ($5!=-1)){
            id=id+1;
            print $1","$2","$3","$4","$5
            if(id==10000){
                exit
```

```
            }
        }
    }' $infile > $outfile
```

下面就可以执行predeal_train.sh脚本文件，截取测试数据集需要预测的数据到train_after.csv，命令如下：

```
chmod +x ./predeal_train.sh
./predeal_train.sh ./train.csv ./train_after.csv
```

预测回头客

将两个数据集分别存取到HDFS中

```
bin/hadoop fs -mkdir -p /dbtaobao/dataset
bin/hadoop fs -put /usr/local/dbtaobao/dataset/train_after.csv /dbtaobao/dataset
bin/hadoop fs -put /usr/local/dbtaobao/dataset/test_after.csv /dbtaobao/dataset
```

进入Mysql：

```
use dbtaobao;
create table rebuy (score varchar(40),label varchar(40));
```

启动SparkShell：

Spark支持通过JDBC方式连接到其他数据库获取数据生成DataFrame。 下载MySQL的JDBC驱动（mysql-connector-java-5.1.40.zip）mysql-connector-java-*.zip是Java连接MySQL的驱动包,默认会下载到”~/下载/”目录执行如下命令（把你有的mysql连接jar包放到spark的jars就好）：

```
cd ~/下载/
unzip mysql-connector-java-5.1.40.zip -d /usr/local/spark/jars
```

接下来正式启动spark-shell

```
cd /usr/local/spark
./bin/spark-shell --jars /usr/local/spark/jars/mysql-connector-java-5.1.40/mysql-
connector-java-5.1.40-bin.jar --driver-class-path /usr/local/spark/jars/mysql-
connector-java-5.1.40/mysql-connector-java-5.1.40-bin.jar
```

支持向量机SVM分类器预测回头客

在spark-shell中执行如下操作：

1.导入需要的包。首先，我们导入需要的包：

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.{Vectors,Vector}
import org.apache.spark.mllib.classification.{SVMModel, SVMWithSGD}
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import java.util.Properties
import org.apache.spark.sql.types._
import org.apache.spark.sql.Row
```

2.读取训练数据

首先，读取训练文本文件；然后，通过map将每行的数据用“,”隔开，在数据集中，每行被分成了5部分，前4部分是用户交易的3个特征(age_range,gender,merchant_id)，最后一部分是用户交易的分类(label)。把这里我们用LabeledPoint来存储标签列和特征列。LabeledPoint在监督学习中常用来存储标签和特征，其中要求标签的类型是double，特征的类型是Vector。

```
val train_data = sc.textFile("/dbtaobao/dataset/train_after.csv")
val test_data = sc.textFile("/dbtaobao/dataset/test_after.csv")
```

3.构建模型

```
val train= train_data.map{line =>
  val parts = line.split(',')
  LabeledPoint(parts(4).toDouble,Vectors.dense(parts(1).toDouble,parts
```

```
(2).toDouble,parts(3).toDouble))
}
val test = test_data.map{line =>
  val parts = line.split(',')
  LabeledPoint(parts(4).toDouble,Vectors.dense(parts(1).toDouble,parts(2).toDouble,parts(3).toDouble))
}
```

接下来，通过训练集构建模型SVMWithSGD。这里的SGD即著名的随机梯度下降算法（Stochastic Gradient Descent）。设置迭代次数为1000，除此之外还有stepSize（迭代步伐大小），regParam（regularization正则化控制参数），miniBatchFraction（每次迭代参与计算的样本比例），initialWeights（weight向量初始值）等参数可以进行设置。

```
val numIterations = 1000
val model = SVMWithSGD.train(train, numIterations)
```

4.评估模型

接下来，我们清除默认阈值，这样会输出原始的预测评分，即带有确信度的结果。

```
model.clearThreshold()
val scoreAndLabels = test.map{point =>
  val score = model.predict(point.features)
  score+" "+point.label
}
scoreAndLabels.foreach(println)
```

如果我们设定了阀值，则会把大于阈值的结果当成正预测，小于阈值的结果当成负预测。

```
model.setThreshold(0.0)
scoreAndLabels.foreach(println)
```

5.把结果添加到mysql数据库中，现在我们上面没有设定阀值的测试集结果存入到MySQL数据中。

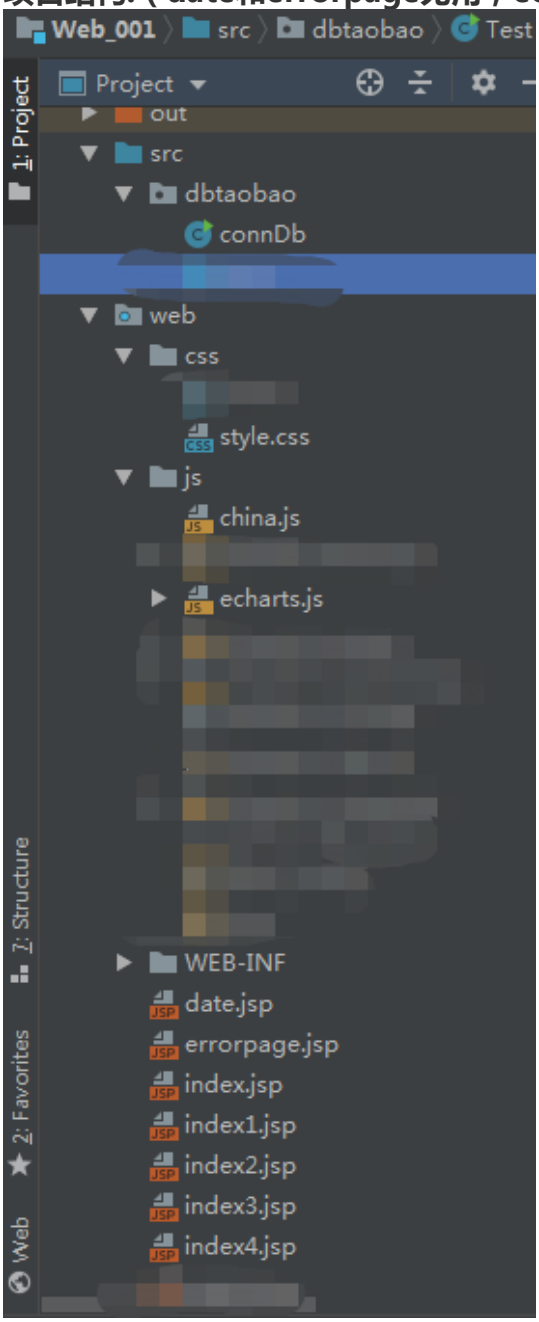
```
model.clearThreshold()
val scoreAndLabels = test.map{point =>
  val score = model.predict(point.features)
  score+" "+point.label
}
//设置回头客数据
val rebuyRDD = scoreAndLabels.map(_.split(" "))
/下面要设置模式信息
val schema = StructType(List(StructField("score", StringType,
true),StructField("label", StringType, true)))
//下面创建Row对象，每个Row对象都是rowRDD中的一行
val rowRDD = rebuyRDD.map(p => Row(p(0).trim, p(1).trim))
//建立起Row对象和模式之间的对应关系，也就是把数据和模式对应起来
val rebuyDF = spark.createDataFrame(rowRDD, schema)
//下面创建一个prop变量用来保存JDBC连接参数
val prop = new Properties()
prop.put("user", "root") //表示用户名是root
prop.put("password", "root") //表示密码是hadoop
prop.put("driver","com.mysql.jdbc.Driver") //表示驱动程序是com.mysql.jdbc.Driver
//下面就可以连接数据库，采用append模式，表示追加记录到数据库dbtaobao的rebuy表中
rebuyDF.write.mode("append").jdbc("jdbc:mysql://localhost:3306/dbtaobao",
"dbtaobao.rebuy", prop)
```

步骤五:利用ECharts进行数据可视化分析

工具

由于ECharts是运行在网页前端，我们选用JSP作为服务端语言，读取MySQL中的数据，然后渲染到前端页面。
本步骤需要涉及以下工具：
操作系统:Linux系统（比如Centos7）
可视化：ECharts（安装在Windows系统下）
数据库：MySQL（安装在Linux系统下）
Web应用服务器：tomcat（windows下）
IDE:IDEA（windows下）

项目结构:（date和errorpage无用，echarts.min.js在echarts.js下）



后端connDb.java全部代码：

```
package dbtaobao;
import java.sql.*;
import java.util.ArrayList;
public class connDb {
    private static Connection con = null;
    private static Statement stmt = null;
    private static ResultSet rs = null;
    //连接数据库方法
    public static void startConn(){
        try{
```

```
            Class.forName("com.mysql.jdbc.Driver");
            //连接数据库中间件
            try{
                con =
DriverManager.getConnection("jdbc:MySQL://localhost:3306/dbtaobao","root","root");
            }catch(SQLException e){
                e.printStackTrace();
            }
        }catch(ClassNotFoundException e){
            e.printStackTrace();
        }
    }
    //关闭连接数据库方法
    public static void endConn() throws SQLException{
        if(con != null){
            con.close();
            con = null;
        }
        if(rs != null){
            rs.close();
            rs = null;
        }
        if(stmt != null){
            stmt.close();
            stmt = null;
        }
    }
    //数据库双11 所有买家消费行为比例
    public static ArrayList index() throws SQLException{
        ArrayList<String[]> list = new ArrayList();
        startConn();
        stmt = con.createStatement();
        rs = stmt.executeQuery("select action,count(*) num from user_log group by
action desc");
        while(rs.next()){
            String[] temp={rs.getString("action"),rs.getString("num")};
            list.add(temp);
        }
            endConn();
        return list;
    }
    //男女买家交易对比
        public static ArrayList index_1() throws SQLException{
            ArrayList<String[]> list = new ArrayList();
            startConn();
            stmt = con.createStatement();
            rs = stmt.executeQuery("select gender,count(*) num from user_log group by
gender desc");
            while(rs.next()){
                String[] temp={rs.getString("gender"),rs.getString("num")};
                list.add(temp);
            }
            endConn();
            return list;
        }
        //男女买家各个年龄段交易对比
        public static ArrayList index_2() throws SQLException{
            ArrayList<String[]> list = new ArrayList();
            startConn();
            stmt = con.createStatement();
            rs = stmt.executeQuery("select gender,age_range,count(*) num from user_log
group by gender,age_range desc");
            while(rs.next()){
                String[] temp=
{rs.getString("gender"),rs.getString("age_range"),rs.getString("num")};
                list.add(temp);
            }
            endConn();
            return list;
        }
        //获取销量前五的商品类别
        public static ArrayList index_3() throws SQLException{
            ArrayList<String[]> list = new ArrayList();
```

```
            startConn();
            stmt = con.createStatement();
            rs = stmt.executeQuery("select cat_id,count(*) num from user_log group by
cat_id order by count(*) desc limit 5");
            while(rs.next()){
                String[] temp={rs.getString("cat_id"),rs.getString("num")};
                list.add(temp);
            }
            endConn();
            return list;
        }
    //各个省份的总成交量对比
    public static ArrayList index_4() throws SQLException{
        ArrayList<String[]> list = new ArrayList();
        startConn();
        stmt = con.createStatement();
        rs = stmt.executeQuery("select province,count(*) num from user_log group by
province order by count(*) desc");
        while(rs.next()){
            String[] temp={rs.getString("province"),rs.getString("num")};
            list.add(temp);
        }
        endConn();
        return list;
    }
}
```

前端代码index

```
<%@ page language="java" import="dbtaobao.connDb,java.util.*" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    connDb conndb=new connDb();
    conndb.startConn();
    ArrayList<String[]> list = conndb.index();
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ECharts 可视化分析淘宝双11</title>
<link href="./css/style.css" type='text/css' rel="stylesheet"/>
<script src="./js/echarts.min.js"></script>
</head>
<body>
   <div class='header'>
        <p>ECharts 可视化分析淘宝双11</p>
    </div>
    <div class="content">
        <div class="nav">
            <ul>
                <li class="current"><a href="#">所有买家各消费行为对比</a></li>
                <li><a href="./index1.jsp">男女买家交易对比</a></li>
                <li><a href="./index2.jsp">男女买家各个年龄段交易对比</a></li>
                <li><a href="./index3.jsp">商品类别交易额对比</a></li>
                <li><a href="./index4.jsp">各省份的总成交量对比</a></li>
            </ul>
        </div>
        <div class="container">
            <div class="title">所有买家各消费行为对比</div>
            <div class="show">
                <div class='chart-type'>饼图</div>
                <div id="main"></div>
            </div>
        </div>
    </div>
<script>
//基于准备好的dom，初始化echarts实例
var myChart = echarts.init(document.getElementById('main'));
// 指定图表的配置项和数据
```

```
option = {
         backgroundColor: '#2c343c',


         title: {
             text: '所有买家消费行为比例图',
             left: 'center',
             top: 20,
             textStyle: {
                 color: '#555555'
             }
         },


         tooltip : {
             trigger: 'item',
             formatter: "{a} <br/>{b} : {c} ({d}%)"
         },


         visualMap: {
             show: false,
             min: 80,
             max: 600,
             inRange: {
                 colorLightness: [0, 1]
             }
         },
         series : [
             {
                 name:'消费行为',
                 type:'pie',
                 radius : '55%',
                 center: ['50%', '50%'],
                 data:[
                    {value:<%=list.get(0)[1]%>, name:'特别关注'},
                     {value:<%=list.get(1)[1]%>, name:'购买'},
                     {value:<%=list.get(2)[1]%>, name:'添加购物车'},
                     {value:<%=list.get(3)[1]%>, name:'点击'},
                 ].sort(function (a, b) { return a.value - b.value}),
                 roseType: 'angle',
                 label: {
                     normal: {
                         textStyle: {
                             color: 'rgba(130,57,53,10)'
                         }
                     }
                 },
                 labelLine: {
                     normal: {
                         lineStyle: {
                             color: 'rgba(255, 100, 0, 1)'
                         },
                         smooth: 0.2,
                         length: 10,
                         length2: 20
                     }
                 },
                 itemStyle: {
                     normal: {
                         color: '#c23531',
                         shadowBlur: 200,
                         shadowColor: 'rgba(0, 0, 0, 0.5)'
                     }
                 },


                 animationType: 'scale',
                 animationEasing: 'elasticOut',
                 animationDelay: function (idx) {
                     return Math.random() * 200;
                 }
             }
```

```
        ]
    };


// 使用刚指定的配置项和数据显示图表。
myChart.setOption(option);
</script>
</body>
</html>
```

index1.jsp

```
<%@ page language="java" import="dbtaobao.connDb,java.util.*" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
ArrayList<String[]> list = connDb.index_1();
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ECharts 可视化分析淘宝双11</title>
<link href="./css/style.css" type='text/css' rel="stylesheet"/>
<script src="./js/echarts.min.js"></script>
</head>
<body>
   <div class='header'>
        <p>ECharts 可视化分析淘宝双11</p>
    </div>
    <div class="content">
        <div class="nav">
            <ul>
                <li><a href="index.jsp">所有买家各消费行为对比</a></li>
                <li class="current"><a href="#">男女买家交易对比</a></li>
                <li><a href="./index2.jsp">男女买家各个年龄段交易对比</a></li>
                <li><a href="./index3.jsp">商品类别交易额对比</a></li>
                <li><a href="./index4.jsp">各省份的总成交量对比</a></li>
            </ul>
        </div>
        <div class="container">
            <div class="title">男女买家交易对比</div>
            <div class="show">
                <div class='chart-type'>饼图</div>
                <div id="main"></div>
            </div>
        </div>
    </div>
<script>
//基于准备好的dom，初始化echarts实例
var myChart = echarts.init(document.getElementById('main'));
// 指定图表的配置项和数据
option = {
         backgroundColor: '#2c343c',


         title: {
             text: '男女买家交易对比',
             left: 'center',
             top: 20,
             textStyle: {
                 color: '#555555'
             }
         },


         tooltip : {
             trigger: 'item',
             formatter: "{a} <br/>{b} : {c} ({d}%)"
         },
```

```
        visualMap: {
            show: false,
            min: 80,
            max: 600,
            inRange: {
                colorLightness: [0, 1]
            }
        },
        series : [
            {
                name:'消费行为',
                type:'pie',
                radius : '55%',
                center: ['50%', '50%'],
                data:[
                   {value:<%=list.get(0)[1]%>, name:'女性'},
                    {value:<%=list.get(1)[1]%>, name:'男性'},
                    {value:<%=list.get(2)[1]%>, name:'未知'},
                ].sort(function (a, b) { return a.value - b.value}),
                roseType: 'angle',
                label: {
                    normal: {
                        textStyle: {
                            color: 'rgba(130,57,53,10)'
                        }
                    }
                },
                labelLine: {
                    normal: {
                        lineStyle: {
                            color: 'rgba(255, 100, 0, 1)'
                        },
                        smooth: 0.2,
                        length: 10,
                        length2: 20
                    }
                },
                itemStyle: {
                    normal: {
                        color: '#c23531',
                        shadowBlur: 200,
                        shadowColor: 'rgba(0, 0, 0, 0.5)'
                    }
                },


                animationType: 'scale',
                animationEasing: 'elasticOut',
                animationDelay: function (idx) {
                    return Math.random() * 200;
                }
            }
        ]
    };


// 使用刚指定的配置项和数据显示图表。
myChart.setOption(option);
</script>
</body>
</html>
```

index2.jsp

```
<%@ page language="java" import="dbtaobao.connDb,java.util.*" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
ArrayList<String[]> list = connDb.index_2();
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ECharts 可视化分析淘宝双11</title>
<link href="./css/style.css" type='text/css' rel="stylesheet"/>
<script src="./js/echarts.min.js"></script>
</head>
<body>
   <div class='header'>
        <p>ECharts 可视化分析淘宝双11</p>
    </div>
    <div class="content">
        <div class="nav">
            <ul>
                <li><a href="./index.jsp">所有买家各消费行为对比</a></li>
                <li><a href="./index1.jsp">男女买家交易对比</a></li>
                <li class="current"><a href="#">男女买家各个年龄段交易对比</a></li>
                <li><a href="./index3.jsp">商品类别交易额对比</a></li>
                <li><a href="./index4.jsp">各省份的总成交量对比</a></li>
            </ul>
        </div>
        <div class="container">
            <div class="title">男女买家各个年龄段交易对比</div>
            <div class="show">
                <div class='chart-type'>散点图</div>
                <div id="main"></div>
            </div>
        </div>
    </div>
<script>
//基于准备好的dom，初始化echarts实例
var myChart = echarts.init(document.getElementById('main'));
// 指定图表的配置项和数据
var data = [];
data[0] = [];
data[1] = [];
<%
   for(String[] a:list){
      if(a[0].equals("0")){
         %>
         data[0].push([<%=a[1]%>,<%=a[2]%>,<%=a[0]%>]);
         <%
      }else if(a[0].equals("1")){
         %>
         data[1].push([<%=a[1]%>,<%=a[2]%>,<%=a[0]%>]);
         <%
      }
   }
%>
option = {
    backgroundColor: new echarts.graphic.RadialGradient(0.3, 0.3, 0.8, [{
        offset: 0,
        color: '#f7f8fa'
    }, {
        offset: 1,
        color: '#cdd0d5'
    }]),
    title: {
        text: '男女买家各个年龄段交易对比'
    },
    legend: {
        right: 10,
        data: ['women', 'men']
    },
    xAxis: {
        splitLine: {
            lineStyle: {
                type: 'dashed'
            }
        }
    },
    yAxis: {
        splitLine: {
            lineStyle: {
```

```
                type: 'dashed'
            }
        },
        scale: true
    },
    series: [{
        name: 'women',
        data: data[0],
        type: 'scatter',
        symbolSize: function (data) {
            return Math.sqrt(((data[1])-360)*25);
        },
        label: {
            emphasis: {
                show: true,
                formatter: function (param) {
                    return param.data[1];
                },
                position: 'top'
            }
        },
        itemStyle: {
            normal: {
                shadowBlur: 10,
                shadowColor: 'rgba(120, 36, 50, 0.5)',
                shadowOffsetY: 5,
                color: new echarts.graphic.RadialGradient(0.4, 0.3, 1, [{
                    offset: 0,
                    color: 'rgb(251, 118, 123)'
                }, {
                    offset: 1,
                    color: 'rgb(204, 46, 72)'
                }])
            }
        }
    }, {
        name: 'men',
        data: data[1],
        type: 'scatter',
        symbolSize: function (data) {
            return Math.sqrt(((data[1])-360)*25);
        },
        label: {
            emphasis: {
                show: true,
                formatter: function (param) {
                    return param.data[1];
                },
                position: 'top'
            }
        },
        itemStyle: {
            normal: {
                shadowBlur: 10,
                shadowColor: 'rgba(25, 100, 150, 0.5)',
                shadowOffsetY: 5,
                color: new echarts.graphic.RadialGradient(0.4, 0.3, 1, [{
                    offset: 0,
                    color: 'rgb(129, 227, 238)'
                }, {
                    offset: 1,
                    color: 'rgb(25, 183, 207)'
                }])
            }
        }
    }]
};


// 使用刚指定的配置项和数据显示图表。
myChart.setOption(option);
</script>
</body>
```

```
</html>
```

index3.jsp

```
<%@ page language="java" import="dbtaobao.connDb,java.util.*" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
ArrayList<String[]> list = connDb.index_3();
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ECharts 可视化分析淘宝双11</title>
<link href="./css/style.css" type='text/css' rel="stylesheet"/>
<script src="./js/echarts.min.js"></script>
</head>
<body>
   <div class='header'>
        <p>ECharts 可视化分析淘宝双11</p>
    </div>
    <div class="content">
        <div class="nav">
            <ul>
                <li><a href="index.jsp">所有买家各消费行为对比</a></li>
                <li><a href="./index1.jsp">男女买家交易对比</a></li>
                <li><a href="./index2.jsp">男女买家各个年龄段交易对比</a></li>
                <li class="current"><a href="#">商品类别交易额对比</a></li>
                <li><a href="./index4.jsp">各省份的总成交量对比</a></li>
            </ul>
        </div>
        <div class="container">
            <div class="title">商品类别交易额对比</div>
            <div class="show">
                <div class='chart-type'>饼图</div>
                <div id="main"></div>
            </div>
        </div>
    </div>
<script>
//基于准备好的dom，初始化echarts实例
var myChart = echarts.init(document.getElementById('main'));
// 指定图表的配置项和数据
var x = []
var y = []
<%
   for(String[] a:list){
      %>
      x.push(<%=a[0]%>);
      y.push(<%=a[1]%>);
      <%
   }
%>
option = {
    color: ['#330000'],
    tooltip : {
        trigger: 'axis',
        axisPointer : {            // 坐标轴指示器，坐标轴触发有效
            type : 'shadow'        // 默认为直线，可选为: 'line' | 'shadow'
        }
    },
    grid: {
        left: '3%',
        right: '4%',
        bottom: '3%',
        containLabel: true
    },
    xAxis : [
        {
            type : 'category',
            data : x,
```

```
            axisTick: {
                alignWithLabel: true
            }
        }
    ],
    yAxis : [
        {
            type : 'value'
        }
    ],
    series : [
        {
            name:'Value',
            type:'bar',
            barWidth: '60%',
            data:y
        }
    ]
};
// 使用刚指定的配置项和数据显示图表。
myChart.setOption(option);
</script>
</body>
</html>
```

index4.jsp

```
<%@ page language="java" import="dbtaobao.connDb,java.util.*" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
ArrayList<String[]> list = connDb.index_4();
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ECharts 可视化分析淘宝双11</title>
<link href="./css/style.css" type='text/css' rel="stylesheet"/>
<script src="./js/echarts.min.js"></script><script src="./js/china.js"></script>
<script src="./js/echarts.js"></script>
</head>
<body>


   <div class='header'>
        <p>ECharts 可视化分析淘宝双11</p>
    </div>
    <div class="content">
        <div class="nav">
            <ul>
                <li><a href="index.jsp">所有买家各消费行为对比</a></li>
                <li><a href="./index1.jsp">男女买家交易对比</a></li>
                <li><a href="./index2.jsp">男女买家各个年龄段交易对比</a></li>
                <li><a href="./index3.jsp">商品类别交易额对比</a></li>
                <li class="current"><a href="#">各省份的总成交量对比</a></li>
            </ul>
        </div>
        <div class="container">
            <div class="title">各省份的总成交量对比</div>
            <div class="show">
                <div class='chart-type'>地图</div>
                <div id="main"></div>
            </div>
        </div>
    </div>
<script type="text/javascript">
    var data = [];
    <%
   for(String[] a:list){
         %>
    data.push({name:"<%=a[0]%>",value:"<%=a[1]%>"});
```

```
    <%}%>
    var myChart = echarts.init(document.getElementById('main'));
    option = {
        tooltip: {
            formatter:function(params,ticket, callback){
                return params.seriesName+'<br />'+params.name+': '+params.value
            }//数据格式化
        },
        visualMap: {
            min: 280,
            max: 340,
            left: 'left',
            top: 'bottom',
            text: ['高','低'],//取值范围的文字
            inRange: {
                color: ['#e0ffff', 'red']//取值范围的颜色
            },
            show:true//图注
        },
        geo: {
            map: 'china',
            roam: false,//不开启缩放和平移
            zoom:1.23,//视角缩放比例
            label: {
                normal: {
                    show: true,
                    fontSize:'10',
                    color: 'rgba(5,2,10,0.7)'
                }
            },
            itemStyle: {
                normal:{
                    borderColor: 'rgba(250, 1, 23, 0.2)'
                },
                emphasis:{
                    areaColor: 'red',//鼠标选择区域颜色
                    shadowOffsetX: 0,
                    shadowOffsetY: 0,
                    shadowBlur: 20,
                    borderWidth: 0,
                    shadowColor: 'rgba(100, 50, 0, 1)'
                }
            }
        },
        series : [
            {
                name: '11.11销售额',
                type: 'map',
                geoIndex: 0,
                data:data
            }
        ]
    };
    myChart.setOption(option);
    myChart.on('click', function (params) {
        alert(params.name);
    });
</script>
</body>
</html>
```

效果：

所有买家各消费行为对比

男女买家交易对比

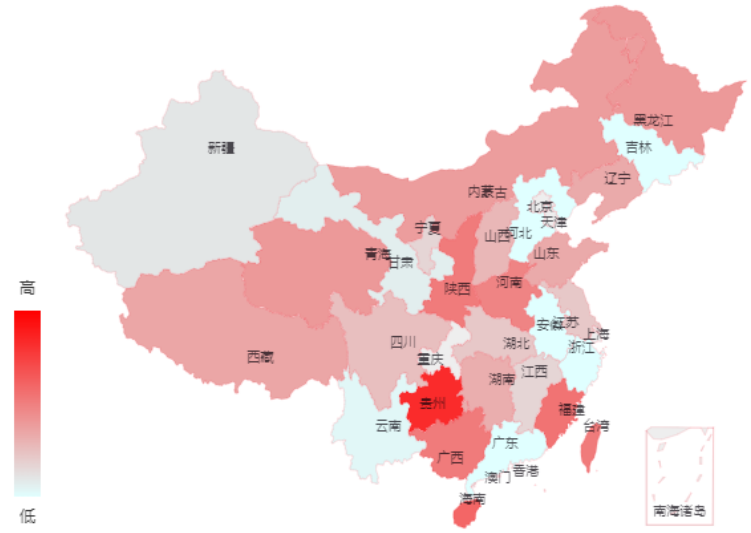
男女买家各个年龄段交易对比

商品类别交易额对比

各省份的总成交量对比

各省份的总成交量对比

地图



ECharts 可视化分析淘宝双11

所有买家各消费行为对比

男女买家交易对比

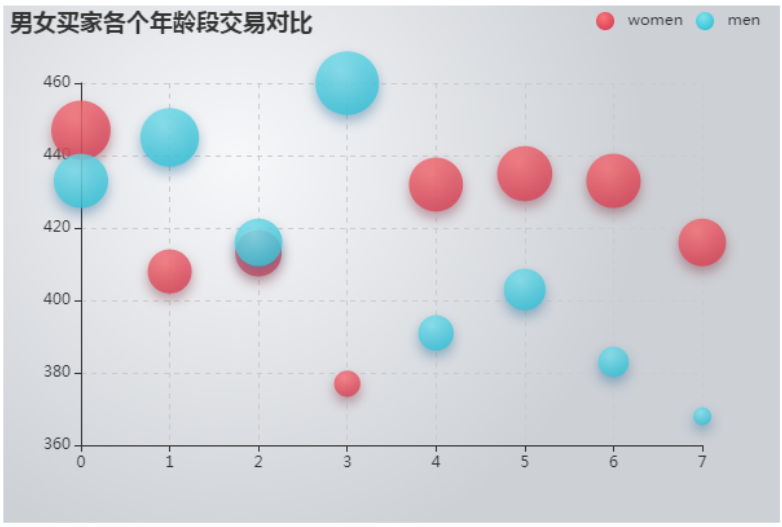
男女买家各个年龄段交易对比

商品类别交易额对比

各省份的总成交量对比

男女买家各个年龄段交易对比

散点图



ECharts 可视化分析淘宝双11

- 所有买家各消费行为对比
- 男女买家交易对比
- 男女买家各个年龄段交易对比
- 商品类别交易额对比
- 各省份的总成交量对比

所有买家各消费行为对比

饼图