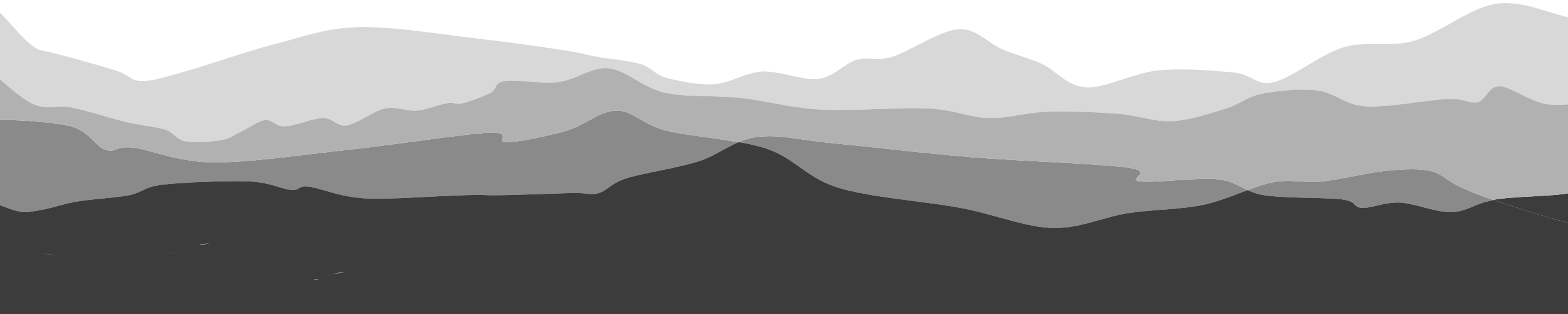
搜索广告转化预测

刘德欣 1901110660

王文祥 1901210258



目录 CONTENTS

01

任务简介

01 任务简介

转化率预测任务

- IJCAI-18 阿里妈妈搜索广告转化预测竞赛
- 给定广告点击相关的用户（user）、广告商品（ad）、检索词（query）、上下文内容（context）、商店（shop）等信息的条件下预测广告产生购买行为的概率（pCVR）

转化率预测任务

- 淘宝应用内广告展示转化效果预估
  - 从五类数据（广告基础数据、广告商品信息、用户信息、上下文信息和店铺信息）推测用户购买此推荐商品的概率
  - 数据来源于淘宝真实交易和广告
- 评测指标：负对数loss

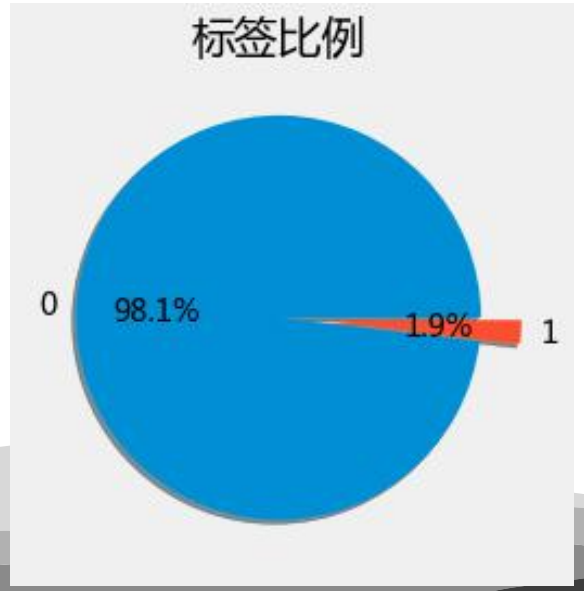
$$\log loss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i))$$

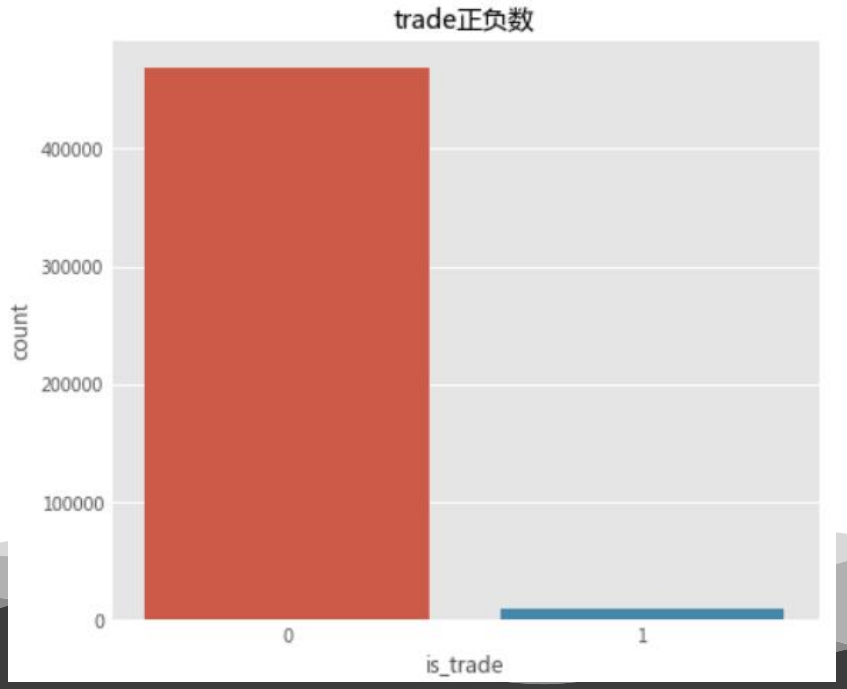
数据集信息及分析——广告基础数据

字段	解释
instance_id	样本编号，Long
is_trade	是否交易的标记位，Int类型；取值是0或者1，其中1 表示这条样本最终产生交易，0 表示没有交易
item_id	广告商品编号，Long类型
user_id	用户的编号，Long类型
context_id	上下文信息的编号，Long类型
shop_id	店铺的编号，Long类型

数据集信息及分析——广告基础数据

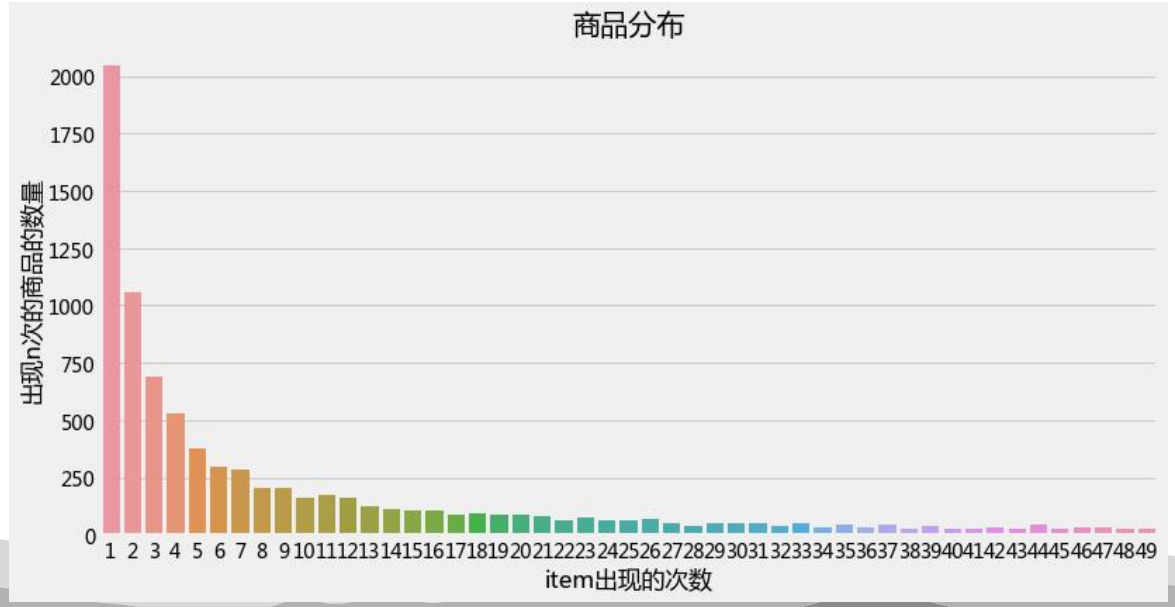
- 共有478,138条数据，非重复条目478,063条，重复条目75条
- 标签比例为1:52.2，正负样本比例很不均匀

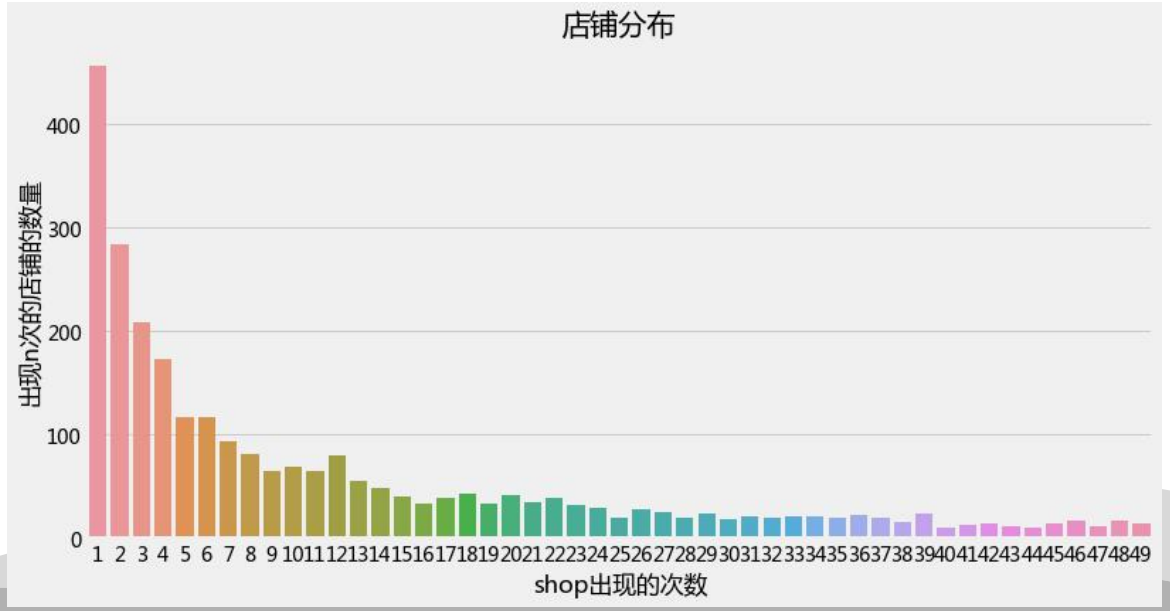




数据集信息及分析——广告基础数据

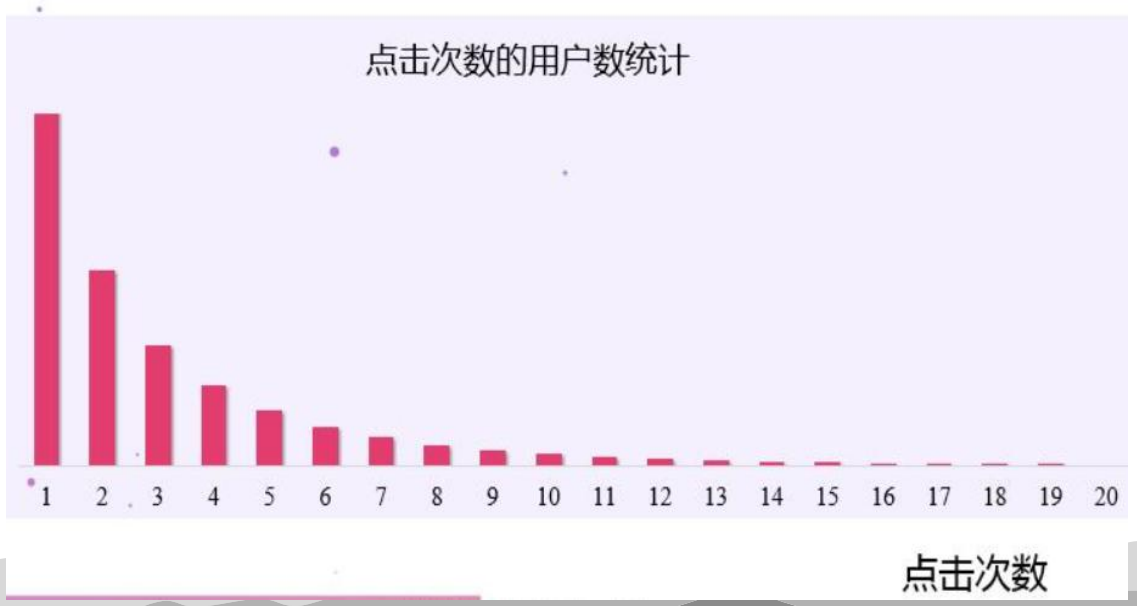
- 广告涉及10,074个不同的商品，来自3,958个不同的店铺
- 商品和店铺都符合长尾分布

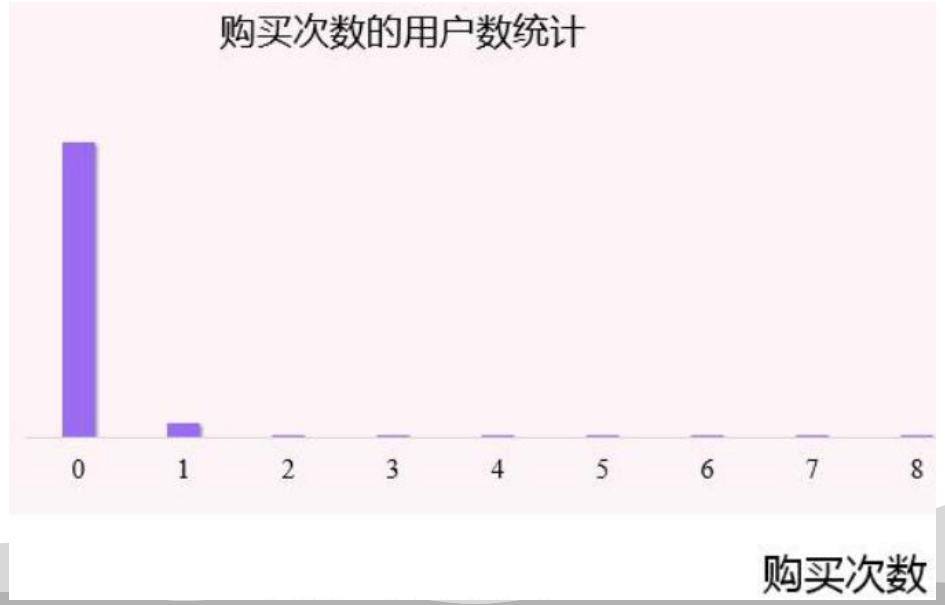




数据集信息及分析——广告基础数据

- 广告推送给197,677个不同的用户
- 大部分（85%）的用户点击广告次数在5次以下



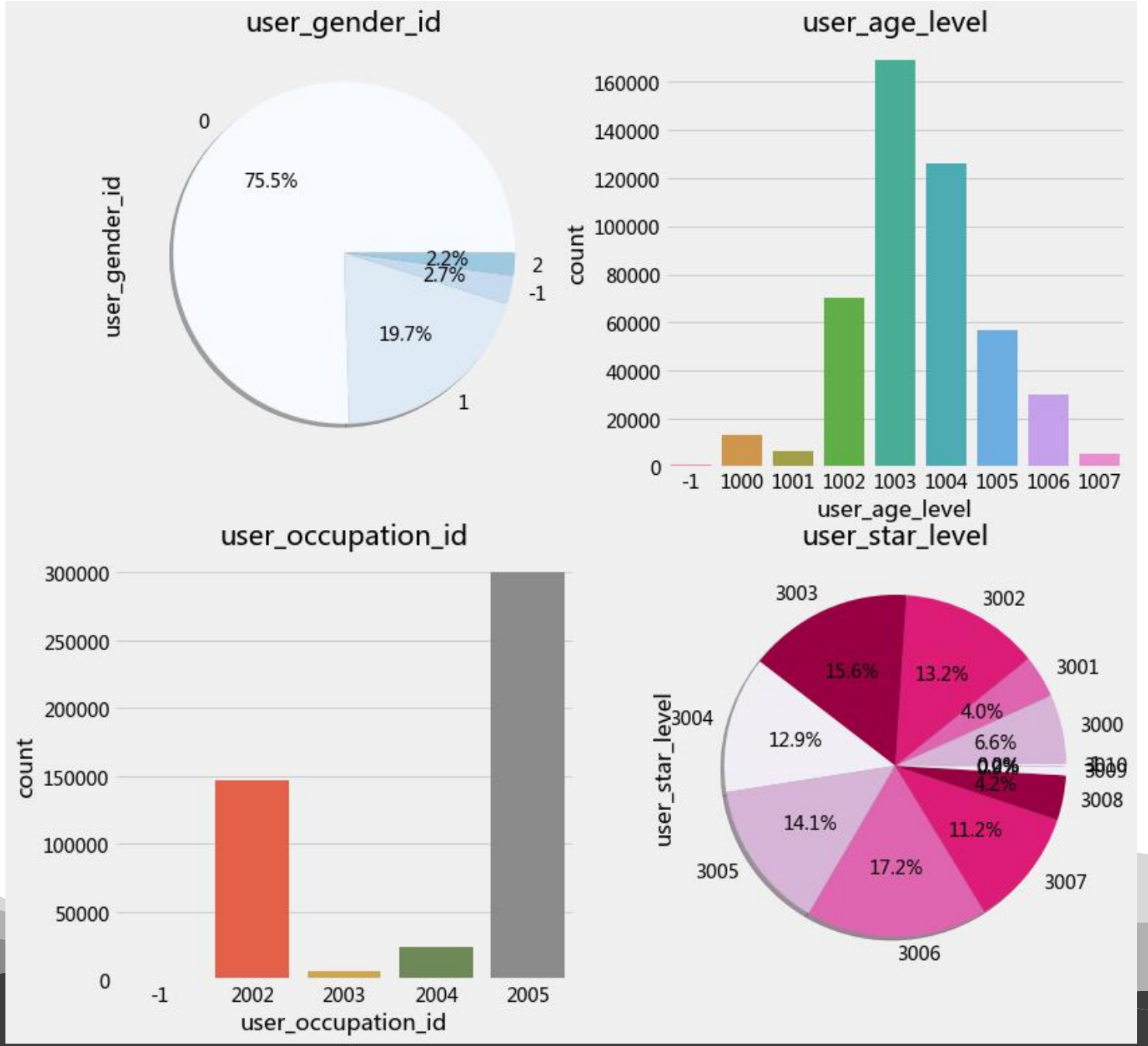


数据集信息及分析——用户信息

字段	解释
user_id	用户的编号，Long类型
user_gender_id	用户的预测性别编号，Int类型；0表示女性用户，1表示男性用户，2表示家庭用户
user_age_level	用户的预测年龄等级，Int类型；数值越大表示年龄越大
user_occupation_id	用户的预测职业编号，Int类型
user_star_level	用户的星级编号，Int类型；数值越大表示用户的星级越高

数据集信息及分析——用户信息

- 女性为广告主要目标群体
- 年龄分布集中在20-50岁

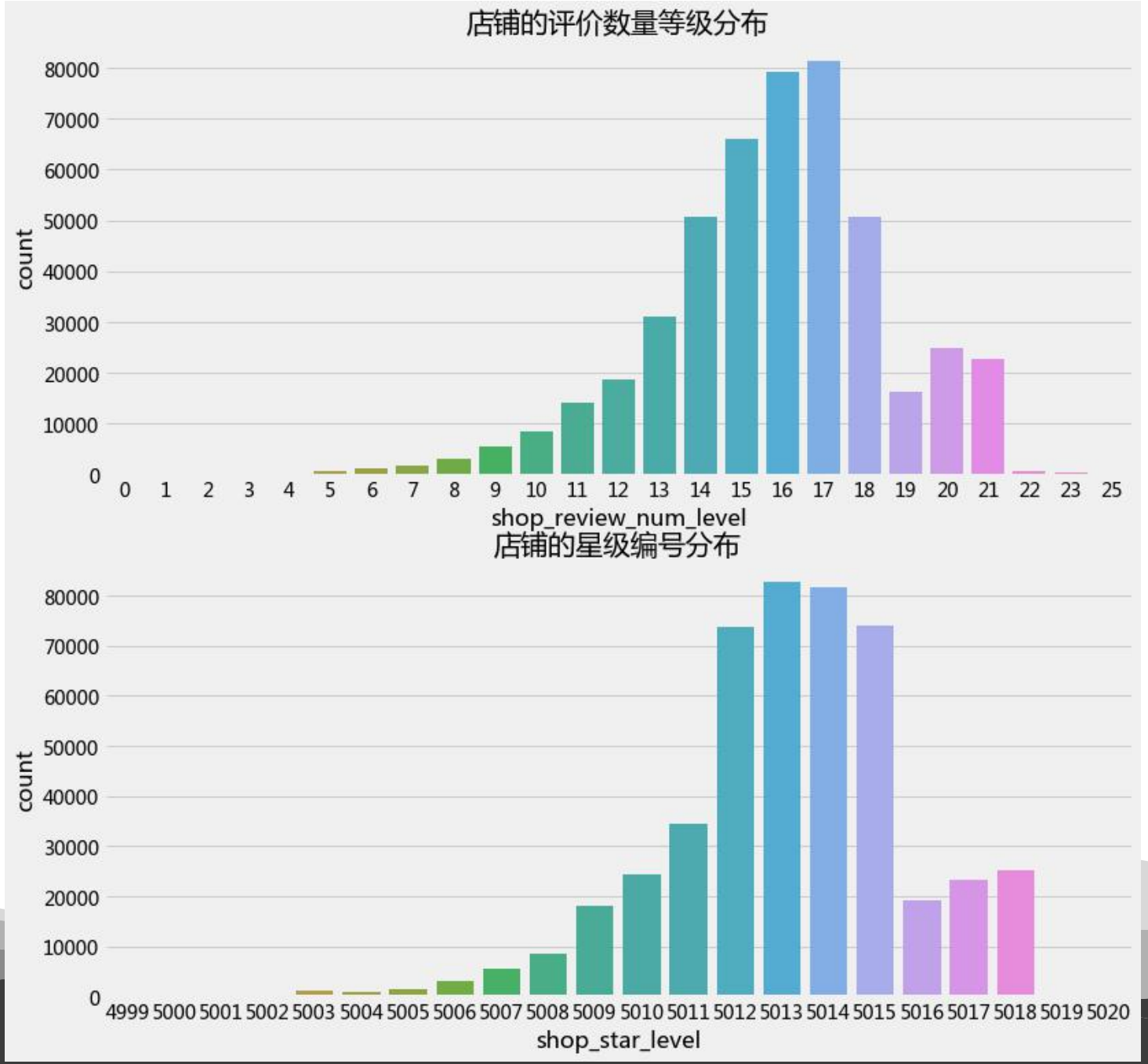


数据集信息及分析——店铺信息

字段	解释
shop_id	店铺的编号，Long类型
shop_review_num_level	店铺的评价数量等级，Int类型；取值从0开始，数值越大表示评价数量越多
shop_review_positive_rate	店铺的好评率，Double类型；取值在0到1之间，数值越大表示好评率越高
shop_star_level	店铺的星级编号，Int类型；取值从0开始，数值越大表示店铺的星级越高
shop_score_service	店铺的服务态度评分，Double类型；取值在0到1之间，数值越大表示评分越高
shop_score_delivery	店铺的物流服务评分，Double类型；取值在0到1之间，数值越大表示评分越高
shop_score_description	店铺的描述相符评分，Double类型；取值在0到1之间，数值越大表示评分越高

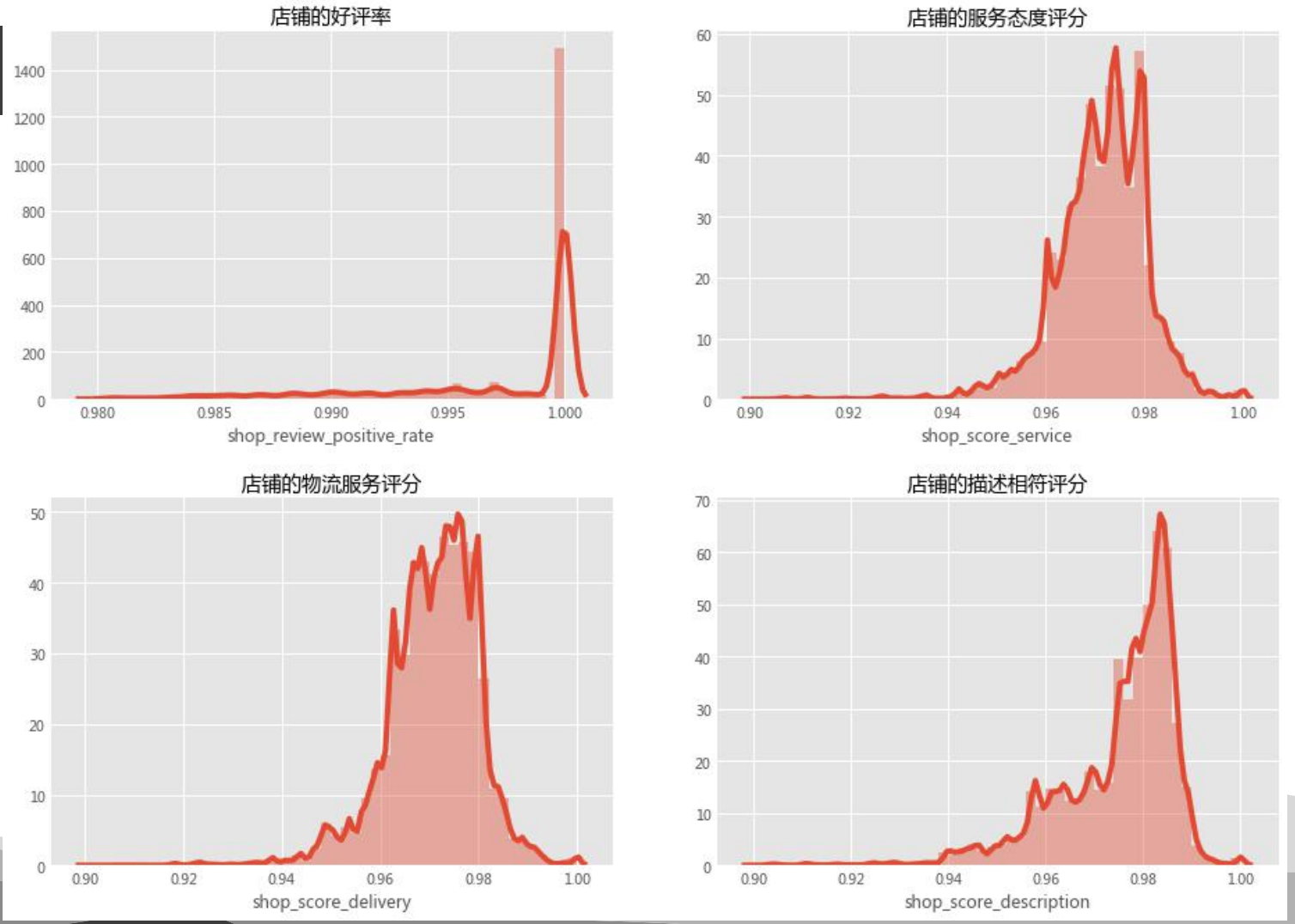
数据集信息及分析——店铺信息

- 店铺评价和星级呈正态分布



数据集信息及分析——店铺信息

- 店铺评价和星级呈正态分布



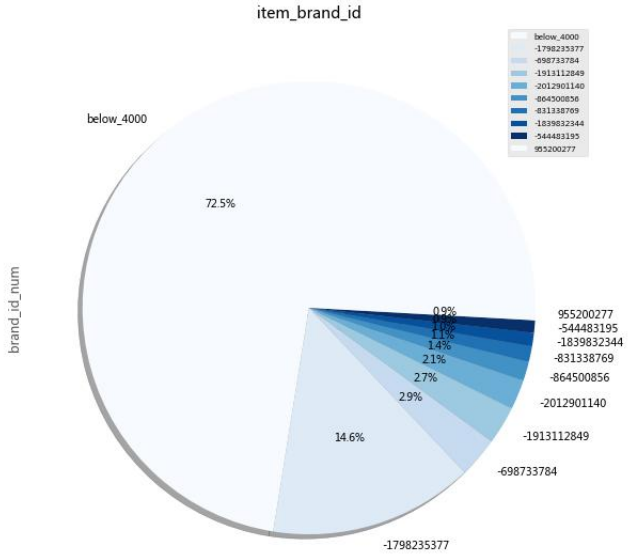
数据集信息及分析——商品信息

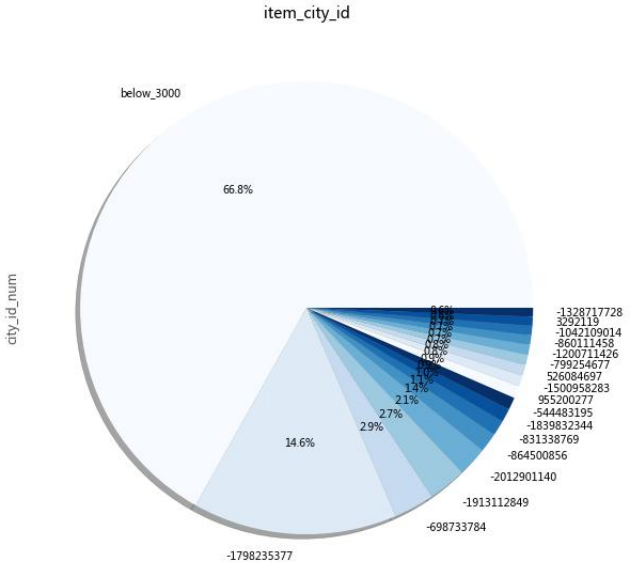
字段	解释
item_id	广告商品编号，Long类型
item_category_list	广告商品的的类目列表，String类型；从根类目（最粗略的一级类目）向叶子类目（最精细的类目）依次排列，数据拼接格式为 "category_0;category_1;category_2"，其中 category_1 是 category_0 的子类目，category_2 是 category_1 的子类目
item_property_list	广告商品的属性列表，String类型；数据拼接格式为 "property_0;property_1;property_2"，各个属性没有从属关系
item_brand_id	广告商品的品牌编号，Long类型
item_city_id	广告商品的城市编号，Long类型
item_price_level	广告商品的价格等级，Int类型；取值从0开始，数值越大表示价格越高
item_sales_level	广告商品的销量等级，Int类型；取值从0开始，数值越大表示销量越大
item_collected_level	广告商品被收藏次数的等级，Int类型；取值从0开始，数值越大表示被收藏次数越大
item_pv_level	广告商品被展示次数的等级，Int类型；取值从0开始，数值越大表示被展示次数越大

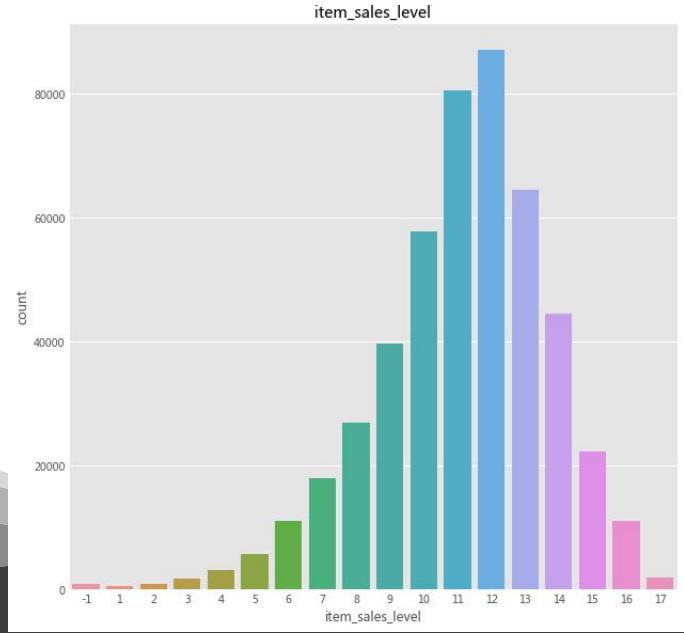
01 任务简介

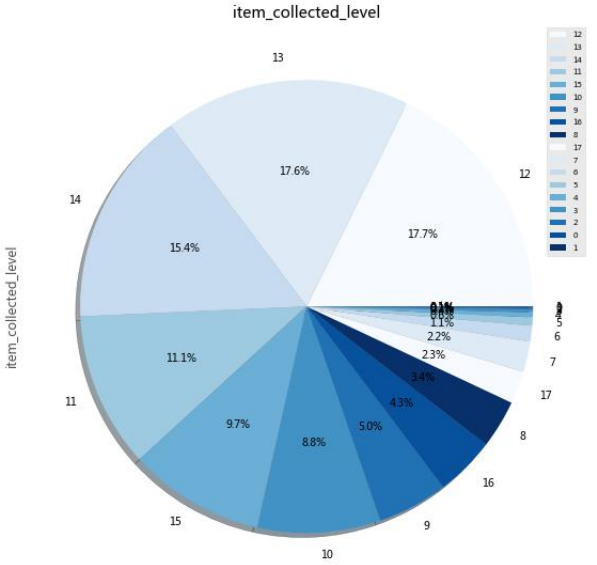
数据集信息及分析——商品信息

- 商品品牌、来源城市亦符合长尾分布，15%的展示商品来自同一品牌







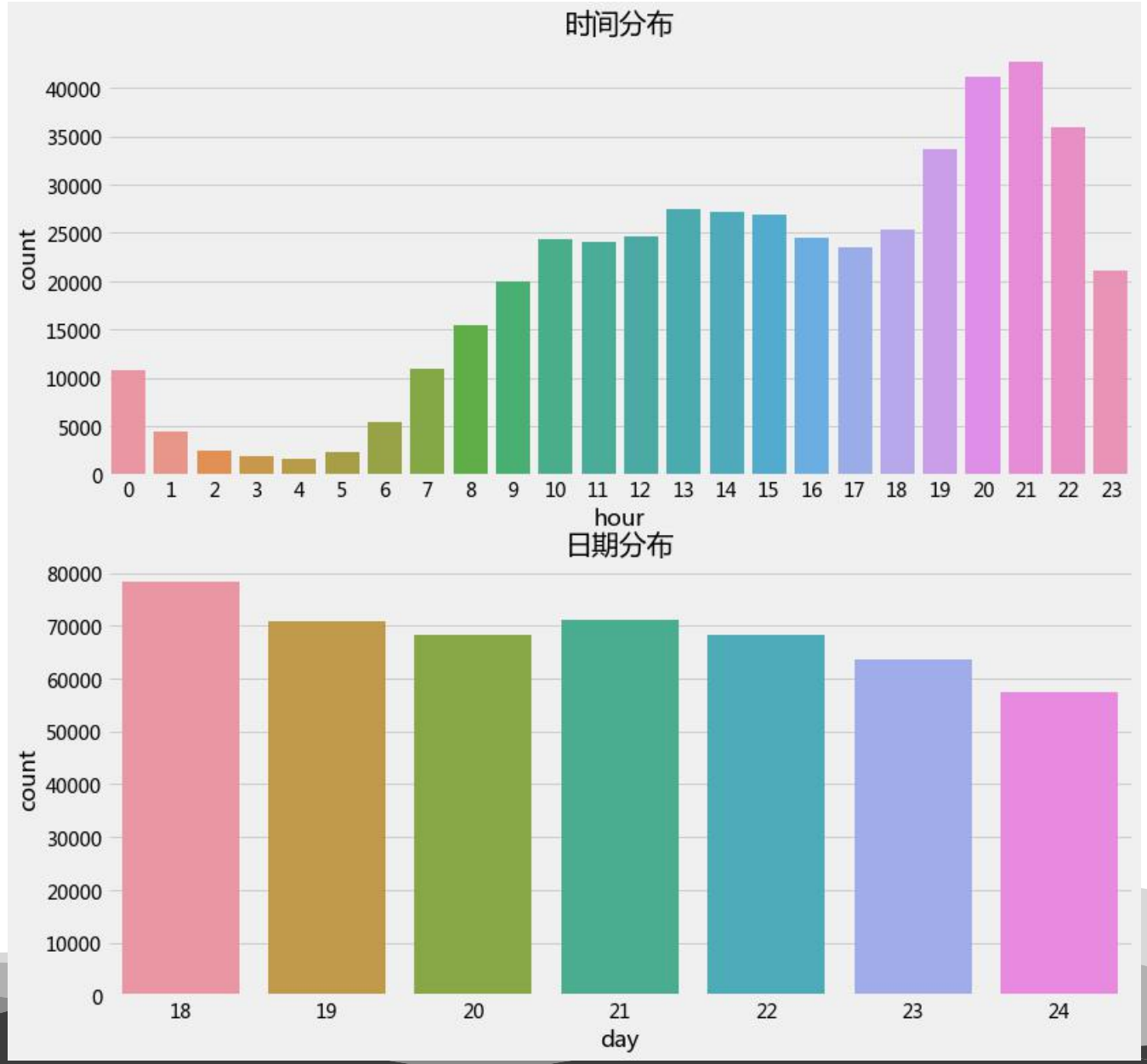


数据集信息及分析——上下文信息

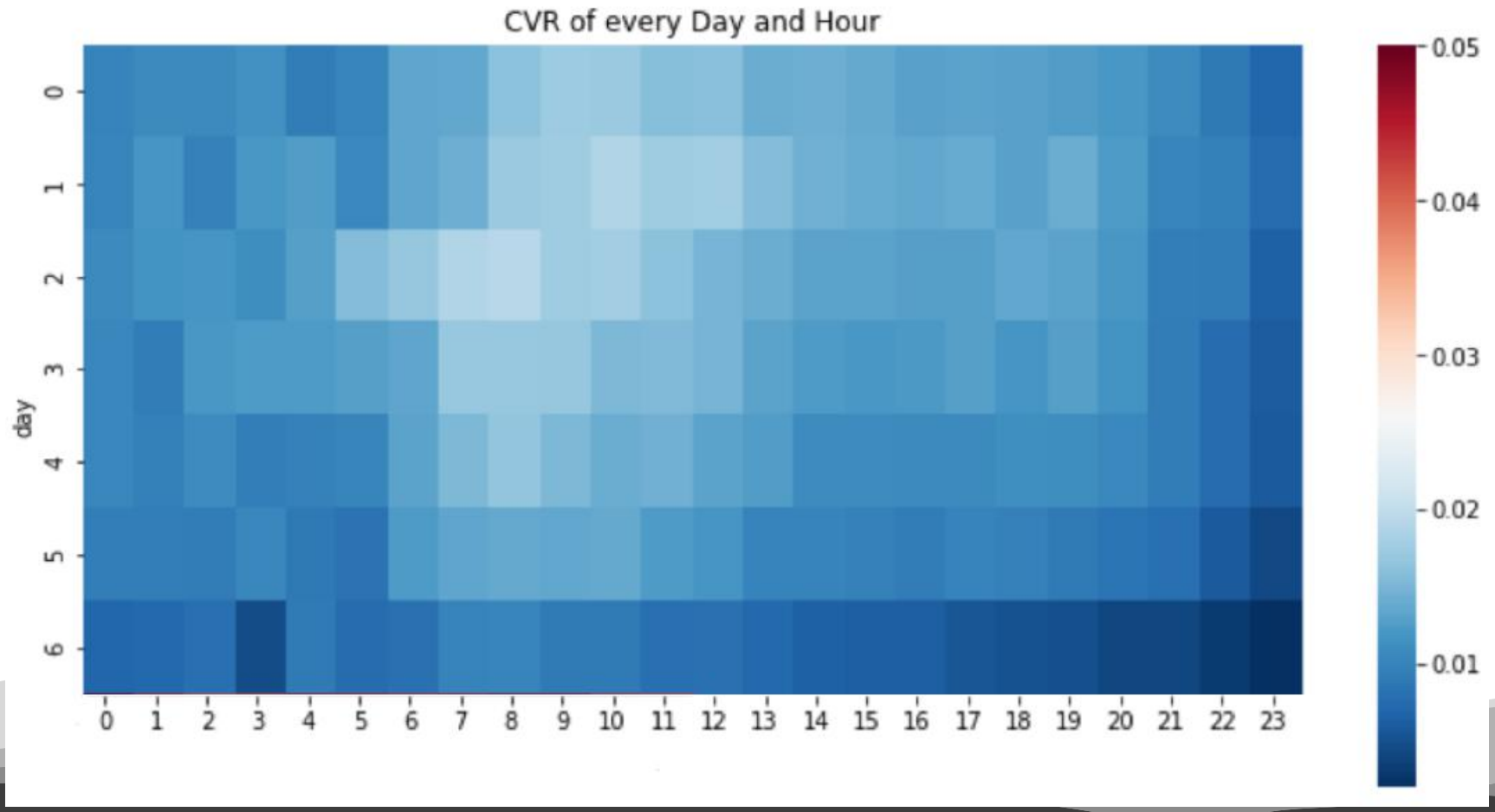
字段	解释
context_id	上下文信息的编号，Long类型
context_timestamp	广告商品的展示时间，Long类型；取值是以秒为单位的Unix时间戳，以1天为单位对时间戳进行了偏移
context_page_id	广告商品的展示页面编号，Int类型；取值从1开始，依次增加；在一次搜索的展示结果中第一屏的编号为1，第二屏的编号为2
predict_category_property	根据查询词预测的类目属性列表，String类型；数据拼接格式为 "category_A:property_A_1,property_A_2,property_A_3;category_B:-1;category_C:property_C_1,property_C_2"，其中 category_A、category_B、category_C 是预测的三个类目；property_B 取值为-1，表示预测的第二个类目 category_B 没有对应的预测属性

数据集信息及分析——上下文信息

- 点击发生时间集中在下午-晚上
- 日期分布较均衡



数据集信息及分析——上下文信息



02

特征处理

02 特征处理

填补缺失值

- 所给数据缺失程度较轻，不必舍去
- 填充缺失值：离散数据用均值填充，连续数据用平均值填充

```python
def __fill_missing_value(self):
    for col in self.int_type_column:
        self.__fill_with_mode(col)
    for col in self.long_type_column:
        self.__fill_with_mode(col)
    for col in self.double_type_column:
        self.__fill_with_mean(col)

def __fill_with_mode(self, column_name):
    mode = self.data[column_name].mode()
    self.data[column_name] = self.data[column_name].replace(-1, mode[0])

def __fill_with_mean(self, column_name):
    mean = self.data[column_name].mean()
    self.data[column_name] = self.data[column_name].replace(-1.0, mean)
```

```
item_id: 0/10074 = 0.0
item_category_list: 0/10074 = 0.0
item_brand_id: 64/10074 = 0.006352987889616835
item_city_id: 22/10074 = 0.002183839587055787
item_price_level: 0/10074 = 0.0
item_sales_level: 236/10074 = 0.02342664284296208
item_collected_level: 0/10074 = 0.0
item_pv_level: 0/10074 = 0.0

user_id: 0/197677 = 0.0
user_gender_id: 5504/197677 = 0.02784340110382088
user_age_level: 439/197677 = 0.0022207945284479225
user_occupation_id: 439/197677 = 0.0022207945284479225
user_star_level: 439/197677 = 0.0022207945284479225

context_id: 0/478043 = 0.0
context_timestamp: 0/478043 = 0.0
context_page_id: 0/478043 = 0.0
predict_category_property: 0/478043 = 0.0

shop_id: 0/3958 = 0.0
shop_review_num_level: 0/3958 = 0.0
shop_review_positive_rate: 5/3958 = 0.0012632642748863063
shop_star_level: 0/3958 = 0.0
shop_score_service: 18/3958 = 0.004547751389590703
shop_score_delivery: 18/3958 = 0.004547751389590703
shop_score_description: 18/3958 = 0.004547751389590703
```

02 特征处理

类别特征处理

- 添加数量特征：类别数目、相同类别的条目数
- 提取最主要的第一个类别
- 类别本身进行label encoding

```
def __process_category(self):
    data = self.data
    data['same_cate'] = data.apply(same_cate, axis=1)  # 相同类别数
    data['same_property'] = data.apply(same_property, axis=1)  # 相同属性数
    data['property_num'] = data['item_property_list'].apply(lambda x: len(x.split(';')))  # 属性的数目
    data['pred_cate_num'] = data['predict_category_property'].apply(lambda x: len(x.split(';')))  # query的类别数目

    def f(x):
        try:
            return len([i for i in reduce((lambda x, y: x + y), [i.split(':')[1].split(',') for i in x.split(';') if
                                                                 len(i.split(':')) > 1]) if i != '-1'])
        except:
            return 0

    data['pred_prop_num'] = data['predict_category_property'].apply(f)  # query的属性数目
    data['predict_query_1'] = data['predict_category_property'].apply(
        lambda x: x.split(';')[0].split(':')[0])  # query第一个类别
    data['predict_query'] = data['predict_category_property'].apply(
        lambda x: '-'.join(sorted([i.split(':')[0] for i in [i for i in x.split(';')]])))  # query的全部类别
    data['item_category_list'] = data['item_category_list'].apply(lambda x: x.split(';')[1])
    self.data = data
```

时间特征处理

- timestamp转化为日期+时间
- 提取时间统计特征：同时段内点击数量、用户平均点击间隔、用户今日点击次数、用户今日点击同类别（同店铺）商品次数
- 泄露特征：距离用户下一次点击的时间（实际实时预测中不可用）

```
data['context_timestamp'] = data['context_timestamp'].apply(
    lambda x: time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(x)))
real_time = pd.to_datetime(data['context_timestamp'])
data['day'] = real_time.dt.day
data['hour'] = real_time.dt.hour
```

03

基础模型

统计学习模型

- 选择多种统计学习基础模型进行试验
- grid_search寻找最佳参数

- 最终选取五种效果最好的模型参与模型融合

```
# %%
lsr = lm.Lasso(alpha=0.0005547)
regr = lm.Ridge(alpha=15.0)
enr = lm.ElasticNet(alpha=0.0009649, l1_ratio=0.2)
svr = svm.SVR(C=200, gamma=0.001)
krr = kernel_ridge.KernelRidge(kernel='polynomial')
gbr = ensemble.GradientBoostingRegressor(
    loss='huber', max_features='sqrt', n_estimators=400)
rfr = ensemble.RandomForestRegressor(n_estimators=90)
xgbr = xgb.XGBRegressor(booster='gbtree',
                        # max_dept=10,
                        learning_rate=0.1,
                        n_estimators=500,
                        subsample=0.9,
                        colsample_bytree=0.8,
                        scale_pos_weight=1)
xgblr = xgb.XGBRegressor(booster='gblinear', n_estimators=300)
lgbr = lgb.LGBMRegressor(num_leaves=6, min_data_in_leaf=12,
                         max_bin=35, learning_rate=0.05, n_estimators=1100)
```

统计学习模型

- 选择的基础模型为Lasso、Ridge、ElasticNet、XGBoost、LightGBM

- svm、tree-based模型计算速度太慢

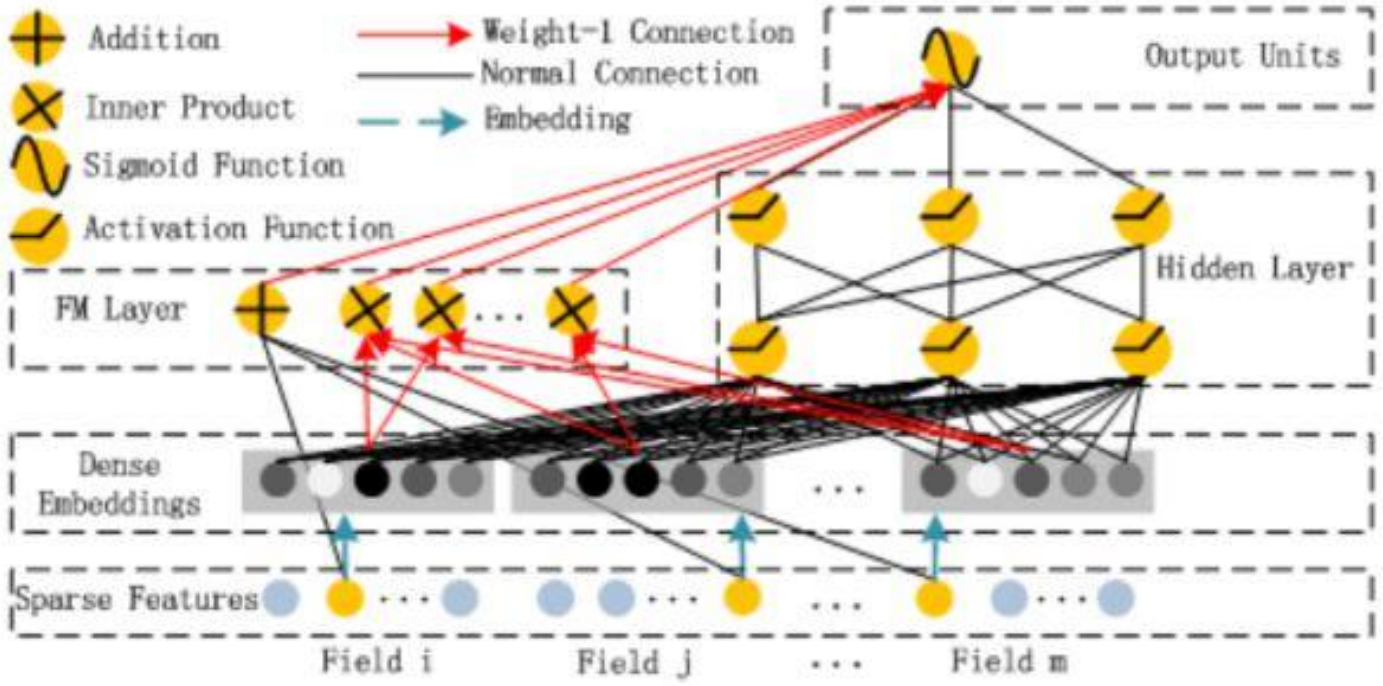
```
[3]  # %% validation...

     lsr  train_log_loss: 0.091530
     lsr  test_log_loss: 0.092977

     regr  train_log_loss: 0.092250
     regr  test_log_loss: 0.093757

     enr  train_log_loss: 0.091917
     enr  test_log_loss: 0.093379

     xgbr  train_log_loss: 0.073725
     xgbr  test_log_loss: 0.095794

     xgblr  train_log_loss: 0.086314
     xgblr  test_log_loss: 0.091017
```

基础pCVR预测模型

- MLP网络
- 输入特征包括三部分的embedding
  - 用户信息（User Profile）： 性别、城市、用户ID
  - 用户行为（User Behavior）：用户最近点击的物品ID序列等
  - 商品信息（Item Profile）：商家ID、品牌ID
- 三者embedding拼接后输入MLP，以PReLU为激活函数，使用sigmod输出预测概率，训练的损失函数为负对数函数

NN-DeepFM

- 将NN的embedding输入部分加入Factorization machines，用于提取一阶特征和二阶交互特征
- Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI’17). AAAI Press, 1725–1731.

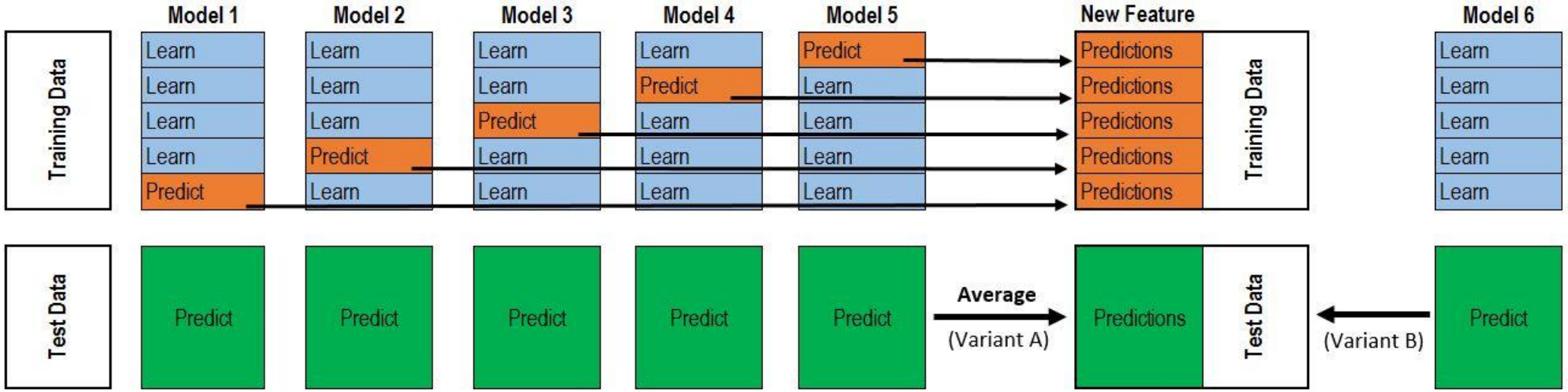


04

模型融合

04 模型融合

Stacking

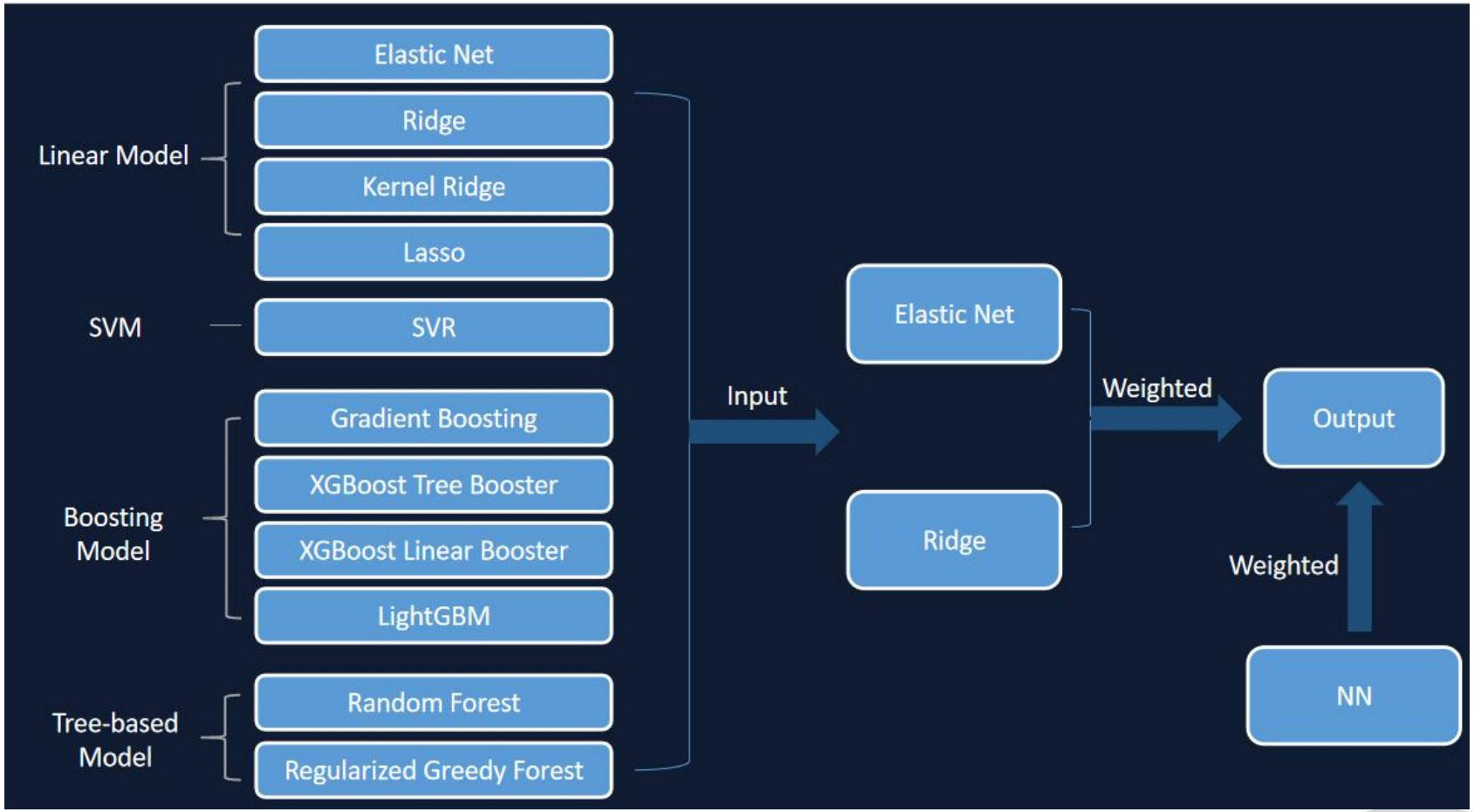


- 单一回归模型效果不够好，使用个三层Stacking模型组合各个模型的结果
  - 整个Stacking的过程类似于CV验证：将训练集分为五份，对每个基本模型进行5轮训练，一次使用其中的4份作为训练集训练，预测余下一份的结果，5轮后得到训练集大小的预测数据，同时在每轮中对测试集进行预测，对每个基本模型来说测试集的预测结果为5轮结果的均值
  - 在第二层中，输入(训练集上的预测结果 * 基本模型数量)的数据进行训练
  - 第三层最终输出为第二层的预测结果

04 模型融合

加入NN结果

- 将NN得到的结果同Stacking模型加权加和（0.2/0.8）
- NN无法参与Stacking（训练时间太久）



05

实验及结论

实验结果

model	neg_log_loss
Lasso	0.09298
Ridge	0.09376
Elastic Net	0.09338
XGBoost-Tree	0.09579
LightGBM	0.091017
NN(DeepFM)	0.09222
Staking(stastic model)	0.08958
Staking(stastic model) + NN(DeepFM)	0.08919

结论与总结

- 就竞赛来说，特征工程还有很多欠缺，交互特征还有许多可以提取的，如
  - 冷启动特征：用户、商品、店铺是否第一次交互等
  - 历史统计特征：用户、商品、品牌等一阶特征及二阶交叉特征的点击、转化、转化率
- 数据中存在大量的leak信息，和实际的pCVR预测有较大的差别
  - 特征工程得到的交互特征工程中无法实时提取
  - NN模型在竞赛中难有好的效果，数据量相对是不够的（400k）
  - DeepFM模型有调优的余地，如embedding方式、regularzation

感谢聆听！

刘德欣 1901110660

王文祥 1901210258