In [ ]:

```
### 推荐服务
'''
- 离线推荐
  - 先召回对召回结果排序
  - 为每一个用户都进行召回并排序的过程并且把拍好顺序的结果放到数据库中
  - 如果需要推荐结果的时候 直接到数据库中按照user_id查询，返回推荐结果
  - 优点 结构比较简单 推荐服务只需要不断计算，把结果保存到数据库中即可
  - 缺点 实时性查 如果数据1天不更新 1天之内推荐结果一样的，不能反映用户的实时兴趣
- 实时推荐
  - 排序的模型加载好
  - 召回阶段的结果缓存
  - 所有用户的特征缓存
  - 所有物品的特征缓存
  - 把推荐的服务暴露出去（django flask）需要推荐结果的服务把 用户id 传递过来
    - 根据id 找到召回结果
    - 根据id 找到缓存的用户特征
    - 根据召回结果的物品id 找到物品的特征
    - 用户特征+物品特征-》逻辑回归模型 就可以预测点击率
    - 所有召回的物品的点记率都预测并排序 推荐topN
    - 实时通过LR模型进行排序的好处
      - 随时修改召回集
      - 随时调整用户的特征
      - 当用户需要推荐服务的时候，获取到最新的召回集和用户特征 得到最新的排序结果 更能
'''
```

实时产生推荐结果

CTR预测模型+特征==>预测结果==>TOP-N列表

- CTR预测模型在离线阶段已经训练完成，此处仅需加载
- 特征是：用户实时特征 + （该用户对应的）召回集（离线召回+在线召回）中物品的特征
  - 在线召回：用户刚买了某个种类的物品，就随机取出 该类中 若干个物品，放入召回集中
- 预测结果：计算出点击率，排序，得到，例如top20

In [2]:

```
import os
# 配置pyspark和spark driver运行时 使用的python解释器
JAVA_HOME = '/root/bigdata/jdk'
PYSPARK_PYTHON = '/miniconda2/envs/py365/bin/python'
# 当存在多个版本时，不指定很可能会导致出错
os.environ['PYSPARK_PYTHON'] = PYSPARK_PYTHON
os.environ['PYSPARK_DRIVER_PYTHON'] = PYSPARK_PYTHON
os.environ['JAVA_HOME'] = JAVA_HOME
# 配置spark信息
from pyspark import SparkConf
from pyspark.sql import SparkSession

SPARK_APP_NAME = 'OnlineRecommendation'
SPARK_URL = 'spark://192.168.58.100:7077'

conf = SparkConf()
config = (
    ('spark.app.name',SPARK_APP_NAME),
    ('spark.executor.memory','2g'),
    ('spark.master',SPARK_URL),
    ('spark.executor.cores','2')
)
conf.setAll(config)

spark = SparkSession.builder.config(conf=conf).getOrCreate()
```

In [ ]:

```
# 以下数据来自第八小节
'''热编码中：
"pvalue_level"特征对应关系:
+-----------+--------------------+
|pvalue_level|pl_onehot_feature     |
+-----------+--------------------+
|          -1|                 0.0|
|           3|                 3.0|
|           1|                 2.0|
|           2|                 1.0|
+-----------+--------------------+

 “new_user_class_level”的特征对应关系：
+-------------------+----------------------+
|new_user_class_level|nucl_onehot_feature     |
+-------------------+----------------------+
|                  -1|                   0.0|
|                   3|                   2.0|
|                   1|                   4.0|
|                   4|                   3.0|
|                   2|                   1.0|
+-------------------+----------------------+
'''
pvalue_level_rela = {-1: 0, 3:3, 1:2, 2:1}
new_user_class_level_rela = {-1:0, 3:2, 1:4, 4:3, 2:1}
'''
"cms_group_id"特征对应关系：
+-----------+------------------------+
|cms_group_id|min(cms_group_id_feature)|
+-----------+------------------------+
|           7|                     9.0|
|          11|                     6.0|
|           3|                     0.0|
|           8|                     8.0|
|           0|                    12.0|
|           5|                     3.0|
|           6|                    10.0|
|           9|                     5.0|
|           1|                     7.0|
|          10|                     4.0|
|           4|                     1.0|
|          12|                    11.0|
|           2|                     2.0|
+-----------+------------------------+
'''
cms_group_id_rela = {
    7: 9,
    11: 6,
    3: 0,
    8: 8,
    0: 12,
    5: 3,
    6: 10,
    9: 5,
    1: 7,
    10: 4,
```

```
    4: 1,
    12: 11,
    2: 2
}
'''
"final_gender_code"特征对应关系：
+-----------------+-----------------------------+
|final_gender_code|min(final_gender_code_feature)|
+-----------------+-----------------------------+
|                1|                          1.0|
|                2|                          0.0|
+-----------------+-----------------------------+
'''
final_gender_code_rela = {1:1, 2:0}
'''
"age_level"特征对应关系：
+---------+---------------------+
|age_level|min(age_level_feature)|
+---------+---------------------+
|        3|                  0.0|
|        0|                  6.0|
|        5|                  2.0|
|        6|                  5.0|
|        1|                  4.0|
|        4|                  1.0|
|        2|                  3.0|
+---------+---------------------+
'''
age_level_rela = {3:0, 0:6, 5:2, 6:5, 1:4, 4:1, 2:3}

'''
"shopping_level"特征对应关系：
|shopping_level|min(shopping_level_feature)|
+--------------+--------------------------+
|             3|                       0.0|
|             1|                       2.0|
|             2|                       1.0|
+--------------+--------------------------+
'''
shopping_level_rela = {3:0, 1:2, 2:1}
'''
"occupation"特征对应关系：
+----------+----------------------+
|occupation|min(occupation_feature)|
+----------+----------------------+
|         0|                   0.0|
|         1|                   1.0|
+----------+----------------------+
'''
occupation_rela = {0:0, 1:1}

pid_rela = {
    "430548_1007": 0,
    "430549_1007": 1
}

# key是原始数据，value是StringIndexe编码后的值1，值1经过onehot编码成为独热编码
```

In [38]:

```
## 特征获取
import redis
import json
import pandas as pd
from pyspark.ml.linalg import DenseVector


def create_datasets(userId, pid):
    client_of_recall = redis.StrictRedis(host="192.168.58.100", port=6379, db=9)
    client_of_features = redis.StrictRedis(host="192.168.58.100", port=6379, db=10)
    # 获取用户特征
    user_feature = json.loads(client_of_features.hget("user_features", userId).decode

    # 获取用户召回集
    recall_sets = client_of_recall.smembers(userId)

    result = []


    # 遍历召回集
    for adgroupId in recall_sets:
        adgroupId = int(adgroupId)
        # 获取该广告的特征值 price
        ad_feature = json.loads(client_of_features.hget("ad_features", adgroupId).dec

        features = {}
        features.update(user_feature)
        features.update(ad_feature)

        for k,v in features.items():
            if v is None:
                features[k] = -1

        features_col = [
            # 特征值
            "price",
            "cms_group_id",
            "final_gender_code",
            "age_level",
            "shopping_level",
            "occupation",
            "pid",
            "pvalue_level",
            "new_user_class_level"
        ]
        '''
        "cms_group_id", 类别型特征，约13个分类 ==> 13维
        "final_gender_code", 类别型特征，2个分类 ==> 2维
        "age_level", 类别型特征，7个分类 ==>7维
        "shopping_level", 类别型特征，3个分类 ==> 3维
        "occupation", 类别型特征，2个分类 ==> 2维
        '''

        price = float(features["price"])

        pid_value = [0 for i in range(2)]#[0,0]
```

```
        cms_group_id_value = [0 for i in range(13)]
        final_gender_code_value = [0 for i in range(2)]
        age_level_value = [0 for i in range(7)]
        shopping_level_value = [0 for i in range(3)]
        occupation_value = [0 for i in range(2)]
        pvalue_level_value = [0 for i in range(4)]
        new_user_class_level_value = [0 for i in range(5)]

        pid_value[pid_rela[pid]] = 1
        cms_group_id_value[cms_group_id_rela[int(features["cms_group_id"])]] = 1
        final_gender_code_value[final_gender_code_rela[int(features["final_gender_code
        age_level_value[age_level_rela[int(features["age_level"])]] = 1
        shopping_level_value[shopping_level_rela[int(features["shopping_level"])]] =
        occupation_value[occupation_rela[int(features["occupation"])]] = 1
        pvalue_level_value[pvalue_level_rela[int(features["pvalue_level"])]] = 1
        new_user_class_level_value[new_user_class_level_rela[int(features["new_user_cl
#         print(pid_value)
#         print(cms_group_id_value)
#         print(final_gender_code_value)
#         print(age_level_value)
#         print(shopping_level_value)
#         print(occupation_value)
#         print(pvalue_level_value)
#         print(new_user_class_level_value)

        vector = DenseVector([price] + pid_value + cms_group_id_value + final_gender_
        + age_level_value + shopping_level_value + occupation_value + pvalue_level_va

        result.append((userId, adgroupId, vector))

    return result
# 举例看看用户88 广告资源位"430548_1007" 对应的、召回的500条广告 的特征向量
create_datasets(88, "430548_1007")
```

...

In [34]:

```
# 加载训练好的逻辑回归模型
from pyspark.ml.classification import LogisticRegressionModel
CTR_model = LogisticRegressionModel.load('/models/CTRModel_AllOneHot.obj')
```

In [37]:
```
1 import pandas as pd
2 pdf = pd.DataFrame(create_datasets(8,'430548_1007'),columns=["userId", "adgroupId", "
3 pdf
```

Out[37]:

	userId	adgroupId	features
0	8	568198	[11.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,...
1	8	284442	[1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
2	8	40366	[4.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
3	8	446656	[32.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,...
4	8	132372	[5.599999904632568, 1.0, 0.0, 1.0, 0.0, 0.0, 0...
...	...	...	...
495	8	238926	[9.800000190734863, 1.0, 0.0, 1.0, 0.0, 0.0, 0...
496	8	13307	[78.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,...
497	8	693981	[23.100000381469727, 1.0, 0.0, 1.0, 0.0, 0.0, ...
498	8	258692	[1.600000023841858, 1.0, 0.0, 1.0, 0.0, 0.0, 0...
499	8	708841	[19.600000381469727, 1.0, 0.0, 1.0, 0.0, 0.0, ...

500 rows × 3 columns

In [41]:
```
datasets = spark.createDataFrame(pdf)
datasets.show()
```

```
+------+---------+--------------------+
|userId|adgroupId|            features|
+------+---------+--------------------+
|     8|   568198|[11.0,1.0,0.0,1.0...|
|     8|   284442|[1.0,1.0,0.0,1.0,...|
|     8|    40366|[4.0,1.0,0.0,1.0,...|
|     8|   446656|[32.0,1.0,0.0,1.0...|
|     8|   132372|[5.59999990463256...|
|     8|   424510|[12.8000001907348...|
|     8|   255632|[0.5,1.0,0.0,1.0,...|
|     8|   136509|[8.60000038146972...|
|     8|   143566|[29.6000003814697...|
|     8|   198254|[2.20000004768371...|
|     8|   103023|[28.0,1.0,0.0,1.0...|
|     8|   262373|[100.0,1.0,0.0,1....|
|     8|    63803|[15.0,1.0,0.0,1.0...|
|     8|    12052|[59.0,1.0,0.0,1.0...|
|     8|   538461|[20.7999992370605...|
|     8|    84047|[9.0,1.0,0.0,1.0,...|
|     8|   184721|[10.0,1.0,0.0,1.0...|
|     8|   157274|[38.0,1.0,0.0,1.0...|
|     8|    77436|[29.0,1.0,0.0,1.0...|
|     8|   252225|[12.5,1.0,0.0,1.0...|
+------+---------+--------------------+
only showing top 20 rows
```

In [42]:
```
prediction = CTR_model.transform(datasets).sort('probability')
prediction.show()
```

```
+------+---------+--------------------+--------------------+--------------------+------
----+
|userId|adgroupId|            features|       rawPrediction|         probability|predic
tion|
+------+---------+--------------------+--------------------+--------------------+------
----+
|     8|   202173|[1888.0,1.0,0.0,1...|[2.69017894066573...|[0.93644463234420...|
0.0|
|     8|   241175|[1800.0,1.0,0.0,1...|[2.69017975515559...|[0.93644468081943...|
0.0|
|     8|   247128|[1350.0,1.0,0.0,1...|[2.69018392016059...|[0.93644492870359...|
0.0|
|     8|   788867|[1220.0,1.0,0.0,1...|[2.69018512338425...|[0.93644500031440...|
0.0|
|     8|   730074|[800.0,1.0,0.0,1....|[2.69018901072224...|[0.93644523167188...|
0.0|
|     8|   397105|[800.0,1.0,0.0,1....|[2.69018901072224...|[0.93644523167188...|
0.0|
|     8|   845130|[520.0,1.0,0.0,1....|[2.69019160228090...|[0.93644538590977...|
0.0|
|     8|   295744|[500.0,1.0,0.0,1....|[2.69019178739224...|[0.93644539692675...|
0.0|
|     8|   296690|[418.0,1.0,0.0,1....|[2.69019254634870...|[0.93644544209635...|
0.0|
|     8|     2267|[350.0,1.0,0.0,1....|[2.69019317572724...|[0.93644547955404...|
0.0|
|     8|   603882|[299.0,1.0,0.0,1....|[2.69019364776113...|[0.93644550764729...|
0.0|
|     8|   627350|[299.0,1.0,0.0,1....|[2.69019364776113...|[0.93644550764729...|
0.0|
|     8|    24364|[278.0,1.0,0.0,1....|[2.69019384212803...|[0.93644551921510...|
0.0|
|     8|   270625|[256.0,1.0,0.0,1....|[2.69019404575050...|[0.93644553133375...|
0.0|
|     8|   176624|[248.0,1.0,0.0,1....|[2.69019411979503...|[0.93644553574053...|
0.0|
|     8|   747336|[245.0,1.0,0.0,1....|[2.69019414756173...|[0.93644553739308...|
0.0|
|     8|    24209|[238.0,1.0,0.0,1....|[2.69019421235070...|[0.93644554124901...|
0.0|
|     8|   289624|[228.0,1.0,0.0,1....|[2.69019430490637...|[0.93644554675749...|
0.0|
|     8|   385931|[195.0,1.0,0.0,1....|[2.69019461034007...|[0.93644556493546...|
0.0|
|     8|    44235|[178.0,1.0,0.0,1....|[2.69019476768470...|[0.93644557429987...|
0.0|
+------+---------+--------------------+--------------------+--------------------+------
----+
only showing top 20 rows
```

In [46]:
```
# 为 8号用户 推荐的top20物品
print(prediction.select('adgroupId').head(20))
print([i.adgroupId for i in prediction.select('adgroupId').head(20)])
```

```
[Row(adgroupId=202173), Row(adgroupId=241175), Row(adgroupId=247128), Row(adgroupId=788867), Row(adgroupId=397105), Row(adgroupId=730074), Row(adgroupId=845130), Row(adgroupId=295744), Row(adgroupId=296690), Row(adgroupId=2267), Row(adgroupId=627350), Row(adgroupId=603882), Row(adgroupId=24364), Row(adgroupId=270625), Row(adgroupId=176624), Row(adgroupId=747336), Row(adgroupId=24209), Row(adgroupId=289624), Row(adgroupId=385931), Row(adgroupId=44235)]
[202173, 241175, 247128, 788867, 397105, 730074, 845130, 295744, 296690, 2267, 603882, 627350, 24364, 270625, 176624, 747336, 24209, 289624, 385931, 600455]
```