ctrip tech 携程技术中心

ctrip tech 携程技术中心 | IT大咖说 知识分享平台

携程技术沙龙

去哪儿酒店算法服务

张中原

张中原

- 2008 哈工大
- 2011 去哪儿网
- 从事交易系统、酒店数据、公司基础平台组件、存储与监控等工作
- 关注工程师效率和质量、系统实际效益

目录
CONTENTS

算法平台简介

— 先解决部署的问题

算法部署

模型上线、迭代

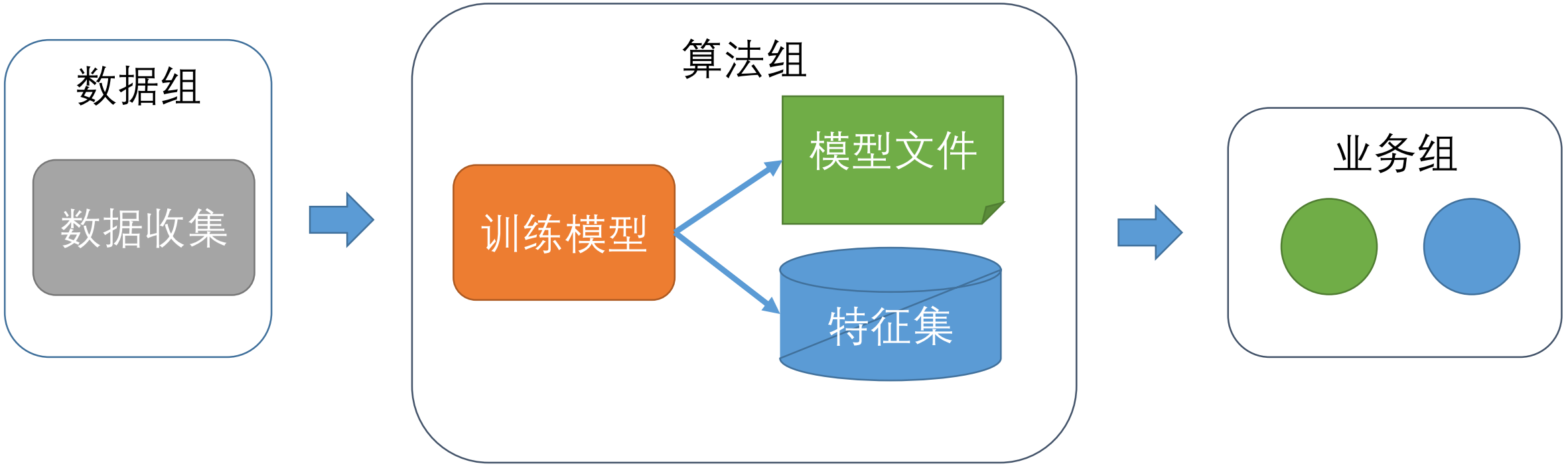
数据集合

用户、酒店等

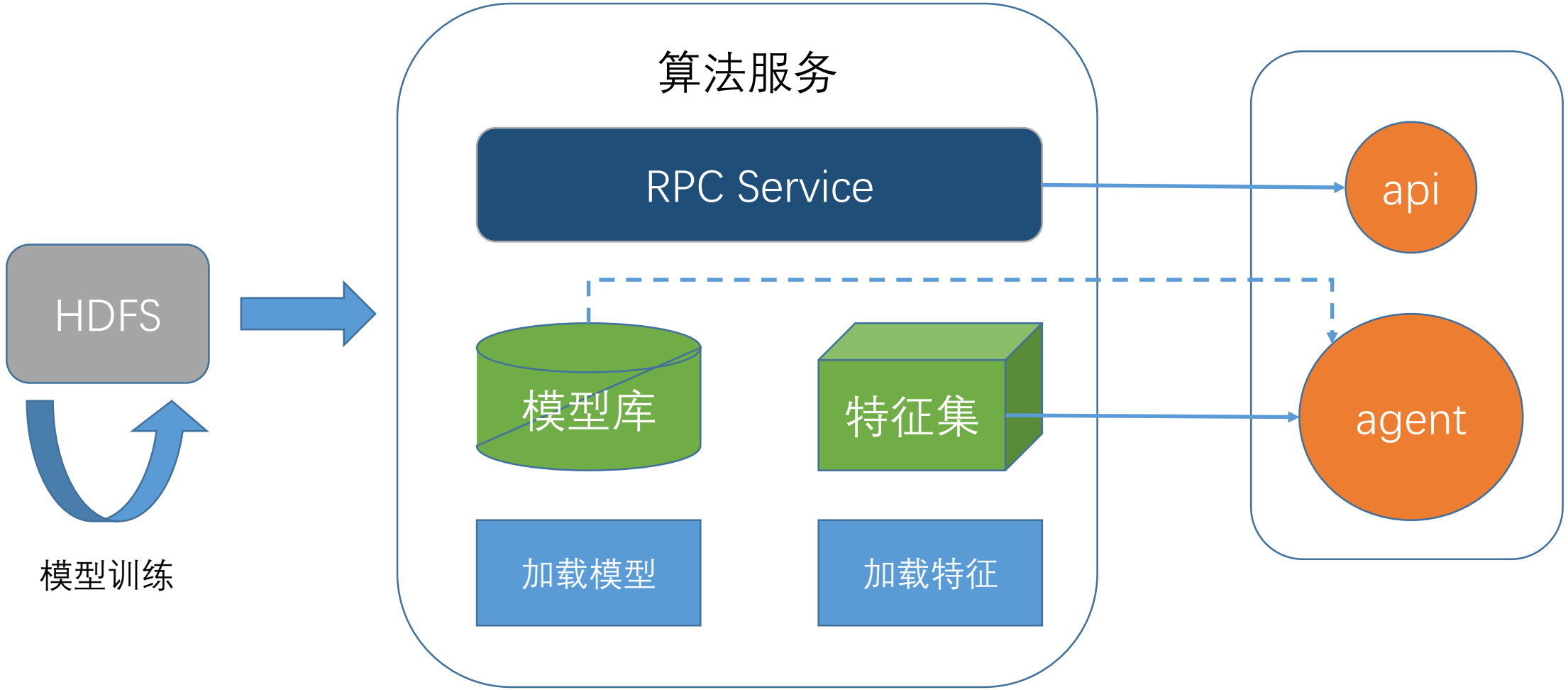
训练平台

算法快速评测

- 9个应用
- 530个模型 共30564个
- 120G 特征数据
- QPS 20+ 由业务系统决定
- P98 10ms MEAN 5ms
- 支持任意类型运算封装，不仅是模型
- 算法一次发布随时变更，对业务零影响
- 特征转换可随时调整，在线离线逻辑统一


数据组
数据收集
算法组
训练模型
模型文件
特征集
业务组

- 没有标准规范，管理乱，效率低
- 训练预测时特征转换独立，重复开发或逻辑不一致
- 模型之间共同功能复用率低
- 算法变更影响大，不兼容
- 跟踪调试复杂


算法服务
RPC Service
HDFS
模型训练
模型库
特征集
加载模型
加载特征
api
agent

- 托管维护模型文件和特征集
- 自动加载模型与特征数据
- 提供统一调用接口
- 使模型预测无缝接入应用
- 算法随时更新迭代对业务系统完全透明

算法定义

- 算法版本迭代
- 城市粒度控制
- 适应不同模型类别
- 特征转换逻辑
- 分类与回归

```
interface Algorithm<R> {
    String getApp()
    String getVersion()
    <P, F> FeatureResolver<P, F> getFeatureResolver()
    <F, R> Evaluator<F, R> getEvaluator()
    ListenableFuture<ResultValue<R>> eval(Request request)
            throws AlgorithmException
}
interface AlgorithmFactory {
    <R> Algorithm<R> getOrCreate(app, version, filter)
}
```

```
// 回归
interface ResultValue<R> {
    R getValue()
    Map<String, Object> getContext()  // 调试用
}

// 分类
interface HashProbability<R> extends ResultValue<R> {
    Set<String> getCategoryValues()
    Double getProbability(String key)
}
```

```
ListenableFuture<ResultValue<T>> eval(Request request) {
    checkUpdate()  // 触发 模型文件 异步更新检查

    return executor.submit(Callable) () -> {
        RequestContext ctx = RequestContext.create(request)
        try {
            return evaluator.eval(resolver.resolve(request.example))
                            .setContxt(ctx.getValue())
        } finally {
            RequestContext.remove()
        }
    }
}
```

模型运算

模型类别	优点	缺点
PMML	标准、平台无关、功能全、用户多	略慢
Vopal Wabbit	快、模型小	C++
DataProc	未知	旧系统兼容
自定义模型	强大、灵活	额外开发

```
interface Evaluator<F, R> {
    ResultValue<R> eval(F resolved)
}

EvaluatorFactory {
    Evalutor create(config, filter, model) {
        Evaluator instance = ... // config.type
        // KVStoreSupport          - 需要访问特征数据
        // AlgorithmFactoryAware - 需要子模型支持
        // ...
    }
}
```

```
class VWEvaluator implements Evaluator<FeatureVector,Double> {

    final VWModel model;  // 自实现Java版

    ResultValue<Double> eval(FeatureVector vector) {
        RequestContext ctx = RequestContext.get();
        ctx.set("trace1", "xxxx") // request.debug=true时有效
        ctx.getRequest()          // 原始Request

        return new SimpleResultValue(model.predict(vector))
    }
```

```
class PMMLEvaluator implements Evaluator<Map, T> {

    final PMML model;  // JPMML

    HashProbability<T> eval(Map map) {
        ModelEvaluator e = new MiningModelEvaluator(pmml);
        …
        return new ProbabilityDistribution(result, probabilities)
    }
}
```

```
class XXXEvaluator implements Evaluator<F, R> {
    final PMML pmml;
    final VWModel vw;

    ResultValue<R> eval(F feature) {
        // ResultValue value = pmml.eval(feature)
        // 附加逻辑
        // vw.eval(value)
        // …
    }
}
```

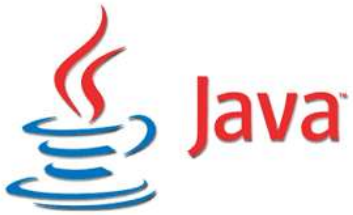
特征处理

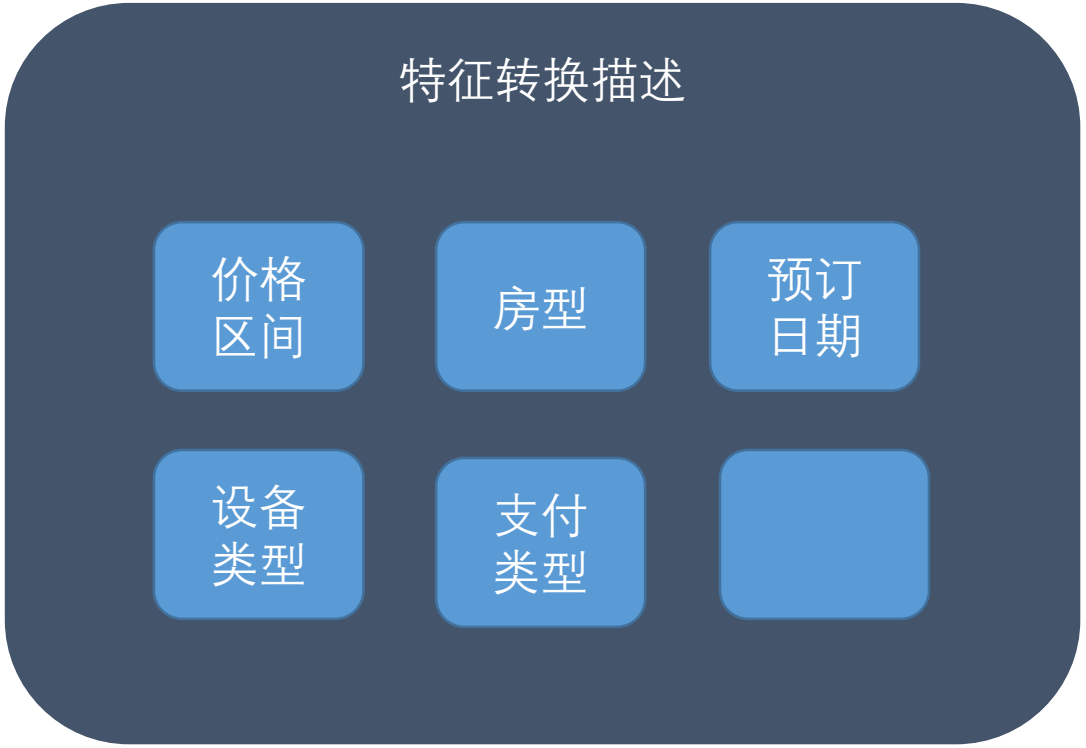
转换场景	技术选型	问题
模型训练	Python, shell, hive等	不易管理 沟通不畅导致应用时逻辑不一致
模型应用	Java	每个模型都要开发 与训练逻辑相同时 无法重用 更新调整慢 影响大

- 特征转换与模型绑定，便于管理
- 统一转换逻辑，使训练预测保持一致
- 在线离线统一转换配置
- 可随时调整生效，降低影响
- 方便跟踪调试


HIVE


APACHE
Spark

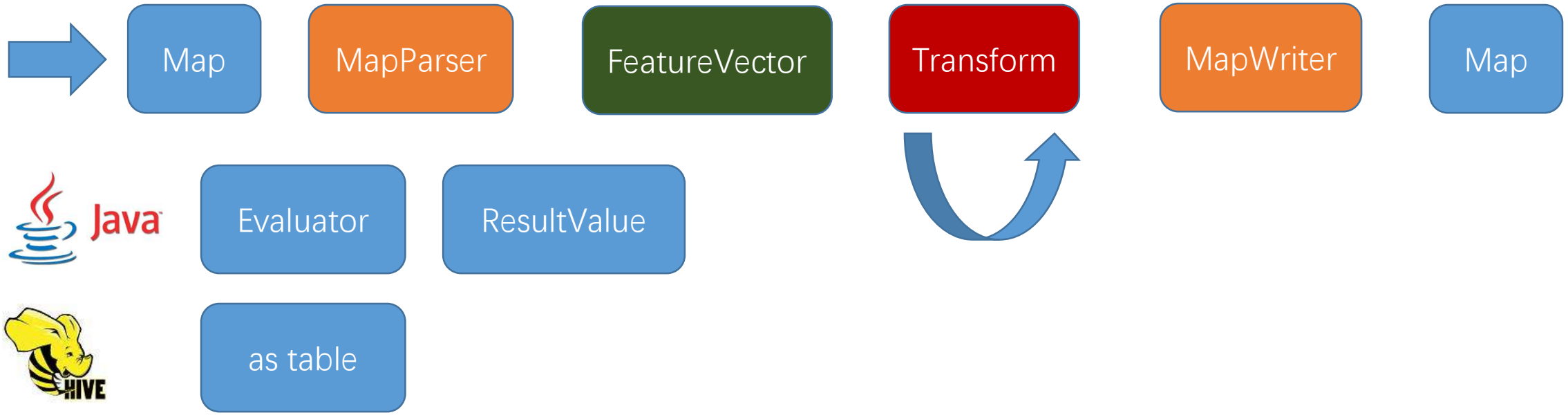

Java


特征转换描述
价格区间
房型
预订日期
设备类型
支付类型

```
FetureVector {
    map<string, set<string>>          stringFeatures
    map<string, map<string, double>> floatFeatures
    map<string, list<double>>         denseFeatures
    map<string, map<string, string>> rawFeatures
}

interface Transform {
    void doTransform(FeatureVector featureVector)
}
```

各种场景使用时，统一转化为FeatureVector

- FeatureVectorParser 支持Map, List(hive用)等
- FeatureVectorWriter 支持None, Map等



```python
def category_temperature(value):
    if not value:
        return '未知'
    if value < 10:
        return '冷'
    if value < 30:
        return '良'
    else:
        return '热'
```

```java
private String category(Double value) {
    if (value == null)
        return "未知";
    if (value < 10):
        return "冷";
    if (value < 30):
        return "良";
    else
        return "热";
}
```

```
category_temperature {
    transform: category
    keys: [temperature]
    output: all
    outputKey: $key          # 支持$key, $value, $category, 自定义
    outputValue: $category # 同上，可根据需要扩展
    categories: {
        ''     : 未知
        '<10'  : 冷
        '<30'  : 良
        '>=30' : 热
    }
}
```

转换器	作用
default	设置缺省值
category	分类转换，数值归一等
cross	向量交叉运算
store	访问特征集进行数值展开
encode	模型运算前置编码转换
log	输出转换中间状态
trace ?	反馈用户到模型

```
class TransformFeatureResolver<P, F> {
    Transform transform = TransformFactory.createTransform(config.transform)

    public F resolve(P param) {
        FeatureVector vector = parser.parse(P param)
        transform.doTransform(vector)
        return writer.write(vector)
    }
}
```

```
add jar hdfs://.../udfs/algorithm-hive-1.1.0-jar-with-dependencies.jar;
CREATE TEMPORARY FUNCTION transform_func
    AS 'com.qunar.hotel.algorithm.store.udf.FeatureTransformToStringUDF';

INSERT overwrite table dm_user_features_v2 partition(dt='$DATE')
select uid, transform_func(
'remote:///vw_detail_price     # 开发测试时可用 file:/// , hdfs:///
?transformer=transformer_user
&writer=writer_user
&parser=parser_user',*,dt)
from tmp_user_portrait_v2
where dt='$DATE' and uid is not null and trim(uid)<>'';
```

代理与服务

方式	优点	缺点
代理	性能略好 可独立部署	消耗业务系统资源 监控联调复杂 变更影响大
服务	系统边界清晰 变更扩展灵活 应用接入便捷 跟踪监控统一	RPC消耗

- 节省内存
- 提升性能

```
abstract AlgorithmCache implements AlgorithmFactory {
    LoadingCache<AV, Config> configCache
    LoadingCache<AVF, Algorithm> algorithmCache
    LoaindgCache<AVF, Algorithm> fallbackAlgorithmCache

    abstract Algorithm createAlgorithm(app, version, filter)
}
```

```
AlgorithmBuilder extends AlgorithmCache {
  Algorithm createAlgorithm(app, version, filter) {
    Config config ...                 // cached remote config
    FeatureResolver resolver ...      // config.transform
    Algorithm algorithm ...           // config, resolver, executor
    CachedRemoteFile model ...        // app, version, filter

    model.addListener(cached => { // auto update
        algorithm.setEvaluator (
            createEvaluator(config.type, cached.file))
    }
    if (!lazyInit)                    // lazy init
        model.update()
    algorithm.model = model
```

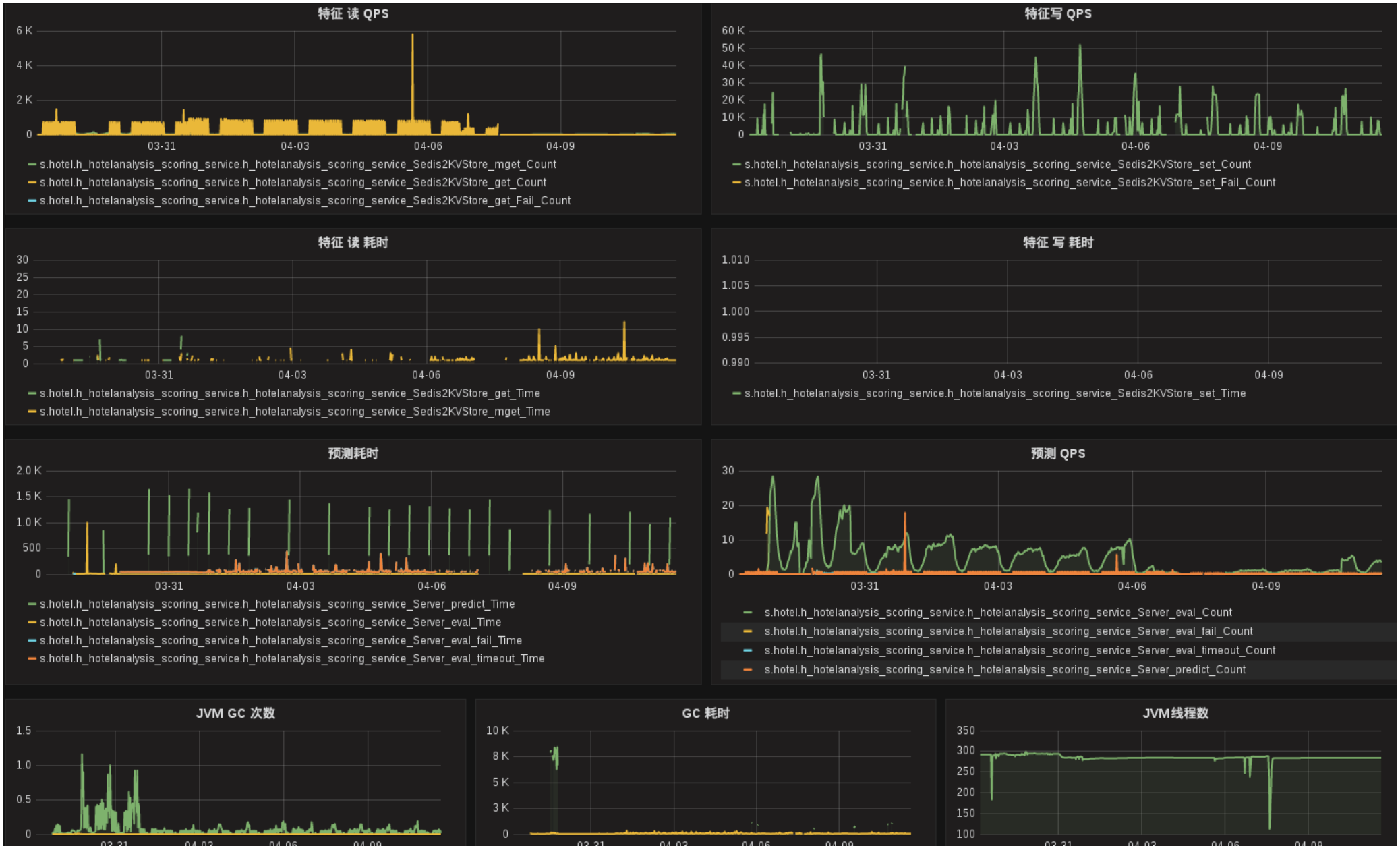
```
AlgorithmBuilder builder = AlgorithmBuilder.create()
                .setServer(..) // config server
                .setStore(..)  // kv store (redis)
                .setLazyInit(..)
                .setExecutor(..)

Algorithm algorithm = builder.getOrCreate(app, version, filter)
```

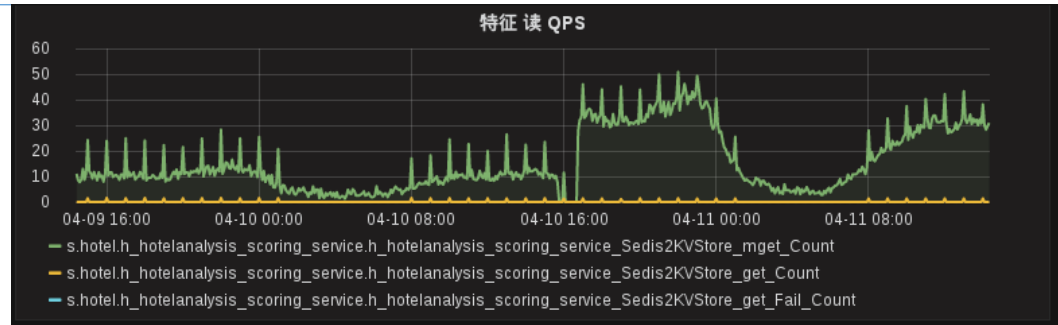
```
interface AlgorithmService {
    <R> ResultValue<R> eval(Request request);
}

<dubbo:reference interface='com.qunar.hotel.algoritym.AlgorithmService'/>

// 同步
AlgorithmService sevice = ..
ResultValue<R> result = service.eval(request)

// 异步
AlgorithmServiceAsync = ..
ListenableFuture<ResultValue<R>> future = service.evalAsync(request)
```

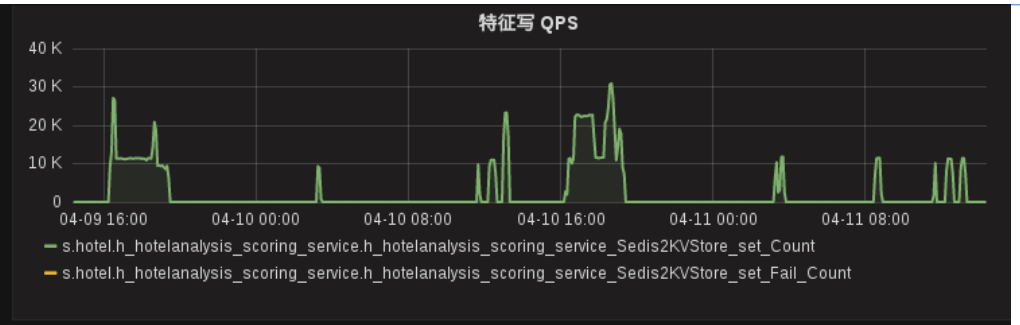
性能与调试

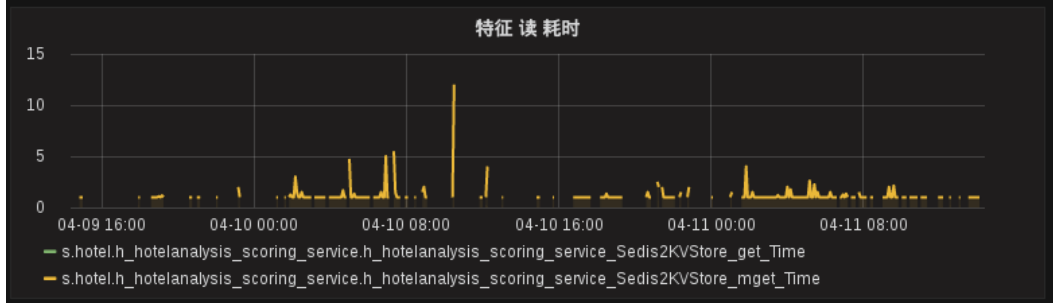
- 异步并发（guava）
- 基础数据类型（koloboke, trove4j）
- Vowpal Wabbit模型 <0.2ms
- 复合模型 <30ms
- IO占90%以上
- RequestContext上下文

```
Request (
    app,
    version,
    filter,
    featureVersion,   // 用于ab测试，默认用version
    debug,            // DebugContext开关
    context,          // 请求附加参数，如id
    T example         // 可自定义，通常为Map，List<Map>
)
```


特征 读 QPS
6 K
4 K
2 K
0
03-31
04-03
04-06
04-09
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_mget_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_get_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_get_Fail_Count
特征写 QPS
60 K
50 K
40 K
30 K
20 K
10 K
0
03-31
04-03
04-06
04-09
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_set_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_set_Fail_Count
特征 读 耗时
30
25
20
15
10
5
0
03-31
04-03
04-06
04-09
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_get_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_mget_Time
特征 写 耗时
1.010
1.005
1.000
0.995
0.990
03-31
04-03
04-06
04-09
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_set_Time
预测耗时
2.0 K
1.5 K
1.0 K
500
0
03-31
04-03
04-06
04-09
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_predict_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_fail_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_timeout_Time
预测 QPS
30
20
10
0
03-31
04-03
04-06
04-09
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_fail_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_timeout_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_predict_Count
JVM GC 次数
1.5
1.0
0.5
0
GC 耗时
10 K
8 K
5 K
3 K
0
JVM线程数
350
300
250
200
150
100

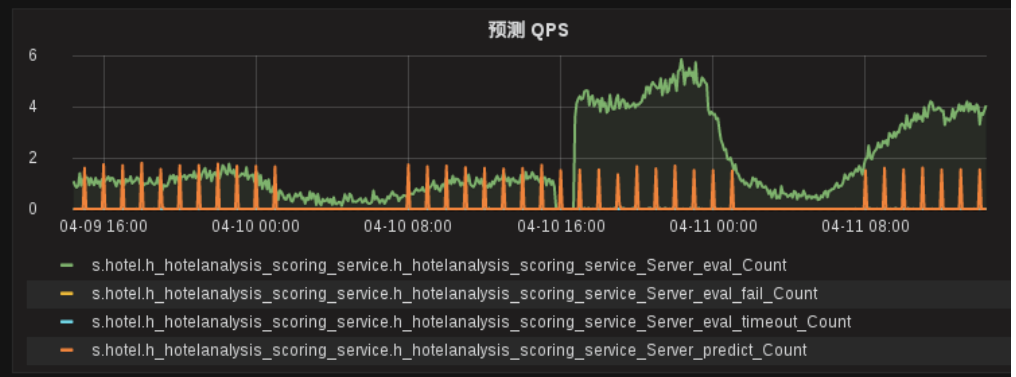

特征 读 QPS
60
50
40
30
20
10
0
04-09 16:00
04-10 00:00
04-10 08:00
04-10 16:00
04-11 00:00
04-11 08:00
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_mget_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_get_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_get_Fail_Count

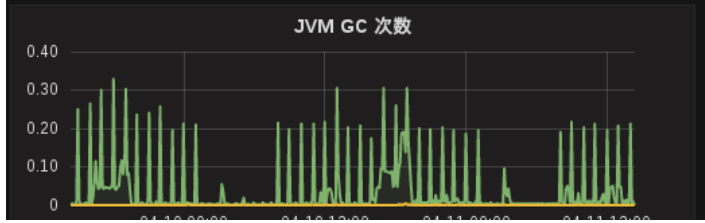

特征写 QPS
40 K
30 K
20 K
10 K
0
04-09 16:00
04-10 00:00
04-10 08:00
04-10 16:00
04-11 00:00
04-11 08:00
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_set_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_set_Fail_Count


特征 读 耗时
15
10
5
0
04-09 16:00
04-10 00:00
04-10 08:00
04-10 16:00
04-11 00:00
04-11 08:00
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_get_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_mget_Time

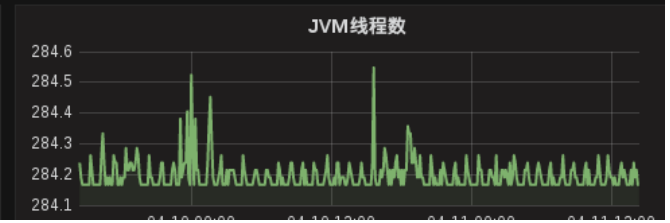

特征 写 耗时
1.010
1.005
1.000
0.995
0.990
04-09 16:00
04-10 00:00
04-10 08:00
04-10 16:00
04-11 00:00
04-11 08:00
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Sedis2KVStore_set_Time


预测耗时
6.5
6.0
5.5
5.0
4.5
4.0
3.5
04-09 16:00
04-10 00:00
04-10 08:00
04-10 16:00
04-11 00:00
04-11 08:00
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_predict_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_fail_Time
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_timeout_Time


预测 QPS
6
4
2
0
04-09 16:00
04-10 00:00
04-10 08:00
04-10 16:00
04-11 00:00
04-11 08:00
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_fail_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_eval_timeout_Count
s.hotel.h_hotelanalysis_scoring_service.h_hotelanalysis_scoring_service_Server_predict_Count


JVM GC 次数
0.40
0.30
0.20
0.10
0


GC 耗时
1.5 K
1.0 K
500
0


JVM线程数
284.6
284.5
284.4
284.3
284.2
284.1

系统展示

查询算法

查询 | +创建算法

查询结果

名称	版本	模型类型	测试	状态	创建时间
inst-conf-rf	v1.0	PMML		在线	2016-08-23 18:35:07
price	v1.0	DataProc		在线	2016-10-11 11:51:54
hotel_price_change	v1.0	DataProc		在线	2016-10-11 11:52:07
vw_detail_price	v1.0	VW		在线	2016-11-17 20:58:26
price-huazhu	v1.0	DataProc		在线	2017-02-20 14:03:18
vw-roomtype-choice	v1.0	VW		在线	2017-02-21 14:04:50
vw_detail_price	v2.0	VW		在线	2017-03-02 17:14:33
coupon-label	v1.0	PMML		在线	2017-03-20 11:57:50
room-detail-price	1.0	自定义类		在线	2017-03-22 16:18:26

测试算法 vw_detail_price / v2.0

luoyang | 开启调试 | 测试

配置详情

应用	vw_detail_price	版本	v1.0
类别	vw	创建时间	2016-11-17 20:58:26
附加属性			

更新 下线 返回

模型文件 参数列表 特征文件

beijing_city 查询 + 新建模型文件

每页10条, 共28919条 1/2,892 最首页 下一页

模型地址	过滤	状态	日志	创建时间
/user/qhstats/dm/dm_product_recommend_trained_mode	yantai	加载完成		2017-04-10 01:49:31
/user/qhstats/dm/dm_product_recommend_trained_mode	nanping	加载完成		2017-04-10 01:49:31
/user/qhstats/dm/dm_product_recommend_trained_mode	shanghai_city	加载完成		2017-04-10 01:49:31
/user/qhstats/dm/dm_product_recommend_trained_mode	maanshan	加载完成		2017-04-10 01:49:31
/user/qhstats/dm/dm_product_recommend_trained_mode	dandong	加载完成		2017-04-10 01:49:31
/user/qhstats/dm/dm_product_recommend_trained_mode	yangjiang	加载完成		2017-04-10 01:49:31
/user/qhstats/dm/dm_product_recommend_trained_mode	bayinguoleng	加载完成		2017-04-10 01:49:31

特征转换

编辑

F11 - 全屏

```
# 客户端类别转换
platform_category : {
    transform: tag_category
    keys: [platform]
    categories : {
        "" : "$tag"              # 默认使用原值
        "1": platform_adr_phone
        "2" : platform_ios_phone
    }
}

# 价格转换，结果输出到 特征空间 R
oprice_category : {
    transform: float_category
    keys: [oprice]
    output: R
    key: "$category"
    value: 1
    categories : {
        "<5.0"  : oprice_code_0
        "<10.0" : oprice_code_1
        "<15.0" : oprice_code_2
        "<20.0" : oprice_code_3
        "<30.0" : oprice_code_4
        "<40.0" : oprice_code_5
        "<50.0" : oprice_code_6
        "<75.0" : oprice_code_7
        "<100.0": oprice_code_8
        "<150.0": oprice_code_9
        "<200.0": oprice_code_10
        ">200.0": oprice_code_11
    }
}
```

模型详情

URL地址: /user/qhstats/dm/dm_product_recommend_trained_model/city=shanghai_city/dt=20170402/sgd.rmodel

过滤器: shanghai_city

状态: 加载完成

模型内容 (只读)

```
Version 8.2.0
Id
Min label:-50
Max label:50
bits:24
lda:0
0 ngram:
0 skip:
options: --quadratic yY --quadratic pP --quadratic sS --quadratic bB --quadratic dD --ignore G --ignore H --ignore I --ignore J --ignore K --ignore L
Checksum: 1962715590
:1
initial_t 0
norm normalizer 1.72991e+10
t 7.63483e+08
sum_loss 8.61849e+06
sum_loss_since_last_dump 3.36396e+06
dump_interval 1.07374e+09
min_label -50
max_label 50
weighted_examples 7.63483e+08
weighted_labels -7.60999e+08
weighted_unlabeled_examples 0
example_number 763482540
total_features 114417671636
gd::total_weight 4.62724e+08
sd::oec.weighted_examples 5.36871e+08
current pass 201
```

更新 取消

模型详情

URL地址 /user/qhstats/sparkModel/instantConfirm/v1.0/modelDate=20161114/inst-conf.pmml

过滤器

状态 加载完成

模型内容 (只读)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PMML xmlns="http://www.dmg.org/PMML-4_2" version="4.2">
    <Header>
        <Application name="JPMML-SparkML" version="1.0-SNAPSHOT"/>
        <Timestamp>2016-12-19T10:34:11Z</Timestamp>
    </Header>
    <DataDictionary>
        <DataField name="final_price_change" optype="continuous" dataType="double"/>
        <DataField name="over_room_type_lowest" optype="continuous" dataType="double"/>
        <DataField name="pay_type" optype="categorical" dataType="integer">
            <Value value="0.0"/>
            <Value value="1.0"/>
            <Value value="2.0"/>
        </DataField>
        <DataField name="hotel_rfm_success_order_total" optype="continuous" dataType="integer"/>
        <DataField name="consumption_ability" optype="continuous" dataType="double"/>
        <DataField name="hotel_rfm_overall_avg_payment_per_room_night" optype="continuous" dataType="double"/>
        <DataField name="sell_price" optype="continuous" dataType="double"/>
        <DataField name="user_hotel_distance" optype="continuous" dataType="double"/>
        <DataField name="ever_buy" optype="categorical" dataType="double">
```

更新 取消

ctrip tech 携程技术中心

ctrip tech 携程技术中心 | IT大咖说 知识分享平台

THANK YOU!

Q&A