派生集团
Paisheng Technology Group, Inc.

在生产环境中使用R语言构建机器学习项目

派生集团　黄庆昌

2018 中国R语言会议（北京）

分享内容

以R语言为主要开发语言，实现一个完整的机器学习项目：

- 数据抽取和处理
- 模型训练
- 模型部署
  - 数据接入
  - 业务流程（决策引擎）
  - A/B测试
  - 提供在线服务
  - 模型/服务器的预警和监控
- 统计分析报表

以具体项目为例...

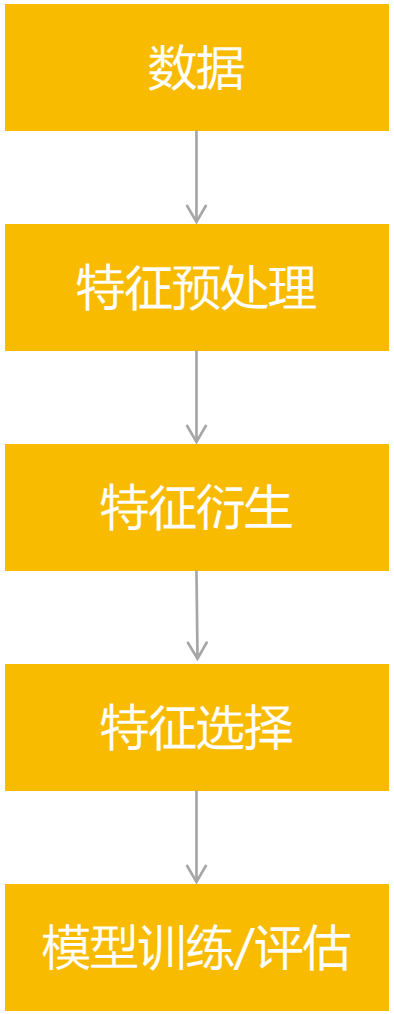
项目背景：信贷产品的贷前风控

功能要求：实现反欺诈模型、审批模型和授信模型

模型框架


决策引擎（机器学习模型 + 专家规则）
数据
反欺诈模型
审批模型
授信模型
结果
• 个人信息
• 行为数据
• 设备信息
• …
二分类模型，
识别欺诈风险
• 团伙欺诈
• 黑中介
• 虚假资料
…
二分类模型，
识别信用风险
最优化模型，
风险定价

可解释的模型 - 以审批模型为例



缺失值处理、异常值处理、标准化、有监督/无监督离散化等

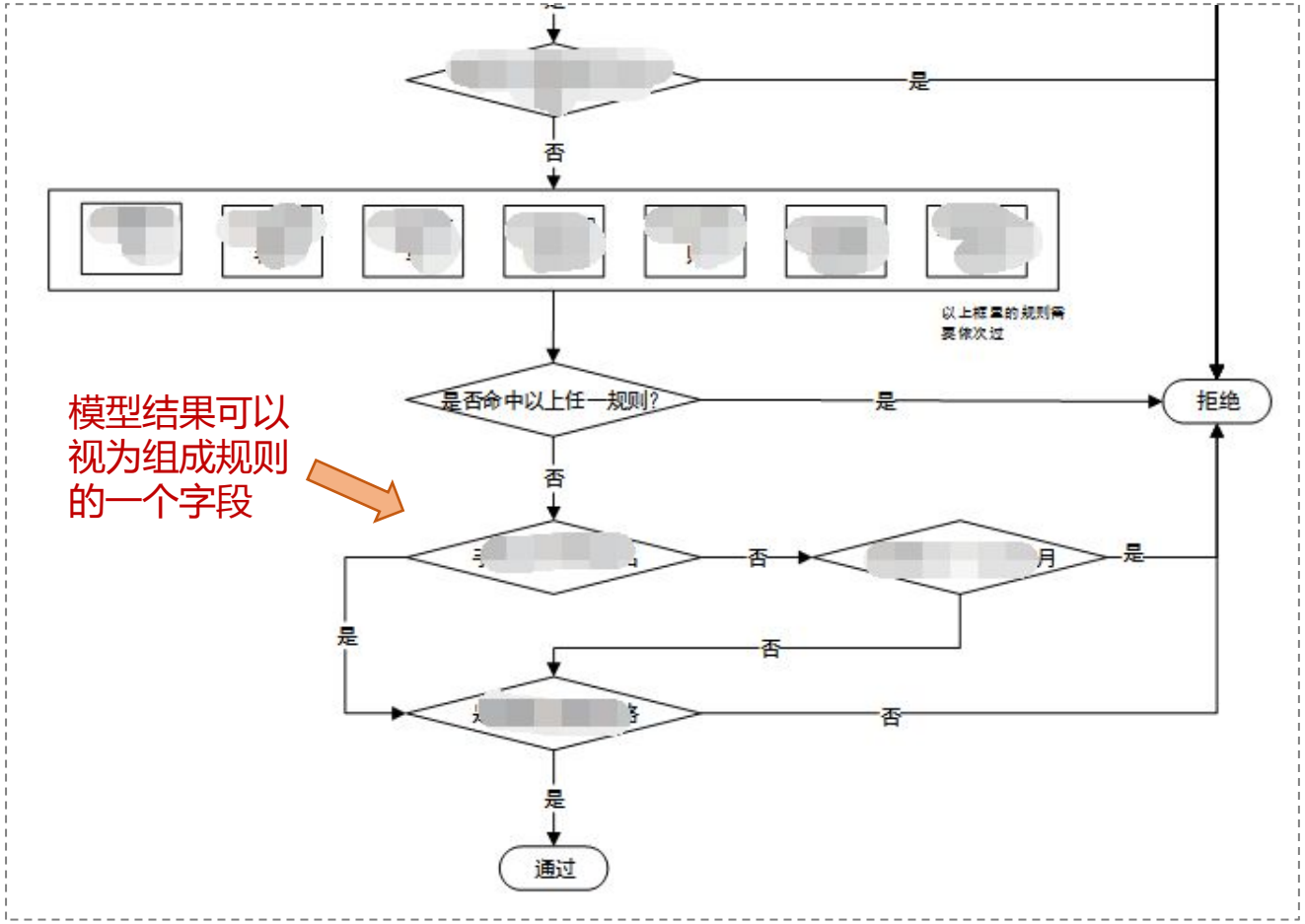
特征交叉，WOE转换，利用xgboost每棵树的路径作为特征等

IV值、卡方、相关系数等；基于boost的模型选择

代价敏感的LR模型（可解释性强，简单，响应速度快）

业务流程（决策引擎）

实际应用场景中，模型需要和业务规则（专家规则）结合使用：



通过算法得到规则

除了先验经验，还可以通过基于规则的分类器得到规则，例如：

- 关联规则

```
library(arules)

rules = apriori(dat,  parameter = list(minlen = 2, supp = 90, conf = 0.7, maxlen = 4, maxtime = 0), appearance = list(rhs = c("object=1"), default = "lhs"))
```

- 规则学习器

```
library(OneR) # one rule

OneR(object ~., data = dat)

library(Rweka) # RIPPER

JRip(object ~., data = dat)
```

- 决策树

```
library(C50)

C5.0(object ~., data = dat, rules = TRUE)
```

业务流程（决策引擎）

关键代码：

```
# 规则集（自上而下执行）
ruleList_main = list()
ruleList_main$rejectRules = list(
    list(result = '拒绝', rule = "`age` < 18 | `age` > 55", ruleName = '年龄<18岁或年龄>55岁'),
    l....（略）
   )
ruleList_main$expertRules = list(
    ....（略）
   )

# 定义执行规则的函数
runRules = function(dat, ruleList) ...（略）
    # 通过循环或lapply遍历ruleList
    ....（略）
    # 通过eval(parse(text = 规则的表达式), 数据集)获取单个规则的命中情况
    idx = which(eval(parse(text = ruleList[[i]][[j]]), dat))
    ....（略）
}

# 执行规则
runRules(dat, ruleList_main)
```

项目流程


离线训练
数据源
训练数据
特征提取
特征
训练
模型
评估
上线
在线服务
API接口
传入数据
实际数据
特征提取
特征
代入
模型
预测结果
返回模型结果
数据储存
运维监控
模型监控

数据源与数据储存

JDBC, ODBC, DBI
RMySQL, RPostgreSQL, mongolite, rredis, RNeo4j, rkafka, Sparklyr, ……

特征工程与模型训练

- 数据处理

  data.table, tidyverse(dplyr + tidyr + lubridate + stringr + jsonlite + xml2 + ...) , plyr + reshape2, mice, DMwR, ...

- 可视化

  ggplot2, recharts, rCharts, plotly, DT, ...

- 建模

  caret, mlr, h2o, RWeka, ...
  e1071, randomForest, tree, rpart, C50, arules, OneR, nnet, glmnet, survfit, ...
  xgboost, lightGBM, gbm, mboost, ...
  Boruta, ROCR, ...

A/B测试

为了衡量不同模型、不同规则组合的效果，A/B测试必不可少：

```
# 配置
flowList = list(
  list(prob = 0.8, flowName = 'model_a',  ruleList = ruleList_main),
  list(prob = 0.1, flowName = 'model_b',  ruleList = ruleList_t1),
  list(prob = 0.1, flowName = 'model_b',  ruleList = ruleList_t2)
)
# 流量分配
set.seed(1218)
prob = sapply(flowList, function(x) x$prob)
testPlan = sample(seq_along(p), nrow(dat), prob = prob, replace = TRUE)

# 执行
 ....（略）
for(i in seq_along(testPlan)){
    k = which(testPlan == i)
    if(length(k) > 0) r = getFlowResult(dat[k, ], flowList[[i]])
    ....（略）
 }
```

模型部署上线

- **提供模型服务接口（web API）**

- 用其他语言直接调用R

```
import rpy2.robjects as robjects
robjects.r('predirctFunction =  function(x){ ... }')
robjects.r['predirctFunction'](predictData)
```

- 在R中把模型生成PMML文件供其他语言使用

```
library(xgboost)
library(pmml)
data(agaricus.train, package = 'xgboost')
model_xg = xgboost(data = agaricus.train$data, label = agaricus.train$label, max_depth = 2, eta = 1, nrounds = 2,
                   nthread = 2, objective = 'binary:logistic')
xgb.dump(model_xg, 'model_xg.dumped.trees')
model_xg.pmml = pmml(model_xg,
                     inputFeatureNames = colnames(agaricus.train$data),
                     outputLabelName = 'prediction_xg',
                     outputCategories = c('0', '1'),
                     xgbDumpFile = 'model_xg.dumped.trees')
saveXML(model_xg.pmml, file = 'model_xg.pmml')
```

提供模型服务接口（web API）

可实现的包：

- httpuv
- jug
- opencpu
- plumber
- Rserve
- fiery
- RestRserve

```
RCurl::postForm('127.0.0.1:1124',
          style = 'post',
          .params = list(jsonDat = '{"v1":1,"v2":2,"v3":3}')
          )
```

```
[1] "{\"message\":\"suceess\",\"result\":0.3}"
attr(,"Content-Type")

"json"
```

```
library(httpuv)
app = list(call = function(req){
 # 获取POST的参数
 postdata = req$rook.input$read_lines()
 qs = httr:::parse_query(gsub("^\\?", "", postdata))
 dat = jsonlite::fromJSON(qs$jsonDat)
 print(dat)
 # 计算返回结果
 r = 0.3 + 0.1 * dat$v1 - 0.2 * dat$v2 + 0.1 * dat$v3
 output = jsonlite::toJSON(list(message = 'suceess', result = r), auto_unbox = T)
 res = list(status = 200L, headers = list('Content-Type' = ''application/json'), body = output)
 return(res)
})
# 启动服务
server = startServer("0.0.0.0", 1124L, app = app)
while(TRUE) {
 service()
 Sys.sleep(0.001)
}
# stopServer(server)
```

提供模型服务接口（web API）

关于API接口的安全性：

- 使用https
- 使用http + 签名验证 + 内容加密

```
# 生成签名函数
# 基于apikey, token, time拼接字符串，先进行sha1加密再md5加密
# library(digest)
formRequest = function(..., apikey, token, time = as.integer(Sys.time())){
  x = digest(paste0(requestid, apikey, token, time), algo = "sha1", serialize = F)
  output = list(apikey = apikey, time = time,
    sign = digest(y, algo = "md5", serialize = F), ...
  )
  output
}
# 验证签名函数
# 根据传输过来的appName+token+time加密结果与sign对比是否一致
tokenList = list(app1 = 'token1', app2 = 'token2')
verifySign = function(appName, token, time, sign){
  tryCatch({
    token = tokenList[[appName]]
    s = digest(paste0(appName, token, time), algo = "sha1", serialize = F)
    sign == digest(s, algo = "md5", serialize = F)
  }, error = function() F)
}
```

```
library(PKI)
library(base64enc)
# 使用公钥加密
pub_txt = "-----BEGIN PUBLIC KEY-----
MIGeMA0GCSqGSIb3DQEBAQUAA4GMADCBiAKBgGzkpZeOZcFZ02Nj80GJ3s4CSaTs
crmgXMYRr3/jkE4n9dNcCGW+ht2ssXySbRkqONtEXkwJ0EYIZvPFQvAbioXqfzJo
Aw+UfnYmrijiil1f3eXvglSNnnQuau1Bh8ib110YkhgHAr3ER7sMCz8yeFIoOYsT
5JOyDO/ADeWBCN5fAgMBAAE=
-----END PUBLIC KEY-----"
pub.pem = PKI.load.key(strsplit(pub_txt, '\n')[[1]])
str = base64encode(PKI.encrypt(charToRaw("hello word"), pub.pem))
str
# 使用私钥解密
priv_txt = "-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgGzkpZeOZcFZ02Nj80GJ3s4CSaTscrmgXMYRr3/jkE4n9dNcCGW+
ht2ssXySbRkqONtEXkwJ0EYIZvPFQvAbioXqfzJoAw+UfnYmrijiil1f3eXvglSN
nnQuau1Bh8ib110YkhgHAr3ER7sMCz8yeFIoOYsT5JOyDO/ADeWBCN5fAgMBAAEC
gYBPjNXgGxoBH2MLV+hMt7C6JPex8T56l0lPNXpVXmR8hDrU9oX39jFGXjOnQr1K
xgqVPKP6vqyVjOvRKAmXqTcPVwOAscKBXYyITgMRxyA9Ei/TzL9k83+TpgAqkY78
5C/xGokxEtd6TV8neScZ4E4DC6HqYnfse+Tfk1ryZOBucQJBAMvkn/QrX8HLbEtL
T0jEhYsCAB+k17lXKygUp+nwezt7kBGItBf1GSlEwjCuwzT+k1VvSQ2unfCBF+P3
R3+vn+cCQQCIuM5Hce09IfVLcoHa0yPekHuybcCSRZbmtzLpkqMhlzRz376Q3MEY
ZwmPM0vEtkhcmtGkwQq+fx1gE73pdt7JAkAJsbkZNua6pB1mBxKh9xYSYen3lzLa
kRZwNWs2aESzs1BKRSGq8fBcUfSZs/V8E46VxVDH4cGIqdqk8CDqJUOJAkEAgmpW
SgxAUln8E9XMTCGvS3PiqlbKpDxBLx59MBQyC66h2A4LRz9r6Y0Pr0ss8R03dS4w
38lMKF40dRsfwn0jEQJAb+Fz13/JEJ0Fgw5+vYmf3VeC7GEmxFzG/wDVrWNHeJy3
rHRwBs8t3xAM1HHRC/vYQbjrhaI2fnt+geDCTSzXcQ==
-----END RSA PRIVATE KEY-----"
priv.pem = PKI.load.key(strsplit(priv_txt, '\n')[[1]])
rawToChar(PKI.decrypt(base64decode(str), priv.pem))
```

API数据接入

- 很多情况下，模型所需的全部或部分数据需要通过API接口从业务系统获取
- 可用httr、RCurl等包实现；

接口地址：192.168.1.1/getUserInfo
请求参数示例：

```
{
    "apikey":"app1",
    "sign":"cae1cac1a2021d573e0a52adc80bcb8d",
    "time":"1510911597",
    "txtId":"TXT2017101712100"
}
```

返回参数示例：

```
{
    "code":200,
    "message":"请求成功",
    "txtId":"TXT2017101712100",
    // data为业务数据
    "data":{
                "deviceData":"{
                "ModelNumber":"iPhone6S",
                "DeviceId":"28426796-8E27-xxxxxxxx",
                "Ip":"114.114.114",
                "DeviceBrand":"Apple",
                "Os":"iOS",
                "OsVer":"10.2.1"}"
              }
}
```

```r
# 请求数据接口示例
api_host = '192.168.1.1'
api_getData = function(api_path, txtId, requestid , ...){
 # 构造请求参数（自定义函数）
 jsondat = formRequest(txtId = txtId, requestid = requestid, apikey = 'app1', token = 'token1', ...)
 reqBody = jsonlite::toJSON(jsondat, auto_unbox = T)
 # POST请求数据接口
 res = httr::POST(sprintf('%s/%s', api_host, api_path),  httr::content_type('application/json'), body
= reqBody)
 # 记录请求日志
 apilog_apiGet = data.frame(recordId = recordId, queryString = as.character(reqBody))
 if(res$status_code == 200L){
  output = httr::content(res)
  apilog_apiGet$resultCode = output$code
  apilog_apiGet$result = jsonlite::toJSON(output, auto_unbox = T)
 } else output = NULL
 # 保存日志（自定义函数）
 db_writeFun('apilog_apiGet', apilog_apiGet)
 return(output)
}

api_getData(api_path = 'getUserInfo', txtId = 'user001')
```

日志

模型服务在运行时应记录相关信息，以便于分析运行情况和定位问题。可记录的信息例如：

- 模型接口的外部请求情况
- 业务数据API接口请求情况
- 各个环节的耗时、错误信息
- 服务器资源消耗情况
- ……

日志示例：

log_apiQuery (11×12)

id	apikey	method	requestid	queryString	result	resultCode	ip	userAgent	insertTime
14	wec	/	99090e31-50cd-4bc7-a14b-8fdb...	{"apikey":","sign":"0e1ead2dbd3f...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 22:23:18
15	wec	/v	99090e31-50cd-4bc7-a14b-8fdb...	{"apikey":","sign":"0e1ead2dbd3f...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 22:23:19
16	wec	/w	99090e31-50cd-4bc7-a14b-8fdb...	{"apikey":","sign":"0e1ead2dbd3f...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 22:24:57
17	we	/w	99090e31-50cd-4bc7-a14b-8fdb...	{"apikey":","sign":"0e1ead2dbd3f...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 22:34:08
18	we	/v	d5a49326-d83c-11e7-a325-e149...	{"apikey":","sign":"8fcc2a7decb5...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 23:15:55
19	w	/	df4acf12-d83c-11e7-a325-e149...	{"apikey":","sign":"173604c2be2c...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 23:16:11
20	w	/	f5089e2e-d83c-11e7-a325-e149...	{"apikey":","sign":"29fb46d477d3...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 23:16:47
21	w	/	0e1a38aa-d83d-11e7-a325-e14...	{"apikey":","sign":"1ccce016bc13...	{"code":200,"message":"提交成...	200	119.145.111.230		2017-12-03 23:17:29
25	w	/	e057125c-d88d-11e7-9b2f-000c...	{"apikey":","sign":"248c0ae8a95e...	{"code":200,"message":"提交成...	200	121.13.249.210		2017-12-04 08:55:52
44	hu	/	4274a128-4799-11e8-84e6-d09...	{"apikey":","sign":"a826c...	{"code":-1,"message":"发送失败"}	-1	172.18.4.6		2018-04-24 16:27:00
45	hu	/	5b02cdd2-4799-11e8-84e6-d094...	{"apikey":","sign":"3753...	{"code":200,"message":"发送成...	200	172.18.4.6	libcurl/7.29.0 r-curl/3.1...	2018-04-24 16:27:49
46	hu	/	b8bb541e-479c-11e8-8ee8-8018...	{"apikey":","sign":"54b1...	{"code":200,"message":"发送成...	200	172.18.4.0	libcurl/7.29.0 r-curl/2.8...	2018-04-24 16:51:49

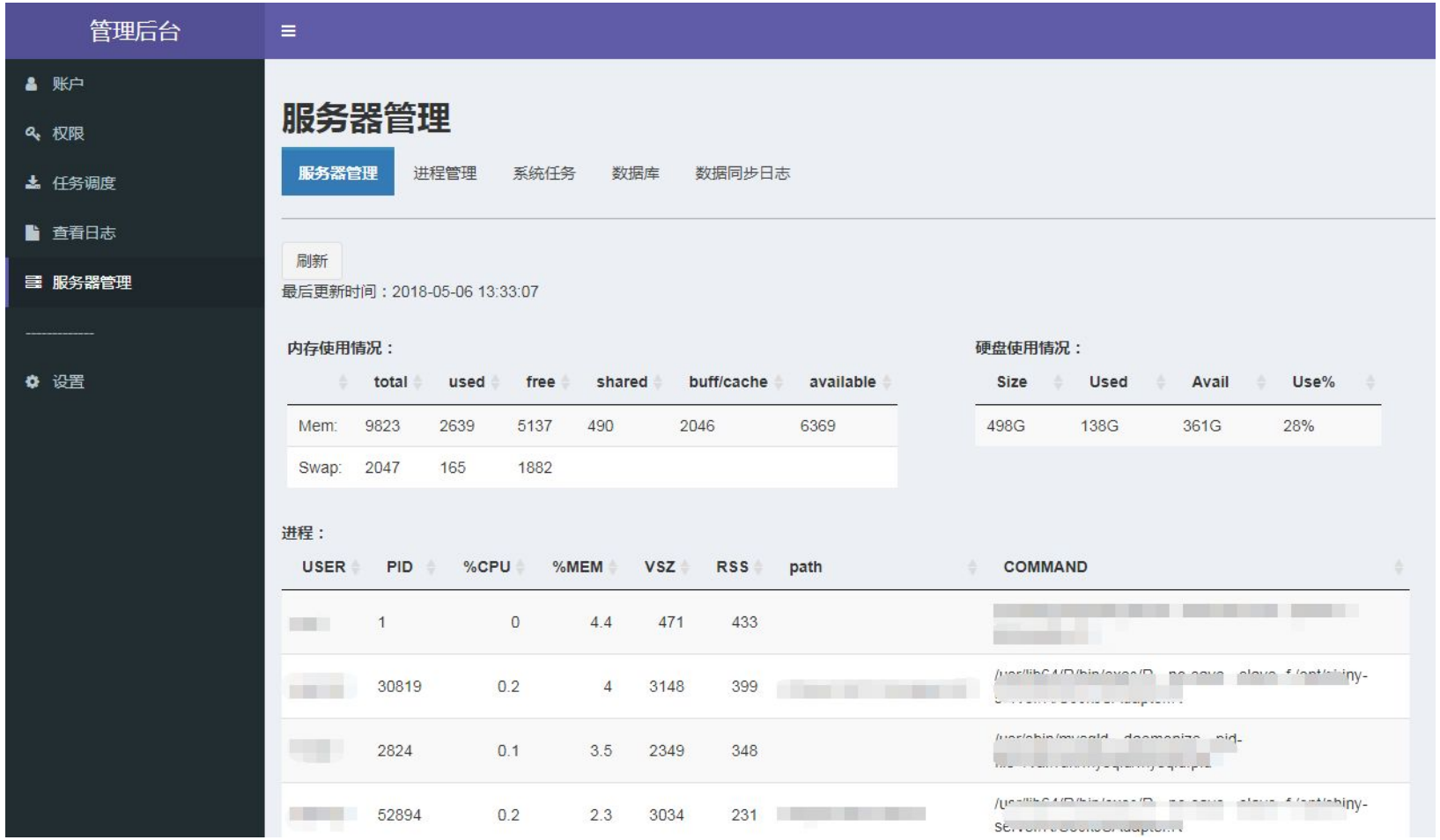
运维监控

模型服务在运行时需要监控服务器运行情况

- 可以开源软件如zabix等；
- 如果需要在R中实现，可以通过system命令获取服务器信息，再用shiny做可视化。

以centos7为例：

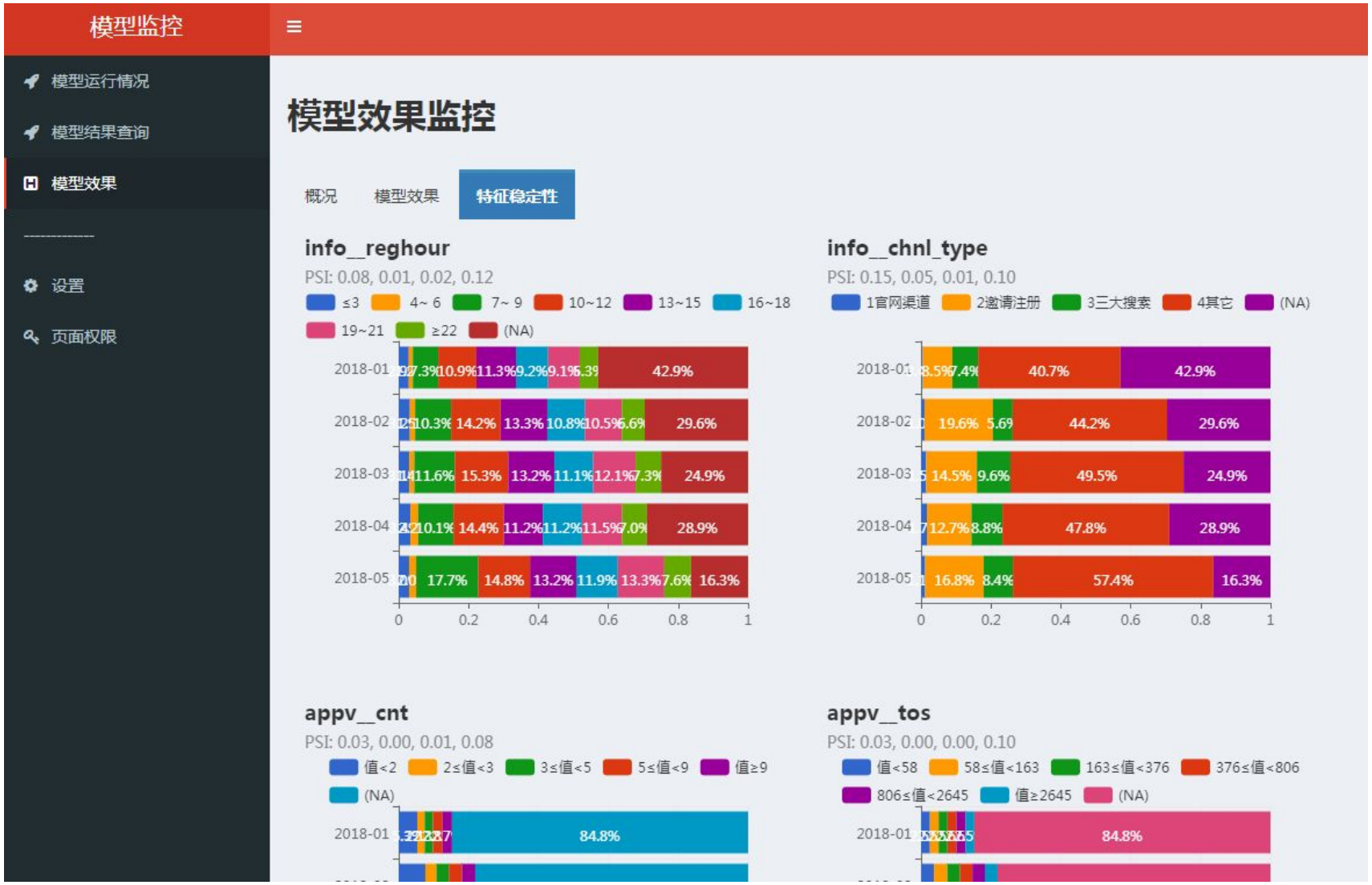
```
# 获取内存
sytem('free -m')
# 获取硬盘
sytem('df -h')
# 获取前30个进程（按内存倒序）
sytem('ps aux | sort -k4nr | head -30')
```



模型监控

除了确保模型服务运行良好，还需要监控模型的稳定性和实际效果。

例如重要特征/模型结果在不同时间窗的分布是否有明显差异，特征与目标变量的相关系数、IV值等是否有明显变化、模型是否达到预期目标等等。

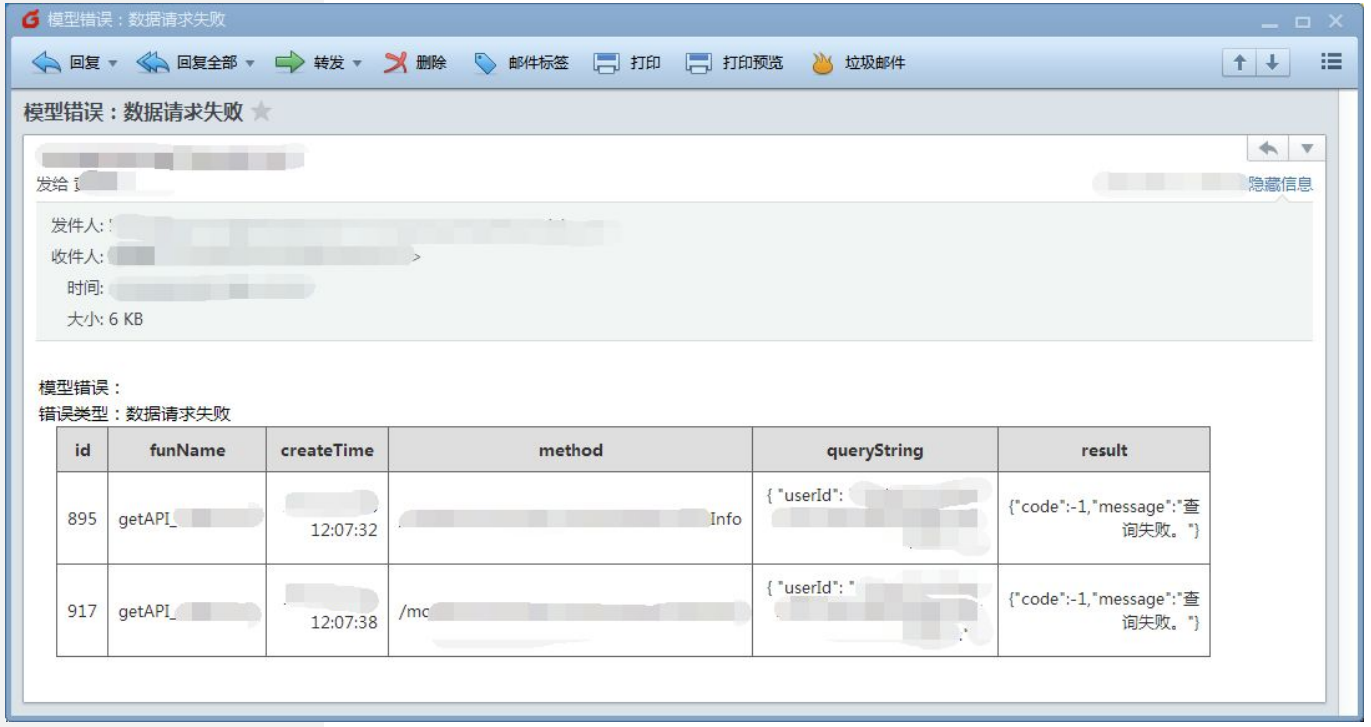


预警

对服务器运行情况和模型的关键评价指标作监控

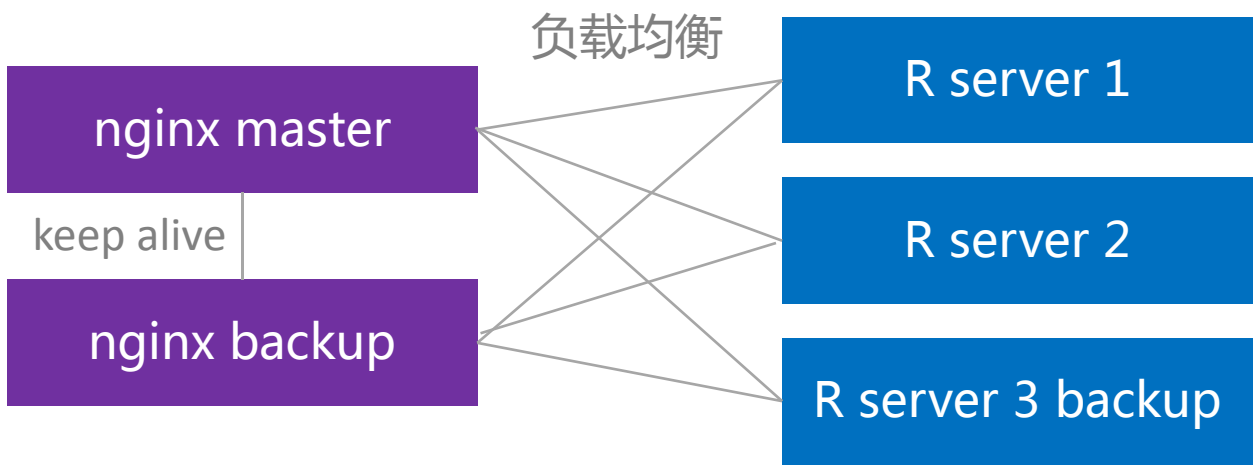
除了报表，出现异常时可通过mailR包自动发送邮件：

```
mailR::send.mail(
  from = 'sender@tuandai.com', # 发送人
  to = 'sendee@tuandai.com', # 接收人
  cc = 'carboncopy@tuandai.com', # 抄送人
  subject = '邮件标题',
  body = as.character(
'<div style = "color:red">邮件正文，可以为HTML格式</div>'
),
  attach.files = NULL, # 附件的路径
  encoding = "utf-8",
  smtp = list(
          host.name = 'smtp.exmail.qq.com', # 邮件服务器IP地址
          port = 465, # 邮件服务器端口
          user.name = 'senderName', # 发送人名称
          passwd = 'yourpassword', # 密码
          ssl = T),
  html = T, inline = T, authenticate = T, send = T, debug = F
)
```



高可用的R服务（HA）

如果需要增强模型服务的稳定性，可以利用nginx：



* 可以使用docker替换上述的server

配置/etc/nginx/nginx.conf

```
# 设定负载均衡的服务器列表
upstream RModelName {
        server 192.168.1.1:9001;
        server 192.168.1.2:9001;
        server 192.168.1.3:9001 backup;
  }
...
server {
      ...
      location /
       {
         ...
         proxy_pass http://RModelName;
        }
  }
```

（实际部署可咨询负责运维的同学）

派生集团
Paisheng Technology Group, Inc.

Q & A