

ATEC实训-互联网交易欺诈识别-任务指南

快速开始

[整体流程](#)

[数据集所在位置](#)

[操作步骤](#)

- 1、登录阿里云ECS
- 2、下载解压demo文件到自己的目录
- 3、本地运行
- 4、制作镜像及push
- 5、触发运行
- 6、查看运行情况及日志
- 7、模型与预测结果下载

[其他需要关注的内容](#)

[激活虚拟环境并定制自己的配置](#)

[训练预测示例代码编写](#)

[任务数据](#)

[数据介绍](#)

[字段介绍](#)

[交易主键](#)

[交易主体](#)

[交易属性](#)

[数值型变量](#)

[行为序列编码](#)

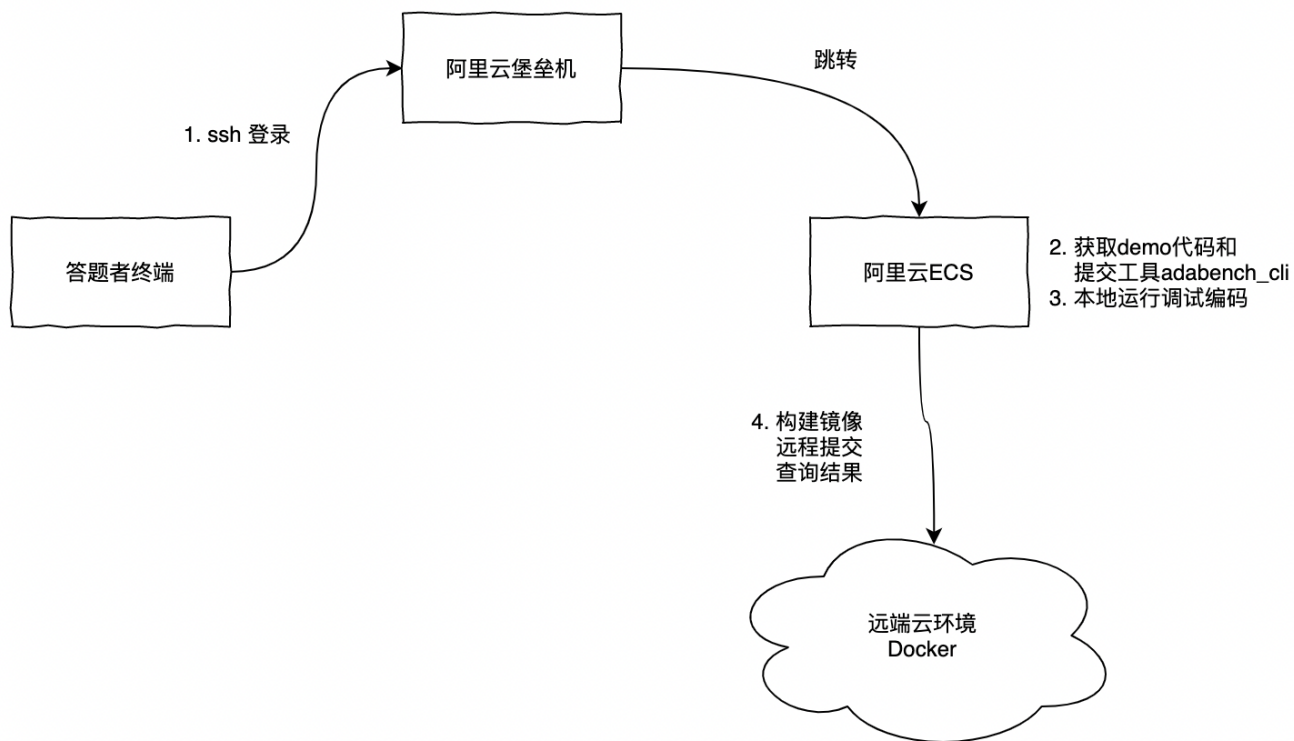
[是否欺诈的标签](#)

[评估标准](#)

快速开始

本文介绍如何通过一个demo方案进行训练与预测的结果提交，答题者熟悉整个流程后，可修改自己的方案进行提交。

整体流程



数据集所在位置

训练/验证/测试集（不带标签），均放置于：`/home/admin/workspace/job/input`

操作步骤

1、登录阿里云ECS

- 1) 答题者有可通过ssh命令登录的终端
- 2) 请答题者使用ssh命令，通过官网提供的堡垒机/ECS口令，登录到自己的ECS下

2、下载解压demo文件到自己的目录

1) 下载demo

```
Bash | 复制代码
1  wget http://atec-code.oss-cn-beijing-internal.aliyuncs.com/demo/atec_fraud_
    trade_dataset/demo.tgz
2  mkdir demo
3  tar xzf demo.tgz -C ~/demo/
```

2) 下载提交工具 (adabench_cli)

```
Bash | 复制代码
1  wget http://atec-zark.oss-cn-beijing-internal.aliyuncs.com/adabench/adabenc
    h_cli
2
3  chmod +x adabench_cli
```

3、本地运行

```
Bash | 复制代码
1  cd ~/demo/atec_project
2  LOCAL=1 sh run.sh
```

训练模型和预测结果会放置于 run.sh同级的demo_output下。

注意：此处使用LOCAL=1标示是本地训练，云端的模型与预测结果存放路径略有不同。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
import time
import json
import lightgbm as lgb
import sys
import os

from sklearn.metrics import roc_auc_score
from datetime import datetime

if os.environ.get('LOCAL'):
    output_model_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'demo_output/your_model') # 本地产出模型路径
    output_predictions_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'demo_output/predictions.jsonl') # 本地产出预测结果
else:
    output_model_path = '/home/admin/workspace/job/output/data/your_model' # 云端产出模型路径
    output_predictions_path = '/home/admin/workspace/job/output/predictions/predictions.jsonl' # 云端产出预测结果
```

```

[101] valid_0's auc: 0.847528
[102] valid_0's auc: 0.847512
[103] valid_0's auc: 0.847209
[104] valid_0's auc: 0.847431
[105] valid_0's auc: 0.847417
[106] valid_0's auc: 0.847431
[107] valid_0's auc: 0.847615
[108] valid_0's auc: 0.847561
[109] valid_0's auc: 0.847823
Early stopping, best iteration is:
[59] valid_0's auc: 0.849683
auc: 0.8496829364628478
start read data
start produce var
start predict
start write data
(your_name_env) [iml56etld4@match-machine-rvd7p89dt atec_project]$ ll
总用量 28
drwxrwxr-x  2 iml56etld4 iml56etld4  4096 11月 11 16:12 demo_output ←
-rw-rw-r--  1 iml56etld4 iml56etld4 13646 11月 11 15:48 run.py
-rw-rw-r--  1 iml56etld4 iml56etld4   196 11月  7 18:13 run.sh
drwxrwxr-x 12 iml56etld4 iml56etld4  4096 11月  7 15:04 your_name_env

```

4、制作镜像及push

制作镜像：

Bash | 复制代码

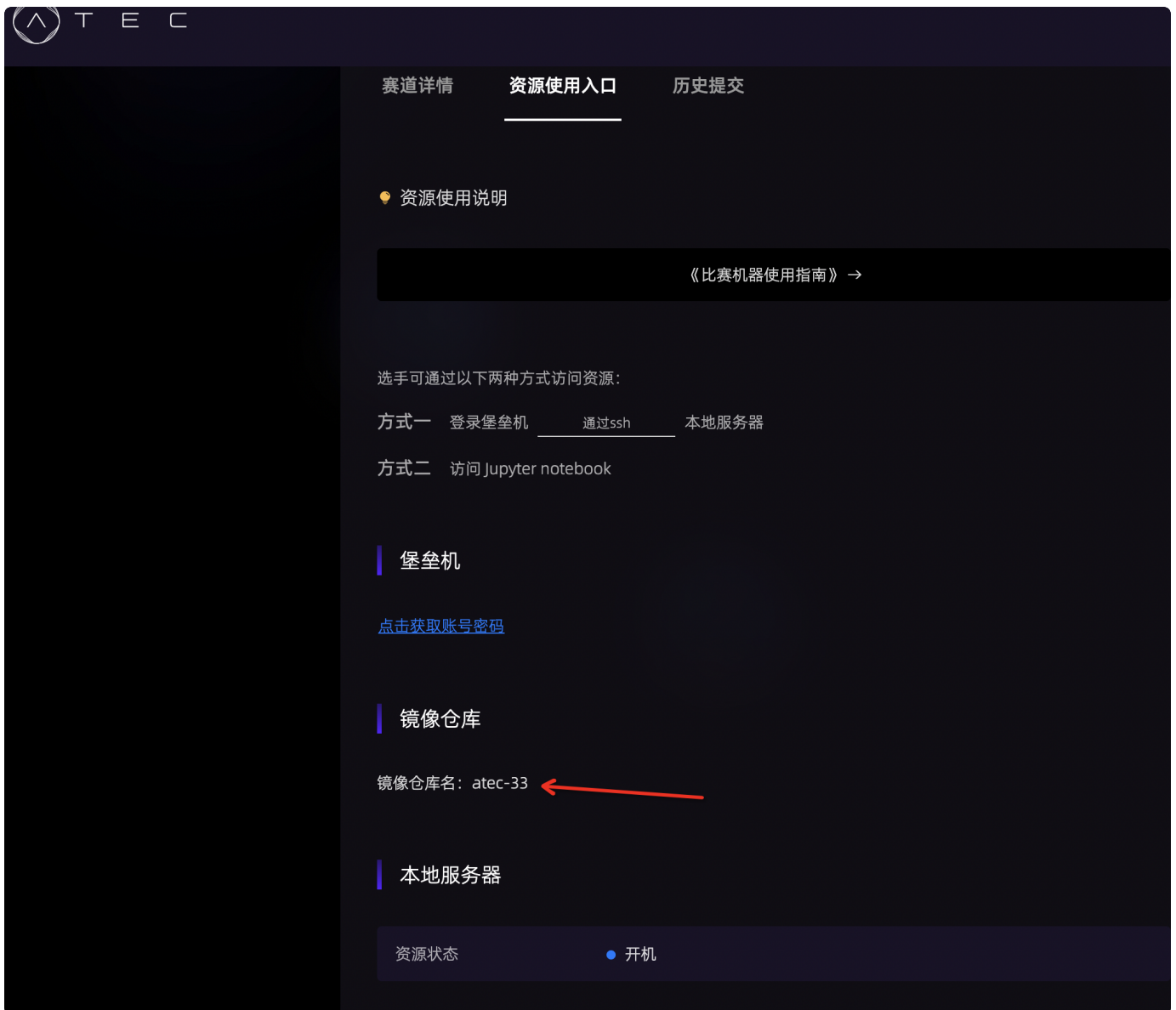
```

1 cd ~/demo
2 docker build -f Dockerfile --network=host -t image_id:image_tag .

```

image_tag: 镜像标签，每次push不能重复

image_id: zark-image-repo-registry-vpc.cn-beijing.cr.aliyuncs.com/zark/仓库名，仓库名在“资源使用入口”页查看。




镜像提交

1) docker登录账号获取

登录前，需要通过cli获得登录账号和密码：

▼

Plain Text |  复制代码

```
1 cd ~
2 ./adabench_cli get-docker-token
```

```
(my_env) [9eys3qws8@match-machine-8td9sda0 atec_project]$ cd
(my_env) [9eys3qws8@match-machine-8td9sda0 atec_project]$ adabench_cli auth get-docker-token
[*] {
  "username": "cr_temp_user",
  "password": "eyJhdXRoVG9rZW5JZGVudGlmYWVYIjoIYWZYTzZkYmYyYzU2NGExN2I4MWU2NmJhMTY4MjMxM2IiLCJpbmN0YW5jZUlkIjoIY3JpLWx6bWY5ZWZgejdyZWlscXIiLCJ0aW50IjoIiMTYzNDZANzNTY1OTAwMCIiInR5cGUiOiJzdHMiLCJ1c2VySWQiOiIiXmJlNTI4NTJhcnxMzExMzc4In0iOiI19965679f082f9aa73228663d06032fb55b7aa9",
  "expiretime": "2021-10-12 18:47:39"
}
```

每个答题者的账号都固定为：cr_temp_user

password每次申请都不同，且只有1小时有效期。建议答题者在每次push镜像前，获取这个账号用于提交镜像。

如果超过1小时，docker push提交镜像会报错！

2) Docker 登录

▼ Plain Text 复制代码

```
1 docker login --username=cr_temp_user zark-image-repo-registry-vpc.cn-beijing.cr.aliyuncs.com
2
```

输入1) 中的password

3) Docker镜像提交

▼ Plain Text 复制代码

```
1 docker push zark-image-repo-registry-vpc.cn-beijing.cr.aliyuncs.com/zark/atec-412:v1
```

```
[iml56etld4@match-machine-rvd7p89dt ~]$ docker login --username=cr_temp_user zark-image-repo-registry-vpc.cn-beijing.cr.aliyuncs.com
Password:
WARNING! Your password will be stored unencrypted in /home/iml56etld4/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[iml56etld4@match-machine-rvd7p89dt ~]$ docker push zark-image-repo-registry-vpc.cn-beijing.cr.aliyuncs.com/zark/atec-412:v1
The push refers to repository [zark-image-repo-registry-vpc.cn-beijing.cr.aliyuncs.com/zark/atec-412]
a392dc64228b: Pushed
174f56854903: Pushed
v1: digest: sha256:4f984dd095f8db2a756563ebf182eab5f3015dc144fc668a2dfe88bf14a3622 size: 742
[iml56etld4@match-machine-rvd7p89dt ~]$
```

5、触发运行

▼ Bash 复制代码

```
1 ./adabench_cli run -t image:///home/admin/atec_project --dataset atec_fraud
   _trade_dataset --image image_id:image_tag
```

例如：

```
./adabench_cli run -t image:///home/admin/atec_project --dataset atec_fraud_trade_dataset --image zark-image-repo-registry-vpc.cn-beijing.cr.aliyuncs.com/zark/atec-412:v1
```

一般答题者就修改--image后面的内容，前面保持不变。

这一步会获取到对应的run id；

6、查看运行情况及日志

通过run id可以查看任务的运行情况及日志：

▼

Bash | 复制代码

```
1  --查看触发镜像运行情况
2  ./adabench_cli run-desc --run_id run_id
3  --查看日志
4  ./adabench_cli run-log --run_id run_id
```

注：任务未running时，run-log没法看到日志

7、模型与预测结果下载

▼

Bash | 复制代码

```
1  --下载模型
2  ./adabench_cli run-download --run_id run_id --download_type model
3  --下载预测结果
4  ./adabench_cli run-download --run_id run_id --download_type predictions
```

其他需要关注的内容

激活虚拟环境并定制自己的配置

demo里面已经提供了一个python3.7的环境，可以通过激活这个环境安装配置自己需要的一些组件：

▼

Bash | 复制代码

```
1  source ~/demo/atec_project/your_name_env/bin/activate
2  pip install ...
```

注：

1. 后续制作Docker镜像时，会将这个python环境(your_name_env)打包到镜像，云端运行的刷榜逻辑将会使用这个环境里python模块。
2. 如果要取消激活，运行source ~/demo/atec_project/your_name_env/bin/deactivate

训练预测示例代码编写

- 1) demo里面提供了一个run.py的代码，其中有train()和rank()的函数，分别对应训练和预测，**所以你想训练自己的模型，修改demo里面的run.py的代码即可：**
- 2) demo里面也提供了一个run.sh的文件，用于执行run.py文件，这个文件不用修改：

```
1  #!/bin/bash
2  DIR1=""`dirname $BASH_SOURCE`"
3  MYDIR=`readlink -f "$DIR1"`
4  cd ${MYDIR}
5  source your_name_env/bin/activate
6  export PYTHONHOME=${MYDIR}/your_name_env
7  echo "start python code"
8  python run.py
```

Python | 复制代码

任务数据

本次任务数据来源于用户转账到卡的交易数据，每行数据代表用户发起的一笔支付。此次建模任务需要将那些因用户被电信诈骗而发起的转账交易识别出来，从而减少资金损失，保障用户资金安全。

数据介绍

本次每个数据集的数据源有3份：

- driver_data，建模样本表，每行数据是一笔交易，数据可以分为几部分：交易主键，交易主体，交易属性，数值型变量，行为序列编码，是否欺诈的标签（只有训练数据集才有）；
- event_data_card，driver数据中收益卡近15天(从driver表中的交易时间算起)的历史收款数据，包含

交易主键，交易主体，交易属性；

- event_data_user, driver数据中用户近15天的历史付款数据，包含交易主键，交易主体，交易属性（注意这里的付款场景不止转账到卡的交易数据）；

本次提供的测试集("test_"开头)没有提供标签，需要你提供预测代码对其进行打分；

字段介绍

交易主键

字段	字段含义	字段类型
e_id	交易主键	STRING

交易主体

字段	字段含义	字段类型
a_k_id	用户主键	STRING
c_k_id	收款卡主键；收款方为卡时才可能有该属性	STRING

交易属性

字段	字段含义	字段类型
time	发生时间	STRING
amt	交易金额	DOUBLE
a_id_1	用户相关id_1	STRING
b_id_1	收款方相关id_1；当收款方为体	STRING

	系内账号时才可能有值，卡为体系外	
c_id_1	收款卡相关id_1；当收款方为卡时才可能有该属性	STRING
a_info_1	用户信息_1；与用户基本属性有关	STRING
b_info_1	收款方信息_1；当收款方为体系内账号时才可能有值，与收款方基本属性有关	STRING
c_info_1	收款卡信息_1；当收款方为卡时才可能有该属性，与收款卡的基本属性有关	STRING
a_info_2	用户信息_2；与用户基本属性有关	STRING
b_info_2	收款方信息_2；当收款方为体系内账号时才可能有值，与收款方基本属性有关	STRING
c_info_2	收款卡信息_2；当收款方为卡时才可能有该属性，与收款卡的基本属性有关	STRING
e_info_2	交易信息_2；与设备位置信息相关	STRING
e_info_3	交易信息_3；与设备位置信息相关	STRING
a_info_3	用户信息_3；与用户基本属性相关	STRING
e_info_4	交易信息_4；与设备位置信息相关	STRING
e_info_5	交易信息_5；与设备位置信息相关	STRING
a_info_4	用户信息_4；与用户基本属性	STRING

	相关	
a_info_5	用户信息_5；与用户基本属性相关	DOUBLE
hh	发生时间-小时	BIGINT
week_day	发生时间-周几	BIGINT
c_id_2	收款卡相关id；衍生于收款卡主键	STRING
c_info_3	收款卡信息_3；当收款方为卡时才可能有该属性，与收款卡的基本属性有关	STRING
e_info_6	交易信息_6；反映交易类型	STRING
a_info_6	用户信息_6；与用户基本属性相关	STRING
a_info_7	用户信息_7；与用户基本属性相关	STRING
c_info_4	收款卡信息_4；当收款方为卡时才可能有该属性，与收款卡的基本属性有关	STRING
c_info_5	收款卡信息_5；当收款方为卡时才可能有该属性，与收款卡的基本属性有关	STRING
if_succ	交易是否完成，即交易可能被中止；收款方为卡时才有意义	BIGINT
e_info_13	交易信息_13；与金额相关	BIGINT
dt	日期	STRING

数值型变量

其中var1–var21是可直接输入模型的数值型变量；event_data_card和event_data_user中没有该数据；

行为序列编码

event_data_card和event_data_user中没有该数据；

var_23:收款卡行为序列编码

var_24:用户行为序列编码

序列编码数据支持输入神经网络进行使用；

序列生成规则如下：

比如某条样本(a)发生时间是"2021-07-01 12:00:00"，这条样本的a_k_id在"2021-07-01 11:00:00"发生了另外一笔交易（b），我们将这两条交易串成序列，因为交易有很多属性（信息，如交易类型，付款方式等），我们其实是将这部分属性串成序列，假设我们有3个属性，如"e_info_1, e_info_2, e_info_3"

将该序列的属性取值串成序列[[e_info_1_b, e_info_2_b, e_info_3_b], [e_info_1_a, e_info_2_a, e_info_3_a]] 由于现在模型计算只能接受数值型数据，所以我们将属性取值进行id化(数值型的则分bin)，这里id化是按单个属性进行编码的，即e_info_1_a和e_info_1_b共用一套编码字典，而e_info_2_a与e_info_2_b共用一套编码字典；编码后，假设变成：[[1,3,1],[2,3,5]],最后去掉其他符号，只保留逗号分隔：1,3,1,2,3,5 此时可称这个序列的长度为2，编码属性为3；

两个序列的属性个数：

- num_attributes_var_23 = 14
- num_attributes_var_24 = 16

最大序列长度(超过长度的序列会被截断):

- max_seq_length=256

组内编码, 每个属性是1...N:

- min_ids_var_23 = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
- max_ids_var_24 = [4, 501, 432, 442, 377, 35, 108, 34, 24, 7, 32, 367, 8, 9]
- min_ids_var_23 = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
- max_ids_var_24 = [9, 18, 107, 4, 33, 331, 440, 501, 24, 7, 32, 333, 8, 9, 11, 10]

是否欺诈的标签

注意：测试集中的driver表中没有该字段，提交的预测代码需要注意该列不存在的风险

字段	字段含义	字段类型
is_fraud	是否欺诈风险，0代表正常交易，1代表有风险，-1代表未知，即一部分被策略管控失败的交易（只有训练集有）	BIGINT

评估标准

提交结果为每个测试样本是1的概率，也就是is_fraud为1的概率。评价方法为3个打扰率下欺诈金额的召回比例的加权（越大越好），打扰率为0.005, 0.01, 0.05，加权系数分别为0.4, 0.3, 0.3。

举例：

- 假设我们有10000个样本，100个欺诈样本，欺诈金额200，干扰率0.01相当于欺诈概率靠前的 $10000 \times 0.01 = 100$ 个样本入围，假设这100个样本中有50个欺诈样本，涉及欺诈金额150，则干扰率0.01的覆盖欺诈金额比例为 $150/200 = 0.75$