FATE

FederatedAI

Mar 01, 2022

DEPLOY

FATE (Federated AI Technology Enabler) is an open-source project initiated by Webank's AI Department to provide a secure computing framework to support the federated AI ecosystem. It implements secure computation protocols based on homomorphic encryption and multi-party computation (MPC). It supports federated learning architectures and secure computation of various machine learning algorithms, including logistic regression, tree-based algorithms, deep learning and transfer learning.

https://fate.fedai.org

CHAPTER

ONE

FATE STAND-ALONE DEPLOYMENT GUIDE

Server Configuration;

The stand-alone version provides 2 deployment methods, which can be selected according to your actual situation:

- Install FATE using Docker *(Recommended)*
- Install FATE in Host

You can also refer to Chinese guide

1.1 1) Install FATE using Docker*(Recommended)*

It is strongly recommended to use docker, which greatly reduces the possibility of encountering problems.

1. The host needs to be able to access the external network,pull the installation package and docker image from the public network.
2. Dependent on docker , docker recommended version is 18.09, you can use the following command to verify the docker environment: docker –version , docker start and stop and other Please refer to: docker –help.
3. Keep the 8080 port accessible before executing. If you want to execute again, please delete the previous container and image with the docker command.

   please follow the below step:

```
#Get code
wget https://webank-ai-1251170195.cos.ap-guangzhou.myqcloud.com/docker_standalone_fate_1.
↪5.1.tar.gz
tar -xzvf docker_standalone_fate_1.5.1.tar.gz

#Execute the command
cd docker_standalone_fate_1.5.1
bash install_standalone_docker.sh
```

1. Test

   - Unit Test

   ```
   CONTAINER_ID=`docker ps -aqf "name=fate"`
   docker exec -t -i ${CONTAINER_ID} bash
   bash ./python/federatedml/test/run_test.sh
   ```

   If success, the screen shows like blow:

```
there are 0 failed test
```

- Toy_example Test

```
CONTAINER_ID=`docker ps -aqf "name=fate"`
docker exec -t -i ${CONTAINER_ID} bash
python ./examples/toy_example/run_toy_example.py 10000 10000 0
```

If success, the screen shows like blow:

```
success to calculate secure_sum, it is 2000.0
```

2. Install FATE-Client and FATE-Test

   To conveniently interact with FATE, we provide tools FATE-Client and FATE-Test.

   Install FATE-Client and FATE-Test with the following commands:

```
pip install fate-client
pip install fate-test
```

There are a few algorithms under examples folder, try them out!

You can also experience the fateboard access via a browser: Http://hostip:8080.

1.2 2) Install FATE in Host

1. Check whether the local 8080,9360,9380 port is occupied.

```
netstat -apln|grep 8080
netstat -apln|grep 9360
netstat -apln|grep 9380
```

2. Download the compressed package of stand-alone version and decompress it.

```
wget https://webank-ai-1251170195.cos.ap-guangzhou.myqcloud.com/standalone_fate_
↪master_1.5.1.tar.gz
tar -xzvf  standalone_fate_master_1.5.1.tar.gz
```

3. Enter FATE directory and execute the init.sh.

```
cd standalone_fate_master_1.5.1
sh init.sh init
```

4. Test

   - Unit Test

```
cd standalone_fate_master_1.5.1
source bin/init_env.sh
bash ./python/federatedml/test/run_test.sh
```

If success, the screen shows like blow:

```
there are 0 failed test
```

- Toy_example Test

```
cd standalone_fate_master_1.5.1
source bin/init_env.sh
python ./examples/toy_example/run_toy_example.py 10000 10000 0
```

If success, the screen shows like blow:

```
success to calculate secure_sum, it is 2000.0
```

5. Install FATE-Client and FATE-Test

   To conveniently interact with FATE, we provide tools FATE-Client and FATE-Test.

   Install FATE-Client and FATE-Test with the following commands:

```
python -m pip install fate-client
python -m pip install fate-test
```

There are a few algorithms under examples folder, try them out!

You can also experience the fateboard access via a browser: Http://hostip:8080.

CHAPTER

TWO

FATE CLUSTER DEPLOYMENT GUIDE

The Cluster version provides four deployment methods, which can be selected according to your actual situation:

- Install Cluster Chinese guide
- Install AllinOne Chinese guide
- Install Exchange Step By Step Chinese guide

2.1 1. General Introduction

2.1.1 1.1 System Introduction

1FATE

FATE (Federated AI Technology Enabler) is an open source project initiated by the AI department of WeBank. It provides a secure computing framework based on data privacy protection and provides strong secure computing support for machine learning, deep learning, and migration learning algorithms. The security bottom layer supports a variety of multi-party secure computer mechanisms such as homomorphic encryption, secret sharing, and hashing. The algorithm layer supports logistic regression, boosting, and federated migration learning in the multi-party secure computing mode.

2EggRoll

Eggroll is a large-scale distributed architecture suitable for machine learning and deep learning, including computing, storage and communication modules. Provide bottom support for FATE framework.

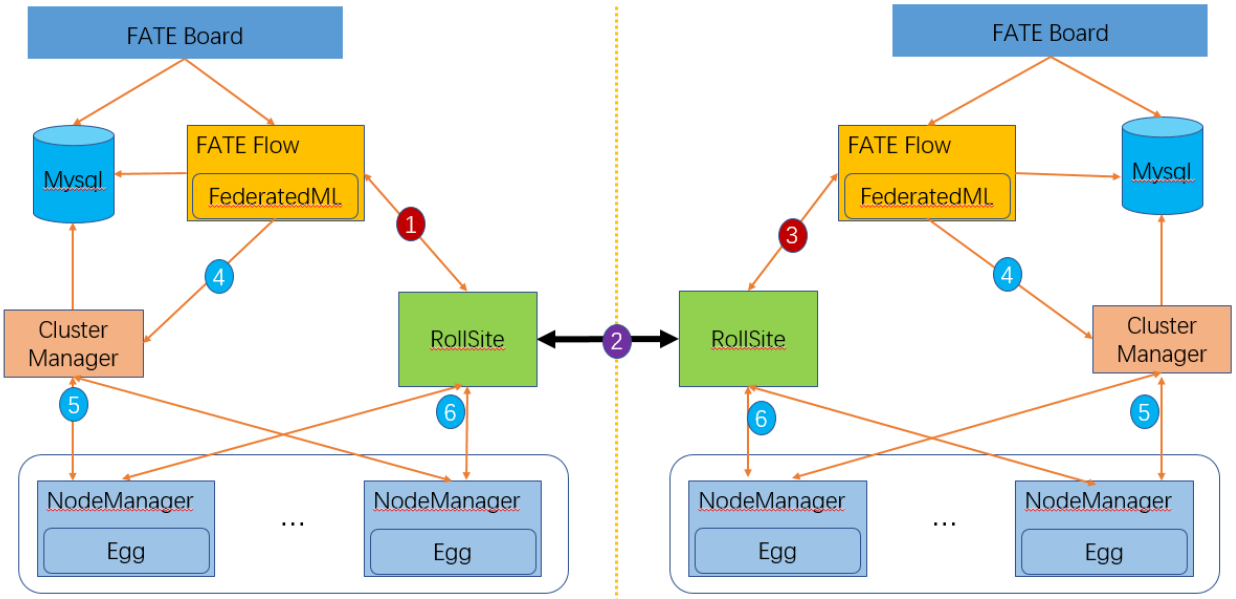
3FATE Official Website https://fate.fedai.org/

This article will introduce the deployment of FATE cluster using ansible deployment script.

2.1.2 1.2. Component Description

2.1.3 1.3. System Structure

Example deployment in two parties



1. Identify 1, 2, 3, fateflow to initiate tasks, synchronize task information to all parties through rollsite, during which fateflow will detect the running status of all parties.
2. Identify 4, 5, 6, eggroll task initiation and scheduling calculation, and communicate with the peer through rollsite.

2.2 2.Detailed Design

2.2.1 2.1.Deployment Planning

In this example, there is only one host on each end, and multiple hosts on each end. Currently, only nodemanager multi-node deployment is supported, and other components are single-node.

Remarks:Involving exchange instructions will use 192.168.0.88 to represent its IP, but this example does not involve exchange deployment.

2.2.2 2.2.Host Resources And Operating System Requirements

2.2.3 2.3.Network Requirements

2.3 3. Basic Environment Configuration

2.3.1 3.1 Hostname Configuration

---

1) Modify the host name

Execute under the 192.168.0.1 root user:

hostnamectl set-hostname VM_0_1_centos

Execute under the 192.168.0.2 root user:

hostnamectl set-hostname VM_0_2_centos

2) Add host mapping

Execute under the root user of the target server (192.168.0.1 192.168.0.2):

vim /etc/hosts

192.168.0.1 VM_0_1_centos

192.168.0.2 VM_0_2_centos

2.3.2 3.2 Close selinux

Execute under the root user of the target server (192.168.0.1 192.168.0.2):

Confirm whether selinux is installed

Centos system execution: rpm -qa | grep selinux

If selinux has been installed, execute: setenforce 0

2.3.3 3.3 Modify Linux System Parameters

Execute under the root user of the target server (192.168.0.1 192.168.0.2):

1. Clean up the 20-nproc.conf file

cd /etc/security/limits.d

ls -lrt 20-nproc.conf

Existence: mv 20-nproc.conf 20-nproc.conf_bak

1. vim /etc/security/limits.conf

* soft nofile 65535

* hard nofile 65535

* soft nproc 65535

* hard nproc 65535

Log in again, ulimit -a to check whether it takes effect

2.3.4 3.4 Turn Off The Firewall

Execute under the root user of the target server (192.168.0.1 192.168.0.2)

Centos System:

systemctl disable firewalld.service

systemctl stop firewalld.service

systemctl status firewalld.service

2.3.5 3.5 Software Environment Initialization

1) Create User

Execute under the root user of the target server (192.168.0.1 192.168.0.2)

```
groupadd -g 6000 apps
useradd -s /bin/bash -g apps -d /home/app app
passwd app
```

2) Configure sudo

Execute under the root user of the target server (192.168.0.1 192.168.0.2)

vim /etc/sudoers.d/app

app ALL=(ALL) ALL

app ALL=(ALL) NOPASSWD: ALL

Defaults !env_reset

2.3.6 3.6 Increase Virtual Memory

Target server (192.168.0.1 192.168.0.2) root user execution

When used in a production environment, 128G of virtual memory needs to be added for memory calculations. Check whether the storage space is sufficient before execution.

Note: dd takes a long time to execute, please be patient

```
cd /data
dd if=/dev/zero of=/data/swapfile128G bs=1024 count=134217728
mkswap /data/swapfile128G
swapon /data/swapfile128G
cat /proc/swaps
echo '/data/swapfile128G swap swap defaults 0 0' >> /etc/fstab
```

2.3.7 3.7 Install Dependent Packages

Target server (192.168.0.1 192.168.0.2) root user execution

```
#Install basic dependencies
#centos
yum install -y gcc gcc-c++ make openssl-devel gmp-devel mpfr-devel libmpc-devel libaio␣
↪numactl autoconf automake libtool libffi-devel
#ubuntu
apt-get install -y gcc g++ make openssl libgmp-dev libmpfr-dev libmpc-dev libaio1 libaio-
↪dev numactl autoconf automake libtool libffi-dev
#If there is an error, you need to solve the yum source problem.
```

(continues on next page)

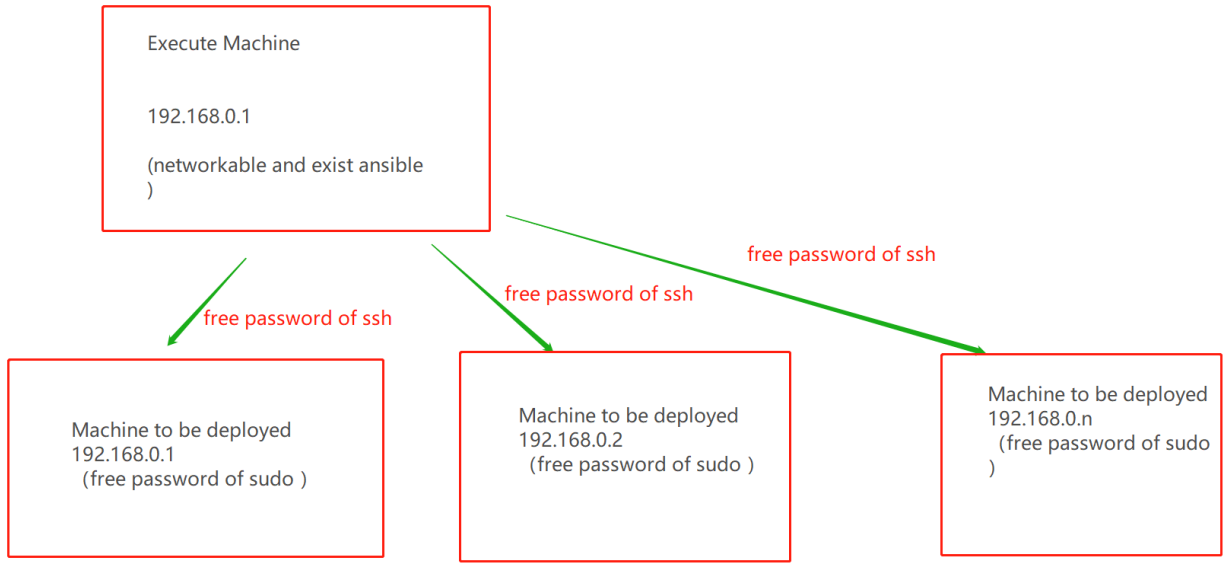
(continued from previous page)

```
#Install ansible and process management dependency packages
#centos
yum install -y ansible
#ubuntu
apt-get install -y ansible

#If there is an error and the server has an external network, there is no need to solve␣
↪the problem of incomplete yum source, execute:
#centos
yum install -y epel-release
#Add a more comprehensive third-party source, and then reinstall ansible jq supervisor
```

2.4 4.Project Deployment

2.4.1 4.1 Deployment Diagram



2.4.2 4.2 System Check

Execute under the target server (192.168.0.1 192.168.0.2) app user

```
#Virtual memory, size is not less than 128G, if not satisfied, please refer to section 3.
↪6 to reset
cat /proc/swaps
Filename                Type    Size      Used  Priority
```

(continues on next page)

(continued from previous page)

```
/data/swapfile128G file 134217724   384    -1

#File handle number, not less than 65535, if not satisfied, please refer to chapter 3.3␣
↪to reset
ulimit -n
65535

#The number of user processes, not less than 64000, if not satisfied, please refer to␣
↪chapter 3.3 to reset
ulimit -u
65535

#Check whether the process has fate process residue, if any, you need to stop the service
ps -ef| grep -i fate

netstat -tlnp | grep 4670
netstat -tlnp | grep 4671
netstat -tlnp | grep 9370
netstat -tlnp | grep 9371
netstat -tlnp | grep 9360
netstat -tlnp | grep 8080
netstat -tlnp | grep 3306

#Check the deployment directory, if necessary, mv first
ls -ld /data/projects/fate
ls -ld /data/projects/data
ls -ld /data/projects/snmp

#Check the supervisord configuration file, if any, you need to mv or delete it
ls -lrt /data/projects/common/supervisord/supervisord.d/fate-*.conf
```

2.4.3 4.3 Obtain The Project

Execute under the app user of the target server (192.168.0.1 with external network environment)

Enter the /data/projects/ directory of the execution node and execute:

```
#Note: URL links have line breaks, please make sure to arrange them in one line when␣
↪copying
cd /data/projects/
wget https://webank-ai-1251170195.cos.ap-guangzhou.myqcloud.com/ansible_nfate_1.5.2_
↪release-1.0.0.tar.gz
tar xzf ansible_nfate_1.5.2_release-1.0.0.tar.gz
```

2.4.4 4.4 Configuration File Modification And Example

4.4.1 Initial Configuration File

```
cd ansible-nfate-*
#init.sh file does not need to be modified, mainly to assist in generating some␣
↪configuration files

#Production environment plus prod parameter execution
  sh ./tools/init.sh prod

>sh ./tools/init.sh prod
clean old config
init environments/prod
init var_files/prod
init project_prod.yml
```

4.4.2 Certificate Production Configuration (optional)

1) Certificate Production

```
vi /data/projects/ansible-nfate-1.*/tools/make.sh

#1. The custom security certificate needs to be deployed at both ends at the same time,␣
↪and only one end needs to be manually processed. The manual processing part will not␣
↪be introduced temporarily.
#2. The security certificate supports the following deployment methods:
     1) Deploy host+guest, host and guest use secure certificate communication.
     2) Deploy host+exchange+guest, where host and exchange use secure certificates to␣
↪communicate, and guest and exchange communicate normally.
     3) Deploy host+exchange+guest, where guest and exchange use secure certificates to␣
↪communicate, and host and exchange communicate normally.

guest_host="192.168.0.1" ---Modify according to the actual IP
host_host="192.168.0.2" ---Modify according to the actual IP
exchange_host="192.168.0.88" --- Modify according to the actual IP, this example does␣
↪not need to be modified without deployment
```

2) Execute Script To Make Certificate

```
cd tools
sh ./make.sh

Certificate files will be generated in the keys/host and guest directories.
```

3) Copy The Certificate To The Deployment Directory

```
sh cp-keys.sh host guest

The certificate file will be copied to the roles/eggroll/files/keys directory
```

(continues on next page)

(continued from previous page)

```
Special Note:
1. Currently, script deployment only supports 2 parties to set up certificate␣
↪authentication. (host&guest, host&exchange, guest&exchange)
```

4.4.3 Modify Configuration File

1. **ModifyThe Initialization Host IP**

```
vi /data/projects/ansible-nfate-1.*/environments/prod/hosts

#ansible format configuration file
[fate] ---Fill the host IP to be deployed into the fate group
192.168.0.1
192.168.0.2

[deploy_check] --- Fill in the deploy_check group with the local IP that executes ansible
192.168.0.1

[all:vars]
ansible_connection=ssh
ansible_ssh_port=22 --- Modify according to actual situation
ansible_ssh_user=app
#ansible_ssh_pass=test ---If you have not done a password-free login, you need to␣
↪provide a password
##method: sudo or su
ansible_become_method=sudo
ansible_become_user=root
ansible_become_pass= --- If each host has not done secret-free sudo, the root password␣
↪must be filled
```

1. **Modify The Deployment Mode**

```
vi /data/projects/ansible-nfate-1.*/var_files/prod/fate_init

#Only modify the following parameters, other parameters remain unchanged by default
deploy_mode: "install" --- The default is empty, modified to install, which means a new␣
↪deployment
```

3) Modify Host Side Parameters

Note: The default is not to enable the configuration of the security certificate. If you enable the security certificate communication, you need to set server_secure, client_secure, is_secure to true, and the port corresponding to is_secure to 9371

```
#If you don't deploy host, you don't need to modify
#Except nodemanger can set multiple IPs, all others are single IP
vi /data/projects/ansible-nfate-1.*/var_files/prod/fate_host

host:
    partyid: 10000 ---host partyid, modify according to actual plan
    rollsite:
```

(continues on next page)

(continued from previous page)

```
      enable: True
      ips: ---IP list, currently rollsite only supports deployment to one server
      -192.168.0.1
      port: 9370 --- grpc port
      secure_port: 9371 ---grpcs port
      pool_size: 600 ---thread pool size,Recommended as: min(1000 + len(party_ids) * 200,
↪ 5000)
      max_memory: ---rollsite process JVM memory parameter, the default is 1/4 of the␣
↪physical memory, which can be set according to the actual situation, such as 12G, if␣
↪it is a machine dedicated to rollsite, configure it to 75% of the physical memory.
      server_secure: False ---As a server, turn on the security certificate verification,
↪ do not use the security certificate by default
      client_secure: False ---As a client, use a certificate to initiate a security␣
↪request, not using a security certificate by default
      default_rules: ---This party points to the IP and port routing configuration of␣
↪exchange or other parties
      -name: default
        ip: 192.168.0.2 ---exchange or peer party rollsite IP
        port: 9370 ---exchange or opposite party rollsite port, generally default 9370,␣
↪which means no security certificate deployment; if you need to enable security␣
↪certificate communication, it should be set to 9371;
        is_secure: False ---Whether to use secure authentication communication; it needs␣
↪to be used in combination with server_secure or client_secure. When all three are true,
↪ it means to use secure authentication communication with the next hop rollsite. At␣
↪the same time, the previous parameter port needs to be set to 9371; not used The␣
↪default security certificate is sufficient.
      rules: ---The party's own routing configuration
      -name: default
        ip: 192.168.0.1
        port: 9370
      -name: fateflow
        ip: 192.168.0.1
        port: 9360
    clustermanager:
      enable: True
      ips:
      -192.168.0.1 --- Only support the deployment of one host
      port: 4670
      cores_per_node: 16 ---Nodemanager node CPU core number, multiple nodemanager nodes␣
↪are set according to the minimum number of CPU cores
    nodemanager:
      enable: True
      ips: ---Support multiple deployment
      -192.168.0.1
      -192.168.0.x
      port: 4671
    eggroll:
      dbname: "eggroll_meta"
      egg: 2
    fate_flow:
      enable: True
      ips:
```

(continues on next page)

(continued from previous page)

```
      -192.168.0.1 ---Only support the deployment of one host
      grpcPort: 9360
      httpPort: 9380
      dbname: "fate_flow"
    fateboard:
      enable: True
      ips:
      -192.168.0.1 ---Only support the deployment of one host
      port: 8080
      dbname: "fate_flow"
    mysql:
      enable: True
      ips:
      -192.168.0.1 ---Only support the deployment of one host
      port: 3306
      dbuser: "fate"
      dbpasswd: "fate_deV2999"
    zk:
      enable: False
      lists:
      -ip: 192.168.0.1
        port: 2181
      use_acl: false
      user: "fate"
      passwd: "fate"
    servings:
     ip: 192.168.0.2
     port: 8000
```

4) Modify The Guest Parameters

Note: The default is not to enable the configuration of the security certificate. If you enable the security certificate communication, you need to set server_secure, client_secure, is_secure to true, and the port corresponding to is_secure to 9371.

```
#If you don't deploy the guest party, you don't need to modify it
#Except nodemanger can set multiple IPs, all others are single IP
vi /data/projects/ansible-nfate-1.*/var_files/prod/fate_guest

guest:
    partyid: 9999 ---Modify according to actual plan
    rollsite:
      enable: True
      ips: ---IP list, currently rollsite only supports deployment to one server
      -192.168.0.2
      port: 9370 --- grpc port
      secure_port: 9371 ---grpcs port
      pool_size: 600 ---thread pool size,Recommended as: min(1000 + len(party_ids) * 200,
↪ 5000)
      max_memory: ---rollsite process JVM memory parameter, the default is 1/4 of the␣
↪physical memory, which can be set according to the actual situation, such as 12G, if␣
↪it is a machine dedicated to rollsite, configure it to 75% of the physical memory.
      server_secure: False ---As a server, turn on the security certificate verification,
↪ do not use the security certificate by default
```

(continues on next page)

(continued from previous page)

```
      client_secure: False ---As a client, use a certificate to initiate a security␣
↪request, not using a security certificate by default
      default_rules: ---This party points to the IP and port routing configuration of␣
↪exchange or other parties
      -name: default
        ip: 192.168.0.1 ---exchange or peer party rollsite IP
        port: 9370 ---exchange or opposite party rollsite port, generally default 9370,␣
↪which means no security certificate deployment; if you need to enable security␣
↪certificate communication, it should be set to 9371;
        is_secure: False ---server_secure or client_secure is true, the next hop␣
↪rollsite pointed to is also turned on security authentication, this parameter needs to␣
↪be set to true, the previous parameter port needs to be set to 9371, and the default␣
↪is not to use a security certificate.
      rules: ---The party's own routing configuration
      -name: default
        ip: 192.168.0.2
        port: 9370
      -name: fateflow
        ip: 192.168.0.2
        port: 9360
    clustermanager:
      enable: True
      ips: ---Only support the deployment of one host
      -192.168.0.2
      port: 4670
      cores_per_node: 16 ---Nodemanager node CPU core number, multiple nodemanager nodes␣
↪are set according to the minimum number of CPU cores
    nodemanager:
      enable: True
      ips: ---Support the deployment of multiple hosts
      -192.168.0.2
      -192.168.0.x
      port: 4671
    eggroll:
      dbname: "eggroll_meta"
      egg: 2
    fate_flow:
      enable: True
      ips: ---Only support the deployment of one host
      -192.168.0.2
      grpcPort: 9360
      httpPort: 9380
      dbname: "fate_flow"
    fateboard:
      enable: True
      ips: ---Only support the deployment of one host
      -192.168.0.2
      port: 8080
      dbname: "fate_flow"
    mysql:
      enable: True
      ips: ---Only support the deployment of one host
```

(continues on next page)

(continued from previous page)

```
      -192.168.0.2
      port: 3306
      dbuser: "fate"
      dbpasswd: "fate_deV2999"
    zk:
      enable: False
      lists:
      -ip: 192.168.0.2
        port: 2181
      use_acl: false
      user: "fate"
      passwd: "fate"
    servings:
     ip: 192.168.0.2
     port: 8000
```

5) Modify Exchange Parameters

Note: The default is not to enable the configuration of the security certificate. If you enable the security certificate communication, you need to set server_secure, client_secure, is_secure to true, and the port corresponding to is_secure to 9371.

```
#Do not deploy exchange without modification
vi /data/projects/ansible-nfate-1.*/var_files/prod/fate_exchange

exchange:
  enable: False - The deployment of exchange needs to be modified to True
  rollsite:
    ips:
    -192.168.0.88
    port: 9370
    secure_port: 9371 ---grpcs port
    pool_size: 600 ---thread pool size,Recommended as: min(1000 + len(party_ids) * 200,␣
↪5000)
    max_memory: ---rollsite process JVM memory parameter, the default is 1/4 of the␣
↪physical memory, which can be set according to the actual situation, such as 12G, if␣
↪it is a machine dedicated to rollsite, configure it to 75% of the physical memory.
    server_secure: False ---As a server, turn on the security certificate verification,␣
↪do not use the security certificate by default
    client_secure: False ---As a client, use a certificate to initiate a security␣
↪request, not using a security certificate by default
  partys: --- Routing configuration pointing to each party
  -id: 10000
    rules:
    -name: default
      ip: 192.168.0.1
      port: 9370 ---corresponding to the party rollsite port, generally the default is␣
↪9370, which means that there is no security certificate communication; if you need to␣
↪open the security certificate communication, it should be set to 9371;
      is_secure: False ---server_secure or client_secure is true, the next hop rollsite␣
↪pointed to is also turned on security authentication, this parameter needs to be set␣
↪to true, the previous parameter port needs to be set to 9371, and the default is not␣
↪to use a security certificate.
```

(continues on next page)

(continued from previous page)

```
  -id: 9999
    rules:
    -name: default
      ip: 192.168.0.2
      port: 9370 ---corresponding to the party rollsite port, generally the default is␣
↪9370, that is, communication without a security certificate; if you need to enable␣
↪communication with a security certificate, set it to 9371;
      is_secure: False ---server_secure or client_secure is true, the next hop rollsite␣
↪pointed to is also turned on security authentication, this parameter needs to be set␣
↪to true, the previous parameter port needs to be set to 9371, and the default is not␣
↪to use a security certificate.
```

2.4.5 4.5 Deployment

After modifying the corresponding configuration items according to the above configuration meaning, then execute the deployment script:

```
#Relative ansible-nfate-* directory
cd /data/projects/ansible-nfate-1.*

#Production environment plus prod parameter execution
nohup sh ./boot.sh prod -D> logs/boot.log 2>&1 &
```

The deployment log is output in the logs directory, and check whether there is an error in real time:

```
#Relative ansible-nfate-* directory
cd logs
tail -f ansible.log (check the deployment in real time, if there is no such log file,␣
↪you need to check whether ansible is installed)

List of check items fail prompt:
1. "Warning: now swap is 0, need to turn up"
    ---The virtual memory is not set, please refer to the previous chapter to set it,␣
↪not less than 128G.
2. "Warning: key fate process exists, please has a check and clean"
    ---The environment has not been cleaned up, and the previously deployed fate process␣
↪needs to be stopped.
3. "Warning: these ports: 4670 4671 9360 9370 9380 have been used"
    ---The environment has not been cleaned up, and the previously deployed fate process␣
↪needs to be stopped.
4. "Warning: if reinstall mysql, please stop mysql, and rename /etc/my.cnf"
    ---mysql did not stop, it needs to be stopped. If there is a /etc/my.cnf file, mv␣
↪needs to be renamed.
5. "Waring: please rename /data/projects/fate"
    ---The fate directory exists, you need to mv first.
6. "Warning: please rename /data/projects/data/fate/mysql"
    ---/data/projects/data exists, mv is required.
7. "Warning: supervisor_fate_conf exists, please remove ls /data/projects/common/
↪supervisord/supervisord.d/fate-*.conf"
  --- The /data/projects/common directory exists, and mv is required.
```

Restart after fateflow deployment:

```
#Because fate_flow depends on more components, there may be exceptions in startup. The␣
↪processing is as follows:
netstat -tlnp | grep 9360
If there is no port, restart fateflow:
sh service.sh stop fate-fateflow
sh service.sh start fate-fateflow
```

2.4.6 4.6 Problem Location

1. Eggroll log

/data/logs/fate/eggroll/bootstrap.clustermanager.err

/data/logs/fate/eggroll/logs/eggroll/clustermanager.jvm.err.log

/data/logs/fate/eggroll/logs/eggroll/nodemanager.jvm.err.log

/data/logs/fate/eggroll/logs/eggroll/bootstrap.nodemanager.err

/data/logs/fate/eggroll/logs/eggroll/bootstrap.rollsite.err

/data/logs/fate/eggroll/logs/eggroll/rollsite.jvm.err.log

1. fateflow log

/data/logs/fate/python/logs/fate_flow/

1. fateboard log

/data/logs/fate/fate/fateboard/logs

2.5 5.Test

2.6 5.1 Toy_example Deployment Verification

For this test you need to set 3 parameters: guest_partyid, host_partyid, work_mode.

2.6.1 5.1.1 Unilateral Test

1) Execute on 192.168.0.1, guest_partyid and host_partyid are both set to 10000:

```
source /data/projects/fate/bin/init_env.sh
cd /data/projects/fate/examples/toy_example/
python run_toy_example.py 10000 10000 1
```

Note: If there is no output for more than 1 minute, it indicates that there is a problem with the deployment. You need to look at the log to locate the problem.

A result similar to the following indicates success:

"2020-04-28 18:26:20,789-secure_add_guest.py[line:126]-INFO: success to calculate secure_sum, it is 1999.9999999999998"

2) Execute on 192.168.0.2, guest_partyid and host_partyid are both set to 9999:

```
source /data/projects/fate/bin/init_env.sh
cd /data/projects/fate/examples/toy_example/
python run_toy_example.py 9999 9999 1
```

Note: If there is no output for more than 1 minute, it indicates that there is a problem with the deployment. You need to look at the log to locate the problem.

A result similar to the following indicates success:

"2020-04-28 18:26:20,789-secure_add_guest.py[line:126]-INFO: success to calculate secure_sum, it is 1999.9999999999998"

2.6.2 5.1.2 Bilateral Testing

Select 9999 as the guest and execute on 192.168.0.2:

```
source /data/projects/fate/bin/init_env.sh
cd /data/projects/fate/examples/toy_example/
python run_toy_example.py 9999 10000 1
```

A result similar to the following indicates success:

"2020-04-28 18:26:20,789-secure_add_guest.py[line:126]-INFO: success to calculate secure_sum, it is 1999.9999999999998"

2.7 5.2 Minimize Test

2.7.1 5.2.1 Upload Preset Data:

Execute on 192.168.0.1 and 192.168.0.2 respectively:

```
source /data/projects/fate/bin/init_env.sh
cd /data/projects/fate/examples/scripts/
python upload_default_data.py -m 1
```

For more details, please refer to Script README

2.7.2 5.2.2 Fast Mode:

Please make sure that both the guest and host have uploaded the preset data through the given script.

In the fast mode, the minimization test script will use a relatively small data set, that is, the breast data set containing 569 data.

Select 9999 as the guest and execute on 192.168.0.2:

```
source /data/projects/fate/bin/init_env.sh
cd /data/projects/fate/examples/toy_example/
#Unilateral test
python run_task.py -m 1 -gid 9999 -hid 9999 -aid 9999 -f fast
#Bilateral test
python run_task.py -m 1 -gid 9999 -hid 10000 -aid 10000 -f fast
```

Some other parameters that may be useful include:

1. -f: File type used. “fast” represents the breast data set, and “normal” represents the default credit data set.
2. –add_sbt: If it is set to 1, the secureboost task will be started after lr is run. If it is set to 0, the secureboost task will not be started. If this parameter is not set, the system defaults to 1.

If the word “success” is displayed in the result after a few minutes, it indicates that the operation has run successfully. If “FAILED” appears or the program is stuck, it means the test has failed.

2.7.3 5.2.3 Normal Mode

Just replace “fast” with “normal” in the command, and the rest is the same as in fast mode.

2.8 5.3 Fateboard Testing

Fateboard is a web service. If the fateboard service is successfully started, you can view the task information by visiting http://192.168.0.1:8080 and http://192.168.0.2:8080. If there is a firewall between the local office computer and the server, you need to enable it.

2.9 6.System Operation And Maintenance

2.9.1 6.1 Service Management

Execute under the target server (192.168.0.1 192.168.0.2) app user

2.9.2 6.1.1 Service Management

```
cd /data/projects/common/supervisord
```

Start/Stop/Restart/View all:

```
#Note: Because mysql is a basic component, the startup is slow. It is recommended to␣
↪restart all components first, then start mysql first, and then start other components
sh service.sh start/stop/restart/status all

#Note: Because fateflow depends on many components, restarting all operations may cause␣
↪fateflow to start abnormally. The processing is as follows:
netstat -tlnp | grep 9360
If there is no port, restart fateflow:
sh service.sh stop fate-fateflow
sh service.sh start fate-fateflow
```

Start/stop/restart/view a single module (optional: clustermanager, nodemanager, rollsite, fateflow, fateboard, mysql):

```
sh service.sh start/stop/restart/status fate-clustermanager
```

2.10 6.2 View Process And Port

Execute under the target server (192.168.0.1 192.168.0.2) app user

2.10.1 6.2.1 View Process

```
#Check whether the process is started according to the deployment plan
ps -ef | grep -i clustermanager
ps -ef | grep -i nodemanager
ps -ef | grep -i rollsite
ps -ef | grep -i fate_flow_server.py
ps -ef | grep -i fateboard
```

2.10.2 6.2.2 View Process Port

```
#Check whether the process port exists according to the deployment plan
#clustermanager
netstat -tlnp | grep 4670
#nodemanager
netstat -tlnp | grep 4671
#rollsite
netstat -tlnp | grep 9370
#fate_flow_server
netstat -tlnp | grep 9360
#fateboard
netstat -tlnp | grep 8080
```

2.11 6.2.3 Service Log

2.11.1 6.2.4 File Directory Description

CHAPTER

THREE

7. UNINSTALL

3.1 7.1 Description

Support the uninstallation of all services and the uninstallation of a single service.

3.2 7.2 Perform uninstall

```
cd /data/projects/ansible-nfate-1.*
sh ./uninstall.sh prod all

#Uninstall command description
sh ./uninstall.sh $arg1 $arg2
- The $arg1 parameter is the same as the parameter executed by init in step 4.4.1, and␣
↪is test|prod.
- The $arg2 parameter is the selected service, the optional parameter is␣
↪(all|mysql|eggroll|fate_flow|fateboard), all means uninstall all services.
```

3.3 8. Appendix

3.4 8.1 Eggroll Parameter Tuning

Configuration file path: /data/projects/fate/eggroll/conf/eggroll.properties

Configuration parameter: eggroll.session.processors.per.node

Assuming that the number of CPU cores (cpu cores) is c, the number of Nodemanagers is n, and the number of tasks that need to run simultaneously is p, then:

egg_num=eggroll.session.processors.per.node = c * 0.8 / p

partitions (roll pair partition number) = egg_num * n

CHAPTER

FOUR

PIPELINE EXAMPLES

4.1 Introduction

We provide some example scripts of running FATE jobs with FATE-Pipeline.

Please refer to the document linked above for details on FATE-Pipeline and FATE-Flow CLI v2. DSL version of provided Pipeline examples can be found here.

4.2 Quick Start

Here is a general guide to quick start a FATE job.

1. (optional) create virtual env

```
python -m venv venv
source venv/bin/activate
pip install -U pip
```

2. install fate_client

```
# this step installs FATE-Pipeline, FATE-Flow CLI v2, and FATE-Flow SDK
pip install fate_client
pipeline init --help
```

3. configure server information

```
# configure by conf file
pipeline init -c pipeline/config.yaml
# alternatively, input real ip address and port info to initialize pipeline
# optionally, set log directory for Pipeline
pipeline init --ip 127.0.0.1 --port 9380 --log-directory ./logs
```

4. upload data with FATE-Pipeline

```
#  upload demo data to FATE data storage, optionally provide path to where deployed␣
↪examples/data locates

python demo/pipeline-upload.py --base /data/projects/fate
```

   If upload job is invoked correctly, job id will be printed to terminal and an upload bar is shown. If FATE-Board is available, job progress can be monitored on Board as well.

```
UPLOADING:||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
↪00%
2020-11-02 15:37:01.030 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:121 - Job id is 2020110215370091210977
Job is still waiting, time elapse: 0:00:01
Running component upload_0, time elapse: 0:00:09
2020-11-02 15:37:13.410 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:129 - Job is success!!! Job id is 2020110215370091210977
```

4. run a FATE-Pipeline fit job

```
python demo/pipeline-quick-demo.py
```

This quick demo shows how to build to a heterogeneous SecureBoost job. Progress of job execution will be printed as modules run. A message indicating final status ("success") will be printed when job finishes. The script queries final model information when model training completes.

```
2020-11-02 10:45:29.875 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:121 - Job id is 2020110210452959882932
Job is still waiting, time elapse: 0:00:01
Running component reader_0, time elapse: 0:00:07
Running component dataio_0, time elapse: 0:00:10
Running component intersection_0, time elapse: 0:00:14
Running component hetero_secureboost_0, time elapse: 0:00:46
Running component evaluation_0, time elapse: 0:00:50
2020-11-02 10:46:21.889 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:129 - Job is success!!! Job id is 2020110210452959882932
2020-11-02 10:46:21.890 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:130 - Total time: 0:00:52
```

5. (another example) run FATE-Pipeline fit and predict jobs

```
python demo/pipeline-mini-demo.py
```

This script trains a heterogeneous logistic regression model and then runs prediction with the trained model.

```
2020-11-02 15:40:43.907 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:121 - Job id is 2020110215404362914679
Job is still waiting, time elapse: 0:00:01
Running component reader_0, time elapse: 0:00:08
Running component dataio_0, time elapse: 0:00:10
Running component intersection_0, time elapse: 0:00:15
Running component hetero_lr_0, time elapse: 0:00:42
2020-11-02 15:41:27.622 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:129 - Job is success!!! Job id is 2020110215404362914679
2020-11-02 15:41:27.622 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:130 - Total time: 0:00:43
```

Once fit job completes, demo script will print coefficients and training information of model.

After having completed the fit job, script will invoke a predict job with the trained model. Note that `Evaluation` component is added to the prediction workflow. For more information on using FATE-Pipeline, please refer to this guide.

```
2020-11-02 15:41:28.255 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:121 - Job id is 202011021541276444328O
Job is still waiting, time elapse: 0:00:02
Running component reader_1, time elapse: 0:00:08
Running component dataio_0, time elapse: 0:00:11
Running component intersection_0, time elapse: 0:00:15
Running component hetero_lr_0, time elapse: 0:00:20
Running component evaluation_0, time elapse: 0:00:25
2020-11-02 15:41:54.605 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:129 - Job is success!!! Job id is 202011021541276444328O
2020-11-02 15:41:54.605 | INFO     | pipeline.utils.invoker.job_submitter:monitor_
↪job_status:130 - Total time: 0:00:26
```

CHAPTER

FIVE

UPLOAD DATA GUIDE

[]

Before start a modeling task, the data to be used should be uploaded. Typically, a party is usually a cluster which include multiple nodes. Thus, when we upload these data, the data will be allocated to those nodes.

5.1 Accepted Data Type

Data IO module accepts the following input data format and transforms them to desired output DTable.

dense input format input DTable's value item is a list of single element, e.g.

```
1.0,2.0,3.0,4.5
1.1,2.1,3.4,1.3
2.4,6.3,1.5,9.0
```

svm-light input format first item of input DTable's value is label, following by a list of complex "feature_id:value" items, e.g.

```
1 1:0.5 2:0.6
0 1:0.7 3:0.8 5:0.2
```

tag input format the input DTable's value is a list of tag, data io module first aggregates all tags occurred in input table, then changes all input line to one-hot representation in sorting the occurred tags by lexicographic order, e.g. assume values is

```
a c
a b d
```

after processing, the new values became:

```
1 0 1 0
1 1 0 1
```

tag:value input format the input DTable's value is a list of tag:value, like a mixed svm-light and tag input-format. data io module first aggregates all tags occurred in input table, then changes all input line to one-hot representation in sorting the occurred tags by lexicographic order, then fill the occur item with value. e.g. assume values is

```
a:0.2 c:1.5
a:0.3 b:0.6 d:0.7
```

after processing, the new values became:

```
0.2 0 0.5 0
0.3 0.6 0 0.7
```

5.2 Define upload data config file

Here is an example showing how to create a upload config file:

```
{
  "file": "examples/data/breast_hetero_guest.csv",
  "table_name": "hetero_breast_guest",
  "namespace": "experiment",
  "head": 1,
  "partition": 8,
  "work_mode": 0,
  "backend": 0
}
```

Field Specifications:

1. file: file path
2. table_name & namespace: Indicators for stored data table.
3. head: Specify whether your data file include a header or not
4. partition: Specify how many partitions used to store the data
5. work_mode: Specify current work mode: 0 for standalone, 1 for cluster
6. backend: Specify backend for job: 0 for EGGROLL, 1 for SPARK

5.3 Upload Command

We use fate-flow to upload data. Starting at FATE ver1.5, FATE-Flow Client Command Line is recommended for interacting with FATE-Flow.

The command is as follows:

```
flow data upload -c dsl_test/upload_data.json
```

Meanwhile, user can still upload data using python script as in the older versions:

```
python ${your_install_path}/fate_flow/fate_flow_client.py -f upload -c dsl_test/upload_
↪data.json
```

Note: This step is needed for every data-provide party(i.e. Guest and Host).

After running this command, the following information is shown if it is success.

```
{
  "data": {
      "board_url": "http://127.0.0.1:8080/index.html#/dashboard?job_
↪id=20201013110207536321 7&role=local&party_id=0",
      "job_dsl_path": "/data/projects/fate/jobs/20201013110207536321 7/job_dsl.json",
      "job_runtime_conf_path": "/data/projects/fate/jobs/20201013110207536321 7/job_
↪runtime_conf.json",
      "logs_directory": "/data/projects/fate/logs/20201013110207536321 7",
      "namespace": "experiment",
      "table_name": "breast_hetero_guest"
  },
  "jobId": "20201013110207536321 7",
  "retcode": 0,
  "retmsg": "success"
}
```


And as this output shown, table_name and namespace have been listed, which can be taken as input config in submit-runtime conf.

CHAPTER

SIX

DSL & TASK SUBMIT RUNTIME CONF SETTING V1

[]

To make the modeling task more flexible, currently, FATE use its own domain-specific language(DSL) to describe modeling task. With usage of this DSL, modeling components such as data-io, feature-engineering and classification/regression module etc. can be combined as a Directed Acyclic Graph(DAG). Therefore, user can take and combine the algorithm components flexibly according to their needs.

In addition, each component has their own parameters to be configured. Also, the configuration may differ from party to party. For convenience, FATE configure all parameters for all parties and all components in one file. This guide will show you how to create such a configure file.

6.1 DSL Configure File

We use json file which is actually a dict as a dsl config file. The first level of the dict is always "components," which indicates content in the dict are components in your modeling task.

```
{
  "components" : {
          ...
      }
  }
```

Then each component should be defined on the second level. Here is an example of setting a component:

```
"dataio_0": {
      "module": "DataIO",
      "input": {
          "data": {
              "data": [
                  "args.train_data"
              ]
          }
      },
      "output": {
          "data": ["train"],
          "model": ["dataio"]
      },
      "need_deploy": true
  }
```

As the example shows, user define the component name as key of this module. Note this module name should end up with a "_num" where the num should start with 0.

6.1.1 Field Specification

module Specify which component to use. This field should strictly match the file name in federatedml/conf/setting_conf except the .json suffix.

input There are two types of input, data and model.

1. Data: There are three possible data_input type:
    1. data: typically used in data_io, feature_engineering modules and evaluation.
    2. train_data: Used in homo_lr, hetero_lr and secure_boost. If this field is provided, the task will be parse as a **fit** task
    3. validate_data: If train_data is provided, this field is optional. In this case, this data will be used as validation set. If train_data is not provided, this task will be parsed as a **predict** or **transform** task.
2. Model: There are two possible model-input types:
    1. model: This is a model input by the same type of component. For example, hetero_binning_0 run as a fit component, and hetero_binning_1 take model output of hetero_binning_0 as input so that can be used to transform or predict.

    Here's an example showing this logic:

```
"hetero_feature_binning_1": {
  "module": "HeteroFeatureBinning",
  "input": {
      "data": {
          "data": [
              "dataio_1.validate_data"
          ]
      },
      "model": [
          "hetero_feature_binning_0.fit_model"
      ]
  },
  "output": {
      "data": ["validate_data"],
      "model": ["eval_model"]
  }
}
```

    2. isometric_model: This is used to specify the model input from upstream components.

    For example, feature selection will take feature binning as upstream model, since it will use information value as feature importance. Here's an example of feature selection component:

```
"hetero_feature_selection_0": {
    "module": "HeteroFeatureSelection",
    "input": {
        "data": {
```

(continues on next page)

(continued from previous page)

```
            "data": [
                "hetero_feature_binning_0.train"
            ]
        },
        "isometric_model": [
            "hetero_feature_binning_0.output_model"
        ]
    },
    "output": {
        "data": ["train"],
        "model": ["output_model"]
    }
}
```

3. output: Same as input, two types of output may occur which are data and model.
    1. Data: Specify the output data name
    2. Model: Specify the output model name

   You can take the above case as an example.

6.2 Submit Runtime Conf

Besides the dsl conf, user also need to prepare a submit runtime conf to set the parameters of each component.

initiator To begin with, the initiator should be specified in this runtime conf. Here is an exmaple of setting initiator:

```
"initiator": {
    "role": "guest",
    "party_id": 10000
}
```

role All the roles involved in this modeling task should be specified. Each element in the role should contain role name and their party ids. The reason for ids are with form of list is that there may exist multiple parties in one role.

```
"role": {
    "guest": [
      10000
    ],
    "host": [
      10000
    ],
    "arbiter": [
      10000
    ]
}
```

role_parameters Those parameters that are differ from party to party, should be indicated here. Please note that each parameters should has the form of list. Inside the role_parameters, party names are used as key and parameters of these parties are values. Take the following structure as an example:

```
"guest": {
  "args": {
    "data": {
      "train_data": [
        {
          "name": "1ca0d9eea77e11e9a84f5254005e961b",
          "namespace": "arbiter-10000#guest-10000#host-10000#train_input
↪#guest#10000"
        }
      ]
    }
  },
  "dataio_0": {
    "with_label": [
      true
    ],
    ...
  }
},
"host": {
  "args": {
    "data": {
      "train_data": [
        {
          "name": "3de22bdaa77e11e99c5d5254005e961b",
          "namespace": "arbiter-10000#guest-10000#host-10000#train_input
↪#host#10000"
        }
      ]
    }
  },
  "dataio_0": {
     ...
  }
  ...
}
```

As this example shows, for each party, the input parameters such as train_data, validate_data and so on should be list in args. The name and namespace above are table indicators for uploaded data.

Then, user can config parameters for each components. The component names should match names defined in the dsl config file. The content of each component parameters are defined in Param class located in federatedml/param.

algorithm_parameters If some parameters are the same among all parties, they can be set in algorithm_parameters. Here is an example showing how to do that.

```
"hetero_feature_binning_0": {
    ...
},
"hetero_feature_selection_0": {
    ...
},
"hetero_lr_0": {
```

(continues on next page)

(continued from previous page)

```
  "penalty": "L2",
  "optimizer": "rmsprop",
  "eps": 1e-5,
  "alpha": 0.01,
  "max_iter": 10,
  "converge_func": "diff",
  "batch_size": 320,
  "learning_rate": 0.15,
  "init_param": {
    "init_method": "random_uniform"
  },
  "cv_param": {
    "n_splits": 5,
    "shuffle": false,
    "random_seed": 103,
    "need_cv": false,

  }
}
```

Same with the form in role parameters, each key of the parameters are names of components that are defined in dsl config file.

After setting config files and submitting the task, fate-flow will combine the parameter list in role-parameters and algorithm parameters. If there are still some undefined fields, values in default runtime conf will be used. Then fate-flow will send these config files to their corresponding parties and start the federated modeling task.

6.3 Multi-host configuration

For multi-host modeling case, all the host's party ids should be list in the role field.

```
"role": {
  "guest": [
    10000
  ],
  "host": [
    10000, 10001, 10002
  ],
  "arbiter": [
    10000
  ]
}
```

Each parameter set for host should also be list in a list. The number of elements should match the number of hosts.

```
"host": {
    "args": {
      "data": {
        "train_data": [
          {
            "name": "hetero_breast_host_1",
```

(continues on next page)

(continued from previous page)

```
            "namespace": "hetero_breast_host"
          },
          {
            "name": "hetero_breast_host_2",
            "namespace": "hetero_breast_host"
          },
          {
            "name": "hetero_breast_host_3",
            "namespace": "hetero_breast_host"
          }

        ]
      }
    },
    "dataio_0": {
      "with_label": [false, false, false],
      "output_format": ["dense", "dense", "dense"],
      "outlier_replace": [true, true, true]
    }
```

The parameters set in algorithm parameters need not be copied into host role parameters. Algorithm parameters will be copied for every party.

CHAPTER

SEVEN

DSL & TASK SUBMIT RUNTIME CONF SETTING V2

[]

To make the modeling task more flexible, currently, FATE uses its own domain-specific language(DSL) to describe modeling task. With usage of this DSL, modeling components such as data-io, feature-engineering and classification/regression module etc. can be combined as a Directed Acyclic Graph(DAG). Therefore, user can take and combine the algorithm components flexibly according to their needs.

In addition, parameters of each component need to be configured. Also, the configuration may vary from party to party. For convenience, FATE configure all parameters for all parties and all components in one file. This guide will show you how to create such a configure file.

Starting at FATE-1.5.0, V2 of dsl and submit conf is recommended, but user can still use old configuration method of [V1]

7.1 DSL Configure File

7.1.1 1. Overview

We use json file which is actually a dictionary as a dsl config file.

7.1.2 2. Components

- **definition:** components in your modeling task, always the first level of dsl dict.
- **example:**

```
{
  "components" : {
          ...
      }
  }
```

- **explanation:**

Then each component should be defined on the second level. Here is an example of setting a component:

```
"dataio_0": {
      "module": "DataIO",
      "input": {
          "data": {
```

(continues on next page)

(continued from previous page)

```
                "data": [
                    "reader_0.train_data"
                ]
            }
        },
        "output": {
            "data": ["train"],
            "model": ["dataio"]
        }
    }
```

As the example shows, user define the component name as key of this module.

Please note that in DSL V2, all modeling task config should contain a **Reader** component to reader data from storage service, this component has "output" field only, like the following:

```
"reader_0": {
        "module": "Reader",
        "output": {
            "data": ["train"]
        }
}
```

7.1.3 3. Module

- **definition:** Specify which component to use.
- **explanation:** This field should strictly match the file name in python/federatedml/conf/setting_conf except the `.json` suffix.
- **example:**

```
"hetero_feature_binning_1": {
    "module": "HeteroFeatureBinning",
     ...
}
```

7.1.4 4. Input

- **definition:** There are two types of input, data and model.

4.1 Data Input

- **definition:** Data input from previous modules; there are four possible data_input type: 1. data: typically used in data_io, feature_engineering modules and evaluation. 2. train_data: uses in training components like HeteroLR-HeteroSBT and so on. If this field is provided, the task will be parse as a **fit** task 3. validate_data: If train_data is provided, this field is optional. In this case, this data will be used as validation set. 4. test_data: specify the data used to predict, if this field is set up, the **model** also needs.

4.2 Model Input

- **definition:** Model input from previous modules; there are two possible model-input types:

1. model: This is a model input by the same type of component. For example, hetero_binning_0 run as a fit component, and hetero_binning_1 takes model output of hetero_binning_0 as input so that can be used to transform or predict. Here's an example showing this logic:

```
"hetero_feature_binning_1": {
    "module": "HeteroFeatureBinning",
    "input": {
        "data": {
            "data": [
                "dataio_1.validate_data"
            ]
        },
        "model": [
            "hetero_feature_binning_0.fit_model"
        ]
    },
    "output": {
        "data": ["validate_data"],
      "model": ["eval_model"]
    }
}
```

2. isometric_model: This is used to specify the model input from upstream components. For example, feature selection will take feature binning as upstream model, since it will use information value as feature importance. Here's an example of feature selection component:

```
"hetero_feature_selection_0": {
    "module": "HeteroFeatureSelection",
    "input": {
        "data": {
            "data": [
                "hetero_feature_binning_0.train"
            ]
        },
        "isometric_model": [
            "hetero_feature_binning_0.output_model"
        ]
    },
    "output": {
        "data": ["train"],
        "model": ["output_model"]
    }
}
```

4.3 Model Output

- **definition:** Same as input, two types of output may occur: which are data and model.

5.1 Data Output

- **definition:** data output, there are four types:

1. data: normal data output
2. train_data: only for Data Split
3. validate_data: only for Data Split
4. test_data only for Data Split

5.2 Model Output

- **definition:** model output, only use `model`

7.2 JOB RUNTIME CONFIG Guide (for version 1.5.x and above)

7.2.1 1. Overview

Job Runtime Conf configures job and module settings for all participants. Configurable values include:

7.2.2 2. DSL version

- **definition:** conf version, default 1, 2 is recommended
- **example:**

```
"dsl_version": "2"
```

7.2.3 3. Job Participants

3.1 Initiator

- **definition:** role and party_id of job initiator
- **example:**

```
"initiator": {
    "role": "guest",
    "party_id": 9999
}
```

3.2 Role

- **definition:** Information on all participants
- **explanation:** each key-value pair in `role` represents a role type and corresponding party ids; `party_id` should be specified as list since multiple parties may take the same role in a job
- **examples**

```
"role": {
    "guest": [9999],
    "host": [10000],
    "arbiter": [10000]
}
```

7.2.4 4. System Runtime Parameters

- **definition:** main system configuration when running jobs

4.1 Configuration Applicable Range Policy

- `common`: applies to all participants
- `role`: applies only to specific participant; specify participant in $role$:party_index format; note that `role` configuration takes priority over `common`

```
"common": {
}

"role": {
  "guest": {
    "0": {
    }
  }
}
```

In the example above, configuration inside`common` applies to all participants; configuration inside `role-guest-0` only applies to participant `guest_0`

Note: current version does not perform strict checking on role-specific runtime parameters; `common` is suggested for setting runtime configuration

4.2 Configurable Job Parameters

Table 1: Configurable Job Parameters

Parameter Name	Default Value	Acceptable Values	Information
job_type	train	train, predict	job type
work_mode	0	0, 1	0 for standalone, 1 for cluster
backend	0	0, 1	0 for EGGROLL, 1 for SPARK
task_cores	4	positive integer	total cpu cores requested
task_parallelism	1	positive int	maximum number of tasks allowed to run in parallel
computing_partitions	same as task_cores	positive integer	partition number for table computing
eggroll_run		processors_per_node	configuration specific for EGGROLL computing engine; generally set automatically based on `task_cores`; if specified, `task_cores` value ineffective
spark_run		num-executors, executor-cores	configuration specific for SPARK computing engine; generally set automatically based on `task_cores`; if specified, `task_cores` value ineffective
rabbitmq_run		queue, exchange etc.	parameters for rabbitmq to set up queue, exchange, etc.; generally takes system default
federated_status_collect_type	PUSH	PUSH, PULL	way to collect federated job status; PUSH: participants report to initiator, PULL: initiator regularly queries from all participants
timeout	259200 (3 days)	positive int	time elapse (in second) for a job to timeout
model_id	-	-	id of model, needed for prediction task
model_version	-	-	version of model, needed for prediction task

Note: 1. Some types of `computing_engine`, `storage_engine`, and `federation_engine` are only compatible with each other. For examples, SPARK `computing_engine` only supports HDFS `storage_engine`.

2. Combination of `work_mode` and `backend` automatically determines which three engines will be used.

3. Developer may implement other types of engines and set new engine combinations in runtime conf.

4.3 Non-Configurable Job Parameters

Table 2: Non-configurable Job Parameters

Parameter Name	Default Value	Acceptable Values	Information
computing_engine	set automatically based on work_mode and backend	EGGROLL, SPARK, STANDALONE	engine for computation
storage_engine	set automatically based on work_mode and backend	EGGROLL, HDFS, STANDALONE	engine for storage
federation_engine	set automatically based on work_mode and backend	EGGROLL, RABBITMQ, STANDALONE	engine for communication among parties
federated_mode	set automatically based on work_mode and backend	SINGLE, MULTIPLE	federation mode

4.4 Example Job Parameter Configuration

1. **EGGROLL** conf example with default CPU settings:

```
"job_parameters": {
   "common": {
      "work_mode": 1,
      "backend": 0,
      "task_cores": 4
   }
}
```

2. **EGGROLL** conf example with manually specified CPU settings:

```
"job_parameters": {
   "common": {
        "job_type": "train",
        "work_mode": 1,
        "backend": 0,
        "eggroll_run": {
          "eggroll.session.processors.per.node": 2
        },
        "task_parallelism": 2,
        "computing_partitions": 8,
        "timeout": 36000,
   }
}
```

3. **SPARK** conf example with manually specified CPU settings:

```
"job_parameters": {
   "common": {
       "job_type": "train",
       "work_mode": 1,
       "backend": 1,
       "spark_run": {
           "num-executors": 1,
           "executor-cores": 2
       },
       "task_parallelism": 2,
       "computing_partitions": 8,
       "timeout": 36000,
       "rabbitmq_run": {
           "queue": {
               "durable": true
           },
           "connection": {
               "heartbeat": 10000
           }
       }
   }
}
```

4.5 Resource Management

Starting at version 1.5.0, FATE-Flow implements improved, more fine-grained resource management policy on cpu cores, lifting restrictions on number of parallel tasks in previous versions.

4.5.1 Total Resource Setting

- resource comes from underlying engines; since current version does automatically obtain resource information from engines, FATE-Flow server obtains and register engine information to `t_engine_registry` from user-specified conf file `$PROJECT_BASE/conf/service_conf.yaml`
- fate_on_eggrolltotal_cores=cores_per_node*nodes
- fate_on_sparktotal_cores=cores_per_node*nodes
- standaloneuse **STANDALONE_BACKEND_VIRTUAL_CORES_PER_NODE**from `$PROJECT_BASE/python/fate_flow/settings.py`
- separate computing resources for different engines
- above settings effective after restarting FATE-Flow server

4.5.2 Calculate Computing Resource

Calculate actual `task_run_cores` each task requests at computing engine, may not equal to the amount applied by resource manager

1. only set `task_cores` in job conf:
   - task_run_cores(guest, host)max(task_cores / total_nodes, 1) * total_nodes
   - task_run_cores(arbiter)max(1 / total_nodes, 1) * total_nodes
   - FATE-Flow will automatically convert `task_cores` value into engine-specific configuration: eggroll.session.processors.per.node for EGGROLL, and executor-cores & num-executors for SPARK
2. set eggroll_run in job conf
   - task_run_cores(guest, host, arbiter)eggroll.session.processors.per.node * total_nodes
3. set spark_run in job conf
   - task_run_cores(guest, host, arbiter)executor-cores * num-executors

4.5.3 Resource Manager

1. Apply Resource for Jobs
   - Computing Engine set to EGGROLL, STANDALONE
     - apply_cores(guest, host): task_run_cores * task_parallelism
     - apply_cores(arbiter): 0, because actual resource cost is minimal and EGGROLL currently sets the same cores for all nodes, set to **0** to avoid unnecessary job queueing due to resource need from arbiter
     - note: on EGGROLL cluster, each node always assigns arbiter task_run_cores/nodes cores
   - Computing Engine set to SPARK
     - SPARK supports executor-cores * num-executors; not strongly correlated with number of cluster nodes due to SPARK own resource manager; if the calculated resource different from the one actually applied, jobs may keep waiting on SPARK engine
     - apply_cores(guest, host, arbiter): task_run_cores * task_parallelism
2. Job Management Policy
   - Enqueue by job submission time
   - Currently only support FIFO policy: manager only applies resources for the first job, deque the first job if success, wait for the next round if failure
3. Resource Application Policy
   - Manager selects job following the above guidelines and distribute federated resource application request to all participants
   - If all participants successfully secure resource, i.e.: (total_cores - apply_cores > 0), then the job succeeds in resource application
   - If not all participants succeeds, then send rollback request to succeeded participants, and the job fails in resource application

7.2.5 5. Component Parameter Configuration

5.1 Configuration Applicable Range Policy

- `common`: applies to all participants
- `role`: applies only to specific participant; specify participant in $role:$party_index format; note that `role` configuration takes priority over `common`

```
"commom": {
}

"role": {
  "guest": {
    "0": {
    }
  }
}
```

In the example above, configuration inside``common`` applies to all participants; configuration inside `role-guest-0` only applies to participant *guest_0*

Note: current version now supports checking on both fields of specification.

5.2 Example Component Parameter Configuration

- Configuration of modules `intersection_0`& `hetero_lr_0`are put inside `common`, thus applies to all participants
- Configuration of modules `reader_0`& `dataio_0`are specified for each participant
- Names of the above modules are specified in dsl file

```
"component_parameters": {
  "common": {
    "intersection_0": {
      "intersect_method": "raw",
      "sync_intersect_ids": true,
      "only_output_key": false
    },
    "hetero_lr_0": {
      "penalty": "L2",
      "optimizer": "rmsprop",
      "alpha": 0.01,
      "max_iter": 3,
      "batch_size": 320,
      "learning_rate": 0.15,
      "init_param": {
        "init_method": "random_uniform"
      }
    }
  },
  "role": {
    "guest": {
      "0": {
```

(continues on next page)

(continued from previous page)

```
        "reader_0": {
          "table": {"name": "breast_hetero_guest", "namespace": "experiment"}
        },
        "dataio_0":{
          "with_label": true,
          "label_name": "y",
          "label_type": "int",
          "output_format": "dense"
        }
      }
    },
    "host": {
      "0": {
        "reader_0": {
          "table": {"name": "breast_hetero_host", "namespace": "experiment"}
        },
        "dataio_0":{
          "with_label": false,
          "output_format": "dense"
        }
      }
    }
  }
}
```

5.3 Multi-host configuration

For multi-host modeling case, all the host's party ids should be list in the role field.

```
"role": {
   "guest": [
     10000
   ],
   "host": [
     10000, 10001, 10002
   ],
   "arbiter": [
     10000
   ]
}
```

Each parameter set for host should also be config The number of elements should match the number of hosts.

```
"component_parameters": {
   "role": {
      "host": {
         "0": {
            "reader_0": {
               "table":
                {
                  "name": "hetero_breast_host_0",
```

(continues on next page)

(continued from previous page)

```
                    "namespace": "hetero_breast_host"
                }
            }
        },
        "1": {
            "reader_0": {
                "table":
                {
                    "name": "hetero_breast_host_1",
                    "namespace": "hetero_breast_host"
                }
            }
        },
        "2": {
            "reader_0": {
                "table":
                {
                    "name": "hetero_breast_host_2",
                    "namespace": "hetero_breast_host"
                }
            }
        }
    }
}
```

The parameters set in common parameters need not be copied into host role parameters. Common parameters will be copied for every party.

5.4 Prediction configuration

5.4.1 Overview

Please note that in dsl v2predict dsl is not automatically generated after training. User should first deploy needed components with Flow Client. Please refer to FATE-Flow document for details on using deploy command:

```
flow model deploy --model-id $model_id --model-version $model_version --cpn-list ...
```

Optionally, user can add additional component(s) to predict dsl, like `Evaluation`:

5.4.2 Example

training dsl:

```
"components": {
    "reader_0": {
        "module": "Reader",
        "output": {
            "data": [
                "data"
```

(continues on next page)

(continued from previous page)

```
            ]
        }
    },
    "dataio_0": {
        "module": "DataIO",
        "input": {
            "data": {
                "data": [
                    "reader_0.data"
                ]
            }
        },
        "output": {
            "data": [
                "data"
            ],
            "model": [
                "model"
            ]
        }
    },
    "intersection_0": {
        "module": "Intersection",
        "input": {
            "data": {
                "data": [
                    "dataio_0.data"
                ]
            }
        },
        "output": {
            "data":[
                "data"
            ]
        }
    },
    "hetero_nn_0": {
        "module": "HeteroNN",
        "input": {
            "data": {
                "train_data": [
                    "intersection_0.data"
                ]
            }
        },
        "output": {
            "data": [
                "data"
            ],
            "model": [
                "model"
            ]
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

predict dsl:

```
"components": {
    "reader_0": {
        "module": "Reader",
        "output": {
            "data": [
                "data"
            ]
        }
    },
    "dataio_0": {
        "module": "DataIO",
        "input": {
            "data": {
                "data": [
                    "reader_0.data"
                ]
            }
        },
        "output": {
            "data": [
                "data"
            ],
            "model": [
                "model"
            ]
        }
    },
    "intersection_0": {
        "module": "Intersection",
        "input": {
            "data": {
                "data": [
                    "dataio_0.data"
                ]
            }
        },
        "output": {
            "data":[
                "data"
            ]
        }
    },
    "hetero_nn_0": {
        "module": "HeteroNN",
        "input": {
            "data": {
```

(continues on next page)

(continued from previous page)

```
                "train_data": [
                    "intersection_0.data"
                ]
            }
        },
        "output": {
            "data": [
                "data"
            ],
            "model": [
                "model"
            ]
        }
    },
    "evaluation_0": {
        "module": "Evaluation",
        "input": {
            "data": {
                "data": [
                    "hetero_nn_0.data"
                ]
            }
         },
         "output": {
             "data": [
                 "data"
             ]
          }
    }
```

7.2.6 6. Basic Workflow

1. After job submission, FATE-Flow obtains job dsl and job config and store them inside job folder under corresponding directory `$PROJECT_BASE/jobs/$jobid/`
2. Parse job dsl & job config, generate fine-grained configuration according to provided settings (as mentioned above, backend & work_mode together determines configration for three engines) and fill in default parameter values
3. Distribute and store common configuration to each party, generate and store party-specific **job_runtime_on_party_conf**under jobs directory
4. Each party execute job following **job_runtime_on_party_conf**

CHAPTER

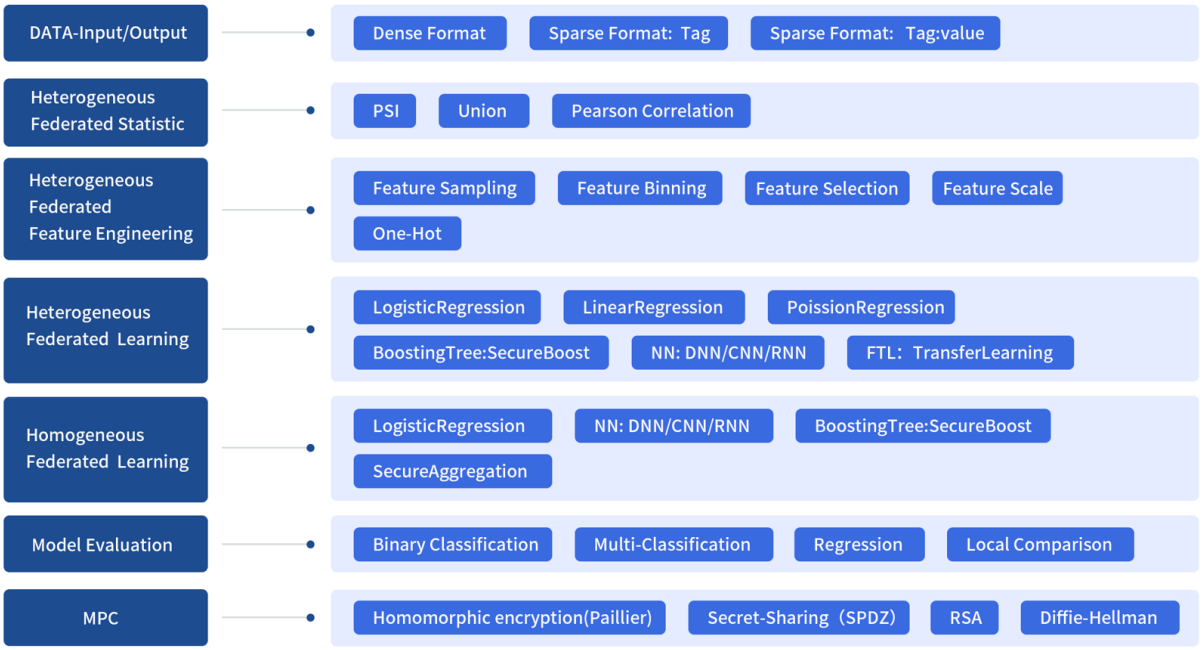
EIGHT

FEDERATED MACHINE LEARNING

[]

FederatedML includes implementation of many common machine learning algorithms on federated learning. All modules are developed in a decoupling modular approach to enhance scalability. Specifically, we provide:

1. Federated Statistic: PSI, Union, Pearson Correlation, etc.
2. Federated Feature Engineering: Feature Sampling, Feature Binning, Feature Selection, etc.
3. Federated Machine Learning Algorithms: LR, GBDT, DNN, TransferLearning, which support Heterogeneous and Homogeneous styles.
4. Model Evaluation: Binary | Multiclass | Regression | Clustering Evaluation, Local vs Federated Comparison.
5. Secure Protocol: Provides multiple security protocols for secure multi-party computing and interaction between participants.



8.1 Algorithm List

Table 1: Algorithm

Algorithm	Module Name	Description	Data Input	Data Output	Model Input	Model Output
Reader	Reader	This component loads and transforms data from storage engine so that data is compatible with FATE computing engine	Original Data	Transformed Data		
DataIO	DataIO	This component transforms user-uploaded date into Instance object.	Table, values are raw data.	Transformed Table, values are data instance defined here		DataIO Model
Intersect	Intersection	Compute intersect data set of multiple parties without leakage of difference set information. Mainly used in hetero scenario task.	Table.	Table with only common instance keys.		Intersect Model
Federated Sampling	FederatedSample	Federated Sampling data so that its distribution become balance in each party.This module supports standalone and federated versions.	Table	Table of sampled data; both random and stratified sampling methods are supported.		
Feature Scale	FeatureScale	module for feature scaling and standardization.	Table-values are instances.	Transformed Table.	Transform factors like min/max, mean/std.	

continues on next page

Table 1 – continued from previous page

Algorithm	Module Name	Description	Data Input	Data Output	Model Input	Model Output
Hetero Feature Binning	Hetero Feature Binning	With binning input data, calculates each column's iv and woe and transform data according to the binned information.	Table, values are instances.	Transformed Table.		iv/woe, split points, event count, non-event count etc. of each column.
OneHot Encoder	OneHotEncoder	Transfer a column into one-hot format.	Table, values are instances.	Transformed Table with new header.		Feature-name mapping between original header and new header.
Hetero Feature Selection	Hetero-Feature-Selection	Provide 5 types of filters. Each filters can select columns according to user config	Table	Transformed Table with new header and filtered data instance.	If iv filters used, hetero_binning model is needed.	Whether each column is filtered.
Union	Union	Combine multiple data tables into one.	Tables.	Table with combined values from input Tables.		
Hetero-LR	HeteroLR	Build hetero logistic regression model through multiple parties.	Table, values are instances	Table, values are instances.		Logistic Regression Model, consists of model-meta and model-param.

continues on next page

Table 1 – continued from previous page

Algorithm	Module Name	Description	Data Input	Data Output	Model Input	Model Output
Local Baseline	LocalBaseline	Wrapper that runs sklearn(scikit-learn) Logistic Regression model with local data.	Table, values are instances.	Table, values are instances.		
HeteroLinR	HeteroLinR	Build hetero linear regression model through multiple parties.	Table, values are instances.	Table, values are instances.		Linear Regression Model, consists of model-meta and model-param.
HeteroPoisson	HeteroPoisson	Build hetero poisson regression model through multiple parties.	Table, values are instances.	Table, values are instances.		Poisson Regression Model, consists of model-meta and model-param.
HomoLR	HomoLR	Build homo logistic regression model through multiple parties.	Table, values are instances.	Table, values are instances.		Logistic Regression Model, consists of model-meta and model-param.
HomoNN	HomoNN	Build homo neural network model through multiple parties.	Table, values are instances.	Table, values are instances.		Neural Network Model, consists of model-meta and model-param.

continues on next page

Table 1 – continued from previous page

Algorithm	Module Name	Description	Data Input	Data Output	Model Input	Model Output
Hetero Secure Boosting	HeteroSecureBoost	Build hetero secure boosting model through multiple parties	Table, values are instances.	Table, values are instances.		SecureBoost Model, consists of model-meta and model-param.
Hetero Fast Secure Boosting	HeteroFastSecureBoost	Build hetero secure boosting model through multiple parties in layered/mix manners.	Table, values are instances.	Table, values are instances.		FastSecureBoost Model, consists of model-meta and model-param.
Evaluation	Evaluation	Output the model evaluation metrics for user.	Table(s), values are instances.			
Hetero Pearson	HeteroPearson	Calculate hetero correlation of features from different parties.	Table, values are instances.			
HeteroNN	HeteroNN	Build hetero neural network model.	Table, values are instances.	Table, values are instances.		Hetero Neural Network Model, consists of model-meta and model-param.

continues on next page

Table 1 – continued from previous page

Algorithm	Module Name	Description	Data Input	Data Output	Model Input	Model Output
Homo Secure Boosting	HomoSecureBoost	Build homo secure boosting model through multiple parties	Table, values are instances.	Table, values are instances.		SecureBoost Model, consists of model-meta and model-param.
Homo OneHot Encoder	HomoOneHotEncoder	Build homo onehot encoder model through multiple parties.	Table, values are instances.	Transformed Table with new header.		Feature-name mapping between original header and new header.
Data Split	Data Split	Split one data table into 3 tables by given ratio or count	Table, values are instances.	3 Tables, values are instance.		
Column Expand	Column Expand	Add arbitrary number of columns with user-provided values.	Table, values are raw data.	Transformed Table with added column(s) and new header.		Column Expand Model
Secure Information Retrieval	Secure Information Retrieval	Securely retrieves information from host through oblivious transfer	Table, values are instance	Table, values are instance		
Hetero Federated Transfer Learning	Hetero FTL	Build Hetero FTL Model Between 2 party	Table, values are instance			Hetero FTL Model

continues on next page

Table 1 – continued from previous page

Algorithm	Module Name	Description	Data Input	Data Output	Model Input	Model Output
Hetero KMeans	Hetero KMeans	Build Hetero KMeans model through multiple parties	Table, values are instance	Table, values are instance; Arbier outputs 2 Tables		Hetero KMeans Model
PSI	PSI module	Compute PSI value of features between two table	Table, values are instance			PSI Results
Data Statistics	Data Statistics	This component will do some statistical work on the data, including statistical mean, maximum and minimum, median, etc.	Table, values are instance	Table		Statistic Result
Scorecard	Scorecard	Scale predict score to credit score by given scaling parameters	Table, values are predict score	Table, values are score results		
Feldman Verifiable Sum	Feldman Verifiable Sum	This component will sum multiple privacy values without exposing data	Table, values are addend	Table, values are sum results		

8.2 Secure Protocol

- Encrypt
  - Paillier encryption
  - RSA encryption
  - Fake encryption
- Encode
- Diffne Hellman Key Exchange
- SecretShare MPC Protocol(SPDZ)
- Oblivious Transfer
- Feldman Verifiable Secret Sharing

8.3 Params

Classes:

BoostingParam([task_type, objective_param, ...])	Basic parameter for Boosting Algorithms
CrossValidationParam([n_splits, mode, role, ...])	Define cross validation params
DataIOParam([input_format, delimitor, ...])	Define dataio parameters that used in federated ml.
DataSplitParam([random_state, test_size, ...])	Define data split param that used in data split.
DecisionTreeParam([criterion_method, ...])	Define decision tree parameters that used in federated ml.
EncryptParam([method, key_length])	Define encryption method that used in federated ml. :param method: If method is 'Paillier', Paillier encryption will be used for federated ml. To use non-encryption version in HomoLR, set this to None. For detail of Paillier encryption, please check out the paper mentioned in README file. :type method: {'Paillier'} :param key_length: Used to specify the length of key in this encryption method. :type key_length: int, default: 1024.
EncryptedModeCalculatorParam([mode, ...])	Define the encrypted_mode_calulator parameters.
FeatureBinningParam([method, ...])	Define the feature binning method
FeatureSelectionParam([select_col_indexes, ...])	Define the feature selection parameters.
HeteroNNParam([task_type, config_type, ...])	Parameters used for Homo Neural Network.
HomoNNParam(secure_aggregate, ...[, ...])	Parameters used for Homo Neural Network.
HomoOneHotParam([transform_col_indexes, ...])	**param transform_col_indexes** Specify which columns need to calculated. -1 represent for all columns.
InitParam([init_method, init_const, ...])	Initialize Parameters used in initializing a model.
IntersectParam(intersect_method[, ...])	Define the intersect method
LinearParam([penalty, tol, alpha, ...])	Parameters used for Linear Regression.
LocalBaselineParam([model_name, model_opts, ...])	Define the local baseline model param
LogisticParam([penalty, tol, alpha, ...])	Parameters used for Logistic Regression both for Homo mode or Hetero mode.
ObjectiveParam([objective, params])	Define objective parameters that used in federated ml.
OneVsRestParam([need_one_vs_rest, has_arbiter])	Define the one_vs_rest parameters.
PoissonParam([penalty, tol, alpha, ...])	Parameters used for Poisson Regression.
PredictParam([threshold])	Define the predict method of HomoLR, HeteroLR, SecureBoosting
RsaParam([rsa_key_n, rsa_key_e, rsa_key_d, ...])	Define the sample method
SampleParam([mode, method, fractions, ...])	Define the sample method
ScaleParam([method, mode, ...])	Define the feature scale parameters.
StatisticsParam([statistics, column_names, ...])	Define statistics params
StepwiseParam([score_name, mode, role, ...])	Define stepwise params
StochasticQuasiNewtonParam([...])	Parameters used for stochastic quasi-newton method.
UnionParam([need_run, allow_missing, ...])	Define the union method for combining multiple dTables and keep entries with the same id

class BoostingParam(*task_type='classification'*, *objective_param=<federatedml.param.boosting_param.ObjectiveParam object>*, *learning_rate=0.3*, *num_trees=5*, *subsample_feature_rate=1*, *n_iter_no_change=True*, *tol=0.0001*, *bin_num=32*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *validation_freqs=None*, *metrics=None*, *subsample_random_seed=None*, *binning_error=0.0001*)

Basic parameter for Boosting Algorithms

Parameters

- **task_type** (*str, accepted 'classification', 'regression' only, default: 'classification'*) –
- **objective_param** (*ObjectiveParam Object, default: ObjectiveParam()*) –
- **learning_rate** (*float, accepted float, int or long only, the learning rate of secure boost. default: 0.3*) –
- **num_trees** (*int, accepted int, float only, the max number of boosting round. default: 5*) –
- **subsample_feature_rate** (*float, a float-number in [0, 1], default: 0.8*) –
- **n_iter_no_change** (*bool,*) – when True and residual error less than tol, tree building process will stop. default: True
- **bin_num** (*int, positive integer greater than 1, bin number use in quantile. default: 32*) –
- **validation_freqs** (*None or positive integer or container object in python. Do validation in training process or Not.*) – if equals None, will not do validation in train process; if equals positive integer, will validate data every validation_freqs epochs passes; if container object in python, will validate data if epochs belong to this container.

  e.g. validation_freqs = [10, 15], will validate data when epoch equals to 10 and 15.

  Default: None

class CrossValidationParam(*n_splits=5*, *mode='hetero'*, *role='guest'*, *shuffle=True*, *random_seed=1*, *need_cv=False*)

Define cross validation params

Parameters

- **n_splits** (*int, default: 5*) – Specify how many splits used in KFold
- **mode** (*str, default: 'Hetero'*) – Indicate what mode is current task
- **role** (*str, default: 'Guest'*) – Indicate what role is current party
- **shuffle** (*bool, default: True*) – Define whether do shuffle before KFold or not.
- **random_seed** (*int, default: 1*) – Specify the random seed for numpy shuffle
- **need_cv** (*bool, default True*) – Indicate if this module needed to be run

class DataIOParam(*input_format='dense'*, *delimitor=','*, *data_type='float64'*, *exclusive_data_type=None*, *tag_with_value=False*, *tag_value_delimitor=':'*, *missing_fill=False*, *default_value=0*, *missing_fill_method=None*, *missing_impute=None*, *outlier_replace=False*, *outlier_replace_method=None*, *outlier_impute=None*, *outlier_replace_value=0*, *with_label=False*, *label_name='y'*, *label_type='int'*, *output_format='dense'*, *need_run=True*)

Define dataio parameters that used in federated ml.

Parameters

- **input_format** (*str, accepted 'dense','sparse' 'tag' only in this version. default: 'dense'.*) – please have a look at this tutorial at "DataIO" section of federatedml/util/README.md. Formally,

  dense input format data should be set to "dense", svm-light input format data should be set to "sparse", tag or tag:value input format data should be set to "tag".

- **delimitor** (*str, the delimitor of data input, default: ','*) –
- **data_type** (*str, the data type of data input, accepted 'float','float64', 'int','int64','str','long'*) – "default: "float64"
- **exclusive_data_type** (*dict, the key of dict is col_name, the value is data_type, use to specified special data type*) – of some features.
- **tag_with_value** (*bool, use if input_format is 'tag', if tag_with_value is True,*) – input column data format should be tag[delimitor]value, otherwise is tag only
- **tag_value_delimitor** (*str, use if input_format is 'tag' and 'tag_with_value' is True,*) – delimitor of tag[delimitor]value column value.
- **missing_fill** (*bool, need to fill missing value or not, accepted only True/False, default: False*) –
- **default_value** (*None or single object type or list, the value to replace missing value.*) – if None, it will use default value define in federatedml/feature/imputer.py, if single object, will fill missing value with this object, if list, it's length should be the sample of input data' feature dimension,

  means that if some column happens to have missing values, it will replace it the value by element in the identical position of this list.

  default: None

- **missing_fill_method** (*None or str, the method to replace missing value, should be one of [None, 'min', 'max', 'mean', 'designated'], default: None*) –
- **missing_impute** (*None or list, element of list can be any type, or auto generated if value is None, define which values to be consider as missing, default: None*) –
- **outlier_replace** (*bool, need to replace outlier value or not, accepted only True/False, default: True*) –
- **outlier_replace_method** (*None or str, the method to replace missing value, should be one of [None, 'min', 'max', 'mean', 'designated'], default: None*) –
- **outlier_impute** (*None or list, element of list can be any type, which values should be regard as missing value, default: None*) –

- **outlier_replace_value** (*None or single object type or list, the value to replace outlier.*) – if None, it will use default value define in federatedml/feature/imputer.py, if single object, will replace outlier with this object, if list, it's length should be the sample of input data' feature dimension,

  means that if some column happens to have outliers, it will replace it the value by element in the identical position of this list.

  default: None

- **with_label** (*bool, True if input data consist of label, False otherwise. default: 'false'*) –
- **label_name** (*str, column_name of the column where label locates, only use in dense-inputformat. default: 'y'*) –
- **label_type** (*object, accepted 'int','int64','float','float64','long','str' only,*) – use when with_label is True. default: 'false'
- **output_format** (*str, accepted 'dense','sparse' only in this version. default: 'dense'*) –

class DataSplitParam(*random_state=None*, *test_size=None*, *train_size=None*, *validate_size=None*, *stratified=False*, *shuffle=True*, *split_points=None*, *need_run=True*)

Define data split param that used in data split.

Parameters

- **random_state** (*None, int, default: None*) – Specify the random state for shuffle.
- **test_size** (*None, float, int, default: 0.0*) – Specify test data set size. float value specifies fraction of input data set, int value specifies exact number of data instances
- **train_size** (*None, float, int, default: 0.8*) – Specify train data set size. float value specifies fraction of input data set, int value specifies exact number of data instances
- **validate_size** (*None, float, int, default: 0.2*) – Specify validate data set size. float value specifies fraction of input data set, int value specifies exact number of data instances
- **stratified** (*boolean, default: False*) – Define whether sampling should be stratified, according to label value.
- **shuffle** (*boolean, default : True*) – Define whether do shuffle before splitting or not.
- **split_points** (*None, list, default : None*) – Specify the point(s) by which continuous label values are bucketed into bins for stratified split. eg.[0.2] for two bins or [0.1, 1, 3] for 4 bins
- **need_run** (*bool, default: True*) – Specify whether to run data split

class DecisionTreeParam(*criterion_method='xgboost'*, *criterion_params=[0.1]*, *max_depth=3*, *min_sample_split=2*, *min_impurity_split=0.001*, *min_leaf_node=1*, *max_split_nodes=65536*, *feature_importance_type='split'*, *n_iter_no_change=True*, *tol=0.001*, *use_missing=False*, *zero_as_missing=False*)

Define decision tree parameters that used in federated ml.

Parameters

- **criterion_method** (*str, accepted "xgboost" only, the criterion function to use, default: 'xgboost'*) –

- **criterion_params** (*list, should be non empty and first element is float-number, default: 0.1.*) –
- **max_depth** (*int, positive integer, the max depth of a decision tree, default: 3*) –
- **min_sample_split** (*int, least quantity of nodes to split, default: 2*) –
- **min_impurity_split** (*float, least gain of a single split need to reach, default: 1e-3*) –
- **min_leaf_node** (*int, when samples no more than min_leaf_node, it becomes a leave, default: 1*) –
- **max_split_nodes** (*int, positive integer, we will use no more than max_split_nodes to*) – parallel finding their splits in a batch, for memory consideration. default is 65536
- **feature_importance_type** (*str, support 'split', 'gain' only.*) – if is 'split', feature_importances calculate by feature split times, if is 'gain', feature_importances calculate by feature split gain. default: 'split'
- **use_missing** (*bool, accepted True, False only, use missing value in training process or not. default: False*) –
- **zero_as_missing** (*bool, accepted True, False only, regard 0 as missing value or not,*) – will be use only if use_missing=True, default: False

class **EncryptParam**(*method='Paillier'*, *key_length=1024*)

Define encryption method that used in federated ml. :param method: If method is 'Paillier', Paillier encryption will be used for federated ml.

To use non-encryption version in HomoLR, set this to None. For detail of Paillier encryption, please check out the paper mentioned in README file.

Parameters **key_length** (*int, default: 1024*) – Used to specify the length of key in this encryption method.

class **EncryptedModeCalculatorParam**(*mode='strict'*, *re_encrypted_rate=1*)

Define the encrypted_mode_calulator parameters.

Parameters

- **mode** (*str, support 'strict', 'fast', 'balance', 'confusion_opt', ' only, default: strict*) –
- **re_encrypted_rate** (*float or int, numeric number in [0, 1], use when mode equals to 'balance, default: 1*) –

class **FeatureBinningParam**(*method='quantile'*, *compress_thres=10000*, *head_size=10000*, *error=0.0001*, *bin_num=10*, *bin_indexes=-1*, *bin_names=None*, *adjustment_factor=0.5*, *transform_param=<federatedml.param.feature_binning_param.TransformParam object>*, *optimal_binning_param=<federatedml.param.feature_binning_param.OptimalBinningParam object>*, *local_only=False*, *category_indexes=None*, *category_names=None*, *need_run=True*, *skip_static=False*)

Define the feature binning method

Parameters

- **method** (*str, 'quantile' 'bucket' or 'optimal', default: 'quantile'*) – Binning method.

- **compress_thres** (*int, default: 10000*) – When the number of saved summaries exceed this threshold, it will call its compress function
- **head_size** (*int, default: 10000*) – The buffer size to store inserted observations. When head list reach this buffer size, the QuantileSummaries object start to generate summary(or stats) and insert into its sampled list.
- **error** (*float, 0 <= error < 1 default: 0.001*) – The error of tolerance of binning. The final split point comes from original data, and the rank of this value is close to the exact rank. More precisely, floor((p - 2 * error) * N) <= rank(x) <= ceil((p + 2 * error) * N) where p is the quantile in float, and N is total number of data.
- **bin_num** (*int, bin_num > 0, default: 10*) – The max bin number for binning
- **bin_indexes** (*list of int or int, default: -1*) – Specify which columns need to be binned. -1 represent for all columns. If you need to indicate specific cols, provide a list of header index instead of -1.
- **bin_names** (*list of string, default: []*) – Specify which columns need to calculated. Each element in the list represent for a column name in header.
- **adjustment_factor** (*float, default: 0.5*) – the adjustment factor when calculating WOE. This is useful when there is no event or non-event in a bin. Please note that this parameter will NOT take effect for setting in host.
- **category_indexes** (*list of int or int, default: []*) – Specify which columns are category features. -1 represent for all columns. List of int indicate a set of such features. For category features, bin_obj will take its original values as split_points and treat them as have been binned. If this is not what you expect, please do NOT put it into this parameters.

  The number of categories should not exceed bin_num set above.
- **category_names** (*list of string, default: []*) – Use column names to specify category features. Each element in the list represent for a column name in header.
- **local_only** (*bool, default: False*) – Whether just provide binning method to guest party. If true, host party will do nothing.
- **transform_param** (*TransformParam*) – Define how to transfer the binned data.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run
- **skip_static** (*bool, default False*) – If true, binning will not calculate iv, woe etc. In this case, optimal-binning will not be supported.

class FeatureSelectionParam(*select_col_indexes=-1*, *select_names=None*, *filter_methods=None*, *unique_param=<federatedml.param.feature_selection_param.UniqueValueParam object>*, *iv_value_param=<federatedml.param.feature_selection_param.IVValueSelectionParam object>*, *iv_percentile_param=<federatedml.param.feature_selection_param.IVPercentileSelectionParam object>*, *iv_top_k_param=<federatedml.param.feature_selection_param.IVTopKParam object>*, *variance_coe_param=<federatedml.param.feature_selection_param.VarianceOfCoeSelectionParam object>*, *outlier_param=<federatedml.param.feature_selection_param.OutlierColsSelectionParam object>*, *manually_param=<federatedml.param.feature_selection_param.ManuallyFilterParam object>*, *percentage_value_param=<federatedml.param.feature_selection_param.PercentageValueParam object>*, *iv_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *statistic_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *psi_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *sbt_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *need_run=True*)

Define the feature selection parameters.

Parameters

- **select_col_indexes** (*list or int, default: -1*) – Specify which columns need to calculated. -1 represent for all columns.
- **select_names** (*list of string, default: []*) – Specify which columns need to calculated. Each element in the list represent for a column name in header.
- **filter_methods** (*list, ["manually", "iv_filter", "statistic_filter",*) –

  "psi_filter", "hetero_sbt_filter", "homo_sbt_filter", "hetero_fast_sbt_filter", "percentage_value"],

  default: ["manually"]

  The following methods will be deprecated in future version: "unique_value", "iv_value_thres", "iv_percentile", "coefficient_of_variation_value_thres", "outlier_cols"

  Specify the filter methods used in feature selection. The orders of filter used is depended on this list. Please be notified that, if a percentile method is used after some certain filter method, the percentile represent for the ratio of rest features.

  e.g. If you have 10 features at the beginning. After first filter method, you have 8 rest. Then, you want top 80% highest iv feature. Here, we will choose floor(0.8 * 8) = 6 features instead of 8.

- **unique_param** (*filter the columns if all values in this feature is the same*) –
- **iv_value_param** (*Use information value to filter columns. If this method is set, a float threshold need to be provided.*) – Filter those columns whose iv is smaller than threshold. Will be deprecated in the future.

- **iv_percentile_param** (*Use information value to filter columns. If this method is set, a float ratio threshold*) – need to be provided. Pick floor(ratio * feature_num) features with higher iv. If multiple features around the threshold are same, all those columns will be keep. Will be deprecated in the future.
- **variance_coe_param** (*Use coefficient of variation to judge whether filtered or not.*) – Will be deprecated in the future.
- **outlier_param** (*Filter columns whose certain percentile value is larger than a threshold.*) – Will be deprecated in the future.
- **percentage_value_param** (*Filter the columns that have a value that exceeds a certain percentage.*) –
- **iv_param** (*Setting how to filter base on iv. It support take high mode only. All of "threshold",*) – "top_k" and "top_percentile" are accepted. Check more details in CommonFilterParam. To use this filter, hetero-feature-binning module has to be provided.
- **statistic_param** (*Setting how to filter base on statistic values. All of "threshold",*) – "top_k" and "top_percentile" are accepted. Check more details in CommonFilterParam. To use this filter, data_statistic module has to be provided.
- **psi_param** (*Setting how to filter base on psi values. All of "threshold",*) – "top_k" and "top_percentile" are accepted. Its take_high properties should be False to choose lower psi features. Check more details in CommonFilterParam. To use this filter, data_statistic module has to be provided.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class HeteroNNParam(*task_type='classification'*, *config_type='keras'*, *bottom_nn_define=None*, *top_nn_define=None*, *interactive_layer_define=None*, *interactive_layer_lr=0.9*, *optimizer='SGD'*, *loss=None*, *epochs=100*, *batch_size=-1*, *early_stop='diff'*, *tol=1e-05*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *encrypted_mode_calculator_param=<federatedml.param.encrypted_mode_calculation_param.EncryptedMode object>*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *validation_freqs=None*, *early_stopping_rounds=None*, *metrics=None*, *use_first_metric_only=True*)

Parameters used for Homo Neural Network.

Parameters

- **task_type** – str, task type of hetero nn model, one of 'classification', 'regression'.
- **config_type** – str, accept "keras" only.
- **bottom_nn_define** – a dict represents the structure of bottom neural network.
- **interactive_layer_define** – a dict represents the structure of interactive layer.
- **interactive_layer_lr** – float, the learning rate of interactive layer.
- **top_nn_define** – a dict represents the structure of top neural network.
- **optimizer** – optimizer method, accept following types: 1. a string, one of "Adadelta", "Adagrad", "Adam", "Adamax", "Nadam", "RMSprop", "SGD" 2. a dict, with a required key-value pair keyed by "optimizer",

  with optional key-value pairs such as learning rate.

  defaults to "SGD"

- **loss** – str, a string to define loss function used
- **early_stopping_rounds** – int, default: None
- **rounds** (*Will stop training if one metric doesn't improve in last early_stopping_round*) –
- **metrics** – list, default: None Indicate when executing evaluation during train process, which metrics will be used. If not set, default metrics for specific task type will be used. As for binary classification, default metrics are ['auc', 'ks'], for regression tasks, default metrics are ['root_mean_squared_error', 'mean_absolute_error'], [ACCURACY, PRECISION, RECALL] for multi-classification task
- **use_first_metric_only** – bool, default: False Indicate whether to use the first metric in *metrics* as the only criterion for early stopping judgement.
- **epochs** – int, the maximum iteration for aggregation in training.
- **batch_size** – int, batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy. defaults to -1.
- **early_stop** – str, accept 'diff' only in this version, default: 'diff' Method used to judge converge or not.

  a) diff Use difference of loss between two iterations to judge whether converge.
- **validation_freqs** – None or positive integer or container object in python. Do validation in training process or Not. if equals None, will not do validation in train process; if equals positive integer, will validate data every validation_freqs epochs passes; if container object in python, will validate data if epochs belong to this container.

  e.g. validation_freqs = [10, 15], will validate data when epoch equals to 10 and 15.

  Default: None The default value is None, 1 is suggested. You can set it to a number larger than 1 in order to speed up training by skipping validation rounds. When it is larger than 1, a number which is divisible by "epochs" is recommended, otherwise, you will miss the validation scores of last training epoch.

class HomoNNParam(*secure_aggregate: bool = True*, *aggregate_every_n_epoch: int = 1*, *config_type: str = 'nn'*, *nn_define: typing.Optional[dict] = None*, *optimizer: typing.Union[str, dict, types.SimpleNamespace] = 'SGD'*, *loss: typing.Optional[str] = None*, *metrics: typing.Optional[typing.Union[str, list]] = None*, *max_iter: int = 100*, *batch_size: int = -1*, *early_stop: typing.Union[str, dict, types.SimpleNamespace] = 'diff'*, *encode_label: bool = False*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*)

Parameters used for Homo Neural Network.

Parameters **Args** – secure_aggregate: enable secure aggregation or not, defaults to True. aggregate_every_n_epoch: aggregate model every n epoch, defaults to 1. config_type: one of "nn", "keras", "tf" nn_define: a dict represents the structure of neural network. optimizer: optimizer method, accept following types:

1. a string, one of "Adadelta", "Adagrad", "Adam", "Adamax", "Nadam", "RMSprop", "SGD"
2. **a dict, with a required key-value pair keyed by "optimizer",** with optional key-value pairs such as learning rate.

defaults to "SGD"

loss: a string metrics: max_iter: the maximum iteration for aggregation in training. batch_size : batch size when updating model.

-1 means use all data in a batch. i.e. Not to use mini-batch strategy. defaults to -1.

early_stop [str, 'diff', 'weight_diff' or 'abs', default: 'diff']

Method used to judge converge or not.

a) diff Use difference of loss between two iterations to judge whether converge.

b) weight_diff: Use difference between weights of two consecutive iterations

c) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < eps, it is converged.

encode_label : encode label to one_hot.

class HomoOneHotParam(*transform_col_indexes=- 1*, *transform_col_names=None*, *need_run=True*, *need_alignment=True*)

Parameters

- **transform_col_indexes** (*list or int, default: -1*) – Specify which columns need to calculated. -1 represent for all columns.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run
- **need_alignment** (*bool, default True*) – Indicated whether alignment of features is turned on

class InitParam(*init_method='random_uniform'*, *init_const=1*, *fit_intercept=True*, *random_seed=None*)

Initialize Parameters used in initializing a model.

Parameters

- **init_method** (*str, 'random_uniform', 'random_normal', 'ones', 'zeros' or 'const'. default: 'random_uniform'*) – Initial method.
- **init_const** (*int or float, default: 1*) – Required when init_method is 'const'. Specify the constant.
- **fit_intercept** (*bool, default: True*) – Whether to initialize the intercept or not.

class IntersectParam(*intersect_method: str = 'rsa'*, *random_bit=128*, *sync_intersect_ids=True*, *join_role='guest'*, *with_encode=False*, *only_output_key=False*, *encode_params=<federatedml.param.intersect_param.EncodeParam object>*, *intersect_cache_param=<federatedml.param.intersect_param.IntersectCache object>*, *repeated_id_process=False*, *repeated_id_owner='guest'*, *allow_info_share: bool = False*, *info_owner='guest'*)

Define the intersect method

Parameters

- **intersect_method** (*str, it supports 'rsa' and 'raw', default by 'rsa'*) –
- **random_bit** (*positive int, it will define the encrypt length of rsa algorithm. It effective only for intersect_method is rsa*) –
- **sync_intersect_ids** (*bool. In rsa, 'synchronize_intersect_ids' is True means guest or host will send intersect results to the others, and False will not.*) – while in raw, 'synchronize_intersect_ids' is True means the role of "join_role" will send intersect results and the others will get them. Default by True.

- **join_role** (*str, it supports "guest" and "host" only and effective only for raw. If it is "guest", the host will send its ids to guest and find the intersection of*) – ids in guest; if it is "host", the guest will send its ids. Default by "guest".
- **with_encode** (*bool, if True, it will use encode method for intersect ids. It effective only for "raw".*) –
- **encode_params** (*EncodeParam, it effective only for with_encode is True*) –
- **only_output_key** (*bool, if false, the results of intersection will include key and value which from input data; if true, it will just include key from input*) – data and the value will be empty or some useless character like "intersect_id"
- **repeated_id_process** (*bool, if true, intersection will process the ids which can be repeatable*) –
- **repeated_id_owner** (*str, which role has the repeated ids*) –

class LinearParam(*penalty='L2'*, *tol=0.0001*, *alpha=1.0*, *optimizer='sgd'*, *batch_size=-1*, *learning_rate=0.01*, *init_param=<federatedml.param.init_model_param.InitParam object>*, *max_iter=20*, *early_stop='diff'*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *sqn_param=<federatedml.param.sqn_param.StochasticQuasiNewtonParam object>*, *encrypted_mode_calculator_param=<federatedml.param.encrypted_mode_calculation_param.EncryptedModeCa object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *decay=1*, *decay_sqrt=True*, *validation_freqs=None*, *early_stopping_rounds=None*, *stepwise_param=<federatedml.param.stepwise_param.StepwiseParam object>*, *metrics=None*, *use_first_metric_only=False*)

Parameters used for Linear Regression.

Parameters

- **penalty** (*str, 'L1' or 'L2'. default: 'L2'*) – Penalty method used in LinR. Please note that, when using encrypted version in HeteroLinR, 'L1' is not supported.
- **tol** (*float, default: 1e-4*) – The tolerance of convergence
- **alpha** (*float, default: 1.0*) – Regularization strength coefficient.
- **optimizer** (*str, 'sgd', 'rmsprop', 'adam', 'sqn', or 'adagrad', default: 'sgd'*) – Optimize method
- **batch_size** (*int, default: -1*) – Batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy.
- **learning_rate** (*float, default: 0.01*) – Learning rate
- **max_iter** (*int, default: 20*) – The maximum iteration for training.
- **init_param** (*InitParam object, default: default InitParam object*) – Init param method object.
- **early_stop** (*str, 'diff' or 'abs' or 'weight_dff', default: 'diff'*) –

  Method used to judge convergence.

  a) diff Use difference of loss between two iterations to judge whether converge.

  b) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < tol, it is converged.

c) weight_diff: Use difference between weights of two consecutive iterations

- **predict_param** (*PredictParam object, default: default PredictParam object*) –
- **encrypt_param** (*EncryptParam object, default: default EncryptParam object*) –
- **encrypted_mode_calculator_param** (*EncryptedModeCalculatorParam object, default: default EncryptedModeCalculatorParam object*) –
- **cv_param** (*CrossValidationParam object, default: default CrossValidationParam object*) –
- **decay** (*int or float, default: 1*) – Decay rate for learning rate. learning rate will follow the following decay schedule. lr = lr0/(1+decay*t) if decay_sqrt is False. If decay_sqrt is True, lr = lr0 / sqrt(1+decay*t) where t is the iter number.
- **decay_sqrt** (*Bool, default: True*) – lr = lr0/(1+decay*t) if decay_sqrt is False, otherwise, lr = lr0 / sqrt(1+decay*t)
- **validation_freqs** (*int, list, tuple, set, or None*) – validation frequency during training, required when using early stopping. The default value is None, 1 is suggested. You can set it to a number larger than 1 in order to speed up training by skipping validation rounds. When it is larger than 1, a number which is divisible by "max_iter" is recommended, otherwise, you will miss the validation scores of the last training iteration.
- **early_stopping_rounds** (*int, default: None*) – If positive number specified, at every specified training rounds, program checks for early stopping criteria. Validation_freqs must also be set when using early stopping.
- **metrics** (*list or None, default: None*) – Specify which metrics to be used when performing evaluation during training process. If metrics have not improved at early_stopping rounds, trianing stops before convergence. If set as empty, default metrics will be used. For regression tasks, default metrics are ['root_mean_squared_error', 'mean_absolute_error']
- **use_first_metric_only** (*bool, default: False*) – Indicate whether to use the first metric in *metrics* as the only criterion for early stopping judgement.

class LocalBaselineParam(*model_name='LogisticRegression'*, *model_opts=None*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *need_run=True*)

Define the local baseline model param

Parameters

- **model_name** (*str, sklearn model used to train on baseline model*) –
- **model_opts** (*dict or none, default None*) – Param to be used as input into baseline model
- **predict_param** (*PredictParam object, default: default PredictParam object*) –
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class LogisticParam(*penalty='L2'*, *tol=0.0001*, *alpha=1.0*, *optimizer='rmsprop'*, *batch_size=-1*, *learning_rate=0.01*, *init_param=<federatedml.param.init_model_param.InitParam object>*, *max_iter=100*, *early_stop='diff'*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *decay=1*, *decay_sqrt=True*, *multi_class='ovr'*, *validation_freqs=None*, *early_stopping_rounds=None*, *stepwise_param=<federatedml.param.stepwise_param.StepwiseParam object>*, *metrics=None*, *use_first_metric_only=False*)

Parameters used for Logistic Regression both for Homo mode or Hetero mode.

Parameters

- **penalty** (*str, 'L1', 'L2' or None. default: 'L2'*) – Penalty method used in LR. Please note that, when using encrypted version in HomoLR, 'L1' is not supported.
- **tol** (*float, default: 1e-4*) – The tolerance of convergence
- **alpha** (*float, default: 1.0*) – Regularization strength coefficient.
- **optimizer** (*str, 'sgd', 'rmsprop', 'adam', 'nesterov_momentum_sgd', 'sqn' or 'adagrad', default: 'rmsprop'*) – Optimize method, if 'sqn' has been set, sqn_param will take effect. Currently, 'sqn' support hetero mode only.
- **batch_size** (*int, default: -1*) – Batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy.
- **learning_rate** (*float, default: 0.01*) – Learning rate
- **max_iter** (*int, default: 100*) – The maximum iteration for training.
- **early_stop** (*str, 'diff', 'weight_diff' or 'abs', default: 'diff'*) –

  Method used to judge converge or not.

  a) diff Use difference of loss between two iterations to judge whether converge.

  b) weight_diff: Use difference between weights of two consecutive iterations

  c) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < eps, it is converged.

  Please note that for hetero-lr multi-host situation, this parameter support "weight_diff" only.
- **decay** (*int or float, default: 1*) – Decay rate for learning rate. learning rate will follow the following decay schedule. lr = lr0/(1+decay*t) if decay_sqrt is False. If decay_sqrt is True, lr = lr0 / sqrt(1+decay*t) where t is the iter number.
- **decay_sqrt** (*Bool, default: True*) – lr = lr0/(1+decay*t) if decay_sqrt is False, otherwise, lr = lr0 / sqrt(1+decay*t)
- **encrypt_param** (*EncryptParam object, default: default EncryptParam object*) –
- **predict_param** (*PredictParam object, default: default PredictParam object*) –
- **cv_param** (*CrossValidationParam object, default: default CrossValidationParam object*) –
- **multi_class** (*str, 'ovr', default: 'ovr'*) – If it is a multi_class task, indicate what strategy to use. Currently, support 'ovr' short for one_vs_rest only.

- **validation_freqs** (*int, list, tuple, set, or None*) – validation frequency during training.
- **early_stopping_rounds** (*int, default: None*) – Will stop training if one metric doesn't improve in last early_stopping_round rounds
- **metrics** (*list or None, default: None*) – Indicate when executing evaluation during train process, which metrics will be used. If set as empty, default metrics for specific task type will be used. As for binary classification, default metrics are ['auc', 'ks']
- **use_first_metric_only** (*bool, default: False*) – Indicate whether use the first metric only for early stopping judgement.

class ObjectiveParam(*objective='cross_entropy'*, *params=None*)

Define objective parameters that used in federated ml.

Parameters

- **objective** (*None or str, accepted None,'cross_entropy','lse','lae', 'log_cosh','tweedie','fair','huber' only,*) – None in host's config, should be str in guest'config. when task_type is classification, only support cross_entropy, other 6 types support in regression task. default: None
- **params** (*None or list, should be non empty list when objective is 'tweedie','fair','huber',*) – first element of list shoulf be a float-number large than 0.0 when objective is 'fair','huber', first element of list should be a float-number in [1.0, 2.0) when objective is 'tweedie'

class OneVsRestParam(*need_one_vs_rest=False*, *has_arbiter=True*)

Define the one_vs_rest parameters.

Parameters has_arbiter (*bool. For some algorithm, may not has arbiter, for instances, secureboost of FATE,*) – for these algorithms, it should be set to false. default true

class PoissonParam(*penalty='L2'*, *tol=0.0001*, *alpha=1.0*, *optimizer='rmsprop'*, *batch_size=-1*, *learning_rate=0.01*, *init_param=<federatedml.param.init_model_param.InitParam object>*, *max_iter=20*, *early_stop='diff'*, *exposure_colname=None*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *encrypted_mode_calculator_param=<federatedml.param.encrypted_mode_calculation_param.EncryptedModeC object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *stepwise_param=<federatedml.param.stepwise_param.StepwiseParam object>*, *decay=1*, *decay_sqrt=True*, *validation_freqs=None*, *early_stopping_rounds=None*, *metrics=None*, *use_first_metric_only=False*)

Parameters used for Poisson Regression.

Parameters

- **penalty** (*str, 'L1' or 'L2'. default: 'L2'*) – Penalty method used in Poisson. Please note that, when using encrypted version in HeteroPoisson, 'L1' is not supported.
- **tol** (*float, default: 1e-4*) – The tolerance of convergence
- **alpha** (*float, default: 1.0*) – Regularization strength coefficient.
- **optimizer** (*str, 'sgd', 'rmsprop', 'adam' or 'adagrad', default: 'rmsprop'*) – Optimize method
- **batch_size** (*int, default: -1*) – Batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy.

- **learning_rate** (*float, default: 0.01*) – Learning rate
- **max_iter** (*int, default: 20*) – The maximum iteration for training.
- **init_param** (*InitParam object, default: default InitParam object*) – Init param method object.
- **early_stop** (*str, 'weight_diff', 'diff' or 'abs', default: 'diff'*) –

  Method used to judge convergence.

  a) diff Use difference of loss between two iterations to judge whether converge.

  b) weight_diff: Use difference between weights of two consecutive iterations

  c) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < eps, it is converged.
- **exposure_colname** (*str or None, default: None*) – Name of optional exposure variable in dTable.
- **predict_param** (*PredictParam object, default: default PredictParam object*) –
- **encrypt_param** (*EncryptParam object, default: default EncryptParam object*) –
- **encrypted_mode_calculator_param** (*EncryptedModeCalculatorParam object, default: default EncryptedModeCalculatorParam object*) –
- **cv_param** (*CrossValidationParam object, default: default CrossValidationParam object*) –
- **stepwise_param** (*StepwiseParam object, default: default StepwiseParam object*) –
- **decay** (*int or float, default: 1*) – Decay rate for learning rate. learning rate will follow the following decay schedule. lr = lr0/(1+decay*t) if decay_sqrt is False. If decay_sqrt is True, lr = lr0 / sqrt(1+decay*t) where t is the iter number.
- **decay_sqrt** (*Bool, default: True*) – lr = lr0/(1+decay*t) if decay_sqrt is False, otherwise, lr = lr0 / sqrt(1+decay*t)
- **validation_freqs** (*int, list, tuple, set, or None*) – validation frequency during training, required when using early stopping. The default value is None, 1 is suggested. You can set it to a number larger than 1 in order to speed up training by skipping validation rounds. When it is larger than 1, a number which is divisible by "max_iter" is recommended, otherwise, you will miss the validation scores of the last training iteration.
- **early_stopping_rounds** (*int, default: None*) – If positive number specified, at every specified training rounds, program checks for early stopping criteria. Validation_freqs must also be set when using early stopping.
- **metrics** (*list or None, default: None*) – Specify which metrics to be used when performing evaluation during training process. If metrics have not improved at early_stopping rounds, trianing stops before convergence. If set as empty, default metrics will be used. For regression tasks, default metrics are ['root_mean_squared_error', 'mean_absolute_error']
- **use_first_metric_only** (*bool, default: False*) – Indicate whether to use the first metric in *metrics* as the only criterion for early stopping judgement.

class PredictParam(*threshold=0.5*)

Define the predict method of HomoLR, HeteroLR, SecureBoosting

Parameters **threshold** (*float or int, The threshold use to separate positive and negative class. Normally, it should be (0,1)*) –

class RsaParam(*rsa_key_n=None*, *rsa_key_e=None*, *rsa_key_d=None*, *save_out_table_namespace=None*, *save_out_table_name=None*)

Define the sample method

Parameters

- **rsa_key_n** (*integer, RSA modulus, default: None*) –
- **rsa_key_e** (*integer, RSA public exponent, default: None*) –
- **rsa_key_d** (*integer, RSA private exponent, default: None*) –
- **save_out_table_namespace** (*str, namespace of dtable where stores the output data. default: None*) –
- **save_out_table_name** (*str, name of dtable where stores the output data. default: None*) –

class SampleParam(*mode='random'*, *method='downsample'*, *fractions=None*, *random_state=None*, *task_type='hetero'*, *need_run=True*)

Define the sample method

Parameters

- **mode** (*str, accepted 'random','stratified" only in this version, specify sample to use, default: 'random'*) –
- **method** (*str, accepted 'downsample','upsample' only in this version. default: 'downsample'*) –
- **fractions** (*None or float or list, if mode equals to random, it should be a float number greater than 0,*) – otherwise a list of elements of pairs like [label_i, sample_rate_i], e.g. [[0, 0.5], [1, 0.8], [2, 0.3]]. default: None
- **random_state** (*int, RandomState instance or None, default: None*) –
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class ScaleParam(*method=None*, *mode='normal'*, *scale_col_indexes=-1*, *scale_names=None*, *feat_upper=None*, *feat_lower=None*, *with_mean=True*, *with_std=True*, *need_run=True*)

Define the feature scale parameters.

Parameters

- **method** (*str, like scale in sklearn, now it support "min_max_scale" and "standard_scale", and will support other scale method soon.*) – Default None, which will do nothing for scale
- **mode** (*str, the mode support "normal" and "cap". for mode is "normal", the feat_upper and feat_lower is the normal value like "10" or "3.1" and for "cap", feat_upper and*) – feature_lower will between 0 and 1, which means the percentile of the column. Default "normal"
- **feat_upper** (*int or float, the upper limit in the column. If the scaled value is larger than feat_upper, it will be set to feat_upper. Default None.*) –
- **feat_lower** (*int or float, the lower limit in the column. If the scaled value is less than feat_lower, it will be set to feat_lower. Default None.*) –

- **scale_col_indexes** (*list,the idx of column in scale_column_idx will be scaled, while the idx of column is not in, it will not be scaled.*) –
- **scale_names** (*list of string, default: [].Specify which columns need to scaled. Each element in the list represent for a column name in header.*) –
- **with_mean** (*bool, used for "standard_scale". Default False.*) –
- **with_std** (*bool, used for "standard_scale". Default False.*) – The standard scale of column x is calculated as : z = (x - u) / s, where u is the mean of the column and s is the standard deviation of the column. if with_mean is False, u will be 0, and if with_std is False, s will be 1.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class StatisticsParam(*statistics='summary'*, *column_names=None*, *column_indexes=-1*, *need_run=True*, *abnormal_list=None*, *quantile_error=0.0001*, *bias=True*)

Define statistics params

Parameters

- **statistics** (*list, string, default "summary"*) – Specify the statistic types to be computed. "summary" represents list: [consts.SUM, consts.MEAN, consts.STANDARD_DEVIATION,

  consts.MEDIAN, consts.MIN, consts.MAX, consts.MISSING_COUNT, consts.SKEWNESS, consts.KURTOSIS]
- **column_names** (*list of string, default []*) – Specify columns to be used for statistic computation by column names in header
- **column_indexes** (*list of int, default -1*) – Specify columns to be used for statistic computation by column order in header -1 indicates to compute statistics over all columns
- **bias** (*bool, default: True*) – If False, the calculations of skewness and kurtosis are corrected for statistical bias.
- **need_run** (*bool, default True*) – Indicate whether to run this modules

class StepwiseParam(*score_name='AIC'*, *mode='hetero'*, *role='guest'*, *direction='both'*, *max_step=10*, *nvmin=2*, *nvmax=None*, *need_stepwise=False*)

Define stepwise params

Parameters

- **score_name** (*str, default: 'AIC'*) – Specify which model selection criterion to be used
- **mode** (*str, default: 'Hetero'*) – Indicate what mode is current task
- **role** (*str, default: 'Guest'*) – Indicate what role is current party
- **direction** (*str, default: 'both'*) – Indicate which direction to go for stepwise. 'forward' means forward selection; 'backward' means elimination; 'both' means possible models of both directions are examined at each step.
- **max_step** (*int, default: '10'*) – Specify total number of steps to run before forced stop.

- **nvmin** (*int, default: '2'*) – Specify the min subset size of final model, cannot be lower than 2. When nvmin > 2, the final model size may be smaller than nvmin due to max_step limit.
- **nvmax** (*int, default: None*) – Specify the max subset size of final model, 2 <= nvmin <= nvmax. The final model size may be larger than nvmax due to max_step limit.
- **need_stepwise** (*bool, default False*) – Indicate if this module needed to be run

class StochasticQuasiNewtonParam(*update_interval_L=3*, *memory_M=5*, *sample_size=5000*, *random_seed=None*)

Parameters used for stochastic quasi-newton method.

Parameters

- **update_interval_L** (*int, default: 3*) – Set how many iteration to update hess matrix
- **memory_M** (*int, default: 5*) – Stack size of curvature information, i.e. y_k and s_k in the paper.
- **sample_size** (*int, default: 5000*) – Sample size of data that used to update Hess matrix

class UnionParam(*need_run=True*, *allow_missing=False*, *keep_duplicate=False*)

Define the union method for combining multiple dTables and keep entries with the same id

Parameters

- **need_run** (*bool, default True*) – Indicate if this module needed to be run
- **allow_missing** (*bool, default False*) – Whether allow mismatch between feature length and header length in the result. Note that empty tables will always be skipped regardless of this param setting.
- **keep_duplicate** (*bool, default False*) – Whether to keep entries with duplicated keys. If set to True, a new id will be generated for duplicated entry in the format {id}_{table_name}.

CHAPTER

NINE

EXAMPLE USAGE GUIDE.

We provide here examples of FATE jobs, including FATE-Pipeline scripts, DSL conf files, and modeling quality comparison tasks

We suggest that user use example-runner tool FATE-Test.

Also, for smoother interaction with FATE-Flow, we suggest that user install Flow-Client with FATE-Client.

To quickly start model training and predictions using dsl & pipeline, please refer to

1. DSL v1 train and predict quick tutorial.
2. DSL v2 train and predict quick tutorial.
3. Pipeline train and predict quick tutorial.

Below lists included types of examples.

9.1 FATE-Pipeline

To enhance usability of FATE, starting at FATE-v1.5, FATE provides python APIs. User may develop federated learning models conveniently with FATE-Pipeline. We provide a host of Pipeline examples for each FATE module and a quick start guide for Pipeline here

Below shows how to build and fit a Hetero SecureBoost model with FATE-Pipeline in few lines.

```
import json
from pipeline.backend.config import Backend, WorkMode
from pipeline.backend.pipeline import PipeLine
from pipeline.component import Reader, DataIO, Intersection, HeteroSecureBoost,␣
↪Evaluation
from pipeline.interface import Data
from pipeline.runtime.entity import JobParameters

guest_train_data = {"name": "breast_hetero_guest", "namespace": "experiment"}
host_train_data = {"name": "breast_hetero_host", "namespace": "experiment"}

# initialize pipeline
pipeline = PipeLine().set_initiator(role="guest", party_id=9999).set_roles(guest=9999,␣
↪host=10000)

# define components
reader_0 = Reader(name="reader_0")
reader_0.get_party_instance(role="guest", party_id=9999).component_param(table=guest_
↪train_data)
```

(continues on next page)

(continued from previous page)

```
reader_0.get_party_instance(role="host", party_id=10000).component_param(table=host_
↪train_data)
dataio_0 = DataIO(name="dataio_0", with_label=True)
dataio_0.get_party_instance(role="host", party_id=10000).component_param(with_
↪label=False)
intersect_0 = Intersection(name="intersection_0")
hetero_secureboost_0 = HeteroSecureBoost(name="hetero_secureboost_0",
                                         num_trees=5,
                                         bin_num=16,
                                         task_type="classification",
                                         objective_param={"objective": "cross_entropy"},
                                         encrypt_param={"method": "paillier"},
                                         tree_param={"max_depth": 3})
evaluation_0 = Evaluation(name="evaluation_0", eval_type="binary")

# add components to pipeline, in order of task execution
pipeline.add_component(reader_0)\
    .add_component(dataio_0, data=Data(data=reader_0.output.data))\
    .add_component(intersect_0, data=Data(data=dataio_0.output.data))\
    .add_component(hetero_secureboost_0, data=Data(train_data=intersect_0.output.data))\
    .add_component(evaluation_0, data=Data(data=hetero_secureboost_0.output.data))

# compile & fit pipeline
pipeline.compile().fit(JobParameters(backend=Backend.EGGROLL, work_mode=WorkMode.
↪STANDALONE))

# query component summary
print(f"Evaluation summary:\n{json.dumps(pipeline.get_component('evaluation_0').get_
↪summary(), indent=4)}")

# Evaluation summary:
# {
#   "auc": 0.9971790603033666,
#   "ks": 0.9624094920987263
# }
```

Code for the above job can also be found here.

9.2 DSL

DSL is the first method FATE provides for constructing federated modeling jobs. For more information, please refer this guide on DSL.

Upgraded DSL(DSL v2) by FATE-v1.5 comes with the following major features:

1. Predictions DSL may now be configured through FATE-Flow cli. Please note that new DSL no longer supports automatic formation of prediction DSL; user needs to first form DSL manually with FATE-Flow cli before running prediction task.
2. New components may now be added to prediction DSL; for instance, `evaluation` module may be added to prediction task.
3. Standardize style of `role_parameter` and `algorithm_parameter`.

For DSL v2 examples, please refer dsl/v2. For examples of the older version, please refer dsl/v1. This is the "federatedml-1.x-examples" in older version. Please note that starting at version 1.6, FATE may no longer support DSL v1.

9.3 Benchmark Quality

Starting at FATE-v1.5, FATE provides modeling quality verifier for comparing modeling quality of centralized training and FATE federated modeling. As of v1.5, we have provided quality comparison scripts for the following common models:

- heterogeneous scenario: LogisticRegression(benchmark_quality/hetero_lr), SecureBoost(benchmark_quality/hetero_sbt), FastSecureBoost(benchmark_quality/hetero_fast_sbt), NN(benchmark_quality/hetero_nn).
- homogeneous scenario: LogisticRegression(benchmark_quality/homo_lr), SecureBoost(benchmark_quality/homo_sbt), NN(benchmark_quality/homo_nn.

To run the comparison, please refer to the guide on benchmark_quality.

9.4 Upload Default Data

FATE provides a collection of publicly available data at data directory, along with a utility script for uploading all data sets. User may use the provided script to upload all pre-given data, or modify the corresponding configuration file for uploading arbitrary data. Please refer scripts for details.

9.5 Toy Example

FATE provides simple toy job for quick experiment when user developing FATE modules or testing for deployment. For details, please refer toy_example.

9.6 Min-test

Min-test is used for deployment testing and quick modeling demo. Min-test includes tasks of hetero Logistic Regression and hetero SecureBoost. User only needs to configure few simple parameters to run a full modeling job with FATE. Please refer min_test_task for instructions.

English |

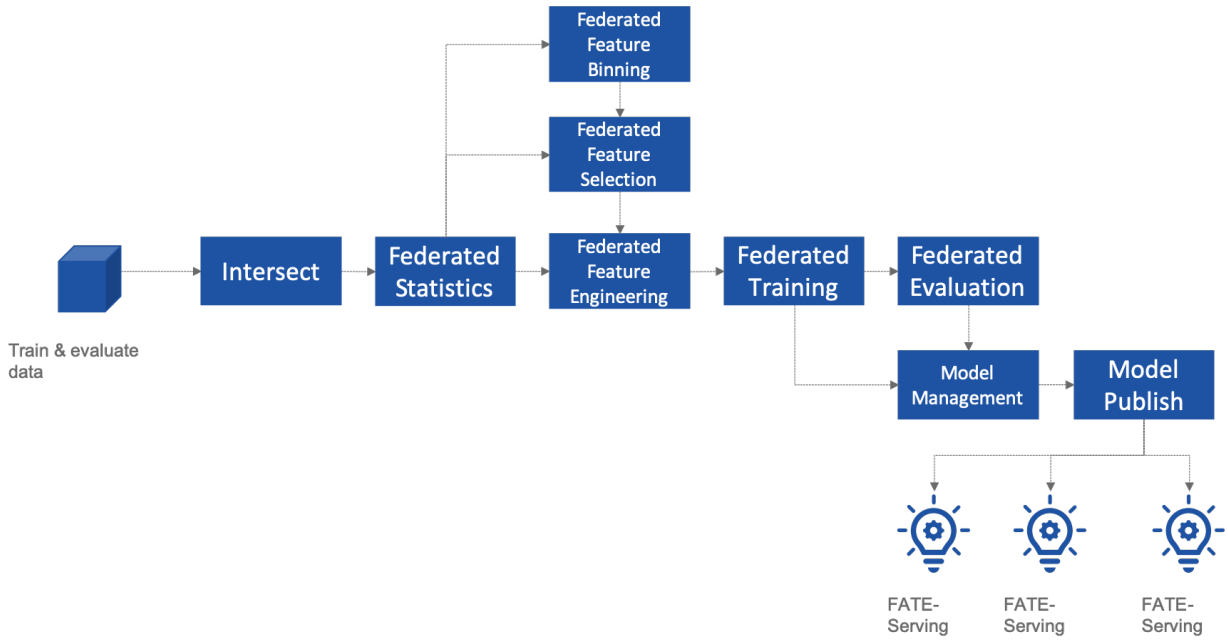
CHAPTER

TEN

FATE FLOW

10.1 Introduction

FATE-Flow is the job scheduling system of the federated learning framework FATE, which realizes the complete management of the federated learning job life cycle, including data input, training job scheduling, indicator tracking, model center and other functions



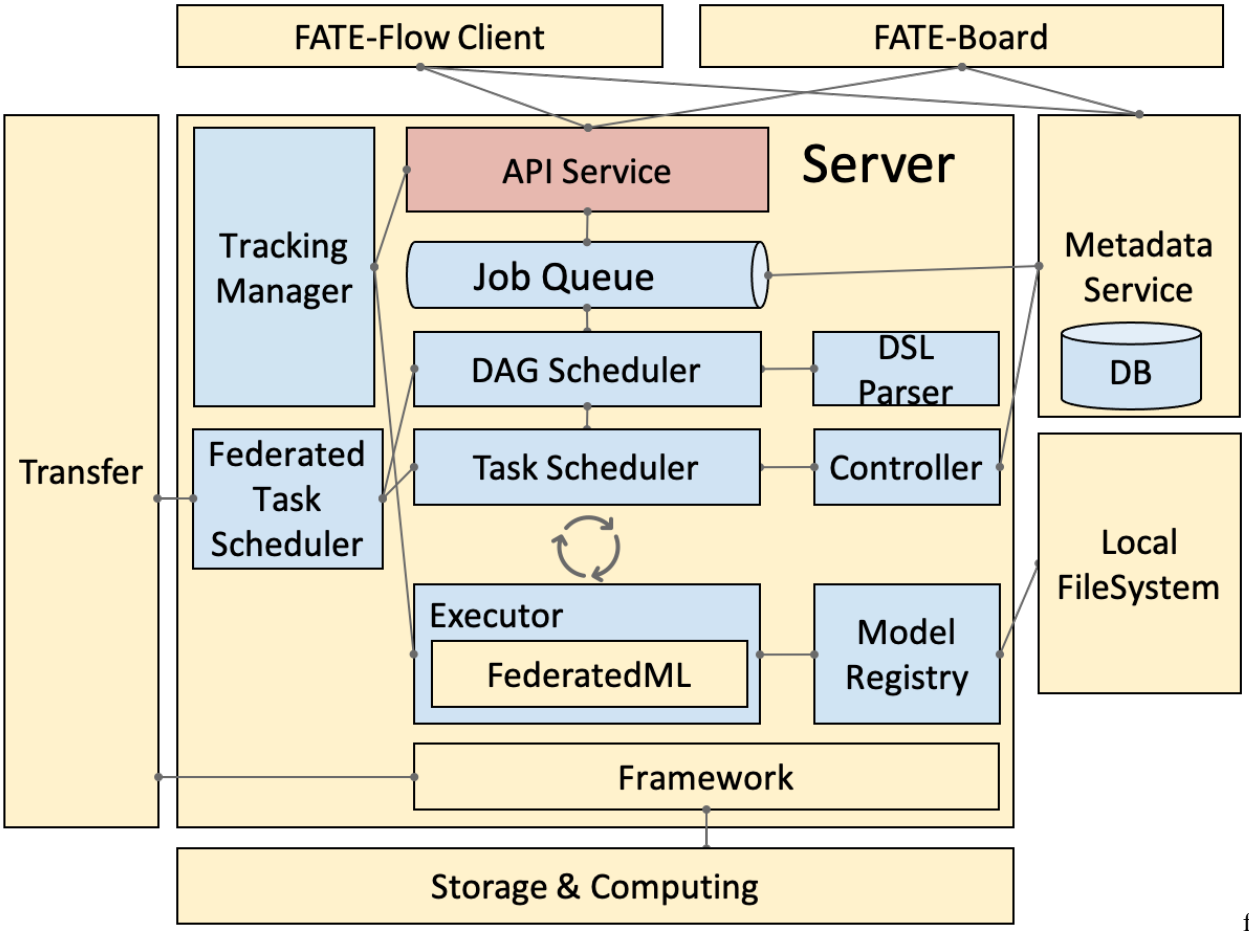
FATE-Flow Federated Learning Pipeline{.align-center}

10.1.1 FATE-Flow now supports

- DAG define Pipeline
- Describe DAG using FATE-DSL in JSON format
- Advanced scheduling framework, based on global state and optimistic lock scheduling, single-party DAG scheduling, multi-party coordinated scheduling, and support for multiple schedulers
- Flexible scheduling strategy, support start/stop/rerun, etc.
- Fine-grained resource scheduling capabilities, supporting core, memory, and working node strategies based on different computing engines

- Realtime tracker, real-time tracking data, parameters, models and indicators during operation
- Federated Learning Model Registry, model management, federated consistency, import and export, migration between clusters
- Provide CLI, HTTP API, Python SDK

10.2 Architecture



fateflow_arch{.ali

center}

10.3 Deploy

README

10.4 Usage

10.4.1 Command Line Interface v2

10.4.2 Python SDK

10.4.3 HTTP API

10.4.4 Training Examples

10.4.5 Online Inference Examples

10.5 Logs

FATE-Flow Server log

`$PROJECT_BASE/logs/fate_flow/`

Job log

`$PROJECT_BASE/logs/$job_id/`

10.6 FAQ

What is the role of FATE FLOW in the FATE?

:

> FATE Flow is a scheduling system that schedules the execution of algorithmic components based on the DSL of the job submitted by the user.

ModuleNotFoundError

: No module named "arch":

> Set PYTHONPATH to the parent directory of fate_flow.

Why does the task show success when submitting the task, but the task fails on the dashboard page?

:

- Submit success just means that the job was submitted and not executed. If the job fails, you need to check the log.
- You can view the logs through the board.

What meaning and role do the guest, host, arbiter, and local roles represent in fate?

:

- Arbiter is used to assist multiple parties to complete joint modeling. Its main role is to aggregate gradients or models. For example, in vertical lr, each party sends half of its gradient to arbiter, and then arbiter jointly optimizes, etc.
- Guest represents the data application party.
- Host is the data provider.
- Local refers to local, only valid for upload and download.

Error about"cannot find xxxx" when killing a waiting job

:

Fate_flow currently only supports kill on the job initiator, kill will report "cannot find xxx".

What is the upload data doing?

:

Upload data is uploaded to eggroll and becomes a DTable format executable by subsequent algorithms.

How to download the generated data in the middle of the algorithm?

:

You can use

: `python fate_flow_client.py -f component_output_data -j $job_id -r $role -p $party_id -cpn $component_name -o $output_path`

If the same file upload is executed twice, will fate delete the first data and upload it again?

:

It will be overwritten if the keys are the same in the same table.

What is the reason for the failure of this job without error on the board?

:

The logs in these places will not be displayed on the board:`$job_id/fate_flow_schedule.log`, `logs/error.log`, `logs/fate_flow/ERROR.log` .

What is the difference between the load and bind commands?

:

Load can be understood as a model release, and bind is the default model version.

CHAPTER

ELEVEN

FATE-FLOW CLIENT COMMAND LINE INTERFACE

11.1 Usage

```
python fate_flow_client.py -f $command
```

11.2 JOB_OPERATE

11.2.1 submit_job

- description: submit a pipeline job
- parameter:
  - -c -config: runtime conf path, Required
  - -d -dsl: dsl path, Required

```
python fate_flow_client.py -f submit_job -c examples/test_hetero_lr_job_conf.json -
↪d examples/test_hetero_lr_job_dsl.json
```

11.2.2 stop_job

- description: cancel job or stop job
- parameter:
  - -j -job_id: job id, Required

```
python fate_flow_client.py -f stop_job -j $job_id
```

11.2.3 query_job

- description: query job information by filters
- parameter:
  - -j -job_id: filter by job id, Optional
  - -r -role : filter by role, Optional
  - -p -party_id: filter by party id, Optional
  - -s -status: filter by status, Optional

```
python fate_flow_client.py -f query_job -j $job_id
```

11.2.4 clean_job

- description: clean processor,data table and metric data
- parameter:
  - -j -job_id: filter by job id, Optional
  - -r -role : filter by role, Optional
  - -p -party_id: filter by party id, Optional
  - -cpn -component_name: component name, Optional

```
python fate_flow_client.py -f clean_job -j $job_id
```

11.2.5 data_view_query

- description: query data view information by filters
- **parameter:**
  - j -job_id: filter by job id, Optional
  - r -role : filter by role, Optional
  - p -party_id: filter by party id, Optional
  - s -status: filter by status, Optional

```
python fate_flow_client.py -f data_view_query -j $job_id
```

11.3 JOB

11.3.1 job_config

- description: download the configuration of this job
- parameter:
  - -j -job_id: job id, Required

  - -r -role : role, Required
  - -p -party_id: party id, Required
  - -o -output_path: config output directory path, Required

```
python fate_flow_client.py -f job_config -j $job_id -r $role -p $party_id -o
↪$output_path
```

11.3.2 job_log

- description: download the log of this job
- parameter:
  - -j -job_id: job id, Required
  - -o -output_path: config output directory path, Required

```
python fate_flow_client.py -f job_log -j $job_id -o $output_path
```

11.4 TASK_OPERATE

11.4.1 query_task

- description: query task information by filters
- parameter:
  - -j -job_id: filter by job id, Optional
  - -cpn -component_name: filter by component name, Optional
  - -r -role : filter by role, Optional
  - -p -party_id: filter by party id, Optional
  - -s -status: filter by status, Optional

```
python fate_flow_client.py -f query_task -j $job_id
```

11.5 TRACKING

11.5.1 component_parameters

- description: query the parameters of this component
- parameter:
  - -j -job_id: job id, Required
  - -cpn -component_name: component name, Required
  - -r -role: role, Required
  - -p -party_id: party id, Required

```
python fate_flow_client.py -f component_parameters -j $job_id -r $role -p $party_id␣
↪-cpn $component_name
```

11.5.2 component_metric_all

- description: query all metric data
- parameter:
  - -j –job_id: job id, Required
  - -cpn –component_name: component name, Required
  - -r –role: role, Required
  - -p –party_id: party id, Required

```
python fate_flow_client.py -f component_metric_all -j $job_id -r $role -p $party_id␣
↪-cpn $component_name
```

11.5.3 component_metrics

- description: query the list of metrics
- parameter:
  - -j –job_id: job id, Required
  - -cpn –component_name: component name, Required
  - -r –role: role, Required
  - -p –party_id: party id, Required

```
python fate_flow_client.py -f component_metrics -j $job_id -r $role -p $party_id -
↪cpn $component_name
```

11.5.4 component_output_model

- description: query this component model
- parameter:
  - -j –job_id: job id, Required
  - -cpn –component_name: component name, Required
  - -r –role: role, Required
  - -p –party_id: party id, Required

```
python fate_flow_client.py -f component_output_model -j $job_id -r $role -p $party_
↪id -cpn $component_name
```

11.5.5 component_output_data

- description: download the output data of this component
- parameter:
  - -j –job_id: job id, Required
  - -cpn –component_name: component name, Required
  - -r –role: role, Required
  - -p –party_id: party id, Required
  - -o –output_path: config output path, Required
  - -limit –limit: Limit quantity, Optional

```
python fate_flow_client.py -f component_output_data -j $job_id -r $role -p $party_
↪id -cpn $component_name -o $output_path
```

11.5.6 component_output_data_table

- description: view table name and namespace
- parameter:
  - -j –ob_id: job id, Required
  - -cpn –component_name: component name, Required
  - -r –role: role, Required
  - -p –party_id: party id, Required

```
python fate_flow_client.py -f component_output_data_table -j $job_id -r $role -p
↪$party_id -cpn $component_name
```

11.5.7 DATA

download

- description: download table
- parameter:
  - -c –config: config path, Required

```
python fate_flow_client.py -f download -c examples/download_guest.json
```

upload

- description: upload table
- parameter:
  - -c –config: config path, Required
  - -drop –drop: Operations before file upload, Optional

```
python fate_flow_client.py -f upload -c examples/upload_guest.json
python fate_flow_client.py -f upload -c examples/upload_guest.json -drop 0(or1)
```

upload_history

- description: upload table history
- parameter:
  - -j –job_id: job id, Optional
  - -limit –limit: Limit quantity, Optional

```
python fate_flow_client.py -f upload_history -j $job_id
python fate_flow_client.py -f upload_history -limit 5
```

11.5.8 Table

table_info

- description: query table information
- parameter:
  - -n –namespace: namespace, Required
  - -t –table_name: table name, Required

```
python fate_flow_client.py -f table_info -n $namespace -t $table_name
```

table_delete

- description: delete table
- parameter:
  - -n –namespace: namespace, Optional
  - -t –table_name: table name, Optional
  - -j –job_id: job id, Optional
  - -cpn –component_name: component name, Optional
  - -r –role: role, Optional
  - -p –party_id: party id, Optional

```
python fate_flow_client.py -f table_delete -n $namespace -t $table_name
python fate_flow_client.py -f table_delete -j $job_id
```

11.5.9 Model

load

- description: load model. Need to deploy model first if *dsl_version* == *2*.
- parameter:
  - -c –config: config path, Required

```
python fate_flow_client.py -f load -c $conf_path
```

- response example:


```
{
    "data": {
        "detail": {
            "guest": {
                "9999": {
                    "retcode": 0,
                    "retmsg": "xxx"
                }
            },
            "host": {
                "10000": {
                    "retcode": 0,
                    "retmsg": "xxx"
                }
            }
        },
        "guest": {
            "9999": 0
        },
        "host": {
            "10000": 0
        }
    },
    "jobId": "xxxxxxxxxxxxxxx",
    "retcode": 0,
    "retmsg": "success"
}
```

bind

- description: bind model. Need to deploy model first if *dsl_version* == *2*.
- parameter:
  - -c –config: config path, Required

```
python fate_flow_client.py -f bind -c $conf_path
```

- response example:

```
{
    "retcode": 0,
    "retmsg": "service id is xxx"
}
```

store

- description: store model
- parameter:
  - -c –config: config path, Required

```
python fate_flow_client.py -f store -c $conf_path
```

restore

- description: restore mode
- parameter:
  - -c –config: config path, Required

```
python fate_flow_client.py -f restore -c $conf_path
```

export

- description: export model
- parameter:
  - -c –config: config path, Required

```
python fate_flow_client.py -f export -c $conf_path
```

import

- description: import model
- parameter:
  - -c –config: config path, Required

```
python fate_flow_client.py -f import -c $conf_path
```

CHAPTER

TWELVE

FATE-FLOW REST API

- HTTP Method: POST
- Content-Type: application/json

12.1 DataAccess

12.1.1 /v1/data/upload

- request structure
  - namespace: Required,String: upload data table namespace
  - table_name: Required,String: upload data table name
  - work_mode: Required,Integer: eggroll's working mode
  - file: Required, String: upload file location
  - head: Required,Integer: determine if there is a data header
  - partition: Required,Integer: set the number of partitions to save data
  - module: Optional,String: If you need to use the data of the machine where the FATE-Flow server is located, this value is not empty.
  - use_local_data: Optional,String: If you need to use the data of the machine where the FATE-Flow server is located, this value is 0.
  - drop: Optional, Integer: When the cluster deployment uses the same table to upload data, it is necessary to carry the drop parameter,0 represents overwriting upload, 1 represents deleting the previous data and re-uploading
- response structure
  - job_id: upload job id,String
  - data: return data for submitting job ,Object

12.1.2 /v1/data/download

- request structure
  - namespace: Required,String: download data table namespace
  - table_name: Required,String: download data table name
  - output_path: Required, String: download file location
  - work_mode: Required,Integer:working mode
  - delimitor: Optional,String: download data delimitor
- response structure
  - job_id: download job id,String
  - data: return data for submitting job ,Object

12.1.3 /v1/data/upload/history

- request structure
  - job_id: Optional,String:download job id
  - limit: Optional, Integer:Limit quantity
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: return data for submitting job ,Object

12.2 Job

12.2.1 /v1/job/submit

- request structure
  - job_runtime_conf: Required,Object: configuration information for the currently submitted job
  - job_dsl: Required,Object: dsl of the currently submitted job
- response structure
  - job_id: job id of the currently submitted job,String
  - data: return data for submitting job ,Object

12.2.2 /v1/job/stop

- request structure
  - job_id: Required, String: job id
- response structure
  - job_id: job id of the currently submitted job,String
  - retmsg: return code description,String

12.2.3 /v1/job/query

- request structure
  - job_id: Optional,String: job id
  - name: Optional,String: job name
  - description: Optional,String: job description
  - tag: Optional,String:Optional,String: job tag
  - role: Optional,String: job role
  - party_id: Optional,String: job party id
  - roles: Optional,String: job roles
  - initiator_party_id: Optional,String: initiator's party id
  - is_initiator: Optional,Integer: mark if it is the initiator
  - dsl: Optional,String: job dsl
  - runtime_conf : Optional,String: configuration information for the job
  - run_ip: Optional,String: job run ip
  - status: Optional,String: job status
  - current_steps: Optional,String:record component id in DSL
  - current_tasks: Optional,String: record task id
  - progress: Optional,Integer: job progress
  - create_time: Optional,Integer: job create time
  - update_time: Optional,Integer:job update time
  - start_time: Optional,Integer: job start time
  - end_time: Optional,Integer: job end time
  - elapsed: Optional,Integer: job elapsed time
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: job data, Array

12.2.4 /v1/job/update

- request structure
  - job_id: Required,String: job id
  - role: Required,String: job role
  - party_id: Required,String: job party id
  - notes: Required, String: remark Information
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String

12.2.5 /v1/job/config

- request structure
  - job_id: Optional,String: job id
  - name: Optional,String: job name
  - description: Optional,String: job description
  - tag: Optional,String:Optional,String: job tag
  - role: Optional,String: job role
  - party_id: Optional,String: job party id
  - roles: Optional,String: job roles
  - initiator_party_id: Optional,String: initiator's party id
  - is_initiator: Optional,Integer: mark if it is the initiator
  - dsl: Optional,String: job dsl
  - runtime_conf : Optional,String: configuration information for the job
  - run_ip: Optional,String: job run ip
  - status: Optional,String: job status
  - current_steps: Optional,String:record component id in DSL
  - current_tasks: Optional,String: record task id
  - progress: Optional,Integer: job progress
  - create_time: Optional,Integer: job create time
  - update_time: Optional,Integer:job update time
  - start_time: Optional,Integer: job start time
  - end_time: Optional,Integer: job end time
  - elapsed: Optional,Integer: job elapsed time
- response structure
  - retcode: return code,Integer

  - retmsg: return code description,String
  - data: config data, Object

12.2.6 /v1/job/task/query

- request structure
  - job_id: Optional,String: job id
  - name: Optional,String: job name
  - description: Optional,String: job description
  - tag: Optional,String:Optional,String: job tag
  - role: Optional,String: job role
  - party_id: Optional,String: job party id
  - roles: Optional,String: job roles
  - initiator_party_id: Optional,String: initiator's party id
  - is_initiator: Optional,Integer: mark if it is the initiator
  - dsl: Optional,String: job dsl
  - runtime_conf : Optional,String: configuration information for the job
  - run_ip: Optional,String: job run ip
  - status: Optional,String: job status
  - current_steps: Optional,String:record component id in DSL
  - current_tasks: Optional,String: record task id
  - progress: Optional,Integer: job progress
  - create_time: Optional,Integer: job create time
  - update_time: Optional,Integer:job update time
  - start_time: Optional,Integer: job start time
  - end_time: Optional,Integer: job end time
  - elapsed: Optional,Integer: job elapsed time
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: tasks data, Array

12.2.7 /v1/job/list/job

- request structure
  - limit: Optional, Integer: limitation of number of return records
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: info of jobs, Array

12.2.8 /v1/job/list/task

- request structure
  - limit: Optional, Integer: limitation of number of return records
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: info of tasks, Array

12.2.9 /v1/job/dsl/generate

- request structure
  - train_dsl: Required, String: training dsl
  - cpn_str: Required, String or Array: list of components which are chose to be used
  - filename: Optional, String: generated dsl storing path
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: generated dsl, Array

12.2.10 Tracking

12.2.11 /v1/tracking/job/data_view

- request structure
  - job_id: Required,String: job id
  - role: Required,String: role information
  - party_id: Required,Integer: party id
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String

  - data: job view data,Object

12.2.12 /v1/tracking/component/metric/all

- request structure
  - job_id: Required,String: job id
  - role: Required,String: role information
  - party_id: Required,Integer
  - component_name: Required,String: conponent name
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: all metric data,Object

12.2.13 /v1/tracking/component/metrics

- request structure
  - job_id: Required,String: job id
  - role: Required,String: role information
  - party_id: Required,Integer
  - component_name: Required,String: component name
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: metrics data,Object

12.2.14 /v1/tracking/component/metric_data

- request structure
  - job_id: Required,String: job id
  - role: Required,String: role information
  - party_id: Required,Integer: party id
  - component_name: Required,String: component name
  - meric_name: Required,String: metric name
  - metric_namespace: Required,String: metric namespace
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String

  - data: metric data, Array
  - meta: metric meta, Object

12.2.15 /v1/tracking/component/parameters

- request structure
  - job_id: Required,String: job id
  - role: Required,String: role information
  - party_id: Required,Integer: party id
  - component_name: Required,String: component name
- response structure
  - retcode:return code,Integer
  - retmsg: return code description,String
  - data: output parameters, Object

12.2.16 /v1/tracking/component/output/model

- request structure
  - job_id: Required,String: job id
  - role: Required,String: role information
  - party_id: Required,Integer: party id
  - component_name: Required,String: component name
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: output model, Object
  - meta: component model meta,Object

12.2.17 /v1/tracking/component/output/data

- request structure
  - job_id: Required,String: job id
  - role: Required,String: role information
  - party_id: Required,Integer: party id
  - component_name: Required,String: component name
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String

  - data: output data, Array
  - meta: schema header information, Object

12.2.18 Pipeline

12.2.19 /v1/pipeline/dag/dependency

- request structure
  - job_id: Required,String:job id
  - role: Required,String: role information
  - party_id: Required,Integer: party id
- response structure
  - retcode: return code,Integer
  - retmsg: return code description,String
  - data: pipeline dag dependency data,Object

12.2.20 Model

12.2.21 /v1/model/load

- request structure
  - initiator: Required,Object: job initiator information, including party_id and role
  - job_parameters: Required,Object: job parameters information, including work_mode, model_id and model_version
  - role: Required,Object: role information of the parties
  - servings: Optional,Array: fate serving address and port
- response structure
  - job_id:job id, String
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: status info, Object

12.2.22 /v1/model/bind

- request structure
  - service_id: Required,String: service id
  - initiator: Required,Object: job initiator information, including party_id and role
  - job_parameters: Required,Object: job parameters information, including work_mode, model_id and model_version
  - role: Required,Object: role information of the parties

  - servings: Optional,Array: fate serving address and port
- response structure
  - retcode: return code, Integer

12.2.23 /v1/model/transfer

- request structure
  - name: Requied,String: model version
  - namespace: Requied,String: model id
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: model data, Object

12.2.24 /v1/model/import

- request structure
  - model_version: Required, Integer: model version
  - model_id: Required, String: model id
  - role: Required, String: role
  - party_id: Required, String: party id
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.25 /v1/model/export

- request structure
  - model_version: Required, Integer: model version
  - model_id: Required, String: model id
  - role: Required, String: role
  - party_id: Required, String: party id
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.26 /v1/model/store

- request structure
  - model_version: Required, Integer: model version
  - model_id: Required, String: model id
  - role: Required, String: role
  - party_id: Required, String: party id
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.27 /v1/model/restore

- request structure
  - model_version: Required, Integer: model version
  - model_id: Required, String: model id
  - role: Required, String: role
  - party_id: Required, String: party id
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.28 /v1/model/model_tag/retrieve

- request structure
  - job_id: Required, Integer: a valid job id or model version
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: information of tags related to the specified model

12.2.29 /v1/model/model_tag/create

- request structure
  - job_id: Required, Integer: a valid job id or model version
  - tag_name: Required, String: a valid name of tag
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.30 /v1/model/model_tag/remove

- request structure
  - job_id: Required, Integer: a valid job id or model version
  - tag_name: Required, String: a valid name of tag
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.31 /v1/model/tag/retrieve

- request structure
  - tag_name: Required, String: a valid tag name
  - with_model: Optional, Boolean: choose to show tag info or tag info related to models
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: tag info, Object

12.2.32 /v1/model/tag/create

- request structure
  - tag_name: Required, String: name of tag
  - tag_desc: Optional, String: description of tag
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.33 /v1/model/tag/destroy

- request structure
  - tag_name: Required, String: a valid tag name
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.34 /v1/model/tag/update

- request structure
  - tag_name: Required, String: a valid tag name
  - new_tag_name: Optional, String: a new name to replace previous name
  - new_tag_desc: Optional, String: a new decription to replace previous description
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String

12.2.35 /v1/model/tag/list

- request structure
  - limit: Required, Integer: limitation of number of return records
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: tag info, Object

12.2.36 /v1/model/migrate

- request structure
  - migrate_initiator: Required, Object: indicates which party is the new initiator after migrating
  - unify_model_version: Optional, String: a unitive model version for migrate model
  - role: Required, String: information of roles which participated in model training, including role name and array of party ids
  - migrate_role: Required, Object: information of roles model would be migrated to, including role name and array of party ids
  - model_id: Required, String: original model id
  - model_version: Required, Integer: original model version
  - execute_party: Required, Object: parties that is going to execute model migration task
  - job_parameters: Required, Object: job parameters information, including work_mode, model_id and model_version
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: status info, Object

12.2.37 /v1/model/query

- request structure
  - model_version: Required, Integer: model version
  - model_id: Optional, String: model id
  - role: Optional, String: role
  - party_id: Optional, String: party id
  - query_filters: Optional, Array: features filters
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: model info, Object

12.2.38 /v1/model/deploy

- request structure
  - model_version: Required, Integer: model version
  - model_id: Required, String: model id
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: status info, Object

12.2.39 /v1/model/get/predict/dsl

- request structure
  - model_version: Required, Integer: model version
  - model_id: Optional, String: model id
  - role: Optional, String: role
  - party_id: Optional, String: party id
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: predict dsl of specified model, Object

12.2.40 /v1/model/get/predict/conf

- request structure
  - model_version: Required, Integer: model version
  - model_id: Required, String: model id
  - filename: Optional, String: file storing path
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: predict config of specified model, Object

12.3 Table

12.3.1 /v1/table/table_info

- request structure
  - create: Optional, Boolean: whether to create
  - namespace: Optional,String: download data table namespace, need to be used with table_name
  - table_name: Optional,String: download data table name, need to be used with namespace
  - local: Optional,Object: local configuration
  - role: Optional,Object: role information
  - data_type: Optional,String: download file data type
  - gen_table_info: Optional,Boolean: tag table information
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: table information

12.3.2 /v1/table/delete

- request structure
  - namespace: Optional,String: download data table namespace, need to be used with table_name
  - table_name: Optional,String: download data table name, need to be used with namespace
- response structure
  - retcode: return code, Integer
  - retmsg: return code description, String
  - data: table information

CHAPTER

THIRTEEN

FATE FLOW CLIENT SDK GUIDE

[]

13.1 Usage

```
from flow_sdk.client import FlowClient
# use real ip address to initialize SDK
client = FlowClient('127.0.0.1', 9000, 'v1')
```

13.2 Job Operations

13.2.1 Usage

```
client.job.submit(conf_path, dsl_path)
```

13.2.2 Functions

submit(conf_path, dsl_path)

- *Description*Submit a pipeline job.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	Yes	Runtime configuration file path
2	dsl_path	string	Yes	DSL file path

stop(job_id)

- *Description*Cancel or stop a specified job.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id

query(job_id=None, role=None, party_id=None, status=None)

- *Description*Query job information by filters.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	No	A valid job id
2	role	string	No	Role
3	party_id	integer	No	Party id
4	status	string	No	Job Status

config(job_id, role, party_id, output_path)

- *Description*Download the configuration of a specified job.
- *Arguments*

No	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	output_path	string	Yes	Specified Output Path

log(job_id, output_path)

- *Description*Download log files of a specified job.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	output_path	string	Yes	Specified Output Path

`list(limit=10)`

- *Description*List jobs.
- *Arguments*

No.	Argument	Type	Required	Description
1	limit	integer	No	Limit the number of results, default is 10

`view(job_id=None, role=None, party_id=None, status=None)`

- *Description*List jobs.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	No	A valid job id
2	role	string	No	Role
3	party_id	integer	No	Party id
4	component_name	string	No	Component Name

`generate_dsl(train_dsl_path, cpn_file_path=None, cpn_list = None)`

- *Description*A predict dsl generator.
- *Arguments*

No.	Argument	Type	Required	Description
1	train_dsl_path	string(path)	Yes	User specifies the train dsl file path.
2	version	string	No	Specified version of dsl parser. Default 1.
3	cpn_file_path	string(path)	No	User specifies a file path which records the component list.
4	cpn_list	list	No	User inputs a list of component names.

13.3 Component Operations

13.3.1 Usage

```
client.component.parameters(job_id, role, party_id, component_name)
```

13.3.2 Functions

`parameters(job_id, role, party_id, component_name)`

- *Description*Query the parameters of a specified component.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	component_name	string	Yes	Component Name

metric_all(job_id, role, party_id, component_name)

- *Description*Query all metric data.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	component_name	string	Yes	Component Name

metrics(job_id, role, party_id, component_name)

- *Description*Query all metric data.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	component_name	string	Yes	Component Name

metric_delete(date=None, job_id=None)

- *Description*Delete specified metric.
- *Arguments*

No.	Argument	Type	Required	Description
1	date	integer	Yes	An 8-Digit Valid Date, Format Like 'YYYYMMDD'
2	job_id	integer	Yes	A valid job id

`Notice`: If you input two optional arguments in the mean time, the 'date' argument will be detected in priority while the 'job_id' argument would be ignored.

output_model(job_id, role, party_id, component_name)

- *Description*Query a specified component model.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	component_name	string	Yes	Component Name

output_data(job_id, role, party_id, component_name, output_path, limit=10)

- *Description*Download the output data of a specified component.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	component_name	string	Yes	Component Name
5	output_path	string	Yes	Specified Output directory path
6	limit	integer	No	Limit the number of results, default is 10

output_data_table(job_id, role, party_id, component_name)

- *Description*View table name and namespace.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	component_name	string	Yes	Component Name

list(job_id)

- *Description* List components of a specified job.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id

get_summary(job_id, role, party_id, component_name)

- *Description*Get summary of specified component.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	role	string	Yes	Role
3	party_id	integer	Yes	Party id
4	component_name	string	Yes	Component Name

13.4 Data Operations

13.4.1 Usage

```
client.data.download(conf_path)
```

13.4.2 Functions

download(conf_path)

- *Description*Download Data Table.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	Yes	Configuration file path

upload(conf_path, verbose=0, drop=0)

- *Description*Upload Data Table.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	Yes	Configuration file path
2	verbose	integer	No	Verbose mode, 0 (default) means 'disable', 1 means 'enable'
3	drop	integer	No	If 'drop' is set to be 0 (defualt), when data had been uploaded before, current upload task would be rejected. If 'drop' is set to be 1, data of old version would be replaced by the latest version.

`upload_history(limit=10, job_id=None)`

- *Description*Query Upload Table History.
- *Arguments*

No.	Argument	Type	Required	Description
1	limit	integer	No	Limit the number of results, default is 10
2	job_id	integer	No	A valid job id

13.5 Task Operations

13.5.1 Usage

```
client.task.list(limit=10)
```

13.5.2 Functions

`list(limit=10)`

- *Description* List tasks.
- *Arguments*

No.	Argument	Type	Required	Description
1	limit	integer	No	Limit the number of results, default is 10

`query(job_id=None, role=None, party_id=None, component_name=None, status=None)`

- *Description* Query task information by filters.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	No	A valid job id.
2	role	string	No	Role
3	party_id	integer	No	Party ID
4	component_name	string	No	Component Name
5	status	string	No	Job Status

13.6 Model Operations

13.6.1 Usage

```
client.model.load(conf_path)
```

13.6.2 Functions

load(conf_path=None, job_id=None)

- *Description* Load model. Need to deploy model first if *dsl_version* == *2*.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	No	Configuration file path
2	job_id	string	No	A valid job id

bind(conf_path, job_id=None)

- *Description* Bind model. Need to deploy model first if *dsl_version* == *2*.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	Yes	Configuration file path
2	job_id	string	No	A valid job id

export_model(conf_path, to_database=False)

- *Description* Export model.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	Yes	Configuration file path
2	to_database	bool	No	If specified and there is a valid database environment, fate flow will export model to database which you specified in configuration file.

import_model(conf_path, from_database=False)

- *Description* Import model.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	Yes	Configuration file path
2	from_database	bool	No	If specified and there is a valid database environment, fate flow will import model from database which you specified in configuration file.

migrate(conf_path, to_database=False)

- *Description* Migrate model.
- *Arguments*

No.	Argument	Type	Required	Description
1	conf_path	string	Yes	Configuration file path
2	to_database	bool	No	If specified and there is a valid database environment, fate flow will export model to database which you specified in configuration file.

tag_list(job_id)

- *Description* List tags of model.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id

tag_model(job_id, tag_name, remove=False)

- *Description* Tag model.
- *Arguments*

No.	Argument	Type	Required	Description
1	job_id	integer	Yes	A valid job id
2	tag_name	string	Yes	The name of tag
3	remove	bool	No	If specified, the name of specified model will be removed from the model name list of specified tag.

deploy(model_id, model_version=None, cpn_list=None, predict_dsl=None)

- *Description* Deploy model.
- *Arguments*

No.	Argument	Type	Required	Description
1	model_id	string	Yes	Parent model id
2	model_version	string	Yes	Parent model version
3	cpn_list	list	No	Component list
4	predict_dsl	dict	No	Predict DSL

get_predict_dsl(model_id, model_version)

- *Description* Get predict dsl of model.
- *Arguments*

No.	Argument	Type	Required	Description
1	model_id	string	Yes	Parent model id
2	model_version	string	Yes	Parent model version

get_predict_conf(model_id, model_version)

- *Description* Get predict conf of model.
- *Arguments*

No.	Argument	Type	Required	Description
1	model_id	string	Yes	Parent model id
2	model_version	string	Yes	Parent model version

get_model_info(model_id=None, model_version=None, role=None, party_id=None, query_filters=None, **kwargs)

- *Description* Get information of model.
- *Arguments*

No.	Argument	Type	Required	Description
1	model_id	string	No	model id
2	model_version	string	Yes	model version
3	role	string	No	role name
4	party_id	string	No	party id
5	query_filters	list	No	query filters

13.7 Tag Operations

13.7.1 Usage

```
client.tag.create(tag_name, desc)
```

13.7.2 Functions

create(tag_name, tag_desc=None)

- *Description* Create Tag.
- *Arguments*

No.	Argument	Type	Required	Description
1	tag_name	string	Yes	The name of tag
2	tag_desc	string	No	The description of tag

update(tag_name, new_tag_name=None, new_tag_desc=None)

- *Description* Update information of tag.
- *Arguments*

No.	Argument	Type	Required	Description
1	tag_name	string	Yes	The name of tag
2	new_tag_name	string	No	New name of tag
3	new_tag_desc	string	No	New description of tag

list(limit=10)

- *Description* List recorded tags.
- *Arguments*

No.	Argument	Type	Required	Description
1	limit	integer	No	Number of records to return. (default: 10)

query(tag_name, with_model=False)

- *Description* Retrieve tag.
- *Arguments*

No.	Argument	Type	Required	Description
1	tag_name	string	Yes	The name of tag
2	with_model	bool	No	If specified, the information of models which have the tag custom queried would be displayed

delete(tag_name)

- *Description* Delete tag.
- *Arguments*

No.	Argument	Type	Required	Description
1	tag_name	string	Yes	The name of tag

13.8 Table Operations

13.8.1 Usage

```
client.table.info(namespace, table_name)
```

13.8.2 Functions

info(namespace, table_name)

- *Description* Query table information.
- *Arguments*

No.	Argument	Type	Required	Description
1	namespace	string	Yes	Namespace
2	table_name	string	Yes	Table Name

delete(namespace=None, table_name=None, job_id=None, role=None, party_id=None, component_name=None)

- *Description*Delete table.
- *Arguments*

No.	Argument	Type	Required	Description
1	namespace	string	No	Namespace
2	table_name	string	No	Table Name
3	job_id	integer	No	A valid job id
4	role	string	No	Role
5	party_id	integer	No	Party id
6	component_name	string	No	Component Name

13.9 Queue Operations

13.9.1 Usage

```
client.queue.clean()
```

13.9.2 Functions

clean()

- *Description*Cancel all jobs in queue.
- *Arguments*None

CHAPTER

FOURTEEN

FATE-FLOW CLIENT COMMAND LINE INTERFACE V2 GUIDE

[]

14.1 Usage

Before using fate flow client command line interface (CLI), please make sure that you have activated the virtual environment of FATE. For more details about how to activate virtual environment, please read the documentation of deployment.

In this version of client CLI, commands are separated into several classes, including *job*, *data*, *model*, *component* and etc. And all of these classes have a common parent (CLI entry) named *'flow'*, which means you can type *'flow'* in your terminal window to find out all of these classes and also their sub-commands.

```
[IN]
flow

[OUT]
Usage: flow [OPTIONS] COMMAND [ARGS]...

  Fate Flow Client

Options:
  -h, --help  Show this message and exit.

Commands:
  component   Component Operations
  data        Data Operations
  job         Job Operations
  model       Model Operations
  queue       Queue Operations
  table       Table Operations
  task        Task Operations
```

For more details, please check this documentation or try `flow --help` for help.

14.2 Init

14.2.1 init

- *Description*: Flow CLI Init Command. Custom can choose to provide an absolute path of server conf file, or provide ip address and http port of a valid fate flow server. Notice that, if custom provides both, the server conf would be loaded in priority. In this case, ip address and http port would be ignored.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--server-conf-path	No	Server configuration file absolute path
2	ip		--ip	No	Fate flow server ip address
3	port		--port	No	Fate flow server port
4	reset		--reset	No	If specified, initialization settings of flow CLI would be reset to none.

- *Examples*:

```
flow init -c /data/projects/fate/python/conf/service_conf.yaml
flow init --ip 127.0.0.1 --port 9380
```

14.3 Job

14.3.1 submit

- *Description*: Submit a pipeline job.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	Yes	Runtime configuration file path
2	dsl_path	-d	--dsl-path	Yes	Domain-specific language(DSL) file path. If the type of job is 'predict', you can leave this feature blank, or you can provide a valid dsl file to replace the one that aotumatically generated by fate.

- *Examples*:

```
flow job submit -c fate_flow/examples/test_hetero_lr_job_conf.json -d fate_flow/examples/
↪test_hetero_lr_job_dsl.json
```

14.3.2 stop

- *Description*: Cancel or stop a specified job.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	A valid job id.

- *Examples*:

```
flow job stop -j $JOB_ID
```

14.3.3 query

- *Description*: Query job information by filters.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	No	A valid job id.
2	role	-r	--role	No	Role
3	party_id	-p	--party_id	No	Party ID
4	status	-s	--status	No	Job Status

- *Examples*:

```
flow job query -r guest -p 9999 -s complete
flow job query -j $JOB_ID
```

14.3.4 view

- *Description*: Query data view information by filters.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	No	A valid job id.
2	role	-r	--role	No	Role
3	party_id	-p	--party_id	No	Party ID
4	status	-s	--status	No	Job Status

- *Examples*:

```
flow job view -r guest -p 9999
flow job view -j $JOB_ID -s complete
```

14.3.5 config

- *Description*: Download the configuration of a specified job.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	A valid job id.
2	role	-r	--role	Yes	Role
3	party_id	-p	--party_id	Yes	Party ID
4	output_path	-o	--output-path	Yes	Output Path

- *Examples*

```
flow job config -j $JOB_ID -r host -p 10000 --output-path ./examples/
```

14.3.6 log

- *Description*: Download log files of a specified job.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	A valid job id.
2	output_path	-o	--output-path	Yes	Output Path

- *Examples*:

```
flow job log -j JOB_ID --output-path ./examples/
```

14.3.7 list

- *Description*: List jobs.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	limit	-l	--limit	No	Number of records to return. (default: 10)

- *Examples*:

```
flow job list
flow job list -l 30
```

14.3.8 dsl

- *Description*: A predict dsl generator.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	cpn_list		`--cpn-list`	No	User inputs a string to specify component list.
2	cpn_path		`--cpn-path`	No	User specifies a file path which records the component list.
3	train_dsl_path		`--train-dsl-path`	Yes	User specifies the train dsl file path.
4	output_path	`-o`	`--output-path`	No	User specifies output directory path.

- *Examples*:

```
flow job dsl --cpn-path fate_flow/examples/component_list.txt --train-dsl-path fate_flow/
↪examples/test_hetero_lr_job_dsl.json

flow job dsl --cpn-path fate_flow/examples/component_list.txt --train-dsl-path fate_flow/
↪examples/test_hetero_lr_job_dsl.json -o fate_flow/examples/

flow job dsl --cpn-list "dataio_0, hetero_feature_binning_0, hetero_feature_selection_0,␣
↪evaluation_0" --train-dsl-path fate_flow/examples/test_hetero_lr_job_dsl.json -o fate_
↪flow/examples/

flow job dsl --cpn-list [dataio_0,hetero_feature_binning_0,hetero_feature_selection_0,
↪evaluation_0] --train-dsl-path fate_flow/examples/test_hetero_lr_job_dsl.json -o fate_
↪flow/examples/
```

14.4 Component (TRACKING)

14.4.1 parameters

- *Description*: Query the arguments of a specified component.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	`-j`	`--job_id`	Yes	A valid job id.
2	role	`-r`	`--role`	Yes	Role
3	party_id	`-p`	`--party_id`	Yes	Party ID
4	component_name	`-cpn`	`--component_name`	Yes	Component Name

- *Examples*:

```
flow component parameters -j $JOB_ID -r host -p 10000 -cpn hetero_feature_binning_0
```

14.4.2 metric-all

- *Description*: Query all metric data.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	A valid job id.
2	role	-r	--role	Yes	Role
3	party_id	-p	--party_id	Yes	Party ID
4	component_name	-cpn	--component_name	Yes	Component Name

- *Examples*:

```
flow component metric-all -j $JOB_ID -r host -p 10000 -cpn hetero_feature_binning_0
```

14.4.3 metrics

- *Description*: Query the list of metrics.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	A valid job id.
2	role	-r	--role	Yes	Role
3	party_id	-p	--party_id	Yes	Party ID
4	component_name	-cpn	--component_name	Yes	Component Name

- *Examples*:

```
flow component metrics -j $JOB_ID -r host -p 10000 -cpn hetero_feature_binning_0
```

14.4.4 metric-delete

- *Description*: Delete specified metric.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	date	-d	--date	No	An 8-Digit Valid Date, Format Like 'YYYYMMDD'
2	job_id	-j	--job_id	No	Job ID

- *Examples*:

```
# NOTICE: If you input both two optional arguments, the 'date' argument will be detected␣
↪in priority while the 'job_id' argument would be ignored.
flow component metric-delete -d 20200101
flow component metric-delete -j $JOB_ID
```

14.4.5 output-model

- *Description*: Query a specified component model.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	Job ID
2	role	-r	--role	Yes	Role
3	party_id	-p	--party_id	Yes	Party ID
4	component_name	-cpn	--component_name	Yes	Component Name

- *Examples*:

```
flow component output-model -j $JOB_ID -r host -p 10000 -cpn hetero_feature_binning_
↪0
```

14.4.6 output-data

- *Description*: Download the output data of a specified component.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	Job ID
2	role	-r	--role	Yes	Role
3	party_id	-p	--party_id	Yes	Party ID
4	component_name	-cpn	--component_name	Yes	Component Name
5	output_path	-o	--output-path	Yes	User specifies output directory path
6	limit	-l	--limit	No	Number of records to return, default -1 means return all data

- *Examples*:

```
flow component output-data -j $JOB_ID -r host -p 10000 -cpn hetero_feature_binning_
↪0 --output-path ./examples/
```

14.4.7 output-data-table

- *Description*: View table name and namespace.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	Job ID
2	role	-r	--role	Yes	Role
3	party_id	-p	--party_id	Yes	Party ID
4	component_name	-cpn	--component_name	Yes	Component Name

- *Examples*:

```
flow component output-data-table -j $JOB_ID -r host -p 10000 -cpn hetero_feature_
↪binning_0
```

14.4.8 list

- *Description*: List components of a specified job.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	Job ID

- *Examples*:

```
flow component list -j $JOB_ID
```

14.4.9 get-summary

- *Description*: Download summary of a specified component and save it as a json file.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	Job ID
2	role	-r	--role	Yes	Role
3	party_id	-p	--party_id	Yes	Party ID
4	component_name	-cpn	--component_name	Yes	Component Name
5	output_path	-o	--output-path	No	User specifies output directory path

- *Examples*:

```
flow component get-summary -j $JOB_ID -r host -p 10000 -cpn hetero_feature_binning_0

flow component get-summary -j $JOB_ID -r host -p 10000 -cpn hetero_feature_binning_0 -o .
↪/examples/
```

14.5 Model

14.5.1 load

- *Description*: Load model. Need to deploy model first if *dsl_version* == *2*.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	No	Runtime configuration file path
2	job_id	-j	--job_id	No	Job ID

- *Examples*:

```
flow model load -c fate_flow/examples/publish_load_model.json
flow model load -j $JOB_ID
```

14.5.2 bind

- *Description*: Bind model. Need to deploy model first if *dsl_version* == *2*.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	Yes	Runtime configuration file path
2	job_id	-j	--job_id	No	Job ID

- *Examples*:

```
flow model bind -c fate_flow/examples/bind_model_service.json
flow model bind -c fate_flow/examples/bind_model_service.json -j $JOB_ID
```

14.5.3 import

- *Description*: Import model
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	Yes	Runtime configuration file path
2	from-database		–from-database	No	If specified and there is a valid database environment, fate flow will import model from database which you specified in configuration file.

- *Examples*:

```
flow model import -c fate_flow/examples/import_model.json
flow model import -c fate_flow/examples/restore_model.json --from-database
```

14.5.4 export

- *Description*: Export model
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	Yes	Runtime configuration file path
2	to-database		--to-database	No	If specified and there is a valid database environment, fate flow will export model to database which you specified in configuration file.

- *Examples*:

```
flow model export -c fate_flow/examples/export_model.json
flow model export -c fate_flow/examplse/store_model.json --to-database
```

14.5.5 migrate

- *Description*: Migrate model
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	Yes	Runtime configuration file path

- *Examples*:

```
flow model migrate -c fate_flow/examples/migrate_model.json
```

14.5.6 tag-list

- *Description*: List tags of model.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	Job ID

- *Examples*:

```
flow model tag-list -j $JOB_ID
```

14.5.7 tag-model

- *Description*: Tag model.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	Yes	Job ID
2	tag_name	-t	--tag-name	Yes	The name of tag
3	remove		--remove	No	If specified, the name of specified model will be removed from the model name list of specified tag.

- *Examples*:

```
flow model tag-model -j $JOB_ID -t $TAG_NAME
flow model tag-model -j $JOB_ID -t $TAG_NAME --remove
```

14.5.8 deploy

- *Description*: Deploy model.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	model_id		`--model-id`	Yes	Parent model id.
2	model_version		`--model-version`	Yes	Parent model version.
3	cpn_list		`--cpn-list`	No	User inputs a string to specify component list.
4	cpn_path		`--cpn-path`	No	User specifies a file path which records the component list.
5	dsl_path		`--train-dsl-path`	No	User specified predict dsl file.

- *Examples*:

```
flow model deploy --model_id $MODEL_ID --model_version $MODEL_VERSION
```

14.5.9 get-predict-dsl

- *Description*: Get predict dsl of model.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	model_id		`--model-id`	Yes	Model id
2	model_version		`--model-version`	Yes	Model version
3	output_path	`-o`	`--output-path`	Yes	Output directory path

- *Examples*:

```
flow model get-predict-dsl --model_id $MODEL_ID --model_version $MODEL_VERSION -o ./
↪examples/
```

14.5.10 get-predict-conf

- *Description*: Get predict conf template of model.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	model_id		`--model-id`	Yes	Model id
2	model_version		`--model-version`	Yes	Model version
3	output_path	`-o`	`--output-path`	Yes	Output directory path

- *Examples*:

```
flow model get-predict-conf --model_id $MODEL_ID --model_version $MODEL_VERSION -o ./
↪examples/
```

14.5.11 get-model-info

- *Description*: Get information of model.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	model_id		`--model-id`	No	Model id
2	model_version		`--model-version`	Yes	Model version
3	role	`-r`	`--role`	No	Role
2	party_id	`-p`	`--party-id`	No	Party ID
3	detail		`--detail`	No	Show details

- *Examples*:

```
flow model model-info --model_id $MODEL_ID --model_version $MODEL_VERSION
flow model model-info --model_id $MODEL_ID --model_version $MODEL_VERSION --detail
```

14.6 Tag

14.6.1 create

- *Description*: Create tag.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	tag_name	`-t`	`--tag-name`	Yes	The name of tag
2	tag_description	`-d`	`--tag-desc`	No	The description of tag

- *Examples*:

```
flow tag create -t tag1 -d "This is the description of tag1."
flow tag create -t tag2
```

14.6.2 update

- *Description*: Update information of tag.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	tag_name	`-t`	`--tag-name`	Yes	The name of tag
2	new_tag_name		`--new-tag-name`	No	New name of tag
3	new_tag_description		`--new-tag-desc`	No	New description of tag

- *Examples*:

```
flow tag update -t tag1 --new-tag-name tag2
flow tag update -t tag1 --new-tag-desc "This is the new description."
```

14.6.3 list

- *Description*: List recorded tags.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	limit	`-l`	`--limit`	No	Number of records to return. (default: 10)

- *Examples*:

```
flow tag list
flow tag list -l 3
```

14.6.4 query

- *Description*: Retrieve tag.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	tag_name	`-t`	`--tag-name`	Yes	The name of tag
2	with_model		`--with-model`	No	If specified, the information of models which have the tag custom queried would be displayed

- *Examples*:

```
flow tag query -t $TAG_NAME
flow tag query -t $TAG_NAME --with-model
```

14.6.5 delete

- *Description*: Delete tag.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	tag_name	`-t`	`--tag-name`	Yes	The name of tag

- *Examples*:

```
flow tag delete -t tag1
```

14.7 Data

14.7.1 download

- *Description*: Download Data Table.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	Yes	Configuration file path

- *Examples*:

```
flow data download -c fate_flow/examples/download_host.json
```

14.7.2 upload

- *Description*: Upload Data Table.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	conf_path	-c	--conf-path	Yes	Configuration file path
2	verbose		--verbose	No	If specified, verbose mode will be turn on. Users can have feedback on upload task in progress. (Default: False)
3	drop		--drop	No	If specified, data of old version would be replaced by the current version. Otherwise, current upload task would be rejected. (Default: False)

- *Examples*:

```
flow data upload -c fate_flow/examples/upload_guest.json
flow data upload -c fate_flow/examples/upload_host.json --verbose --drop
```

14.7.3 upload-history

- *Description*: Query Upload Table History.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	limit	-l	--limit	No	Number of records to return. (default: 10)
2	job_id	-j	--job_id	No	Job ID

- *Examples*:

```
flow data upload-history -l 20
flow data upload-history --job-id $JOB_ID
```

14.8 Task

14.8.1 query

- *Description*: Query task information by filters.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	job_id	-j	--job_id	No	Job ID
2	role	-r	--role	No	Role
3	party_id	-p	--party_id	No	Party ID
4	component_name	-cpn	--component_name	No	Component Name
5	status	-s	--status	No	Job Status

- *Examples*:

```
flow task query -j $JOB_ID -p 9999 -r guest
flow task query -cpn hetero_feature_binning_0 -s complete
```

14.8.2 list

- *Description*: List tasks.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	limit	-l	--limit	No	Number of records to return. (default: 10)

- *Examples*:

```
flow task list
flow task list -l 25
```

14.9 Table

14.9.1 info

- *Description*: Query Table Information.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	namespace	-n	--namespace	Yes	Namespace
2	table_name	-t	--table-name	Yes	Table Name

- *Examples*:

```
flow table info -n $NAMESPACE -t $TABLE_NAME
```

14.9.2 delete

- *Description*: Delete A Specified Table.
- *Arguments*:

No.	Argument	Flag_1	Flag_2	Required	Description
1	namespace	-n	--namespace	No	Namespace
2	table_name	-t	--table_name	No	Table name
3	job_id	-j	--job_id	No	A valid job id
4	role	-r	--role	No	Role
5	party_id	-p	--party_id	No	Party ID
6	component_name	-cpn	--component_name	No	Component Name

- *Examples*:

```
flow table delete -n $NAMESPACE -t $TABLE_NAME
flow table delete -j $JOB_ID -r guest -p 9999
```

14.10 Queue

14.10.1 clean

- *Description*: Cancel all jobs in queue.
- *Arguments*: None.
- *Examples*:

```
flow queue clean
```

CHAPTER

FIFTEEN

FATE PIPELINE

Pipeline is a high-level python API that allows user to design, start, and query FATE jobs in a sequential manner. FATE Pipeline is designed to be user-friendly and consistent in behavior with FATE command line tools. User can customize job workflow by adding components to pipeline and then initiate a job with one call. In addition, Pipeline provides functionality to run prediction and query information after fitting a pipeline. Run the mini demo to have a taste of how FATE Pipeline works. Default values of party ids and work mode may need to be modified depending on the deployment setting.

```
python pipeline-mini-demo.py config.yaml
```

For more pipeline demo, please refer to examples.

15.1 A FATE Job is A Directed Acyclic Graph

A FATE job is a dag that consists of algorithm component nodes. FATE pipeline provides easy-to-use tools to configure order and setting of the tasks.

FATE is written in a modular style. Modules are designed to have input and output data and model. Therefore two modules are connected when output of one module is set to be the input of another module. By tracing how one data set is processed through FATE modules, we can see that a FATE job is in fact formed by a sequence of sub-tasks. For example, in the mini demo above, guest's data is first read in by `Reader`, then loaded into `DataIO`. Overlapping ids between guest and host are then found by running data through `Intersection`. Finally, `HeteroLR` model is fit on the data. Each listed modules run a small task with the data, and together they constitute a model training job.

Beyond the given mini demo, a job may include multiple data sets and models. For more pipeline examples, please refer to examples.

15.2 Install Pipeline

15.2.1 Pipeline CLI

After successfully installed FATE Client, user needs to configure server information and log directory for Pipeline. Pipeline provides a command line tool for quick setup. Run the following command for more information.

```
pipeline init --help
```

15.3 Interface of Pipeline

15.3.1 Component

FATE modules are wrapped into `component` in Pipeline API. Each component can take in and output `Data` and `Model`. Parameters of components can be set conveniently at the time of initialization. Unspecified parameters will take default values. All components have a `name`, which can be arbitrarily set. A component's name is its identifier, and so it must be unique within a pipeline. We suggest that each component name includes a numbering as suffix for easy tracking.

Components each may have input and/or output `Data` and/or `Model`. For details on how to use component, please refer to this guide.

An example of initializing a component with specified parameter values:

```
hetero_lr_0 = HeteroLR(name="hetero_lr_0", early_stop="weight_diff", max_iter=10,
                       early_stopping_rounds=2, validation_freqs=2)
```

15.3.2 Input

Input encapsulates all input of a component, including `Data` and `Model` input. To access `input` of a component, reference its `input` attribute:

```
input_all = dataio_0.input
```

15.3.3 Output

Output encapsulates all output result of a component, including `Data` and `Model` output. To access `Output` from a component, reference its `output` attribute:

```
output_all = dataio_0.output
```

15.3.4 Data

`Data` wraps all data-type input and output of components. FATE Pipeline includes five types of `data`, each is used for different scenario. For more information, please refer here.

15.3.5 Model

`Model` defines model input and output of components. Similar to `Data`, the two types of `models` are used for different purposes. For more information, please refer here.

15.4 Build A Pipeline

Below is a general guide to building a pipeline. Please refer to mini demo for an explained demo.

Once initialized a pipeline, job participants and initiator should be specified. Below is an example of initial setup of a pipeline:

```
pipeline = PipeLine()
pipeline.set_initiator(role='guest', party_id=9999)
pipeline.set_roles(guest=9999, host=10000, arbiter=10000)
```

`Reader` is required to read in data source so that other component(s) can process data. Define a `Reader` component:

```
reader_0 = Reader(name="reader_0")
```

In most cases, `DataIO` follows `Reader` to transform data into DataInstance format, which can then be used for data engineering and model training. Some components (such as `Union` and `Intersection`) can run directly on non-DataInstance tables.

All pipeline components can be configured individually for different roles by setting `get_party_instance`. For instance, `DataIO` component can be configured specifically for guest like this:

```
dataio_0 = DataIO(name="dataio_0")
guest_component_instance = dataio_0.get_party_instance(role='guest', party_id=9999)
guest_component_instance.component_param(with_label=True, output_format="dense")
```

To include a component in a pipeline, use `add_component`. To add the `DataIO` component to the previously created pipeline, try this:

```
pipeline.add_component(dataio_0, data=Data(data=reader_0.output.data))
```

15.4.1 Build Fate NN Model In Keras Style

In pipeline, you can build NN structures in a Keras style. Take Homo-NN as an example:

First, import Keras and define your nn structures:

```
from tensorflow.keras import optimizers
from tensorflow.keras.layers import Dense

layer_0 = Dense(units=6, input_shape=(10,), activation="relu")
layer_1 = Dense(units=1, activation="sigmoid")
```

Then, add nn layers into Homo-NN model like using Sequential class in Keras:

```
from pipeline.component.homo_nn import HomoNN

# set parameter
homo_nn_0 = HomoNN(name="homo_nn_0", max_iter=10, batch_size=-1, early_stop={"early_stop
↪": "diff", "eps": 0.0001})
homo_nn_0.add(layer_0)
homo_nn_0.add(layer_1)
```

Set optimizer and compile Homo-NN model:

```
homo_nn_0.compile(optimizer=optimizers.Adam(learning_rate=0.05), metrics=["Hinge",
↪"accuracy", "AUC"],
                  loss="binary_crossentropy")
```

Add it to pipeline:

```
pipeline.add_component(homo_nn, data=Data(train_data=dataio_0.output.data))
```

15.5 Init Runtime JobParameters

To fit or predict, user needs to initialize the runtime environment, like 'backend' and 'work_mode',

```
from pipeline.runtime.entity import JobParameters
job_parameters = JobParameters(backend=Backend.EGGROLL, work_mode=WorkMode.STANDALONE)
```

15.6 Run A Pipeline

Having added all components, user needs to first compile pipeline before running the designed job. After compilation, the pipeline can then be fit(run train job) with appropriate `Backend` and `WorkMode`.

```
pipeline.compile()
pipeline.fit(job_parameters)
```

15.7 Query on Tasks

FATE Pipeline provides API to query component information, including data, model, and summary. All query API have matching name to FlowPy, while Pipeline retrieves and returns query result directly to user.

```
summary = pipeline.get_component("hetero_lr_0").get_summary()
```

15.8 Deploy Components

Once fitting pipeline completes, prediction can be run on new data set. Before prediction, necessary components need to be first deployed. This step marks selected components to be used by prediction pipeline.

```
# deploy select components
pipeline.deploy_component([dataio_0, hetero_lr_0])
# deploy all components
# note that Reader component cannot be deployed. Always deploy pipeline with Reader by␣
↪specified component list.
pipeline.deploy_component()
```

15.9 Predict with Pipeline

First, initiate a new pipeline, then specify data source used for prediction.

```
predict_pipeline = PipeLine()
predict_pipeline.add_component(reader_0)
predict_pipeline.add_component(pipeline,
                               data=Data(predict_input={pipeline.dataio_0.input.data:␣
↪reader_0.output.data}))
```

Prediction can then be initiated on the new pipeline.

```
predict_pipeline.predict(job_parameters)
```

In addition, since pipeline is modular, user may add new components to the original pipeline before running prediction.

```
predict_pipeline.add_component(evaluation_0, data=Data(data=pipeline.hetero_lr_0.output.
↪data))
predict_pipeline.predict(job_parameters)
```

15.10 Save and Recovery of Pipeline

To save a pipeline, just use **dump** interface.

```
pipeline.dump("pipeline_saved.pkl")
```

To restore a pipeline, use **load_model_from_file** interface.

```
from pipeline.backend.pipeline import PineLine
PipeLine.load_model_from_file("pipeline_saved.pkl")
```

15.11 Summary Info of Pipeline

To get the details of a pipeline, use **describe** interface, which prints the "create time" fit or predict state and the constructed dsl if exists.

```
pipeline.describe()
```

15.12 Upload Data

Pipeline provides functionality to upload local data table. Please refer to upload demo for a quick example. Note that uploading data can be added all at once, and the pipeline used to perform upload can be either training or prediction pipeline (or, a separate pipeline as in the demo).

15.13 Pipeline vs. CLI

In the past versions, user interacts with FATE through command line interface, often with manually configured conf and dsl json files. Manual configuration can be tedious and error-prone. FATE Pipeline forms task configure files automatically at compilation, allowing quick experiment with task design.

CHAPTER

SIXTEEN

FATE TEST

A collection of useful tools to running FATE's test.

16.1 quick start

1. (optional) create virtual env

```
python -m venv venv
source venv/bin/activate
pip install -U pip
```

2. install fate_test

```
pip install fate_test
fate_test --help
```

3. edit default fate_test_config.yaml

```
# edit priority config file with system default editor
# filling some field according to comments
fate_test config edit
```

4. configure FATE-Pipeline and FATE-Flow Commandline server setting

```
# configure FATE-Pipeline server setting
pipeline init --port 9380 --ip 127.0.0.1
# configure FATE-Flow Commandline server setting
flow init --port 9380 --ip 127.0.0.1
```

5. run some fate_test suite

```
fate_test suite -i <path contains *testsuite.json>
```

6. run some fate_test benchmark

```
fate_test benchmark-quality -i <path contains *benchmark.json>
```

7. useful logs or exception will be saved to logs dir with namespace shown in last step

16.2 develop install

It is more convenient to use the editable mode during development: replace step 2 with flowing steps

```
pip install -e ${FATE}/python/fate_client && pip install -e ${FATE}/python/fate_test
```

16.3 command types

- suite: used for running testsuites, collection of FATE jobs

  ```
  fate_test suite -i <path contains *testsuite.json>
  ```

- benchmark-quality used for comparing modeling quality between FATE and other machine learning systems

  ```
  fate_test benchmark-quality -i <path contains *benchmark.json>
  ```

16.4 configuration by examples

1. no need ssh tunnel:

   - 9999, service: service_a
   - 10000, service: service_b

   and both service_a, service_b can be requested directly:

   ```
   work_mode: 1 # 0 for standalone, 1 for cluster
   data_base_dir: <path_to_data>
   parties:
     guest: [10000]
     host: [9999, 10000]
     arbiter: [9999]
   services:
     - flow_services:
       - {address: service_a, parties: [9999]}
       - {address: service_b, parties: [10000]}
   ```

2. need ssh tunnel:

   - 9999, service: service_a
   - 10000, service: service_b

   service_a, can be requested directly while service_b don't, but you can request service_b in other node, say B:

   ```
   work_mode: 0 # 0 for standalone, 1 for cluster
   data_base_dir: <path_to_data>
   parties:
     guest: [10000]
     host: [9999, 10000]
     arbiter: [9999]
   services:
   ```

(continues on next page)

(continued from previous page)

```
  - flow_services:
    - {address: service_a, parties: [9999]}
  - flow_services:
    - {address: service_b, parties: [10000]}
    ssh_tunnel: # optional
    enable: true
    ssh_address: <ssh_ip_to_B>:<ssh_port_to_B>
    ssh_username: <ssh_username_to B>
    ssh_password: # optional
    ssh_priv_key: "~/.ssh/id_rsa"
```

16.5 Testsuite

Testsuite is used for running a collection of jobs in sequence. Data used for jobs could be uploaded before jobs are submitted, and are cleaned when jobs finished. This tool is useful for FATE's release test.

16.5.1 command options

```
fate_test suite --help
```

1. include:

   ```
   fate_test suite -i <path1 contains *testsuite.json>
   ```

   will run testsuites in *path1*

2. exclude:

   ```
   fate_test suite -i <path1 contains *testsuite.json> -e <path2 to exclude> -e <path3␣
   ↪to exclude> ...
   ```

   will run testsuites in *path1* but not in *path2* and *path3*

3. glob:

   ```
   fate_test suite -i <path1 contains *testsuite.json> -g "hetero*"
   ```

   will run testsuites in sub directory start with *hetero* of *path1*

4. replace:

   ```
   fate_test suite -i <path1 contains *testsuite.json> -r '{"maxIter": 5}'
   ```

   will find all key-value pair with key "maxIter" in *data conf* or *conf* or *dsl* and replace the value with 5

5. skip-data:

   ```
   fate_test suite -i <path1 contains *testsuite.json> --skip-data
   ```

   will run testsuites in *path1* without uploading data specified in *benchmark.json*.

6. yes:

```
fate_test suite -i <path1 contains *testsuite.json> --yes
```

will run testsuites in *path1* directly, skipping double check

7. skip-dsl-jobs:

```
fate_test suite -i <path1 contains *testsuite.json> --skip-dsl-jobs
```

will run testsuites in *path1* but skip all *tasks* in testsuites. It's would be useful when only pipeline tasks needed.

8. skip-pipeline-jobs:

```
fate_test suite -i <path1 contains *testsuite.json> --skip-pipeline-jobs
```

will run testsuites in *path1* but skip all *pipeline tasks* in testsuites. It's would be useful when only dsl tasks needed.

16.6 Benchmark Quality

Benchmark-quality is used for comparing modeling quality between FATE and other machine learning systems. Benchmark produces a metrics comparison summary for each benchmark job group.

```
fate_test benchmark-quality -i examples/benchmark_quality/hetero_linear_regression
```

```
+-------+-----------------------------------------------------------------+
|  Data |                              Name                               |
+-------+-----------------------------------------------------------------+
| train | {'guest': 'motor_hetero_guest', 'host': 'motor_hetero_host'} |
|  test | {'guest': 'motor_hetero_guest', 'host': 'motor_hetero_host'} |
+-------+-----------------------------------------------------------------+
+-----------------------------------+--------------------+--------------------+---------
↪----------------+---------------------+
|             Model Name            | explained_variance |      r2_score      | root_
↪mean_squared_error |  mean_squared_error |
+-----------------------------------+--------------------+--------------------+---------
↪----------------+---------------------+
| local-linear_regression-regression | 0.9035168452250094 | 0.9035070863155368 |   0.
↪31340413289880553   | 0.09822215051805216 |
| FATE-linear_regression-regression  | 0.903146386539082  | 0.9031411831961411 |    0.
↪3139977881119483   | 0.09859461093919596 |
+-----------------------------------+--------------------+--------------------+---------
↪----------------+---------------------+
+-------------------------+-----------+
|          Metric         | All Match |
+-------------------------+-----------+
|    explained_variance   |    True   |
|         r2_score        |    True   |
| root_mean_squared_error |    True   |
|    mean_squared_error   |    True   |
+-------------------------+-----------+
```

16.6.1 command options

use the following command to show help message

```
fate_test benchmark-quality --help
```

1. include:

   ```
   fate_test benchmark-quality -i <path1 contains *benchmark.json>
   ```

   will run benchmark testsuites in *path1*

2. exclude:

   ```
   fate_test benchmark-quality -i <path1 contains *benchmark.json> -e <path2 to␣
   ↪exclude> -e <path3 to exclude> ...
   ```

   will run benchmark testsuites in *path1* but not in *path2* and *path3*

3. glob:

   ```
   fate_test benchmark-quality -i <path1 contains *benchmark.json> -g "hetero*"
   ```

   will run benchmark testsuites in sub directory start with *hetero* of *path1*

4. tol:

   ```
   fate_test benchmark-quality -i <path1 contains *benchmark.json> -t 1e-3
   ```

   will run benchmark testsuites in *path1* with absolute tolerance of difference between metrics set to 0.001. If absolute difference between metrics is smaller than *tol*, then metrics are considered almost equal. Check benchmark testsuite *writing guide* on setting alternative tolerance.

5. skip-data:

   ```
   fate_test benchmark-quality -i <path1 contains *benchmark.json> --skip-data
   ```

   will run benchmark testsuites in *path1* without uploading data specified in *benchmark.json*.

6. yes:

   ```
   fate_test benchmark-quality -i <path1 contains *benchmark.json> --yes
   ```

   will run benchmark testsuites in *path1* directly, skipping double check

16.6.2 benchmark testsuite

Configuration of jobs should be specified in a benchmark testsuite whose file name ends with "*benchmark.json". For benchmark testsuite example, please refer here.

A benchmark testsuite includes the following elements:

- data: list of local data to be uploaded before running FATE jobs
  - file: path to original data file to be uploaded, should be relative to testsuite or FATE installation path
  - head: whether file includes header
  - partition: number of partition for data storage

  - table_name: table name in storage
  - namespace: table namespace in storage
  - role: which role to upload the data, as specified in fate_test.config; naming format is: "{role_type}_{role_index}", index starts at 0

```
"data": [
    {
        "file": "examples/data/motor_hetero_host.csv",
        "head": 1,
        "partition": 8,
        "table_name": "motor_hetero_host",
        "namespace": "experiment",
        "role": "host_0"
    }
]
```

- job group: each group includes arbitrary number of jobs with paths to corresponding script and configuration
  - job: name of job to be run, must be unique within each group list
    - script: path to *testing script*, should be relative to testsuite
    - conf: path to job configuration file for script, should be relative to testsuite

```
"local": {
     "script": "./local-linr.py",
     "conf": "./linr_config.yaml"
}
```

  - compare_setting: additional setting for quality metrics comparison, currently only takes `relative_tol`

    If metrics *a* and *b* satisfy *abs(a-b) <= max(relative_tol * max(abs(a), abs(b)), absolute_tol)* (from math module), they are considered almost equal. In the below example, metrics from "local" and "FATE" jobs are considered almost equal if their relative difference is smaller than *0.05 * max(abs(local_metric), abs(pipeline_metric)*.

```
"linear_regression-regression": {
    "local": {
        "script": "./local-linr.py",
        "conf": "./linr_config.yaml"
    },
    "FATE": {
        "script": "./fate-linr.py",
        "conf": "./linr_config.yaml"
    },
    "compare_setting": {
        "relative_tol": 0.01
    }
}
```

16.6.3 testing script

All job scripts need to have `Main` function as an entry point for executing jobs; scripts should return two dictionaries: first with data information key-value pairs: {data_type}: {data_name_dictionary}; the second contains {metric_name}: {metric_value} key-value pairs for metric comparison.

By default, the final data summary shows the output from the job named "FATE"; if no such job exists, data information returned by the first job is shown. For clear presentation, we suggest that user follow this general guideline for data set naming. In the case of multi-host task, consider numbering host as such:

```
{'guest': 'default_credit_homo_guest',
 'host_1': 'default_credit_homo_host_1',
 'host_2': 'default_credit_homo_host_2'}
```

Returned quality metrics of the same key are to be compared. Note that only **real-value** metrics can be compared.

- FATE script: `Main` should have three inputs:
  - config: job configuration, JobConfig object loaded from "fate_test_config.yaml"
  - param: job parameter setting, dictionary loaded from "conf" file specified in benchmark testsuite
  - namespace: namespace suffix, user-given *namespace* or generated timestamp string when using *namespace-mangling*
- non-FATE script: `Main` should have one or two inputs:
  - param: job parameter setting, dictionary loaded from "conf" file specified in benchmark testsuite
  - (optional) config: job configuration, JobConfig object loaded from "fate_test_config.yaml"

Note that `Main` in FATE & non-FATE scripts can also be set to take zero input argument.

16.7 data

Data sub-command is used for upload or delete dataset in suite's.

16.7.1 command options

```
fate_test data --help
```

1. include:

   ```
   fate_test data [upload|delete] -i <path1 contains *testsuite.json>
   ```

   will upload/delete dataset in testsuites in *path1*

2. exclude:

   ```
   fate_test data [upload|delete] -i <path1 contains *testsuite.json> -e <path2 to␣
   ↪exclude> -e <path3 to exclude> ...
   ```

   will upload/delete dataset in testsuites in *path1* but not in *path2* and *path3*

3. glob:

```
fate_test data [upload|delete] -i <path1 contains *testsuite.json> -g "hetero*"
```

will upload/delete dataset in testsuites in sub directory start with *hetero* of *path1*

16.8 full command options

16.8.1 fate_test

A collection of useful tools to running FATE's test.

```
fate_test [OPTIONS] COMMAND [ARGS]...
```

Options

-b, --backend <backend>
Manual specify backend, 0 for eggroll, 1 for spark

-w, --work-mode <work_mode>
Manual specify work mode, 0 for local, 1 for cluster

-y, --yes
Skip double check

-nm, --namespace-mangling
Mangling data namespace

-n, --namespace <namespace>
Manual specify fate_test namespace

-c, --config <config>
Manual specify config file

benchmark-quality

process benchmark suite, alias: bq

```
fate_test benchmark-quality [OPTIONS]
```

Options

-i, --include <include>
Required include *benchmark.json under these paths

-e, --exclude <exclude>
exclude *benchmark.json under these paths

-g, --glob <glob>
glob string to filter sub-directory of path specified by <include>

-t, --tol <tol>
tolerance (absolute error) for metrics to be considered almost equal. Comparison is done by evaluating abs(a-b) <= max(relative_tol * max(abs(a), abs(b)), absolute_tol)

--skip-data
skip uploading data specified in benchmark conf

config

fate_test config

```
fate_test config [OPTIONS] COMMAND [ARGS]...
```

check

check connection

```
fate_test config check [OPTIONS]
```

edit

edit fate_test config file

```
fate_test config edit [OPTIONS]
```

new

create new fate_test config temperate

```
fate_test config new [OPTIONS]
```

show

show fate_test default config path

```
fate_test config show [OPTIONS]
```

data

upload or delete data in suite config files

```
fate_test data [OPTIONS] COMMAND [ARGS]...
```

delete

delete data defined in suite config files

```
fate_test data delete [OPTIONS]
```

Options

-i, --include <include>
Required include *benchmark.json under these paths

-e, --exclude <exclude>
exclude *benchmark.json under these paths

-g, --glob <glob>
glob string to filter sub-directory of path specified by <include>

-s, --suite-type <suite_type>
Required suite type

Options testsuite | benchmark

upload

upload data defined in suite config files

```
fate_test data upload [OPTIONS]
```

Options

-i, --include <include>
Required include *benchmark.json under these paths

-e, --exclude <exclude>
exclude *benchmark.json under these paths

-g, --glob <glob>
glob string to filter sub-directory of path specified by <include>

-s, --suite-type <suite_type>
Required suite type

Options testsuite | benchmark

-r, --role <role>
role to process, default to *all*. use option likes: *guest_0*, *host_0*, *host*

suite

process testsuite

```
fate_test suite [OPTIONS]
```

Options

-i, --include <include>
: **Required** include *testsuite.json under these paths

-e, --exclude <exclude>
: exclude *testsuite.json under these paths

-r, --replace <replace>
: a json string represents mapping for replacing fields in data/conf/dsl

-g, --glob <glob>
: glob string to filter sub-directory of path specified by <include>

--skip-dsl-jobs
: skip dsl jobs defined in testsuite

--skip-pipeline-jobs
: skip pipeline jobs defined in testsuite

--skip-data
: skip uploading data specified in testsuite

--data-only
: upload data only

--disable-clean-data

--enable-clean-data

CHAPTER

SEVENTEEN

DEVELOPING GUIDES

[]

17.1 Develop a runnable algorithm module of FATE

In this document, it describes how to develop an algorithm module, which can be callable under the architecture of FATE.

To develop a module, the following 5 steps are needed.

1. define the python parameter object which will be used in this module.
2. define the setting conf json of the module.
3. define the transfer_variable json if the module needs federation.
4. define your module which should inherit model_base class.
5. Define the protobuf file required for model saving.
6. (optional) define Pipeline component for your module.

In the following sections we will describe the 5 steps in detail, with toy_example.

17.1.1 Step 1. Define the parameter object this module will use

Parameter object is the only way to pass user-define runtime parameters to the developing module, so every module has it's own parameter object. In order to define a usable parameter object, three steps will be needed.

a. Open a new python file, rename it as xxx_param.py where xxx stands for your module'name, putting it in folder python/federatedm/param/. The class object defined in xxx_param.py should inherit the BaseParam class that define in python/federatedml/param/base_param.py

b. __init__ of your parameter class should specify all parameters that the module use.

c. Override the check interface of BaseParam, without which will cause not implemented error. Check method is use to validate the parameter variables.

Take hetero lr's parameter object as example, the python file is python/federatedml/param/logistic_regression_param.py

firstly, it inherits BaseParam:

```
class LogisticParam(BaseParam):
```

secondly, define all parameter variable in __init__ method:

```
def __init__(self, penalty='L2',
             eps=1e-5, alpha=1.0, optimizer='sgd', party_weight=1,
             batch_size=-1, learning_rate=0.01, init_param=InitParam(),
             max_iter=100, converge_func='diff',
             encrypt_param=EncryptParam(), re_encrypt_batches=2,
             encrypted_mode_calculator_param=EncryptedModeCalculatorParam(),
             need_run=True, predict_param=PredictParam(), cv_
↪param=CrossValidationParam()):
    super(LogisticParam, self).__init__()
    self.penalty = penalty
    self.eps = eps
    self.alpha = alpha
    self.optimizer = optimizer
    self.batch_size = batch_size
    self.learning_rate = learning_rate
    self.init_param = copy.deepcopy(init_param)
    self.max_iter = max_iter
    self.converge_func = converge_func
    self.encrypt_param = copy.deepcopy(encrypt_param)
    self.re_encrypt_batches = re_encrypt_batches
    self.party_weight = party_weight
    self.encrypted_mode_calculator_param = copy.deepcopy(encrypted_mode_calculator_param)
    self.need_run = need_run
    self.predict_param = copy.deepcopy(predict_param)
    self.cv_param = copy.deepcopy(cv_param)
```

As the example shown above, the parameter can also be a Param class that inherit the BaseParam. The default setting of this kind of parameter is an instance of this class. Then allocated a deepcopy version of this instance to the class attribution. The deepcopy function is used to avoid same pointer risk during the task running.

Once the class defined properly, a provided parameter parser can parse the value of each attribute recursively.

thirdly, override the check interface:

```
def check(self):
    descr = "logistic_param's"

    if type(self.penalty).__name__ != "str":
        raise ValueError(
            "logistic_param's penalty {} not supported, should be str type".format(self.
↪penalty))
    else:
        self.penalty = self.penalty.upper()
        if self.penalty not in ['L1', 'L2', 'NONE']:
            raise ValueError(
                "logistic_param's penalty not supported, penalty should be 'L1', 'L2' or
↪'none'")

    if type(self.eps).__name__ != "float":
        raise ValueError(
            "logistic_param's eps {} not supported, should be float type".format(self.
↪eps))
```

17.1.2 Step 2. Define the setting conf of the new module

The purpose to define a setting conf is that fate_flow module extract this file to get the information of how to start program of the module.

a. Define the setting conf in *python/federatedml/conf/setting_conf/*, name it as xxx.json, where xxx is the module you want to develop. Please note that xxx.json' name "xxx" is very strict, because when fate_flow dsl parser extract the module "xxx" in job dsl, it just concatenates module's name "xxx" with ".json" and retrieve the setting conf in *python/federatedml/conf/setting_conf/xxx.json*.

b. Field Specification of setting conf json.

module_path the path prefix of the developing module's program.

param_class the path to find the param_class define in Step 1, it's a concatenation of path of the parameter python file and parameter object name.

role

guest the path suffix to start the guest program

host the path suffix to start the host program

arbiter the path suffix to start the arbiter program

What's more, if this module does not need federation, which means all parties start a same program file, "guest|host|arbiter" is another way to define the role keys.

Take hetero-lr as an example, users can find it in python/federatedml/conf/setting_conf/HeteroLR.json

```
{
    "module_path":  "federatedml/logistic_regression/hetero_logistic_regression",
    "param_class" : "federatedml/param/logistic_regression_param.py/LogisticParam",
    "role":
    {
        "guest":
        {
            "program": "hetero_lr_guest.py/HeteroLRGuest"
        },
        "host":
        {
            "program": "hetero_lr_host.py/HeteroLRHost"
        },
        "arbiter":
        {
            "program": "hetero_lr_arbiter.py/HeteroLRArbiter"
        }
    }
}
```

Have a look at the above content in HeteroLR.json, HeteroLR is a federation module, its' guest program is define in python/federatedml/logistic_regression/hetero_logistic_regression/hetero_lr_guest.py and HeteroLRGuest is the guest class object. The same rules holds in host and arbiter class too. Fate_flow combine's module_path and role's program to run this module. "param_class" indicates that the parameter class object of HeteroLR is defined in "python/federatedml/param/logistic_regression_param.py", and the class name is LogisticParam.

17.1.3 Step 3. Define the transfer variable json of this module and generate transfer variable object. (Optional)

This step is needed only when this module is federated, which means there exists information interaction between different parties.

Note: this json file should be put under the folder transfer_class

In this python file, you would need to create a "transfer_variable" class and inherit the BaseTransferVariables class. Then, define each transfer variable as its attributes. Here is an example to make it more understandable:

```
from federatedml.transfer_variable.base_transfer_variable import BaseTransferVariables


# noinspection PyAttributeOutsideInit
class HeteroBoostingTransferVariable(BaseTransferVariables):
    def __init__(self, flowid=0):
        super().__init__(flowid)
        self.booster_dim = self._create_variable(name='booster_dim', src=['guest'], dst=[
↪'host'])
        self.stop_flag = self._create_variable(name='stop_flag', src=['guest'], dst=[
↪'host'])
        self.predict_start_round = self._create_variable(name='predict_start_round',␣
↪src=['guest'], dst=['host'])
```

name a string represents variable name

src list, should be some combinations of "guest", "host", "arbiter", it stands for where interactive information is sending from.

dst list, should be some combinations of "guest", "host", "arbiter", defines where the interactive information is sending to.

After setting that, the following command would help you create corresponding json setting file in auth_conf folder where fate_flow can refer to.

```
python fate_arch/federation/transfer_variable/scripts/generate_auth_conf.py federatedml␣
↪federatedml/transfer_variable/auth_conf
```

17.1.4 Step 4. Define your module, it should inherit model_base

The rule of running a module with fate_flow_client is that:

1. retrieves the setting_conf and find the "module" and "role" fields of setting conf.
2. it initializes the running object of every party.
3. calls the fit method of running object.
4. calls the save_data method if needed.
5. calls the export_model method if needed.

In this section, we describe how to do 3-5. Many common interfaces are provided in python/federatedml/model_base.py .

Override fit interface if needed The fit function holds the form of following.

```
  def fit(self, train_data, validate_data):

Both train_data and validate_data(Optional) are Tables from upstream␣
↪components(DataIO for example). This is the file where you fit logic of␣
↪model or feature-engineering components located. When starting a␣
↪training task, this function will be called by model_base automatically.
```

Override predict interface if needed The predict function holds the form of following.

```
  def predict(self, data_inst):

Data_inst is a DTable. Similar to fit function, you can define the␣
↪prediction procedure in the predict function for different roles. When␣
↪starting a predict task, this function will be called by model_base␣
↪automatically. Meanwhile, in training task, this function will also be␣
↪called to predict train data and validation data (if existed). If you␣
↪are willing to use evaluation component to evaluate your predict result,␣
↪it should be designed as the following format:

- for binary, multi-class classification task and regression task, result␣
↪header should be: ["label", "predict_result", "predict_score", "predict_
↪detail", "type"]
    * label: Provided label
    * predict_result: Your predict result.
    * predict_score: For binary classification task, it is the score of␣
↪label "1". For multi-class classification, it is the score of highest␣
↪label. For regression task, it is your predict result.
    * predict_detail: For classification task, it is the detail scores of␣
↪each class. For regression task, it is your predict result.
    * type: The source of you input data, eg. train or test. It will be␣
↪added by model_base automatically.
- There are two Table return in clustering task.
    The format of first Table: ["cluster_sample_count", "cluster_inner_dist
↪", "inter_cluster_dist"]
    * cluster_sample_count: The sample count of each cluster.
    * cluster_inner_dist: The inner distance of each cluster.
    * inter_cluster_dist: The inter distance between each clusters.
    The format of second Table: ["predicted_cluster_index", "distance"]
    * predicted_cluster_index: Your predict label
    * distance: The distance between each sample to its center point.
```

Override transform interface if needed The transform function holds the form of following.

```
def transform(self, data_inst):
```

This function is used for feature-engineering components in predict task.

17.1.5 Step 5. Define the protobuf file required for model saving

Define your save_data interface so that fate-flow can obtain output data through it when needed.

```
def save_data(self):
    return self.data_output
```

To use the trained model through different platform, FATE use protobuf files to save the parameters and model result of a task. When developing your own module, you are supposed to create two proto files which defined your model content in this folder.

For more details of protobuf, please refer to this tutorial

The two proto files are 1. File with "meta" as suffix: Save the parameters of a task. 2. File with "param" as suffix: Save the model result of a task.

After defining your proto files, you can use the following script named proto_generate.sh to create the corresponding python file:

```
bash proto_generate.sh
```

Define export_model interface Similar with part b, define your export_model interface so that fate-flow can obtain output model when needed. The format should be a dict contains both "Meta" and "Param" proto buffer generated. Here is an example showing how to export model.

```
def export_model(self):
    meta_obj = self._get_meta()
    param_obj = self._get_param()
    result = {
        self.model_meta_name: meta_obj,
        self.model_param_name: param_obj
    }
    return result
```

17.1.6 Step 6. Define Pipeline component for your module

One wrapped into a component, module can be used with FATE Pipeline API. To define a Pipeline component, follow these guidelines:

1. all components reside in fate_client/pipeline/component directory
2. components should inherit common base `Component`
3. as a good practice, components should have the same names as their corresponding modules
4. components take in parameters at initialization as defined in fate_client/pipeline/param, where a BaseParam and consts file are provided
5. set attributes of component input and output, including whether module has output model, or type of data output('single' vs. 'multi')

Then you may use Pipeline to construct and initiate a job with the newly defined component. For guide on Pipeline usage, please refer to fate_client/pipeline.

17.2 Start a modeling task

After finished developing, here is a simple example for starting a modeling task.

1. **Upload data** Before starting a task, you need to load data among all the data-providers. To do that, a load_file config is needed to be prepared. Then run the following command:

```
flow data upload -c upload_data.json
```

Note: This step is needed for every data-provide node(i.e. Guest and Host).

2. **Start your modeling task** In this step, two config files corresponding to dsl config file and component config file should be prepared. Please make sure that the table_name and namespace in the conf file match with upload_data conf. Then run the following command:

```
flow job submit -d ${your_dsl_file.json} -c ${your_component_conf_json}
```

If you have defined Pipeline component for your module, you can also make a pipeline script and start your task by:

```
python ${your_pipeline.py}
```

3. **Check log files** Now you can check out the log in the following path: *${your_install_path}/logs/{your jobid}*.

For more detailed information about dsl configure file and parameter configure files, please check out *examples/dsl/v2*.

CHAPTER

EIGHTEEN

COMPUTING API

Most of the time, the federatedml's user does not need to know how to initialize a computing session because fate flow has already cover this for you. Unless, the user is writing unittest, and CTable related functions are involved. Initialize a computing session:

```
from fate_arch.session import computing_session
# initialize
computing_session.init(work_mode=0, backend=0, session_id="a great session")
# create a table from iterable data
table = computing_session.parallelize(range(100), include_key=False, partition=2)
```

class computing_session

static init(*session_id*, *work_mode=0*, *backend=0*)

initialize a computing session

Parameters

- **session_id** (*str*) – session id
- **work_mode** (*int*) – work mode, 0 for standalone, 1 for cluster
- **backend** (*int*) – computing backend, 0 for eggroll, 1 for spark

Returns computing session

Return type instance of concrete subclass of `CSessionABC`

static parallelize(*data: Iterable*, *partition: int*, *include_key: bool*, ***kwargs*) → *fate_arch.abc._computing.CTableABC*

create table from iterable data

Parameters

- **data** (*Iterable*) – data to create table from
- **partition** (*int*) – number of partitions of created table
- **include_key** (*bool*) – `True` for create table directly from data, `False` for create table with generated keys start from 0

Returns a table create from data

Return type instance of concrete subclass fo `CTableABC`

static stop()

stop session

After creating a table using computing session, many distributed computing api available

distributed computing

Classes:

CTableABC()	a table of pair-like data supports distributed processing
CSessionABC()	computing session to load/create/clean tables

class CTableABC

a table of pair-like data supports distributed processing

Attributes:

partitions	get the partitions of table

Methods:

save(address, partitions, schema, **kwargs)	save table
collect(**kwargs)	collect data from table
take([n])	take `n` data from table
first(**kwargs)	take one data from table
count()	number of data in table
map(func)	apply *func* to each data
mapValues(func)	apply *func* to each value of data
mapPartitions(func[, use_previous_behavior, ...])	apply `func` to each partition of table
mapReducePartitions(mapper, reducer, **kwargs)	apply `mapper` to each partition of table and then perform reduce by key operation with *reducer*
applyPartitions(func)	apply `func` to each partitions as a single object
flatMap(func)	apply a flat `func` to each data of table
reduce(func)	reduces all value in pair of table by a binary function *func*
glom()	coalesces all data within partition into a list
sample(*[, fraction, num, seed])	return a sampled subset of this Table.
filter(func)	returns a new table containing only those keys which satisfy a predicate passed in via `func`.
join(other, func)	returns intersection of this table and the other table.
union(other[, func])	returns union of this table and the other table.
subtractByKey(other)	returns a new table containing elements only in this table but not in the other table.

abstract property partitions

get the partitions of table

Returns number of partitions

Return type int

abstract save(*address: fate_arch.abc._address.AddressABC*, *partitions: int*, *schema: dict*, ***kwargs*)

save table

Parameters

- **address** (*AddressABC*) – address to save table to
- **partitions** (*int*) – number of partitions to save as
- **schema** (*dict*) – table schema

abstract collect(***kwargs*) → Generator
collect data from table

Returns generator of data

Return type generator

Notes

no order guarantee

abstract take(*n=1*, ***kwargs*)
take `n` data from table

Parameters **n** (*int*) – number of data to take

Returns a list of `n` data

Return type list

Notes

no order guarantee

abstract first(***kwargs*)
take one data from table

Returns a data from table

Return type object

Notes

no order guarantee

abstract count() → int
number of data in table

Returns number of data

Return type int

abstract map(*func*) → *fate_arch.abc._computing.CTableABC*
apply *func* to each data

Parameters **func** (`typing.Callable[[object, object], typing.Tuple[object, object]]`) – function map (k1, v1) to (k2, v2)

Returns A new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([('k1', 1), ('k2', 2), ('k3', 3)],␣
↪include_key=True, partition=2)
>>> b = a.map(lambda k, v: (k, v**2))
>>> list(b.collect())
[("k1", 1), ("k2", 4), ("k3", 9)]
```

abstract mapValues(*func*)

apply *func* to each value of data

Parameters **func** (typing.Callable[[object], object]) – map v1 to v2

Returns A new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([('a', ['apple', 'banana', 'lemon']), ('b
↪', ['grapes'])], include_key=True, partition=2)
>>> b = a.mapValues(lambda x: len(x))
>>> list(b.collect())
[('a', 3), ('b', 1)]
```

abstract mapPartitions(*func*, *use_previous_behavior=True*, *preserves_partitioning=False*)

apply func to each partition of table

Parameters

- **func** (typing.Callable[[iter], list]) – accept an iterator of pair, return a list of pair
- **use_previous_behavior** (*bool*) – this parameter is provided for compatible reason, if set True, call this func will call applyPartitions instead
- **preserves_partitioning** (*bool*) – flag indicate whether the *func* will preserve partition

Returns a new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([1, 2, 3, 4, 5], include_key=False,␣
↪partition=2)
>>> def f(iterator):
...     s = 0
...     for k, v in iterator:
...             s += v
...         return [(s, s)]
```

(continues on next page)

(continued from previous page)

```
...
>>> b = a.mapPartitions(f)
>>> list(b.collect())
[(6, 6), (9, 9)]
```

abstract mapReducePartitions(*mapper*, *reducer*, ***kwargs*)

apply `mapper` to each partition of table and then perform reduce by key operation with *reducer*

Parameters

- **mapper** (`typing.Callable[[iter], list]`) – accept an iterator of pair, return a list of pair
- **reducer** (`typing.Callable[[object, object], object]`) – reduce v1, v2 to v3

Returns a new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> table = computing_session.parallelize([(1, 2), (2, 3), (3, 4), (4, 5)],␣
↪include_key=False, partition=2)
>>> def _mapper(it):
...     r = []
...     for k, v in it:
...         r.append((k % 3, v**2))
...         r.append((k % 2, v ** 3))
...     return r
>>> def _reducer(a, b):
...     return a + b
>>> output = table.mapReducePartitions(_mapper, _reducer)
>>> collected = dict(output.collect())
>>> assert collected[0] == 3 ** 3 + 5 ** 3 + 4 ** 2
>>> assert collected[1] == 2 ** 3 + 4 ** 3 + 2 ** 2 + 5 ** 2
>>> assert collected[2] == 3 ** 2
```

applyPartitions(*func*)

apply `func` to each partitions as a single object

Parameters **func** (`typing.Callable[[iter], object]`) – accept a iterator, return a object

Returns a new table, with each partition contains a single key-value pair

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([1, 2, 3], partition=3, include_
↪key=False)
>>> def f(it):
...     r = []
...     for k, v in it:
...         r.append(v, v**2, v**3)
...     return r
>>> output = a.applyPartitions(f)
>>> assert (2, 2**2, 2**3) in [v[0] for _, v in output.collect()]
```

abstract flatMap(*func*)

apply a flat func to each data of table

Parameters **func** (typing.Callable[[object, object], typing.List[object, object]]) – a flat function accept two parameters return a list of pair

Returns a new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([(1, 1), (2, 2)], include_key=True,␣
↪partition=2)
>>> b = a.flatMap(lambda x, y: [(x, y), (x + 10, y ** 2)])
>>> c = list(b.collect())
>>> assert len(c) = 4
>>> assert ((1, 1) in c) and ((2, 2) in c) and ((11, 1) in c) and ((12, 4) in␣
↪c)
```

abstract reduce(*func*)

reduces all value in pair of table by a binary function *func*

Parameters **func** (typing.Callable[[object, object], object]) – binary function reduce two value into one

Notes

func should be associative

Returns a single object

Return type object

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize(range(100), include_key=False,␣
↪partition=4)
>>> assert a.reduce(lambda x, y: x + y) == sum(range(100))
```

abstract glom()
coalesces all data within partition into a list

Returns list containing all coalesced partition and its elements. First element of each tuple is chosen from key of last element of each partition.

Return type list

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize(range(5), include_key=False,␣
↪partition=3).glom().collect()
>>> list(a)
[(2, [(2, 2)]), (3, [(0, 0), (3, 3)]), (4, [(1, 1), (4, 4)])]
```

abstract sample(*, *fraction: Optional[float] = None*, *num: Optional[int] = None*, *seed=None*)
return a sampled subset of this Table.

Parameters

- **fraction** (*float*) – Expected size of the sample as a fraction of this table's size without replacement: probability that each element is chosen. Fraction must be [0, 1] with replacement: expected number of times each element is chosen.
- **num** (*int*) – Exact number of the sample from this table's size
- **seed** (*int*) – Seed of the random number generator. Use current timestamp when *None* is passed.

Notes

use one of `fraction` and `num`, not both

Returns a new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> x = computing_session.parallelize(range(100), include_key=False,␣
↪partition=4)
>>>  6 <= x.sample(fraction=0.1, seed=81).count() <= 14
True
```

abstract filter(*func*)
returns a new table containing only those keys which satisfy a predicate passed in via `func`.

Parameters **func** (*Callable[[object, object], bool]*) – Predicate function returning a boolean.

Returns A new table containing results.

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([0, 1, 2], include_key=False,␣
↪partition=2)
>>> b = a.filter(lambda k, v : k % 2 == 0)
>>> list(b.collect())
[(0, 0), (2, 2)]
>>> c = a.filter(lambda k, v : v % 2 != 0)
>>> list(c.collect())
[(1, 1)]
```

abstract join(*other*, *func*)

returns intersection of this table and the other table.

function func will be applied to values of keys that exist in both table.

Parameters

- **other** (CTableABC) – another table to be operated with.
- **func** (typing.Callable[[object, object], object]) – the function applying to values whose key exists in both tables. default using left table's value.

Returns a new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([1, 2, 3], include_key=False,␣
↪partition=2)        # [(0, 1), (1, 2), (2, 3)]
>>> b = computing_session.parallelize([(1, 1), (2, 2), (3, 3)], include_
↪key=True, partition=2)
>>> c = a.join(b, lambda v1, v2 : v1 + v2)
>>> list(c.collect())
[(1, 3), (2, 5)]
```

abstract union(*other*, *func=<function CTableABC.<lambda>>*)

returns union of this table and the other table.

function func will be applied to values of keys that exist in both table.

Parameters

- **other** (CTableABC) – another table to be operated with.
- **func** (typing.Callable[[object, object], object]) – The function applying to values whose key exists in both tables. default using left table's value.

Returns a new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize([1, 2, 3], include_key=False,␣
↪partition=2)         # [(0, 1), (1, 2), (2, 3)]
>>> b = computing_session.parallelize([(1, 1), (2, 2), (3, 3)], include_
↪key=True, partition=2)
>>> c = a.union(b, lambda v1, v2 : v1 + v2)
>>> list(c.collect())
[(0, 1), (1, 3), (2, 5), (3, 3)]
```

abstract subtractByKey(*other*)

returns a new table containing elements only in this table but not in the other table.

Parameters **other** (CTableABC) – Another table to be subtractbykey with.

Returns A new table

Return type *CTableABC*

Examples

```
>>> from fate_arch.session import computing_session
>>> a = computing_session.parallelize(range(10), include_key=False,␣
↪partition=2)
>>> b = computing_session.parallelize(range(5), include_key=False, partition=2)
>>> c = a.subtractByKey(b)
>>> list(c.collect())
[(5, 5), (6, 6), (7, 7), (8, 8), (9, 9)]
```

class CSessionABC

computing session to load/create/clean tables

Methods:

load(address, partitions, schema, **kwargs)	load a table from given address
parallelize(data, partition, include_key, ...)	create table from iterable data
cleanup(name, namespace)	delete table(s)

Attributes:

session_id	get computing session id

abstract load(*address: fate_arch.abc._address.AddressABC*, *partitions*, *schema: dict*, ***kwargs*) → Union[fate_arch.abc._path.PathABC, *fate_arch.abc._computing.CTableABC*]

load a table from given address

Parameters

- **address** (*AddressABC*) – address to load table from

- **partitions** (*int*) – number of partitions of loaded table
- **schema** (*dict*) – schema associate with this table

Returns a table in memory

Return type *CTableABC*

abstract parallelize(*data: collections.abc.Iterable*, *partition: int*, *include_key: bool*, ***kwargs*) → *fate_arch.abc._computing.CTableABC*

create table from iterable data

Parameters

- **data** (*Iterable*) – data to create table from
- **partition** (*int*) – number of partitions of created table
- **include_key** (*bool*) – `True` for create table directly from data, `False` for create table with generated keys start from 0

Returns a table create from data

Return type *CTableABC*

abstract cleanup(*name*, *namespace*)

delete table(s)

Parameters

- **name** (*str*) – table name or wildcard character
- **namespace** (*str*) – namespace

abstract property session_id: str

get computing session id

Returns computing session id

Return type str

CHAPTER

NINETEEN

FEDERATION API

19.1 Low level api

Classes:

FederationABC()	federation, get or remote objects and tables

class FederationABC

federation, get or remote objects and tables

Methods:

get(name, tag, parties, gc)	get objects/tables from `parties`
remote(v, name, tag, parties, gc)	remote object/table to `parties`

abstract get(*name: str*, *tag: str*, *parties: List[fate_arch.common._types.Party]*, *gc: fate_arch.abc._gc.GarbageCollectionABC*) → List

get objects/tables from `parties`

Parameters

- **name** (*str*) – name of transfer variable
- **tag** (*str*) – tag to distinguish each transfer
- **parties** (*List[Party]*) – parties to get objects/tables from
- **gc** (*GarbageCollectionABC*) – used to do some clean jobs

Returns a list of object or a list of table get from parties with same order of *parties*

Return type list

abstract remote(*v*, *name: str*, *tag: str*, *parties: List[fate_arch.common._types.Party]*, *gc: fate_arch.abc._gc.GarbageCollectionABC*) → NoReturn

remote object/table to `parties`

Parameters

- **v** (*object or table*) – object/table to remote
- **name** (*str*) – name of transfer variable
- **tag** (*str*) – tag to distinguish each transfer
- **parties** (*List[Party]*) – parties to remote object/table to

- **gc** (*GarbageCollectionABC*) – used to do some clean jobs

Return type Notes

19.2 user api

remoting or getting an object(table) from other parties is quite easy using apis provided in `Variable`. First to create an instance of BaseTransferVariable, which is simply a collection of Variables:

```
from federatedml.transfer_variable.transfer_class import secure_add_example_transfer_
↪variable
variable = secure_add_example_transfer_variable.SecureAddExampleTransferVariable()
```

Then remote or get object(table) by variable provided by this instance:

```
# remote
variable.guest_share.remote("from guest")

# get
variable.guest_share.get()
```

Classes:

Variable(name, src, dst)	variable to distinguish federation by name

class Variable(*name: str*, *src: Tuple[str, ...]*, *dst: Tuple[str, ...]*)

variable to distinguish federation by name

Methods:

remote_parties(obj, parties[, suffix])	remote object to specified parties
get_parties(parties[, suffix])	get objects/tables from specified parties
remote(obj[, role, idx, suffix])	send obj to other parties.
get([idx, suffix])	get obj from other parties.

remote_parties(*obj*, *parties: Union[List[fate_arch.common._types.Party], fate_arch.common._types.Party]*, *suffix: Union[Any, Tuple] = ()*)

remote object to specified parties

Parameters

- **obj** (*object or table*) – object or table to remote
- **parties** (*List[Party]*) – parties to remote object/table to
- **suffix** (*str or tuple of str*) – suffix used to distinguish federation with in variable

Return type None

get_parties(*parties: Union[List[fate_arch.common._types.Party], fate_arch.common._types.Party]*, *suffix: Union[Any, Tuple] = ()*)

get objects/tables from specified parties

Parameters

- **parties** (*List[Party]*) – parties to remote object/table to
- **suffix** (*str or tuple of str*) – suffix used to distinguish federation with in variable

Returns a list of objects/tables get from parties with same order of `parties`

Return type list

remote(*obj*, *role=None*, *idx=- 1*, *suffix=()*)

send obj to other parties.

Parameters

- **obj** – object to be sent
- **role** – role of parties to sent to, use one of ['Host', 'Guest', 'Arbiter', None]. The default is None, means sent values to parties regardless their party role
- **idx** – id of party to sent to. The default is -1, which means sent values to parties regardless their party id
- **suffix** – additional tag suffix, the default is tuple()

get(*idx=- 1*, *suffix=()*)

get obj from other parties.

Parameters

- **idx** – id of party to get from. The default is -1, which means get values from parties regardless their party id
- **suffix** – additional tag suffix, the default is tuple()

Returns object or list of object

CHAPTER

TWENTY

PARAMS

Classes:

BoostingParam([task_type, objective_param, ...])	Basic parameter for Boosting Algorithms
ObjectiveParam([objective, params])	Define objective parameters that used in federated ml.
DecisionTreeParam([criterion_method, ...])	Define decision tree parameters that used in federated ml.
CrossValidationParam([n_splits, mode, role, ...])	Define cross validation params
DataSplitParam([random_state, test_size, ...])	Define data split param that used in data split.
DataIOParam([input_format, delimitor, ...])	Define dataio parameters that used in federated ml.
EncryptParam([method, key_length])	Define encryption method that used in federated ml. :param method: If method is 'Paillier', Paillier encryption will be used for federated ml. To use non-encryption version in HomoLR, set this to None. For detail of Paillier encryption, please check out the paper mentioned in README file. :type method: {'Paillier'} :param key_length: Used to specify the length of key in this encryption method. :type key_length: int, default: 1024.
EncryptedModeCalculatorParam([mode, ...])	Define the encrypted_mode_calulator parameters.
FeatureBinningParam([method, ...])	Define the feature binning method
FeatureSelectionParam([select_col_indexes, ...])	Define the feature selection parameters.
HeteroNNParam([task_type, config_type, ...])	Parameters used for Homo Neural Network.
HomoNNParam(secure_aggregate, ...[, ...])	Parameters used for Homo Neural Network.
HomoOneHotParam([transform_col_indexes, ...])	**param transform_col_indexes** Specify which columns need to calculated. -1 represent for all columns.
InitParam([init_method, init_const, ...])	Initialize Parameters used in initializing a model.
IntersectParam(intersect_method[, ...])	Define the intersect method
LinearParam([penalty, tol, alpha, ...])	Parameters used for Linear Regression.
LocalBaselineParam([model_name, model_opts, ...])	Define the local baseline model param
LogisticParam([penalty, tol, alpha, ...])	Parameters used for Logistic Regression both for Homo mode or Hetero mode.
OneVsRestParam([need_one_vs_rest, has_arbiter])	Define the one_vs_rest parameters.
PoissonParam([penalty, tol, alpha, ...])	Parameters used for Poisson Regression.
PredictParam([threshold])	Define the predict method of HomoLR, HeteroLR, SecureBoosting
RsaParam([rsa_key_n, rsa_key_e, rsa_key_d, ...])	Define the sample method

continues on next page

Table 1 – continued from previous page

SampleParam([mode, method, fractions, ...])	Define the sample method
ScaleParam([method, mode, ...])	Define the feature scale parameters.
StochasticQuasiNewtonParam([...])	Parameters used for stochastic quasi-newton method.
StatisticsParam([statistics, column_names, ...])	Define statistics params
StepwiseParam([score_name, mode, role, ...])	Define stepwise params
UnionParam([need_run, allow_missing, ...])	Define the union method for combining multiple dTables and keep entries with the same id

class BoostingParam(*task_type='classification'*, *objective_param=<federatedml.param.boosting_param.ObjectiveParam object>*, *learning_rate=0.3*, *num_trees=5*, *subsample_feature_rate=1*, *n_iter_no_change=True*, *tol=0.0001*, *bin_num=32*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *validation_freqs=None*, *metrics=None*, *subsample_random_seed=None*, *binning_error=0.0001*)

Basic parameter for Boosting Algorithms

Parameters

- **task_type** (*str, accepted 'classification', 'regression' only, default: 'classification'*) –
- **objective_param** (*ObjectiveParam Object, default: ObjectiveParam()*) –
- **learning_rate** (*float, accepted float, int or long only, the learning rate of secure boost. default: 0.3*) –
- **num_trees** (*int, accepted int, float only, the max number of boosting round. default: 5*) –
- **subsample_feature_rate** (*float, a float-number in [0, 1], default: 0.8*) –
- **n_iter_no_change** (*bool,*) – when True and residual error less than tol, tree building process will stop. default: True
- **bin_num** (*int, positive integer greater than 1, bin number use in quantile. default: 32*) –
- **validation_freqs** (*None or positive integer or container object in python. Do validation in training process or Not.*) – if equals None, will not do validation in train process; if equals positive integer, will validate data every validation_freqs epochs passes; if container object in python, will validate data if epochs belong to this container.

  e.g. validation_freqs = [10, 15], will validate data when epoch equals to 10 and 15.

  Default: None

class ObjectiveParam(*objective='cross_entropy'*, *params=None*)

Define objective parameters that used in federated ml.

Parameters

- **objective** (*None or str, accepted None,'cross_entropy','lse','lae', 'log_cosh','tweedie','fair','huber' only,*) – None in host's config, should be str in guest'config. when task_type is classification, only support cross_entropy, other 6 types support in regression task. default: None

- **params** (*None or list, should be non empty list when objective is 'tweedie','fair','huber',*) – first element of list shoulf be a float-number large than 0.0 when objective is 'fair','huber', first element of list should be a float-number in [1.0, 2.0) when objective is 'tweedie'

class DecisionTreeParam(*criterion_method='xgboost'*, *criterion_params=[0.1]*, *max_depth=3*, *min_sample_split=2*, *min_impurity_split=0.001*, *min_leaf_node=1*, *max_split_nodes=65536*, *feature_importance_type='split'*, *n_iter_no_change=True*, *tol=0.001*, *use_missing=False*, *zero_as_missing=False*)

Define decision tree parameters that used in federated ml.

Parameters

- **criterion_method** (*str, accepted "xgboost" only, the criterion function to use, default: 'xgboost'*) –
- **criterion_params** (*list, should be non empty and first element is float-number, default: 0.1.*) –
- **max_depth** (*int, positive integer, the max depth of a decision tree, default: 3*) –
- **min_sample_split** (*int, least quantity of nodes to split, default: 2*) –
- **min_impurity_split** (*float, least gain of a single split need to reach, default: 1e-3*) –
- **min_leaf_node** (*int, when samples no more than min_leaf_node, it becomes a leave, default: 1*) –
- **max_split_nodes** (*int, positive integer, we will use no more than max_split_nodes to*) – parallel finding their splits in a batch, for memory consideration. default is 65536
- **feature_importance_type** (*str, support 'split', 'gain' only.*) – if is 'split', feature_importances calculate by feature split times, if is 'gain', feature_importances calculate by feature split gain. default: 'split'
- **use_missing** (*bool, accepted True, False only, use missing value in training process or not. default: False*) –
- **zero_as_missing** (*bool, accepted True, False only, regard 0 as missing value or not,*) – will be use only if use_missing=True, default: False

class CrossValidationParam(*n_splits=5*, *mode='hetero'*, *role='guest'*, *shuffle=True*, *random_seed=1*, *need_cv=False*)

Define cross validation params

Parameters

- **n_splits** (*int, default: 5*) – Specify how many splits used in KFold
- **mode** (*str, default: 'Hetero'*) – Indicate what mode is current task
- **role** (*str, default: 'Guest'*) – Indicate what role is current party
- **shuffle** (*bool, default: True*) – Define whether do shuffle before KFold or not.
- **random_seed** (*int, default: 1*) – Specify the random seed for numpy shuffle
- **need_cv** (*bool, default True*) – Indicate if this module needed to be run

class DataSplitParam(*random_state=None*, *test_size=None*, *train_size=None*, *validate_size=None*, *stratified=False*, *shuffle=True*, *split_points=None*, *need_run=True*)

Define data split param that used in data split.

Parameters

- **random_state** (*None, int, default: None*) – Specify the random state for shuffle.
- **test_size** (*None, float, int, default: 0.0*) – Specify test data set size. float value specifies fraction of input data set, int value specifies exact number of data instances
- **train_size** (*None, float, int, default: 0.8*) – Specify train data set size. float value specifies fraction of input data set, int value specifies exact number of data instances
- **validate_size** (*None, float, int, default: 0.2*) – Specify validate data set size. float value specifies fraction of input data set, int value specifies exact number of data instances
- **stratified** (*boolean, default: False*) – Define whether sampling should be stratified, according to label value.
- **shuffle** (*boolean, default : True*) – Define whether do shuffle before splitting or not.
- **split_points** (*None, list, default : None*) – Specify the point(s) by which continuous label values are bucketed into bins for stratified split. eg.[0.2] for two bins or [0.1, 1, 3] for 4 bins
- **need_run** (*bool, default: True*) – Specify whether to run data split

class DataIOParam(*input_format='dense'*, *delimitor=','*, *data_type='float64'*, *exclusive_data_type=None*, *tag_with_value=False*, *tag_value_delimitor=':'*, *missing_fill=False*, *default_value=0*, *missing_fill_method=None*, *missing_impute=None*, *outlier_replace=False*, *outlier_replace_method=None*, *outlier_impute=None*, *outlier_replace_value=0*, *with_label=False*, *label_name='y'*, *label_type='int'*, *output_format='dense'*, *need_run=True*)

Define dataio parameters that used in federated ml.

Parameters

- **input_format** (*str, accepted 'dense','sparse' 'tag' only in this version. default: 'dense'.*) – please have a look at this tutorial at "DataIO" section of federatedml/util/README.md. Formally,

  dense input format data should be set to "dense", svm-light input format data should be set to "sparse", tag or tag:value input format data should be set to "tag".

- **delimitor** (*str, the delimitor of data input, default: ','*) –
- **data_type** (*str, the data type of data input, accepted 'float', 'float64','int','int64','str','long'*) – "default: "float64"
- **exclusive_data_type** (*dict, the key of dict is col_name, the value is data_type, use to specified special data type*) – of some features.
- **tag_with_value** (*bool, use if input_format is 'tag', if tag_with_value is True,*) – input column data format should be tag[delimitor]value, otherwise is tag only
- **tag_value_delimitor** (*str, use if input_format is 'tag' and 'tag_with_value' is True,*) – delimitor of tag[delimitor]value column value.

- **missing_fill** (*bool, need to fill missing value or not, accepted only True/False, default: False*) –
- **default_value** (*None or single object type or list, the value to replace missing value.*) – if None, it will use default value define in federatedml/feature/imputer.py, if single object, will fill missing value with this object, if list, it's length should be the sample of input data' feature dimension,

  means that if some column happens to have missing values, it will replace it the value by element in the identical position of this list.

  default: None
- **missing_fill_method** (*None or str, the method to replace missing value, should be one of [None, 'min', 'max', 'mean', 'designated'], default: None*) –
- **missing_impute** (*None or list, element of list can be any type, or auto generated if value is None, define which values to be consider as missing, default: None*) –
- **outlier_replace** (*bool, need to replace outlier value or not, accepted only True/False, default: True*) –
- **outlier_replace_method** (*None or str, the method to replace missing value, should be one of [None, 'min', 'max', 'mean', 'designated'], default: None*) –
- **outlier_impute** (*None or list, element of list can be any type, which values should be regard as missing value, default: None*) –
- **outlier_replace_value** (*None or single object type or list, the value to replace outlier.*) – if None, it will use default value define in federatedml/feature/imputer.py, if single object, will replace outlier with this object, if list, it's length should be the sample of input data' feature dimension,

  means that if some column happens to have outliers, it will replace it the value by element in the identical position of this list.

  default: None
- **with_label** (*bool, True if input data consist of label, False otherwise. default: 'false'*) –
- **label_name** (*str, column_name of the column where label locates, only use in dense-inputformat. default: 'y'*) –
- **label_type** (*object, accepted 'int','int64','float','float64','long','str' only,*) – use when with_label is True. default: 'false'
- **output_format** (*str, accepted 'dense','sparse' only in this version. default: 'dense'*) –

class EncryptParam(*method='Paillier'*, *key_length=1024*)

Define encryption method that used in federated ml. :param method: If method is 'Paillier', Paillier encryption will be used for federated ml.

To use non-encryption version in HomoLR, set this to None. For detail of Paillier encryption, please check out the paper mentioned in README file.

Parameters **key_length** (*int, default: 1024*) – Used to specify the length of key in this encryption method.

class EncryptedModeCalculatorParam(*mode='strict'*, *re_encrypted_rate=1*)

Define the encrypted_mode_calulator parameters.

Parameters

- **mode** (*str, support 'strict', 'fast', 'balance', 'confusion_opt', ' only, default: strict*) –
- **re_encrypted_rate** (*float or int, numeric number in [0, 1], use when mode equals to 'balance, default: 1*) –

class FeatureBinningParam(*method='quantile'*, *compress_thres=10000*, *head_size=10000*, *error=0.0001*, *bin_num=10*, *bin_indexes=-1*, *bin_names=None*, *adjustment_factor=0.5*, *transform_param=<federatedml.param.feature_binning_param.TransformParam object>*, *optimal_binning_param=<federatedml.param.feature_binning_param.OptimalBinningParam object>*, *local_only=False*, *category_indexes=None*, *category_names=None*, *need_run=True*, *skip_static=False*)

Define the feature binning method

Parameters

- **method** (*str, 'quantile' 'bucket' or 'optimal', default: 'quantile'*) – Binning method.
- **compress_thres** (*int, default: 10000*) – When the number of saved summaries exceed this threshold, it will call its compress function
- **head_size** (*int, default: 10000*) – The buffer size to store inserted observations. When head list reach this buffer size, the QuantileSummaries object start to generate summary(or stats) and insert into its sampled list.
- **error** (*float, 0 <= error < 1 default: 0.001*) – The error of tolerance of binning. The final split point comes from original data, and the rank of this value is close to the exact rank. More precisely, floor((p - 2 * error) * N) <= rank(x) <= ceil((p + 2 * error) * N) where p is the quantile in float, and N is total number of data.
- **bin_num** (*int, bin_num > 0, default: 10*) – The max bin number for binning
- **bin_indexes** (*list of int or int, default: -1*) – Specify which columns need to be binned. -1 represent for all columns. If you need to indicate specific cols, provide a list of header index instead of -1.
- **bin_names** (*list of string, default: []*) – Specify which columns need to calculated. Each element in the list represent for a column name in header.
- **adjustment_factor** (*float, default: 0.5*) – the adjustment factor when calculating WOE. This is useful when there is no event or non-event in a bin. Please note that this parameter will NOT take effect for setting in host.
- **category_indexes** (*list of int or int, default: []*) – Specify which columns are category features. -1 represent for all columns. List of int indicate a set of such features. For category features, bin_obj will take its original values as split_points and treat them as have been binned. If this is not what you expect, please do NOT put it into this parameters.

  The number of categories should not exceed bin_num set above.
- **category_names** (*list of string, default: []*) – Use column names to specify category features. Each element in the list represent for a column name in header.

- **local_only** (*bool, default: False*) – Whether just provide binning method to guest party. If true, host party will do nothing.
- **transform_param** (*TransformParam*) – Define how to transfer the binned data.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run
- **skip_static** (*bool, default False*) – If true, binning will not calculate iv, woe etc. In this case, optimal-binning will not be supported.

class FeatureSelectionParam(*select_col_indexes=-1*, *select_names=None*, *filter_methods=None*, *unique_param=<federatedml.param.feature_selection_param.UniqueValueParam object>*, *iv_value_param=<federatedml.param.feature_selection_param.IVValueSelectionParam object>*, *iv_percentile_param=<federatedml.param.feature_selection_param.IVPercentileSelectionParam object>*, *iv_top_k_param=<federatedml.param.feature_selection_param.IVTopKParam object>*, *variance_coe_param=<federatedml.param.feature_selection_param.VarianceOfCoeSelectionParam object>*, *outlier_param=<federatedml.param.feature_selection_param.OutlierColsSelectionParam object>*, *manually_param=<federatedml.param.feature_selection_param.ManuallyFilterParam object>*, *percentage_value_param=<federatedml.param.feature_selection_param.PercentageValueParam object>*, *iv_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *statistic_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *psi_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *sbt_param=<federatedml.param.feature_selection_param.CommonFilterParam object>*, *need_run=True*)

Define the feature selection parameters.

Parameters

- **select_col_indexes** (*list or int, default: -1*) – Specify which columns need to calculated. -1 represent for all columns.
- **select_names** (*list of string, default: []*) – Specify which columns need to calculated. Each element in the list represent for a column name in header.
- **filter_methods** (*list, ["manually", "iv_filter", "statistic_filter",]*) –

  "psi_filter", "hetero_sbt_filter", "homo_sbt_filter", "hetero_fast_sbt_filter", "percentage_value"],

  default: ["manually"]

  The following methods will be deprecated in future version: "unique_value", "iv_value_thres", "iv_percentile", "coefficient_of_variation_value_thres", "outlier_cols"

  Specify the filter methods used in feature selection. The orders of filter used is depended on this list. Please be notified that, if a percentile method is used after some certain filter method, the percentile represent for the ratio of rest features.

e.g. If you have 10 features at the beginning. After first filter method, you have 8 rest. Then, you want top 80% highest iv feature. Here, we will choose floor(0.8 * 8) = 6 features instead of 8.

- **unique_param** (*filter the columns if all values in this feature is the same*) –
- **iv_value_param** (*Use information value to filter columns. If this method is set, a float threshold need to be provided.*) – Filter those columns whose iv is smaller than threshold. Will be deprecated in the future.
- **iv_percentile_param** (*Use information value to filter columns. If this method is set, a float ratio threshold*) – need to be provided. Pick floor(ratio * feature_num) features with higher iv. If multiple features around the threshold are same, all those columns will be keep. Will be deprecated in the future.
- **variance_coe_param** (*Use coefficient of variation to judge whether filtered or not.*) – Will be deprecated in the future.
- **outlier_param** (*Filter columns whose certain percentile value is larger than a threshold.*) – Will be deprecated in the future.
- **percentage_value_param** (*Filter the columns that have a value that exceeds a certain percentage.*) –
- **iv_param** (*Setting how to filter base on iv. It support take high mode only. All of "threshold",*) – "top_k" and "top_percentile" are accepted. Check more details in CommonFilterParam. To use this filter, hetero-feature-binning module has to be provided.
- **statistic_param** (*Setting how to filter base on statistic values. All of "threshold",*) – "top_k" and "top_percentile" are accepted. Check more details in CommonFilterParam. To use this filter, data_statistic module has to be provided.
- **psi_param** (*Setting how to filter base on psi values. All of "threshold",*) – "top_k" and "top_percentile" are accepted. Its take_high properties should be False to choose lower psi features. Check more details in CommonFilterParam. To use this filter, data_statistic module has to be provided.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class HeteroNNParam(*task_type='classification'*, *config_type='keras'*, *bottom_nn_define=None*, *top_nn_define=None*, *interactive_layer_define=None*, *interactive_layer_lr=0.9*, *optimizer='SGD'*, *loss=None*, *epochs=100*, *batch_size=-1*, *early_stop='diff'*, *tol=1e-05*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *encrypted_mode_calculator_param=<federatedml.param.encrypted_mode_calculation_param.EncryptedMode object>*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *validation_freqs=None*, *early_stopping_rounds=None*, *metrics=None*, *use_first_metric_only=True*)

Parameters used for Homo Neural Network.

Parameters

- **task_type** – str, task type of hetero nn model, one of 'classification', 'regression'.
- **config_type** – str, accept "keras" only.
- **bottom_nn_define** – a dict represents the structure of bottom neural network.
- **interactive_layer_define** – a dict represents the structure of interactive layer.

- **interactive_layer_lr** – float, the learning rate of interactive layer.
- **top_nn_define** – a dict represents the structure of top neural network.
- **optimizer** – optimizer method, accept following types: 1. a string, one of "Adadelta", "Adagrad", "Adam", "Adamax", "Nadam", "RMSprop", "SGD" 2. a dict, with a required key-value pair keyed by "optimizer",

  with optional key-value pairs such as learning rate.

  defaults to "SGD"
- **loss** – str, a string to define loss function used
- **early_stopping_rounds** – int, default: None
- **rounds** (*Will stop training if one metric doesn't improve in last early_stopping_round*) –
- **metrics** – list, default: None Indicate when executing evaluation during train process, which metrics will be used. If not set, default metrics for specific task type will be used. As for binary classification, default metrics are ['auc', 'ks'], for regression tasks, default metrics are ['root_mean_squared_error', 'mean_absolute_error'], [ACCURACY, PRECISION, RECALL] for multi-classification task
- **use_first_metric_only** – bool, default: False Indicate whether to use the first metric in *metrics* as the only criterion for early stopping judgement.
- **epochs** – int, the maximum iteration for aggregation in training.
- **batch_size** – int, batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy. defaults to -1.
- **early_stop** – str, accept 'diff' only in this version, default: 'diff' Method used to judge converge or not.

  a) diff Use difference of loss between two iterations to judge whether converge.
- **validation_freqs** – None or positive integer or container object in python. Do validation in training process or Not. if equals None, will not do validation in train process; if equals positive integer, will validate data every validation_freqs epochs passes; if container object in python, will validate data if epochs belong to this container.

  e.g. validation_freqs = [10, 15], will validate data when epoch equals to 10 and 15.

  Default: None The default value is None, 1 is suggested. You can set it to a number larger than 1 in order to speed up training by skipping validation rounds. When it is larger than 1, a number which is divisible by "epochs" is recommended, otherwise, you will miss the validation scores of last training epoch.

class HomoNNParam(*secure_aggregate: bool = True*, *aggregate_every_n_epoch: int = 1*, *config_type: str = 'nn'*, *nn_define: typing.Optional[dict] = None*, *optimizer: typing.Union[str, dict, types.SimpleNamespace] = 'SGD'*, *loss: typing.Optional[str] = None*, *metrics: typing.Optional[typing.Union[str, list]] = None*, *max_iter: int = 100*, *batch_size: int = -1*, *early_stop: typing.Union[str, dict, types.SimpleNamespace] = 'diff'*, *encode_label: bool = False*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*)

Parameters used for Homo Neural Network.

Parameters **Args** – secure_aggregate: enable secure aggregation or not, defaults to True. aggregate_every_n_epoch: aggregate model every n epoch, defaults to 1. config_type: one of "nn",

"keras", "tf" nn_define: a dict represents the structure of neural network. optimizer: optimizer method, accept following types:

1. a string, one of "Adadelta", "Adagrad", "Adam", "Adamax", "Nadam", "RMSprop", "SGD"
2. **a dict, with a required key-value pair keyed by "optimizer",** with optional key-value pairs such as learning rate.

defaults to "SGD"

loss: a string metrics: max_iter: the maximum iteration for aggregation in training. batch_size : batch size when updating model.

-1 means use all data in a batch. i.e. Not to use mini-batch strategy. defaults to -1.

early_stop [str, 'diff', 'weight_diff' or 'abs', default: 'diff']

Method used to judge converge or not.

a) diff Use difference of loss between two iterations to judge whether converge.

b) weight_diff: Use difference between weights of two consecutive iterations

c) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < eps, it is converged.

encode_label : encode label to one_hot.

class HomoOneHotParam(*transform_col_indexes=- 1*, *transform_col_names=None*, *need_run=True*, *need_alignment=True*)

Parameters

- **transform_col_indexes** (*list or int, default: -1*) – Specify which columns need to calculated. -1 represent for all columns.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run
- **need_alignment** (*bool, default True*) – Indicated whether alignment of features is turned on

class InitParam(*init_method='random_uniform'*, *init_const=1*, *fit_intercept=True*, *random_seed=None*)

Initialize Parameters used in initializing a model.

Parameters

- **init_method** (*str, 'random_uniform', 'random_normal', 'ones', 'zeros' or 'const'. default: 'random_uniform'*) – Initial method.
- **init_const** (*int or float, default: 1*) – Required when init_method is 'const'. Specify the constant.
- **fit_intercept** (*bool, default: True*) – Whether to initialize the intercept or not.

class IntersectParam(*intersect_method: str = 'rsa'*, *random_bit=128*, *sync_intersect_ids=True*, *join_role='guest'*, *with_encode=False*, *only_output_key=False*, *encode_params=<federatedml.param.intersect_param.EncodeParam object>*, *intersect_cache_param=<federatedml.param.intersect_param.IntersectCache object>*, *repeated_id_process=False*, *repeated_id_owner='guest'*, *allow_info_share: bool = False*, *info_owner='guest'*)

Define the intersect method

Parameters

- **intersect_method** (*str, it supports 'rsa' and 'raw', default by 'rsa'*) –
- **random_bit** (*positive int, it will define the encrypt length of rsa algorithm. It effective only for intersect_method is rsa*) –
- **sync_intersect_ids** (*bool. In rsa, 'synchronize_intersect_ids' is True means guest or host will send intersect results to the others, and False will not.*) – while in raw, 'synchronize_intersect_ids' is True means the role of "join_role" will send intersect results and the others will get them. Default by True.
- **join_role** (*str, it supports "guest" and "host" only and effective only for raw. If it is "guest", the host will send its ids to guest and find the intersection of*) – ids in guest; if it is "host", the guest will send its ids. Default by "guest".
- **with_encode** (*bool, if True, it will use encode method for intersect ids. It effective only for "raw".*) –
- **encode_params** (*EncodeParam, it effective only for with_encode is True*) –
- **only_output_key** (*bool, if false, the results of intersection will include key and value which from input data; if true, it will just include key from input*) – data and the value will be empty or some useless character like "intersect_id"
- **repeated_id_process** (*bool, if true, intersection will process the ids which can be repeatable*) –
- **repeated_id_owner** (*str, which role has the repeated ids*) –

class LinearParam(*penalty='L2'*, *tol=0.0001*, *alpha=1.0*, *optimizer='sgd'*, *batch_size=-1*, *learning_rate=0.01*, *init_param=<federatedml.param.init_model_param.InitParam object>*, *max_iter=20*, *early_stop='diff'*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *sqn_param=<federatedml.param.sqn_param.StochasticQuasiNewtonParam object>*, *encrypted_mode_calculator_param=<federatedml.param.encrypted_mode_calculation_param.EncryptedModeCa object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *decay=1*, *decay_sqrt=True*, *validation_freqs=None*, *early_stopping_rounds=None*, *stepwise_param=<federatedml.param.stepwise_param.StepwiseParam object>*, *metrics=None*, *use_first_metric_only=False*)

Parameters used for Linear Regression.

Parameters

- **penalty** (*str, 'L1' or 'L2'. default: 'L2'*) – Penalty method used in LinR. Please note that, when using encrypted version in HeteroLinR, 'L1' is not supported.
- **tol** (*float, default: 1e-4*) – The tolerance of convergence
- **alpha** (*float, default: 1.0*) – Regularization strength coefficient.
- **optimizer** (*str, 'sgd', 'rmsprop', 'adam', 'sqn', or 'adagrad', default: 'sgd'*) – Optimize method
- **batch_size** (*int, default: -1*) – Batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy.
- **learning_rate** (*float, default: 0.01*) – Learning rate
- **max_iter** (*int, default: 20*) – The maximum iteration for training.

- **init_param** (*InitParam object, default: default InitParam object*) – Init param method object.
- **early_stop** (*str, 'diff' or 'abs' or 'weight_dff', default: 'diff'*) –

  Method used to judge convergence.

  a) diff Use difference of loss between two iterations to judge whether converge.

  b) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < tol, it is converged.

  c) weight_diff: Use difference between weights of two consecutive iterations
- **predict_param** (*PredictParam object, default: default PredictParam object*) –
- **encrypt_param** (*EncryptParam object, default: default EncryptParam object*) –
- **encrypted_mode_calculator_param** (*EncryptedModeCalculatorParam object, default: default EncryptedModeCalculatorParam object*) –
- **cv_param** (*CrossValidationParam object, default: default CrossValidationParam object*) –
- **decay** (*int or float, default: 1*) – Decay rate for learning rate. learning rate will follow the following decay schedule. lr = lr0/(1+decay*t) if decay_sqrt is False. If decay_sqrt is True, lr = lr0 / sqrt(1+decay*t) where t is the iter number.
- **decay_sqrt** (*Bool, default: True*) – lr = lr0/(1+decay*t) if decay_sqrt is False, otherwise, lr = lr0 / sqrt(1+decay*t)
- **validation_freqs** (*int, list, tuple, set, or None*) – validation frequency during training, required when using early stopping. The default value is None, 1 is suggested. You can set it to a number larger than 1 in order to speed up training by skipping validation rounds. When it is larger than 1, a number which is divisible by "max_iter" is recommended, otherwise, you will miss the validation scores of the last training iteration.
- **early_stopping_rounds** (*int, default: None*) – If positive number specified, at every specified training rounds, program checks for early stopping criteria. Validation_freqs must also be set when using early stopping.
- **metrics** (*list or None, default: None*) – Specify which metrics to be used when performing evaluation during training process. If metrics have not improved at early_stopping rounds, trianing stops before convergence. If set as empty, default metrics will be used. For regression tasks, default metrics are ['root_mean_squared_error', 'mean_absolute_error']
- **use_first_metric_only** (*bool, default: False*) – Indicate whether to use the first metric in *metrics* as the only criterion for early stopping judgement.

class LocalBaselineParam(*model_name='LogisticRegression'*, *model_opts=None*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *need_run=True*)

Define the local baseline model param

Parameters

- **model_name** (*str, sklearn model used to train on baseline model*) –
- **model_opts** (*dict or none, default None*) – Param to be used as input into baseline model

- **predict_param** (*PredictParam object, default: default PredictParam object*) –
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class LogisticParam(*penalty='L2'*, *tol=0.0001*, *alpha=1.0*, *optimizer='rmsprop'*, *batch_size=-1*, *learning_rate=0.01*, *init_param=<federatedml.param.init_model_param.InitParam object>*, *max_iter=100*, *early_stop='diff'*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *decay=1*, *decay_sqrt=True*, *multi_class='ovr'*, *validation_freqs=None*, *early_stopping_rounds=None*, *stepwise_param=<federatedml.param.stepwise_param.StepwiseParam object>*, *metrics=None*, *use_first_metric_only=False*)

Parameters used for Logistic Regression both for Homo mode or Hetero mode.

Parameters

- **penalty** (*str, 'L1', 'L2' or None. default: 'L2'*) – Penalty method used in LR. Please note that, when using encrypted version in HomoLR, 'L1' is not supported.
- **tol** (*float, default: 1e-4*) – The tolerance of convergence
- **alpha** (*float, default: 1.0*) – Regularization strength coefficient.
- **optimizer** (*str, 'sgd', 'rmsprop', 'adam', 'nesterov_momentum_sgd', 'sqn' or 'adagrad', default: 'rmsprop'*) – Optimize method, if 'sqn' has been set, sqn_param will take effect. Currently, 'sqn' support hetero mode only.
- **batch_size** (*int, default: -1*) – Batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy.
- **learning_rate** (*float, default: 0.01*) – Learning rate
- **max_iter** (*int, default: 100*) – The maximum iteration for training.
- **early_stop** (*str, 'diff', 'weight_diff' or 'abs', default: 'diff'*) –

  Method used to judge converge or not.

  a) diff Use difference of loss between two iterations to judge whether converge.

  b) weight_diff: Use difference between weights of two consecutive iterations

  c) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < eps, it is converged.

  Please note that for hetero-lr multi-host situation, this parameter support "weight_diff" only.
- **decay** (*int or float, default: 1*) – Decay rate for learning rate. learning rate will follow the following decay schedule. lr = lr0/(1+decay*t) if decay_sqrt is False. If decay_sqrt is True, lr = lr0 / sqrt(1+decay*t) where t is the iter number.
- **decay_sqrt** (*Bool, default: True*) – lr = lr0/(1+decay*t) if decay_sqrt is False, otherwise, lr = lr0 / sqrt(1+decay*t)
- **encrypt_param** (*EncryptParam object, default: default EncryptParam object*) –
- **predict_param** (*PredictParam object, default: default PredictParam object*) –

- **cv_param** (*CrossValidationParam object, default: default CrossValidationParam object*) –
- **multi_class** (*str, 'ovr', default: 'ovr'*) – If it is a multi_class task, indicate what strategy to use. Currently, support 'ovr' short for one_vs_rest only.
- **validation_freqs** (*int, list, tuple, set, or None*) – validation frequency during training.
- **early_stopping_rounds** (*int, default: None*) – Will stop training if one metric doesn't improve in last early_stopping_round rounds
- **metrics** (*list or None, default: None*) – Indicate when executing evaluation during train process, which metrics will be used. If set as empty, default metrics for specific task type will be used. As for binary classification, default metrics are ['auc', 'ks']
- **use_first_metric_only** (*bool, default: False*) – Indicate whether use the first metric only for early stopping judgement.

class OneVsRestParam(*need_one_vs_rest=False*, *has_arbiter=True*)

Define the one_vs_rest parameters.

Parameters **has_arbiter** (*bool. For some algorithm, may not has arbiter, for instances, secureboost of FATE,*) – for these algorithms, it should be set to false. default true

class PoissonParam(*penalty='L2'*, *tol=0.0001*, *alpha=1.0*, *optimizer='rmsprop'*, *batch_size=-1*, *learning_rate=0.01*, *init_param=<federatedml.param.init_model_param.InitParam object>*, *max_iter=20*, *early_stop='diff'*, *exposure_colname=None*, *predict_param=<federatedml.param.predict_param.PredictParam object>*, *encrypt_param=<federatedml.param.encrypt_param.EncryptParam object>*, *encrypted_mode_calculator_param=<federatedml.param.encrypted_mode_calculation_param.EncryptedModeC object>*, *cv_param=<federatedml.param.cross_validation_param.CrossValidationParam object>*, *stepwise_param=<federatedml.param.stepwise_param.StepwiseParam object>*, *decay=1*, *decay_sqrt=True*, *validation_freqs=None*, *early_stopping_rounds=None*, *metrics=None*, *use_first_metric_only=False*)

Parameters used for Poisson Regression.

Parameters

- **penalty** (*str, 'L1' or 'L2'. default: 'L2'*) – Penalty method used in Poisson. Please note that, when using encrypted version in HeteroPoisson, 'L1' is not supported.
- **tol** (*float, default: 1e-4*) – The tolerance of convergence
- **alpha** (*float, default: 1.0*) – Regularization strength coefficient.
- **optimizer** (*str, 'sgd', 'rmsprop', 'adam' or 'adagrad', default: 'rmsprop'*) – Optimize method
- **batch_size** (*int, default: -1*) – Batch size when updating model. -1 means use all data in a batch. i.e. Not to use mini-batch strategy.
- **learning_rate** (*float, default: 0.01*) – Learning rate
- **max_iter** (*int, default: 20*) – The maximum iteration for training.
- **init_param** (*InitParam object, default: default InitParam object*) – Init param method object.
- **early_stop** (*str, 'weight_diff', 'diff' or 'abs', default: 'diff'*) –

Method used to judge convergence.

a) diff Use difference of loss between two iterations to judge whether converge.

b) weight_diff: Use difference between weights of two consecutive iterations

c) abs: Use the absolute value of loss to judge whether converge. i.e. if loss < eps, it is converged.

- **exposure_colname** (*str or None, default: None*) – Name of optional exposure variable in dTable.
- **predict_param** (*PredictParam object, default: default PredictParam object*) –
- **encrypt_param** (*EncryptParam object, default: default EncryptParam object*) –
- **encrypted_mode_calculator_param** (*EncryptedModeCalculatorParam object, default: default EncryptedModeCalculatorParam object*) –
- **cv_param** (*CrossValidationParam object, default: default CrossValidationParam object*) –
- **stepwise_param** (*StepwiseParam object, default: default StepwiseParam object*) –
- **decay** (*int or float, default: 1*) – Decay rate for learning rate. learning rate will follow the following decay schedule. lr = lr0/(1+decay*t) if decay_sqrt is False. If decay_sqrt is True, lr = lr0 / sqrt(1+decay*t) where t is the iter number.
- **decay_sqrt** (*Bool, default: True*) – lr = lr0/(1+decay*t) if decay_sqrt is False, otherwise, lr = lr0 / sqrt(1+decay*t)
- **validation_freqs** (*int, list, tuple, set, or None*) – validation frequency during training, required when using early stopping. The default value is None, 1 is suggested. You can set it to a number larger than 1 in order to speed up training by skipping validation rounds. When it is larger than 1, a number which is divisible by "max_iter" is recommended, otherwise, you will miss the validation scores of the last training iteration.
- **early_stopping_rounds** (*int, default: None*) – If positive number specified, at every specified training rounds, program checks for early stopping criteria. Validation_freqs must also be set when using early stopping.
- **metrics** (*list or None, default: None*) – Specify which metrics to be used when performing evaluation during training process. If metrics have not improved at early_stopping rounds, trianing stops before convergence. If set as empty, default metrics will be used. For regression tasks, default metrics are ['root_mean_squared_error', 'mean_absolute_error']
- **use_first_metric_only** (*bool, default: False*) – Indicate whether to use the first metric in *metrics* as the only criterion for early stopping judgement.

class PredictParam(*threshold=0.5*)

Define the predict method of HomoLR, HeteroLR, SecureBoosting

Parameters **threshold** (*float or int, The threshold use to separate positive and negative class. Normally, it should be (0,1)*) –

class RsaParam(*rsa_key_n=None*, *rsa_key_e=None*, *rsa_key_d=None*, *save_out_table_namespace=None*, *save_out_table_name=None*)

Define the sample method

Parameters

- **rsa_key_n** (*integer, RSA modulus, default: None*) –
- **rsa_key_e** (*integer, RSA public exponent, default: None*) –
- **rsa_key_d** (*integer, RSA private exponent, default: None*) –
- **save_out_table_namespace** (*str, namespace of dtable where stores the output data. default: None*) –
- **save_out_table_name** (*str, name of dtable where stores the output data. default: None*) –

class SampleParam(*mode='random'*, *method='downsample'*, *fractions=None*, *random_state=None*, *task_type='hetero'*, *need_run=True*)

Define the sample method

Parameters

- **mode** (*str, accepted 'random','stratified" only in this version, specify sample to use, default: 'random'*) –
- **method** (*str, accepted 'downsample','upsample' only in this version. default: 'downsample'*) –
- **fractions** (*None or float or list, if mode equals to random, it should be a float number greater than 0,*) – otherwise a list of elements of pairs like [label_i, sample_rate_i], e.g. [[0, 0.5], [1, 0.8], [2, 0.3]]. default: None
- **random_state** (*int, RandomState instance or None, default: None*) –
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class ScaleParam(*method=None*, *mode='normal'*, *scale_col_indexes=-1*, *scale_names=None*, *feat_upper=None*, *feat_lower=None*, *with_mean=True*, *with_std=True*, *need_run=True*)

Define the feature scale parameters.

Parameters

- **method** (*str, like scale in sklearn, now it support "min_max_scale" and "standard_scale", and will support other scale method soon.*) – Default None, which will do nothing for scale
- **mode** (*str, the mode support "normal" and "cap". for mode is "normal", the feat_upper and feat_lower is the normal value like "10" or "3.1" and for "cap", feat_upper and*) – feature_lower will between 0 and 1, which means the percentile of the column. Default "normal"
- **feat_upper** (*int or float, the upper limit in the column. If the scaled value is larger than feat_upper, it will be set to feat_upper. Default None.*) –
- **feat_lower** (*int or float, the lower limit in the column. If the scaled value is less than feat_lower, it will be set to feat_lower. Default None.*) –
- **scale_col_indexes** (*list,the idx of column in scale_column_idx will be scaled, while the idx of column is not in, it will not be scaled.*) –
- **scale_names** (*list of string, default: [].Specify which columns need to scaled. Each element in the list represent for a column name in header.*) –

- **with_mean** (*bool, used for "standard_scale". Default False.*) –
- **with_std** (*bool, used for "standard_scale". Default False.*) – The standard scale of column x is calculated as : z = (x - u) / s, where u is the mean of the column and s is the standard deviation of the column. if with_mean is False, u will be 0, and if with_std is False, s will be 1.
- **need_run** (*bool, default True*) – Indicate if this module needed to be run

class StochasticQuasiNewtonParam(*update_interval_L=3*, *memory_M=5*, *sample_size=5000*, *random_seed=None*)

Parameters used for stochastic quasi-newton method.

Parameters

- **update_interval_L** (*int, default: 3*) – Set how many iteration to update hess matrix
- **memory_M** (*int, default: 5*) – Stack size of curvature information, i.e. y_k and s_k in the paper.
- **sample_size** (*int, default: 5000*) – Sample size of data that used to update Hess matrix

class StatisticsParam(*statistics='summary'*, *column_names=None*, *column_indexes=-1*, *need_run=True*, *abnormal_list=None*, *quantile_error=0.0001*, *bias=True*)

Define statistics params

Parameters

- **statistics** (*list, string, default "summary"*) – Specify the statistic types to be computed. "summary" represents list: [consts.SUM, consts.MEAN, consts.STANDARD_DEVIATION,

  consts.MEDIAN, consts.MIN, consts.MAX, consts.MISSING_COUNT, consts.SKEWNESS, consts.KURTOSIS]
- **column_names** (*list of string, default []*) – Specify columns to be used for statistic computation by column names in header
- **column_indexes** (*list of int, default -1*) – Specify columns to be used for statistic computation by column order in header -1 indicates to compute statistics over all columns
- **bias** (*bool, default: True*) – If False, the calculations of skewness and kurtosis are corrected for statistical bias.
- **need_run** (*bool, default True*) – Indicate whether to run this modules

class StepwiseParam(*score_name='AIC'*, *mode='hetero'*, *role='guest'*, *direction='both'*, *max_step=10*, *nvmin=2*, *nvmax=None*, *need_stepwise=False*)

Define stepwise params

Parameters

- **score_name** (*str, default: 'AIC'*) – Specify which model selection criterion to be used
- **mode** (*str, default: 'Hetero'*) – Indicate what mode is current task
- **role** (*str, default: 'Guest'*) – Indicate what role is current party

- **direction** (*str, default: 'both'*) – Indicate which direction to go for stepwise. 'forward' means forward selection; 'backward' means elimination; 'both' means possible models of both directions are examined at each step.
- **max_step** (*int, default: '10'*) – Specify total number of steps to run before forced stop.
- **nvmin** (*int, default: '2'*) – Specify the min subset size of final model, cannot be lower than 2. When nvmin > 2, the final model size may be smaller than nvmin due to max_step limit.
- **nvmax** (*int, default: None*) – Specify the max subset size of final model, 2 <= nvmin <= nvmax. The final model size may be larger than nvmax due to max_step limit.
- **need_stepwise** (*bool, default False*) – Indicate if this module needed to be run

class UnionParam(*need_run=True*, *allow_missing=False*, *keep_duplicate=False*)

Define the union method for combining multiple dTables and keep entries with the same id

Parameters

- **need_run** (*bool, default True*) – Indicate if this module needed to be run
- **allow_missing** (*bool, default False*) – Whether allow mismatch between feature length and header length in the result. Note that empty tables will always be skipped regardless of this param setting.
- **keep_duplicate** (*bool, default False*) – Whether to keep entries with duplicated keys. If set to True, a new id will be generated for duplicated entry in the format {id}_{table_name}.

CHAPTER

TWENTYONE

MATERIALS

Here are some materials for learning and reference

21.1 Architecture

- FATE_v1.1_ARCH_SIMPLE_201911_v2.pdf

21.2 Workshop and Conference

- SecureBoost-ijcai2019-workshop.pdf
- GDPR_Data_Shortage_and_AI-AAAI_2019_PPT.pdf

21.3 Salon

- FATE.pdf
- Pipeline.pdf

PYTHON MODULE INDEX

f

INDEX

Symbols

A

B

C

D

E

F