ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ «ДИДЖИТРЕК» (ООО «ДТ»)
Адрес: 115114, Москва г., Муниципальный Округ Замоскворечье вн. тер. г., Шлюзовая набережная, д. 8, стр. 1
ИНН: 9705200012, КПП: 770501001, ОГРН: 1237700339788
Тел.: +7 985 770-45-12, Email: info@digi-track.ru

Руководство по использованию программного обеспечения

«DigiTrack Confidential Computing»

г. Москва – 2026

Оглавление

Руководство по использованию программного обеспечения 1

«DigiTrack Confidential Computing» 1

1. Введение 5

1.1. Назначение документа 5

1.2. Назначение программного обеспечения 5

1.3. Область применения 5

1.4. Термины и сокращения 5

1.5. Роли пользователей 6

2. Общая схема эксплуатации ПО 6

2.1. Состав участников 6

2.2. Что задаётся в ПО и как это задаётся 6

В рамках настройки задаются: 6

2.3. Общий порядок выполнения процесса 7

2.4. Разграничение ответственности сторон 7

Координатор выполняет: 7

Партнёр данных выполняет: 7

3. Подготовка окружения 8

3.1. Требования к инфраструктуре 8

3.2. Требования к каталогам 8

Назначение каталогов 8

3.3. Подготовка каналов связи 8

3.4. Передача артефактов установки 9

3.5. Проверка целостности файлов 9

4. Развёртывание серверного компонента VFL 9

4.1. Загрузка Docker-образа 9

Что делает команда 10

4.2. Проверка наличия образа 10

4.3. Подготовка рабочих директорий 10

4.4. Запуск серверного контейнера 10

Базовая команда запуска 10

4.5. Расшифровка команды запуска 10

Часть Docker 10

Часть запуска приложения 11

4.6. Проверка запуска контейнера 11

4.7. Просмотр логов контейнера 11

5. Подготовка данных 11

5.1. Подготовка идентификаторов для PSI 11

ЭТАП 1: Загрузка исходных данных 12

Требования к файлу 12

Пример файла 12
ЭТАП 2: Объединение идентификаторов 12
ЭТАП 3: Хеширование и подготовка к PSI 13
6. Настройка и запуск PSI 14
6.1. Назначение PSI 14
6.2. Что именно задаётся при запуске PSI 14
6.3. Подготовка входного файла 14
6.4. Команда запуска PSI 14
6.5. Описание параметров команды PSI 15
Полная команда 15
Расшифровка параметров 15
6.6. Просмотр всех доступных параметров PSI 15
6.7. Параметры PSI 15
Хэширование идентификаторов 15
Правила преобразования идентификатора 15
6.8. Параметры обработки PSI 15
6.9. Результаты выполнения PSI 16
Пример расположения результатов 16
6.10. Пример содержимого результата PSI 16
7. Настройка и запуск обучения модели (VFL) 17
7.1. Назначение этапа обучения 17
7.2. Подготовка признаков для обучения 17
Пример структуры файла признаков 17
Требования 17
7.3. Подготовка данных для предсказания 17
7.4. Требования к структуре файлов 17
7.5. Что задаётся перед запуском обучения 18
7.6. Команда запуска обучения 18
7.7. Просмотр параметров запуска обучения 18
7.8. Что означает запуск обучения 18
7.9. Параметры модели SecureBoost 19
7.10 Подробное описание параметров обучения 19
`num_trees` 19
`max_depth` 19
`learning_rate` 20
`objective` 20
`task_type` 20
`min_child_weight` 21
`subsample` 21
`colsample_bytree` 21
`random_state` 21
7.11. Результаты выполнения обучения 21
Пример расположения 21
Возможный результат 21

8. Настройка и запуск исполнения модели (скоринг) 22
8.1. Назначение этапа предсказания 22
8.2. Подготовка входных данных 22
8.3. Команда запуска предсказания 22
8.4. Просмотр параметров запуска предсказания 22
8.5. Что делает команда предсказания 23
8.6. Результаты выполнения предсказания 23
Пример расположения 23
Пример содержимого 23
9. Журналирование и контроль выполнения 24
9.1. Какие журналы формируются 24
9.2. Где искать результаты 24
9.3. Какие статусы контролировать 24
10. Администрирование и устранение неполадок 25
10.1. Типовые ошибки PSI 25
Ошибка: удалённый PSI-сервис недоступен 25
Возможные причины: 25
10.2. Типовые ошибки запуска контейнера 25
Проверка статуса контейнера 25
Перезапуск контейнера 25
Удаление контейнера и повторный запуск 25
10.3. Типовые ошибки обучения 25
Возможные причины: 25
Проверки: 26
10.4. Типовые ошибки предсказания 26
Возможные причины: 26
Проверки: 26
10.5. Рекомендации по эксплуатации 26
11. Результаты функционирования ПО 27
Приложение А. Минимальный эксплуатационный сценарий 27
Шаг 1. Создать каталоги 27
Шаг 2. Загрузить Docker-образ 27
Шаг 3. Проверить образ 27
Шаг 4. Запустить контейнер 27
Шаг 5. Подготовить файл идентификаторов 27
Шаг 6. Запустить PSI 27
Шаг 7. Запустить обучение 28
Шаг 8. Запустить предсказание 28
Шаг 9. Проверить результаты 28

1. Введение

1.1. Назначение документа

Настоящее руководство предназначено для администраторов, инженеров сопровождения, DevOps-специалистов и специалистов по анализу данных, выполняющих установку, настройку и эксплуатацию программного обеспечения **«DigiTrack Confidential Computing»** (далее — **ПО**, **Система**).

Документ устанавливает:

- порядок подготовки среды;
- порядок запуска серверных и клиентских компонентов;
- порядок подготовки входных данных;
- порядок выполнения процедур PSI, обучения и исполнения модели;
- порядок контроля результатов;
- действия при возникновении ошибок.

1.2. Назначение программного обеспечения

Программное обеспечение предназначено для реализации процессов защищённого совместного обучения и исполнения моделей машинного обучения в распределённой среде без передачи исходных данных между участниками.

1.3. Область применения

ПО применяется в информационных системах организаций, осуществляющих обработку данных с ограничениями на их передачу, в том числе в рамках межорганизационного взаимодействия.

1.4. Термины и сокращения

Термин	Определение
ПО	Программное обеспечение «DigiTrack Confidential Computing»
Координатор	Сторона, инициирующая и управляющая вычислительным процессом
Партнёр данных	Сторона, владеющая локальными данными и выполняющая вычисления на своей стороне
PSI	Private Set Intersection — протокол безопасной сверки множеств идентификаторов
VFL	Vertical Federated Learning — вертикальное федеративное обучение
SecureBoost	Алгоритм защищённого бустинга деревьев решений
Скоринг	Исполнение обученной модели для получения прогнозов

Термин	Определение
Docker-образ	Упакованный исполняемый артефакт серверного компонента
VPN	Защищённое сетевое соединение, создающее шифрованный туннель между устройством и сервером для скрытия IP и трафика
Python-скрипт	Исполняемый файл с кодом на Python
Python3	Актуальная ветка языка Python
Docker	Платформа для запуска приложений в изолированных контейнерах

1.5. Роли пользователей

Роль	Функции
Администратор системы	Развёртывание контейнеров, подготовка каталогов, контроль доступности сервисов
DevOps-специалист	Настройка окружения, сети, VPN, журналов, автоматизации
Дата-сайентист координатора	Подготовка параметров обучения, запуск обучения и предсказания
Дата-сайентист партнёра	Подготовка локальных данных и участие в вычислительных процессах
Аудитор / ИБ-специалист	Контроль каналов связи, целостности артефактов, журналов и режимов эксплуатации

2. Общая схема эксплуатации ПО

2.1. Состав участников

В минимальном сценарии используются две стороны:

- **Координатор** — сторона, инициирующая процессы;
- **Партнёр данных** — сторона, владеющая локальными признаками или локальными идентификаторами.

На стороне участника могут быть развернуты следующие компоненты:

- **PSI server** — компонент сверки идентификаторов;
- **VFL server** — серверный компонент взаимодействия для обучения/скоринга;
- **FileStorage** — каталог или файловое хранилище для входных/выходных файлов.

2.2. Что задаётся в ПО и как это задаётся

В ПО задаются **не абстрактные роли**, а **конкретные параметры запуска**.

В рамках настройки задаются:

1. роль узла;
2. сетевой адрес и порт взаимодействия;
3. рабочий каталог сервиса;
4. входные файлы идентификаторов;
5. входные файлы признаков;
6. параметры PSI;
7. параметры обучения модели;
8. пути к выходным результатам.

Все указанные параметры задаются:

- через аргументы командной строки;
- через каталоги размещения файлов;
- через команды запуска Docker-контейнеров и Python-скриптов.

2.3. Общий порядок выполнения процесса

Работа с ПО выполняется в следующем порядке:

1. Подготовить защищённый канал связи между сторонами;
2. Передать Docker-образ и вспомогательные файлы;
3. Развернуть серверный компонент на стороне участника;
4. Подготовить идентификаторы для PSI;
5. Выполнить PSI-сверку;
6. Подготовить обучающие признаки;
7. Запустить обучение модели;
8. Подготовить данные для исполнения модели;
9. Запустить предсказание;
10. Получить результаты и проверить журналы.

2.4. Разграничение ответственности сторон

Координатор выполняет:

- запуск PSI;
- запуск обучения;
- запуск скоринга;
- управление параметрами модели;
- агрегацию вычислительных результатов;
- получение итоговых результатов.

Партнёр данных выполняет:

- размещение локальных идентификаторов;
- размещение локальных признаков;
- запуск своей серверной части;
- участие в PSI;

- выполнение локальных вычислений на своей стороне.

3. Подготовка окружения

3.1. Требования к инфраструктуре

Для эксплуатации ПО требуется:

- ОС семейства Linux;
- установленный Docker;
- установленный Python 3;
- сетевой доступ между сторонами через защищённый канал;
- доступ к файловой системе для размещения данных.

3.2. Требования к каталогам

Перед запуском компонентов необходимо создать рабочие директории.

Пример:

```
sudo mkdir -p /data/vfl_server
sudo mkdir -p /data/vfl_server/input
sudo mkdir -p /data/vfl_server/output
sudo mkdir -p /data/vfl_server/models
sudo mkdir -p /data/vfl_server/logs
sudo chmod -R 755 /data/vfl_server
```

Назначение каталогов

Каталог	Назначение
`/data/vfl_server/input`	входные файлы идентификаторов и признаков
`/data/vfl_server/output`	результаты PSI, предсказаний и служебных файлов
`/data/vfl_server/models`	модели и артефакты обучения
`/data/vfl_server/logs`	журналы выполнения

3.3. Подготовка каналов связи

Связь между сторонами должна быть организована через:

- VPN;
- выделенный защищённый канал;
- внутреннюю закрытую сеть.

Перед запуском необходимо проверить доступность целевых узлов.

Пример проверки:

```
ping <адрес_удалённого_узла>
nc -zv <адрес_удалённого_узла> 5672
```

3.4. Передача артефактов установки

На сторону участника передаются:

- Docker-образ серверного компонента;
- Python-клиенты запуска PSI / обучения / предсказания;
- библиотека или модуль хэширования (если поставляется отдельно);
- конфигурационные файлы (если предусмотрены поставкой).

Пример копирования файлов:

```
scp vfl_server-1.0.0.tgz user@remote-host:/tmp/
scp psi_client.py user@remote-host:/opt/digitrack/
scp training_client.py user@remote-host:/opt/digitrack/
scp prediction_client.py user@remote-host:/opt/digitrack/
```

3.5. Проверка целостности файлов

После передачи файлов необходимо проверить их целостность.

Пример:

```
sha512sum vfl_server-1.0.0.tgz
sha512sum psi_client.py
sha512sum training_client.py
sha512sum prediction_client.py
```

Полученные контрольные суммы должны совпадать с эталонными значениями, переданными отдельно.

4. Развёртывание серверного компонента VFL

4.1. Загрузка Docker-образа

Если образ поставляется в архиве `.tgz`, его необходимо загрузить в локальный Docker.

Команда:

```
gunzip -c vfl_server-1.0.0.tgz | docker load
```

Что делает команда

- `gunzip -c` — распаковывает архив в поток вывода;
- `docker load` — загружает Docker-образ в локальный реестр Docker.

4.2. Проверка наличия образа

После загрузки необходимо убедиться, что образ доступен.

Команда:

```
docker images | grep vfl_server
```

Ожидаемый результат — наличие строки вида:

```
vfl_server   1.0.0   <IMAGE_ID>   <DATE>   <SIZE>
```

4.3. Подготовка рабочих директорий

Если директории ещё не созданы, необходимо выполнить:

```
mkdir -p /data/vfl_server/input
mkdir -p /data/vfl_server/output
mkdir -p /data/vfl_server/models
mkdir -p /data/vfl_server/logs
```

4.4. Запуск серверного контейнера

Базовая команда запуска

```
docker run -d \
  --restart always \
  --name vfl_server \
  -v /data/vfl_server:/root/vfl_server/ \
  --network host \
  vfl_server:1.0.0 \
  /vfl/python/server.py 127.0.0.1 5672 /data/vfl_server 1
```

4.5. Расшифровка команды запуска

Часть Docker

Фрагмент команды	Значение
`docker run -d`	запуск контейнера в фоновом режиме
`--restart always`	автоматический перезапуск контейнера
`--name vfl_server`	имя контейнера

Фрагмент команды	Значение
`-v /data/vfl_server:/root/vfl_server/`	монтирование рабочей директории хоста в контейнер
`--network host`	использование сетевого стека хоста

Часть запуска приложения

```
/vfl/python/server.py 127.0.0.1 5672 /data/vfl_server 1
```

Эта строка передаёт параметры в серверный скрипт.

Параметр	Назначение
`127.0.0.1`	адрес, на котором работает сервис или адрес взаимодействия
`5672`	порт взаимодействия
`/data/vfl_server`	рабочая директория, где находятся данные и результаты
`1`	идентификатор или режим/роль узла в вычислительном процессе

В документации поставки рекомендуется отдельно зафиксировать, что означает значение `1` в конкретной поставке: **роль активной стороны / координатора / участника**.

4.6. Проверка запуска контейнера

Проверка факта запуска:

```
docker ps | grep vfl_server
```

Если контейнер запущен, будет отображена строка с его идентификатором и именем.

4.7. Просмотр логов контейнера

Для контроля ошибок и статусов:

```
docker logs vfl_server
```

Для просмотра логов в реальном времени:

```
docker logs -f vfl_server
```

5. Подготовка данных

5.1. Подготовка идентификаторов для PSI

Для PSI подготавливается файл, содержащий идентификаторы объектов, которые необходимо сопоставить между сторонами.

ЭТАП 1: Загрузка исходных данных

Описание этапа: На этом этапе мы загружаем исходный файл, который содержит:

- **Поля для идентификации (ID)** - столбцы, которые однозначно идентифицируют запись
- **Признаки (фичи)** - дополнительные данные, связанные с каждой записью по ID

Что происходит:

- Загружается CSV файл с данными
- Проверяется структура данных
- Определяются столбцы с идентификаторами и признаками

Требования к файлу

- один идентификатор — одна строка;
- файл в текстовом формате;
- кодировка UTF-8;
- без заголовка, если иное не предусмотрено поставкой.

Пример файла

Файл: /data/vfl_server/input/ids.txt

	first_name	last_name	phone	passport_series	passport_number	age	income
0	Anna	Ivanov	75172071727	8765	199796	42	179992

Допустимы идентификаторы следующих типов:

- внутренний ID клиента;
- эл. почта;
- телефон;
- иной согласованный идентификатор.

ЭТАП 2: Объединение идентификаторов

Описание этапа: На этом этапе мы определяем, какие столбцы будут использоваться для идентификации записей, и объединяем их в единый столбец.

Что происходит:

- Партеры данных и Координатор совместно выбирают столбцы, которые формируют уникальный идентификатор
  (например: `phone`, `passport_series`, `passport_number`)
- Значения из этих столбцов объединяются через разделитель `|` в одну строку
- Создается промежуточный столбец с объединенными идентификаторами

Пример:

- Исходные
  данные: `phone=75172071727`, `passport_series=8765`, `passport_number=199796`
- Результат: `75172071727|8765|199796`

ЭТАП 3: Хеширование и подготовка к PSI

Описание этапа: На этом этапе мы хешируем объединенный столбец идентификаторов с использованием криптографической функции SHA-512 для обеспечения безопасности данных.

Что происходит:

- Каждая строка с объединенными идентификаторами преобразуется в хеш (SHA-512)
- Создается столбец только с хешами
- Хеши сохраняются в отдельный CSV файл для отправки на пересечение PSI

Важно:

- Хеширование является односторонней функцией - невозможно восстановить исходные данные из хеша
- Это обеспечивает безопасность при передаче данных для PSI
- Файл с хешами можно безопасно передать другой стороне для выполнения пересечения

Результат: Файл `hashes_only.csv` с одним столбцом хешей, готовый для отправки на PSI.

- row_hashes = flat_rows.apply(lambda x: hashlib.sha512(x.encode("utf-8")).hexdigest())
- # Выводим только столбец хешей
- print("=== ЭТАП 2: Столбец только с хешами ===")
- print(row_hashes)
- print()

- # Сохраняем столбец хешей в CSV файл
- hashes_df = pd.DataFrame({'hash': row_hashes})
- hashes_df.to_csv("hashes_only.csv", index=False)

- print("✅ Файл с одной колонкой хешей сохранен как 'hashes_only.csv'")
- print()

6. Настройка и запуск PSI

6.1. Назначение PSI

Процедура **PSI (Private Set Intersection)** предназначена для определения пересечения записей между сторонами **без передачи полного списка исходных идентификаторов в открытом виде**.

На выходе PSI получается множество **общих объектов**, по которым далее выполняется обучение или предсказание.

6.2. Что именно задаётся при запуске PSI

При запуске PSI в ПО задаются:

- локальный входной файл идентификаторов;
- адрес удалённого PSI-сервиса;
- при необходимости — дополнительные параметры разбиения и обработки.

6.3. Подготовка входного файла

Пример размещения файла:

```
cp ids.txt /data/vfl_server/input/ids.txt
ls -l /data/vfl_server/input/ids.txt
```

6.4. Команда запуска PSI

Переход в директорию приложения:

```
cd /opt/digitrack
```

Запуск PSI:

```
python3 psi_client.py --input_file_name '/data/vfl_server/input/ids.txt' --
psi_address 10.10.10.20
```

6.5. Описание параметров команды PSI

Полная команда

```
python3 psi_client.py --input_file_name '/data/vfl_server/input/ids.txt' --
psi_address 10.10.10.20
```

Расшифровка параметров

Параметр	Значение
`python3 psi_client.py`	запуск клиентского скрипта PSI
`--input_file_name`	путь к локальному файлу идентификаторов
`'/data/vfl_server/input/ids.txt'`	конкретный входной файл
`--psi_address`	адрес PSI-сервиса удалённой стороны
`10.10.10.20`	пример IP-адреса удалённого сервиса

6.6. Просмотр всех доступных параметров PSI

```
python3 psi_client.py help
```

или, в зависимости от реализации:

```
python3 psi_client.py --help
```

6.7. Параметры PSI

Хэширование идентификаторов

Для операций над идентификаторами используется **SHA-512**.

Правила преобразования идентификатора

Перед хэшированием идентификатор:

1. приводится к строковому виду;
2. кодируется в UTF-8;
3. обрабатывается хэш-функцией.

6.8. Параметры обработки PSI

Параметр	Назначение	Значение по умолчанию
`batch_size`	количество идентификаторов в одной обрабатываемой части	1024

Параметр	Назначение	Значение по умолчанию
`max_retries`	число повторных попыток при ошибке передачи	3
`parallelism`	число параллельных потоков обработки	4

6.9. Результаты выполнения PSI

После выполнения PSI формируются файлы пересечения.

Пример расположения результатов

Сторона	Пример файла
Активная / координатор	`/data/vfl_server/output/<session_id>/intersected_keys.csv`
Пассивная / партнёр	`/data/vfl_server/output/<session_id>/intersections.csv`

6.10. Пример содержимого результата PSI

Описание этапа: На этом этапе мы загружаем результаты PSI (столбец хешей после обработки протоколом PSI) и выполняем пересечение с нашим большим файлом хешей.

Что происходит:

- Загружается файл InterSection.csv с результатами PSI (содержит только столбец hash)
- Выполняется пересечение (inner join) по столбцу hash между:
  - Исходным файлом со всеми признаками и хешами
  - Файлом результатов PSI
- Остаются только те записи, у которых хеш присутствует в обоих файлах
- Результат пересечения сохраняется в CSV файл

Результат:

- Файл merged_result.csv содержит только те записи из исходного файла, которые были найдены в результате PSI
- Сохраняются все исходные признаки (фичи) для найденных записей
- Данные готовы для дальнейшего анализа
- ```
  id
  100001
  100004
  100005
  ```
  ```

Это означает, что только эти идентификаторы присутствуют у обеих сторон и будут использованы в следующих этапах.

7. Настройка и запуск обучения модели (VFL)

7.1. Назначение этапа обучения

Этап обучения предназначен для построения распределённой модели машинного обучения по данным, находящимся у разных участников.

7.2. Подготовка признаков для обучения

На каждой стороне размещается локальный файл признаков.

Пример структуры файла признаков

Файл: `/data/vfl_server/input/train_features.csv`

```
id,feature_1,feature_2,feature_3
100001,0.14,21,1
100002,0.09,34,0
100003,0.88,12,1
```

Требования

- обязательное наличие поля идентификатора;
- формат CSV;
- согласованный тип идентификатора;
- отсутствие рассинхронизации с PSI-результатом.

7.3. Подготовка данных для предсказания

Для предсказания подготавливается аналогичный файл признаков.

Файл: `/data/vfl_server/input/predict_features.csv`

```
id,feature_1,feature_2,feature_3
200001,0.22,17,0
200002,0.41,29,1
200003,0.63,31,0
```

7.4. Требования к структуре файлов

Тип файла	Обязательное поле	Пример
Файл PSI	идентификатор	`100001`
Файл обучения	`id` + признаки	`100001,0.14,21,1`
Файл предсказания	`id` + признаки	`200001,0.22,17,0`

7.5. Что задаётся перед запуском обучения

Перед запуском обучения должны быть заданы:

- входные признаки;
- результат PSI (общий набор идентификаторов);
- рабочие каталоги;
- параметры модели SecureBoost.

7.6. Команда запуска обучения

Переход в директорию приложения:

```
cd /opt/digitrack
```

Запуск:

```
python3 training_client.py
```

7.7. Просмотр параметров запуска обучения

```
python3 training_client.py help
```

или:

```
python3 training_client.py --help
```

7.8. Что означает запуск обучения

Команда:

```
python3 training_client.py
```

инициирует:

1. загрузку входных признаков;
2. загрузку пересечения PSI;

3. согласование объектов между сторонами;
4. локальные вычисления на каждой стороне;
5. обмен промежуточными вычислительными параметрами;
6. агрегацию результатов;
7. итеративное обновление модели.

7.9. Параметры модели SecureBoost

Ниже приведены основные параметры, которые должны быть заданы в конфигурации запуска обучения или соответствующем файле параметров.

Параметр	Тип	Значение
`num_trees`	int	количество деревьев
`max_depth`	int	максимальная глубина дерева
`learning_rate`	float	скорость обучения
`objective`	string	функция потерь(оптимизации)
`task_type`	string	тип задачи
`min_child_weight`	float	Минимальный вес дочернего узла
`subsample`	float	доля случайных образцов
`colsample_bytree`	float	доля признаков для дерева
`random_state`	int	фиксация уровня случайности

7.10 Подробное описание параметров обучения

`num_trees`

Количество деревьев в ансамбле.

Пример:

```
num_trees = 100
```

Чем больше значение, тем:

- выше потенциальное качество;
- дольше обучение.

`max_depth`

Максимальная глубина дерева.

Пример:

```
max_depth = 3
```

Влияет на:

- сложность модели;
- вероятность переобучения.

`learning_rate`

Скорость обновления модели на каждой итерации.

Пример:

```
learning_rate = 0.1
```

Меньшее значение:

- делает обучение стабильнее;
- увеличивает длительность обучения.

`objective`

Функция потерь(оптимизации).

Примеры:

```
objective = binary:bce
objective = regression:mse
```

Значение выбирается в зависимости от типа задачи:

- бинарная классификация;
- регрессия.

`task_type`

Тип задачи.

Пример:

```
task_type = classification
```

Возможные значения:

- `classification`
- `regression`

---

min_child_weight

Минимальный вес дочернего узла.

Пример:

```
min_child_weight = 1.0
```

Используется для ограничения слишком мелких разбиений и борьбы с переобучением.

---

subsample

Доля объектов, используемых при построении каждого дерева.

Пример:

```
subsample = 0.8
```

---

colsample_bytree

Доля признаков, используемых при построении дерева.

Пример:

```
colsample_bytree = 0.8
```

---

random_state

Фиксирует генератор случайных чисел.

Пример:

```
random_state = 42
```

Нужен для воспроизводимости результатов.

7.11. Результаты выполнения обучения

После завершения обучения формируются артефакты модели.

Пример расположения

```
/data/vfl_server/models/
```

Возможный результат

```
model.pkl
```

или набор файлов модели, распределённых между сторонами.

8. Настройка и запуск исполнения модели (скоринг)

8.1. Назначение этапа предсказания

Скоринг предназначен для расчёта прогнозных значений на новых объектах без раскрытия исходных признаков между сторонами.

8.2. Подготовка входных данных

До запуска предсказания должен быть размещён файл с признаками.

Пример:

```
cp predict_features.csv /data/vfl_server/input/predict_features.csv
```

Проверка:

```
ls -l /data/vfl_server/input/predict_features.csv
```

8.3. Команда запуска предсказания

Переход в директорию приложения:

```
cd /opt/digitrack
```

Запуск:

```
python3 prediction_client.py
```

8.4. Просмотр параметров запуска предсказания

```
python3 prediction_client.py help
```

или:

```
python3 prediction_client.py --help
```

8.5. Что делает команда предсказания

Команда:

```
python3 prediction_client.py
```

выполняет:

1. загрузку входных признаков;
2. локальные вычисления на стороне участников;
3. обмен промежуточными параметрами;
4. агрегацию вычислений координатором;
5. формирование итогового результата предсказания.

8.6. Результаты выполнения предсказания

Результат сохраняется в файл.

Пример расположения

```
/data/vfl_server/output/predictions.csv
```

Пример содержимого

```
id,prediction
200001,0.812
200002,0.114
200003,0.653
```

9. Журналирование и контроль выполнения

9.1. Какие журналы формируются

В ходе работы ПО формируются:

- журналы запуска контейнера;
- журналы PSI;
- журналы обучения;
- журналы предсказания;
- служебные сообщения об ошибках и исключениях.

9.2. Где искать результаты

Этап	Каталог
PSI	`/data/vfl_server/output/<session_id>/`
Обучение	`/data/vfl_server/models/`
Предсказание	`/data/vfl_server/output/`
Логи контейнера	`docker logs vfl_server`
Логи приложения	`/data/vfl_server/logs/`

9.3. Какие статусы контролировать

При эксплуатации необходимо контролировать:

- факт запуска контейнера;
- доступность сетевого адреса удалённого узла;
- наличие выходного файла PSI;
- наличие файлов модели;
- наличие файла `predictions.csv`;
- отсутствие ошибок в логах.

10. Администрирование и устранение неполадок

10.1. Типовые ошибки PSI

Ошибка: удалённый PSI-сервис недоступен

Проверка:

```
ping <psi_address>
nc -zv <psi_address> 5672
```

Возможные причины:

- недоступен VPN;
- неверный IP-адрес;
- закрыт порт;
- удалённый сервис не запущен.

10.2. Типовые ошибки запуска контейнера

Проверка статуса контейнера

```
docker ps -a | grep vfl_server
```

Перезапуск контейнера

```
docker restart vfl_server
```

Удаление контейнера и повторный запуск

```
docker stop vfl_server
docker rm vfl_server
```

После этого контейнер запускается повторно штатной командой.

10.3. Типовые ошибки обучения

Возможные причины:

- отсутствует результат PSI;
- несогласованные идентификаторы;
- ошибка в структуре CSV-файлов;
- недостаточно прав на рабочие каталоги;
- ошибка параметров модели.

Проверки:

```
ls -l /data/vfl_server/output/
ls -l /data/vfl_server/models/
docker logs vfl_server
```

10.4. Типовые ошибки предсказания

Возможные причины:

- модель не была обучена;
- отсутствует файл входных признаков;
- неверный формат входного файла;
- недоступен удалённый узел.

Проверки:

```
ls -l /data/vfl_server/input/
ls -l /data/vfl_server/models/
docker logs vfl_server
```

10.5. Рекомендации по эксплуатации

Рекомендуется:

- запускать ПО только через защищённый канал связи;
- использовать отдельные каталоги для каждого запуска;
- хранить контрольные суммы передаваемых артефактов;
- сохранять журналы выполнения для аудита;
- не смешивать тестовые и продуктивные данные в одной директории;
- выполнять резервное копирование каталога моделей и журналов.

11. Результаты функционирования ПО

В результате штатной эксплуатации ПО формируются:

1. результат PSI-сверки — множество общих идентификаторов;
2. распределённая обученная модель;
3. файл предсказаний;
4. журналы выполнения процессов.

Приложение А. Минимальный эксплуатационный сценарий

Шаг 1. Создать каталоги

```
mkdir -p /data/vfl_server/input
mkdir -p /data/vfl_server/output
mkdir -p /data/vfl_server/models
mkdir -p /data/vfl_server/logs
```

Шаг 2. Загрузить Docker-образ

```
gunzip -c vfl_server-1.0.0.tgz | docker load
```

Шаг 3. Проверить образ

```
docker images | grep vfl_server
```

Шаг 4. Запустить контейнер

```
docker run -d \
  --restart always \
  --name vfl_server \
  -v /data/vfl_server:/root/vfl_server/ \
  --network host \
  vfl_server:1.0.0 \
  /vfl/python/server.py 127.0.0.1 5672 /data/vfl_server 1
```

Шаг 5. Подготовить файл идентификаторов

```
cp ids.txt /data/vfl_server/input/ids.txt
```

Шаг 6. Запустить PSI

```
cd /opt/digitrack
python3 psi_client.py --input_file_name '/data/vfl_server/input/ids.txt' --
psi_address 10.10.10.20
```

Шаг 7. Запустить обучение

```
cd /opt/digitrack
python3 training_client.py
```

Шаг 8. Запустить предсказание

```
cd /opt/digitrack
python3 prediction_client.py
```

Шаг 9. Проверить результаты

```
ls -l /data/vfl_server/output/
ls -l /data/vfl_server/models/
docker logs vfl_server
```