# Модуль 2. Работа с ClearML.

## В этом модуле

Вы изучите и примените на практике ClearML, один из самых популярных инструментов для решения задач MLOps.

## Содержание юнитов

|  |
| --- |
| 1. Установка и запуск |
| 1. Построение графиков |
| 1. Настройка и использование гиперпараметров |
| 1. Управление экспериментами |
| 1. Тюнинг модели, оптимизация гиперпараметров |
| 1. Создание и использование конвейеров |
| 1. Кэширование данных |

# Модуль 2. Юнит 1. Установка и запуск.

## Введение

В этом юните вы научитесь устанавливать и запускать ClearML.

## Содержание:

Основное назначение программного обеспечения ClearML состоит в управлении ML экспериментами, в том числе

* версионирование, хранение и предоставление по запросу модели ML,
* контроль метрик, гиперпараметров и результатов выполнения отдельных этапов проекта (артефакты),
* сравнение экспериментов, в том числе в удобном графическом виде,
* воспроизведение экспериментов, использование при необходимости кэширования для ускорения процесса,
* визуализация результатов.

Кроме того, ClearML предоставляет инструментарий для работы с данными, позволяющий

* хранить датасеты,
* настраивать конвейеры обработки данных.

При работе ClearML ведется автоматическое логирование всех действий, что облегчает диагностику неисправностей.

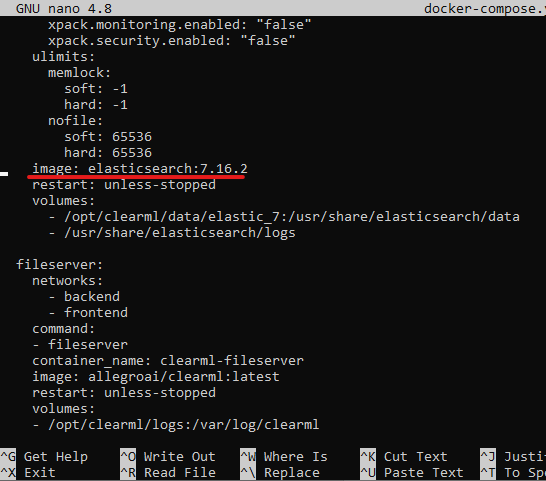
В небольших проектах, например образовательных, можно пользоваться бесплатной облачной версией ClearML. Для промышленных проектов есть версия ClearML, которую можно установить локально в инфраструктуре проекта. Далее в юнитах 1-7 данного модуля рассматривается вариант локальной установки ClearML. Для создания сервера ClearML необходимо выполнить команды

**sudo apt install docker.io docker-compose**

**mkdir /opt/clearml**

**curl** [**https://raw.githubusercontent.com/allegroai/clearml-server/master/docker/docker-compose.yml -o /opt/clearml/docker-compose.yml**](https://raw.githubusercontent.com/allegroai/clearml-server/master/docker/docker-compose.yml%20%20-o%20/opt/clearml/docker-compose.yml)

Обратите внимание, что в файле docker-compose.yml нужно изменить строку (как показано на фото) в противном случае образ elasticsearch будет скачиваться с запрещенного сайта.



Изображение 1

**docker-compose up -d**

После запуска дождитесь окончания запуска docker-контейнеров. После — выключите все контейнеры.

**docker-compose down**

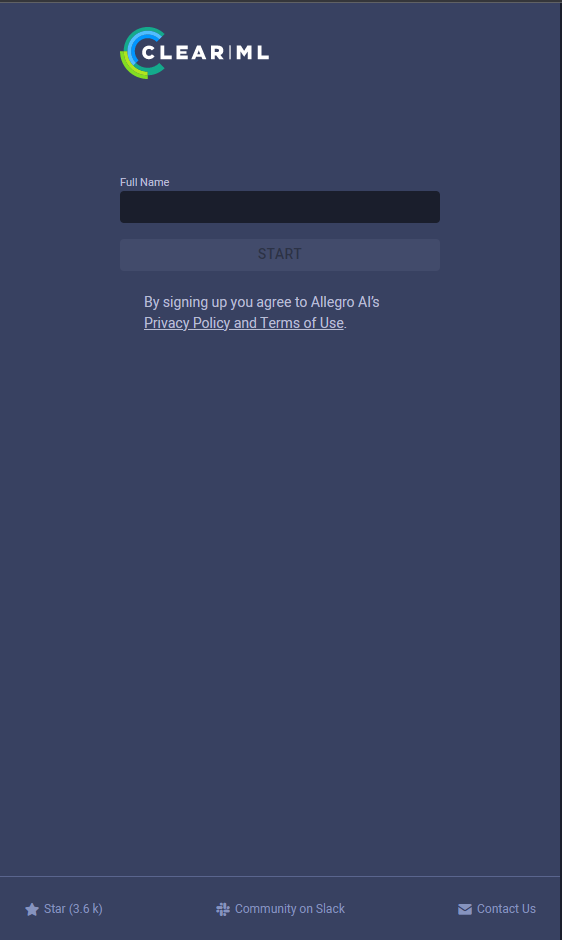
Это необходимо для того, чтобы выдать права на папки, иначе контейнер не сможет корректно запуститься.

**sudo chmod 777 data/ && sudo chmod 777 data**

**docker-compose up -d**

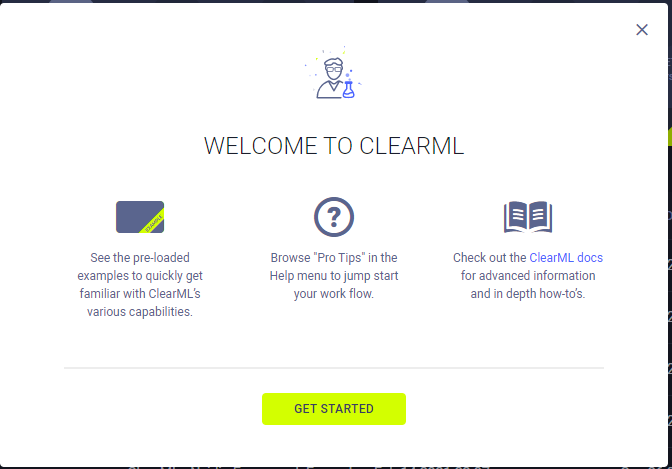
Вместо использования команды sudo лучше создать группу docker и выдавать соответствующие права участникам группы docker.

Теперь выполните авторизацию и сделайте основные настройки для работы с ClearML. Для этого перейдите на страницу [http://IP-Addres:8080](http://IP-Addres:8080/), после чего необходимо выполнить авторизацию



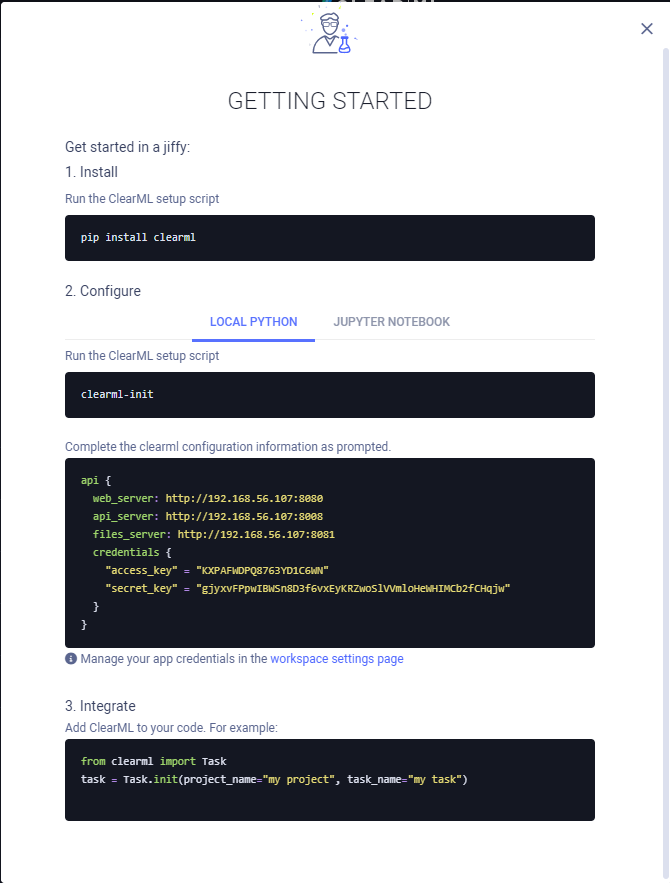
Изображение 2

После успешной авторизации вы увидите окно с приветствием.



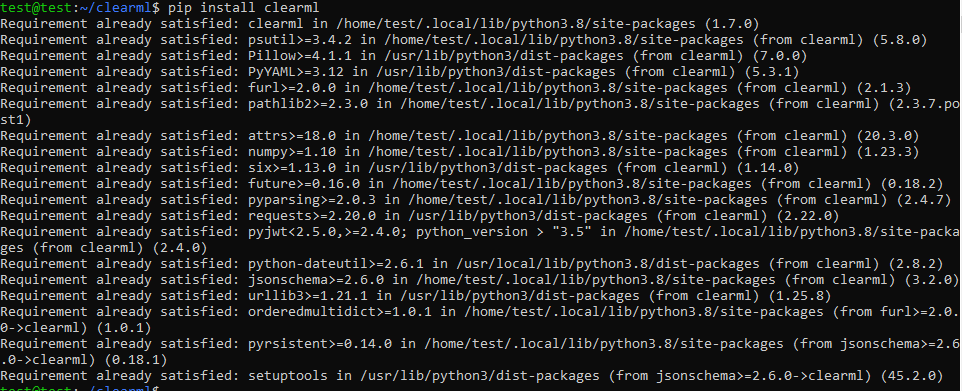
Изображение 3

Здесь необходимо нажать «Get Started», после чего вы будете перенаправлены на форму с инструкцией для создания подключения проекта с ClearML.



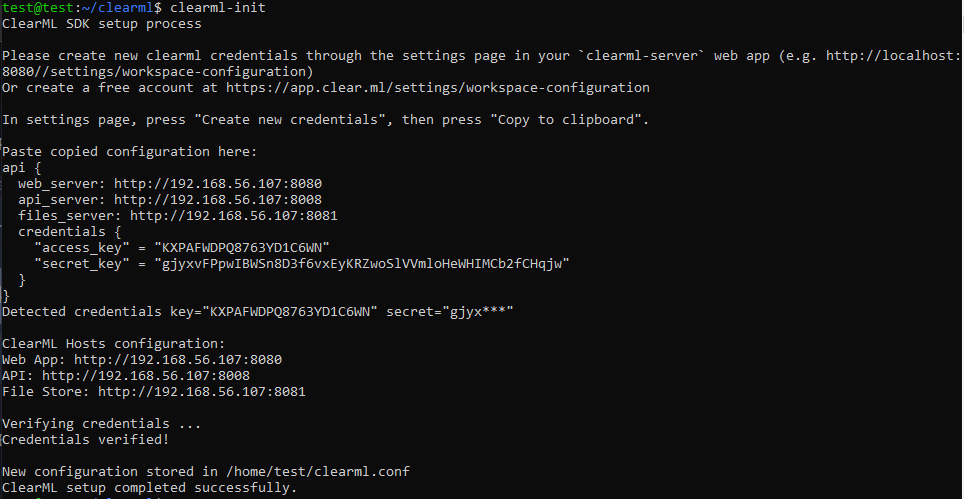
Изображение 4

Выполним последовательно шаги, указанные в этой инструкции. Сначала установим пакеты clearml с использованием установщика пакетов pip.



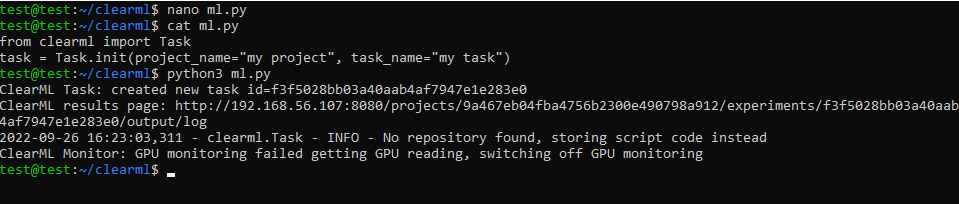
Изображение 5

Дальше выполните команду «clearml-init», введите данные для подключения и нажмите «Enter».



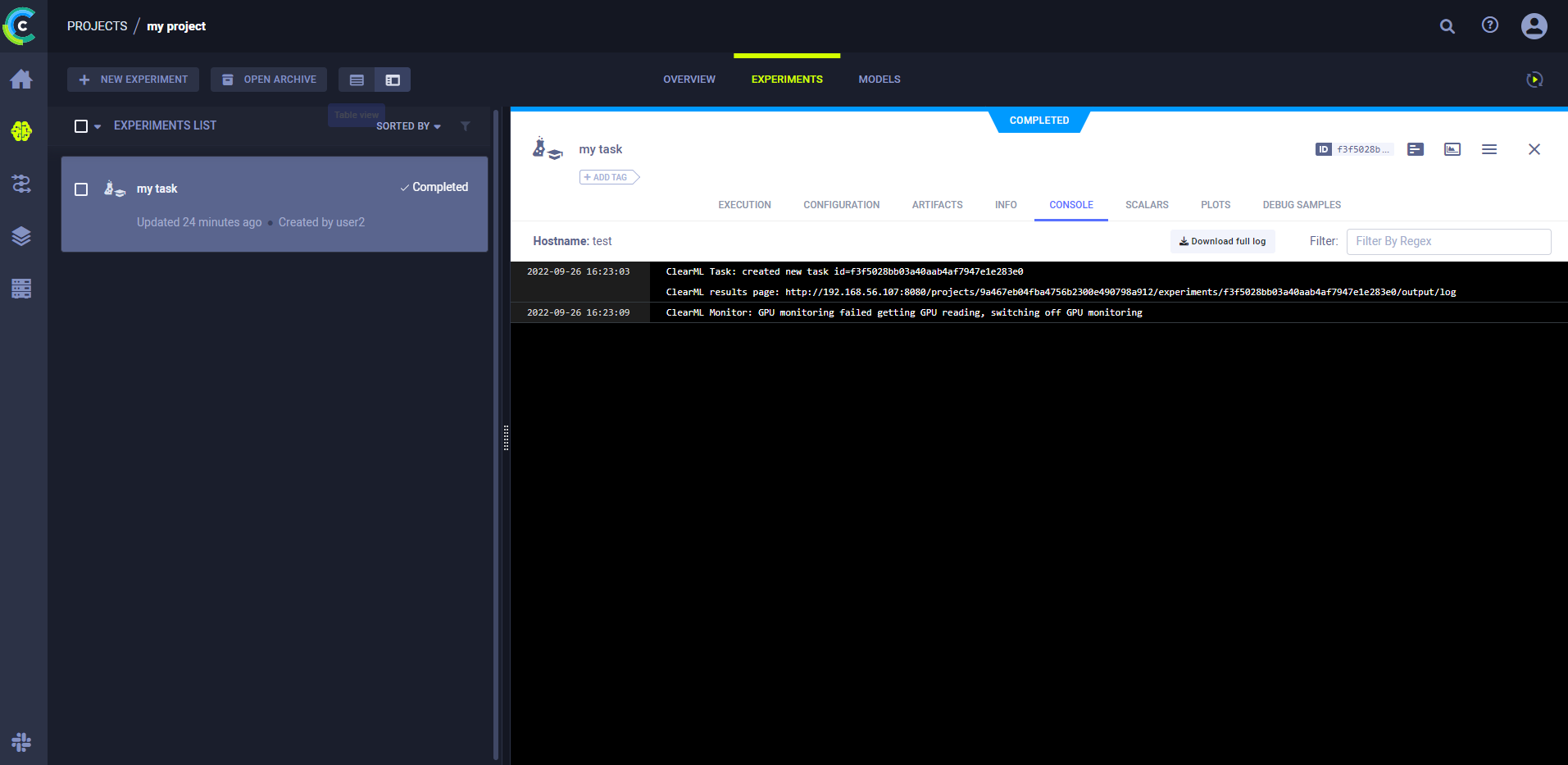
Изображение 6

Теперь все готово для того, чтобы запустить ваш первый скрипт, отображающийся в ClearMl.



Изображение 7

Скопируйте ссылку для просмотра результата работы нашей программы.



Изображение 8

В итоге ваша программа успешно скомпилировалась

## Тест

1. Какое средство для развертывания ClearML использовалось в юните? (0.25)
   1. dvc
   2. virtualbox
   3. kubernetes
   4. **docker**
2. Какой номер порта используется по умолчанию при первичном запуске ClearML для настройки? (0.25)
   1. **8080**
   2. 8000
   3. 21
   4. 80
3. Какая команда устанавливает clearml? (0.25)
   1. apt install clearml
   2. **pip install clearml**
   3. apt-get install clearml
   4. clearml.sh
4. Какая команда используется для инициализации настроек clearml? (0.25)
   1. clearml-start
   2. **clearml-init**
   3. clearml-begin
   4. clearml-clear

## Итоги/выводы

В этом юните вы научились устанавливать и настраивать ClearML.

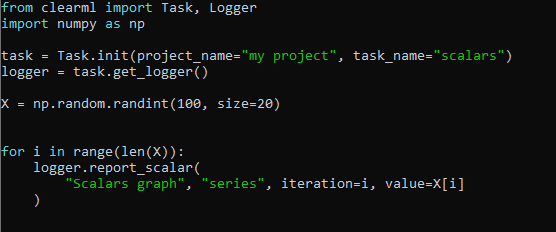
# Модуль 2. Юнит 2. Визуализация.

## Введение

Визуализация — важный инструмент на всех этапах проекта машинного обучения. В этом юните вы научитесь строить графики в ClearML.

## Содержание:

В ClearML есть специальные внутренние структуры, используемые для хранения и обработки данных в процессе выполнения программы, которые называются скаляры (scalars). Вот пример программного кода, который использует эти структуры.



Изображение 9

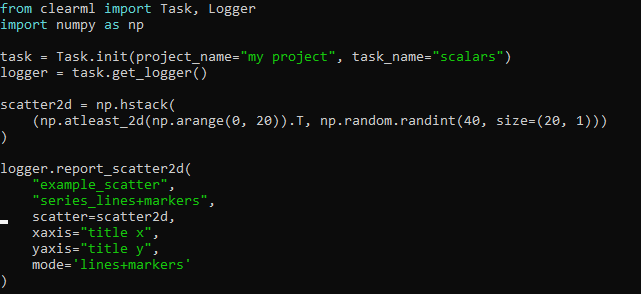
Для начала работы ClearML необходимо инициализировать задачу. Для этого импортируется класс Task, с помощью которого осуществляется инициализация. Этот код создает массив из 20 значений в диапазоне от 0 до 100. Далее с помощью цикла проходим по всем значениям и сохраняем их в специальный лог-файл с помощью импортированного класса Logger. Информацию, записанную в этот служебный файл, можно прочитать с помощью ClearML и построить график.



Изображение 10

Этот график можно открывать в рабочем окружении ClearML и анализировать информацию в реальном времени, например, в процессе обучения модели.

Давайте рассмотрим пример более сложного графика



Изображение 11

В этом примере создается двумерный массив со значениями «x» и «y». После этого в полученный массив передается в параметр «scatter» по которому будет строиться график.



Изображение 12

Общий подход к рисованию графиков в ClearML, иллюстрирующих процесс обучения, очень похож на принципы работы уже известного вам инструмента tensorboard. В процессе работы программа пишет данные в специальный лог-файл, а программа, создающая визуализации, читает с него данные и отрисовывает.

## Тест

1. Для чего в проекте машинного обучения могут быть использованы графики? (0.25)
   1. для максимально полного задействования GPU
   2. **для визуального контроля качества входных данных**
   3. **для визуализации полученных результатов**
   4. **для контроля процесса обучения модели**
2. Какой класс импортируется, чтобы инициализировать работу ClearML? (0.25)
   1. **from clearml import Task**
   2. from clearml import Base
   3. from clearml import Common
   4. from clearml import Setup
3. Какой класс используется для того, чтобы организовать логирование информации для ее последующей визуализации? (0.25)
   1. **from clearml import Logger**
   2. from clearml import Writer
   3. from clearml import Draw
   4. from clearml import Picture
4. Какая команда используется для инициализации настроек clearml? (0.25)
   1. clearml-start
   2. **clearml-init**
   3. clearml-begin
   4. clearml-clear

## Итоги/выводы

В этом юните вы научились устанавливать и настраивать ClearML.

# Модуль 2. Юнит 3. Настройка и использование гиперпараметров.

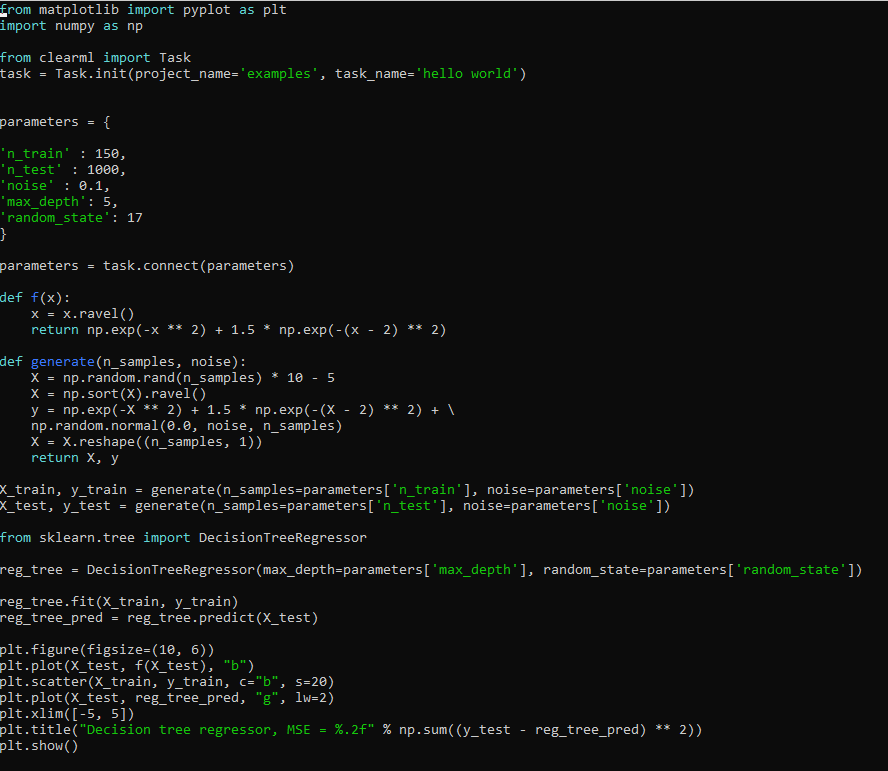
## Введение

В этом юните вы научитесь работать с гиперпараметрами для модели машинного обучения.

## Содержание

Гиперпараметры модели машинного обучения задаются перед началом обучения и затем не изменяются, в отличие от параметров, изменение которых и составляет суть машинного обучения. Удачное сочетание гиперпараметров может существенно ускорить обучение модели или повысить качество.

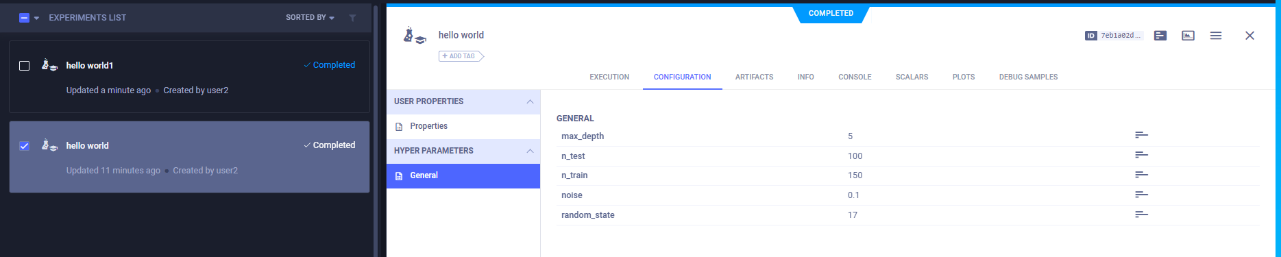
ClearML предоставляет удобный инструмент для работы с гиперпараметрами. Давайте рассмотрим его использование на простом практическом примере.



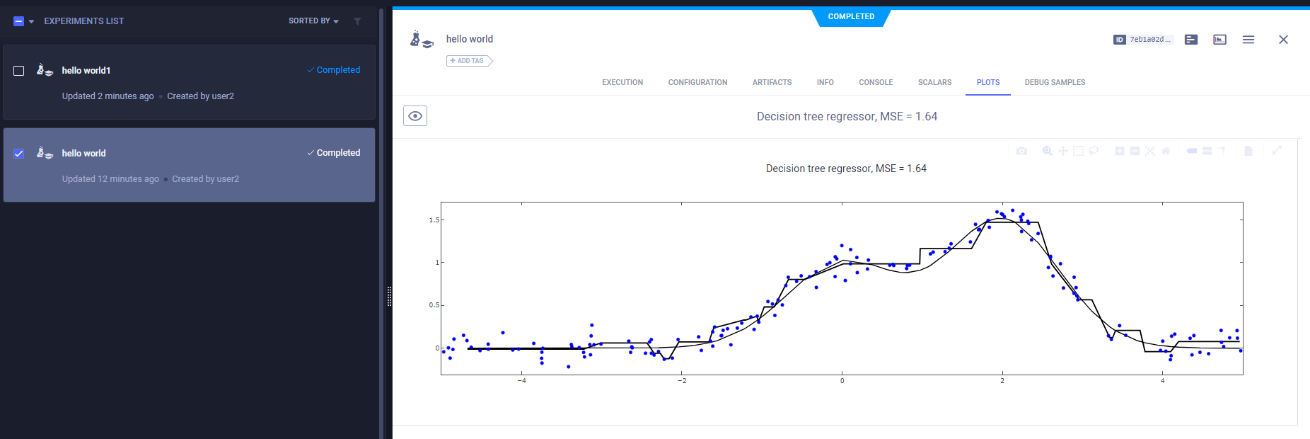
Изображение 13

В приведенном выше коде создается список parameters и в него заносятся гиперпараметры, которые будут в дальнейшем использоваться. Чтобы гиперпараметры сохранились, для повторяемости эксперимента используется функция «task.connect».

Еще в этом случае ClearML понимает, что графики строятся с помощью matplotlib и автоматически загружает в панель «Plots»



Изображени 14



Изображение 15

## Тест

1. Чем параметры модели машинного обучения отличаются от гиперпараметров? (0.25)
   1. Гиперпараметры требуют больше памяти для хранения информации
   2. Ничем не отличаются
   3. **Гиперпараметры устанавливаются до обучения и в процессе обучения не меняются, а параметры меняются в процессе обучения**
   4. Параметры устанавливаются до обучения и в процессе обучения не меняются, а гиперпараметры меняются в процессе обучения
2. Зачем меняют гиперпараметры? (0.25)
   1. **Для ускорения обучения модели**
   2. **Для борьбы с переобучением**
   3. **Для повышения точности работы модели**
   4. Для внесения элемента случайности в эксперимент
3. С помощью какой команды в рабочий скрипт добавляется набор параметров? (0.25)
   1. parameter.add()
   2. **task.connect()**
   3. parameter.set()
   4. task.init()
4. Какая python библиотека используется для рисования графиков в ClearML? (0.25)
   1. seaborn
   2. **matplotlib**
   3. dash
   4. paint

## Итоги/выводы

В этом модуле вы научились загружать и использовать гиперпараметры в проектах ClearML.

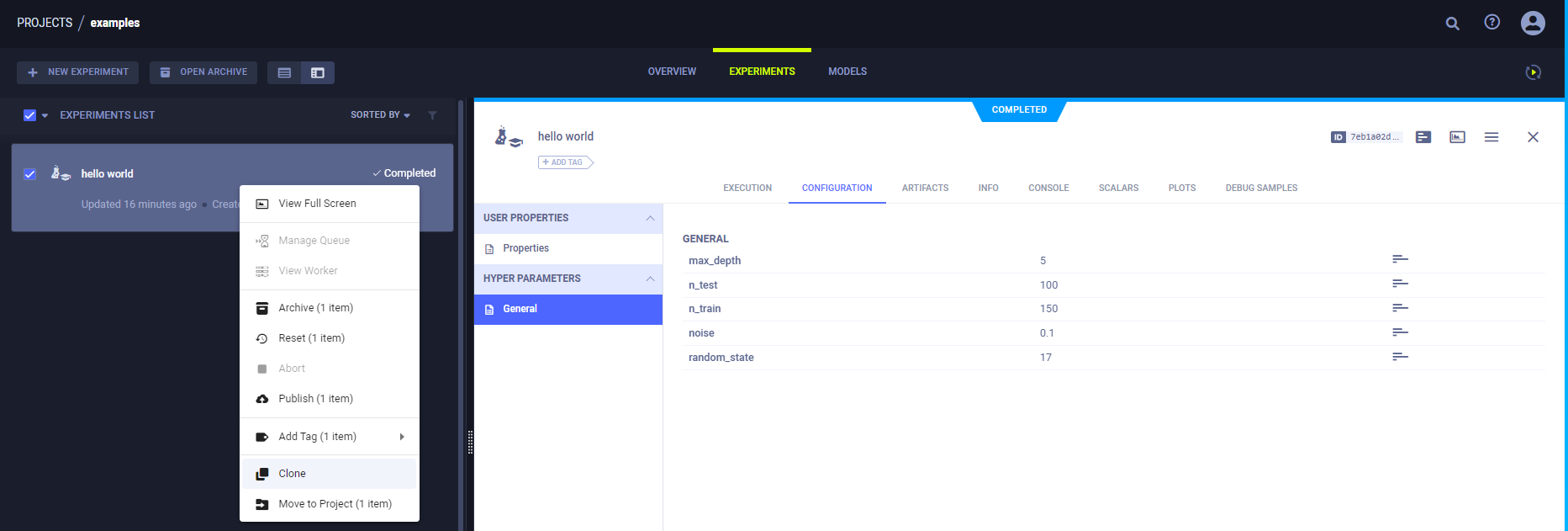
# Модуль 2. Юнит 4. Управление экспериментами.

## Введение

В этом юните вы настроите контроль и управление экспериментами в ClearML.

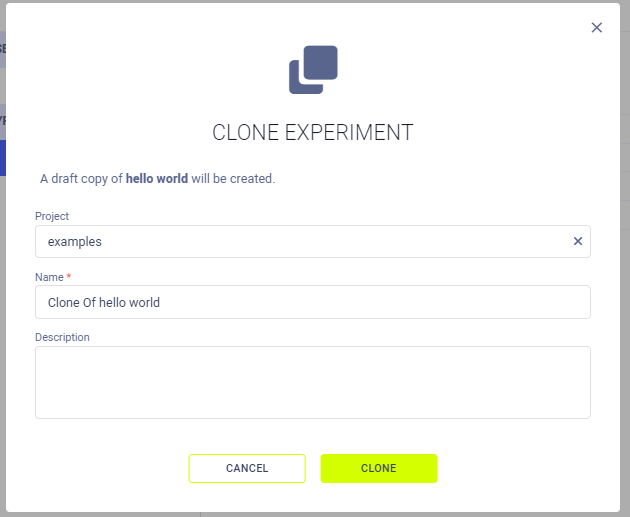
## Содержание

Повторение экспериментов с разными сочетаниями гиперпараметров, моделями, наборами признаков (features) — это основа машинного обучения. Кроме того, часто возникает ситуация, когда исследователь хочет повторить эксперимент другого участника проекта. Давайте научимся создавать и повторять эксперименты в ClearML. Для начала клонируем эксперимент, который хотим повторить.



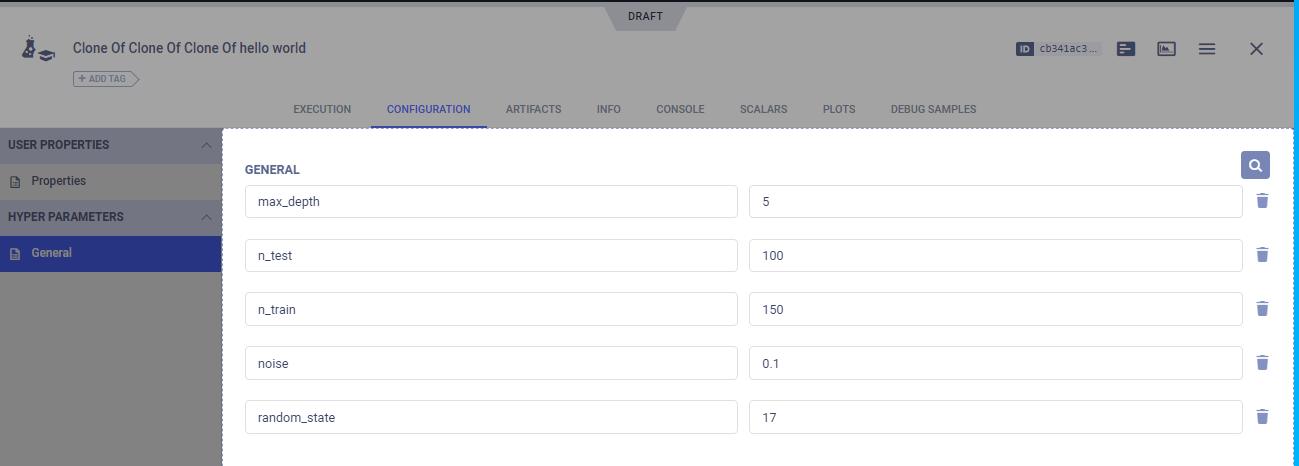
Изображение 16

Дальше вам будет предложено выбрать проект, в котором будет храниться эксперимент, а также задать новое имя для эксперимента. Кроме этого, можно задать описание для эксперимента.



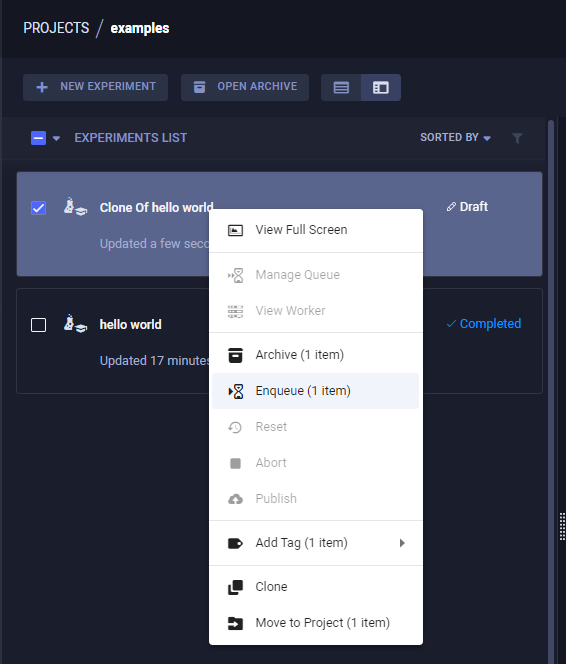
Изображение 17

Вы можете изменить гиперпараметры для создания нового эксперимента либо оставить те же данные, чтобы повторить эксперимент.



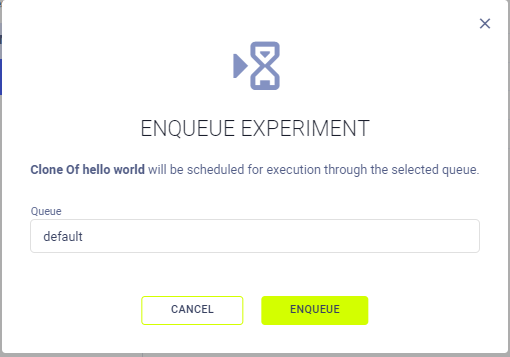
Изображение 18

После окончания подготовки эксперимента необходимо его запустить.



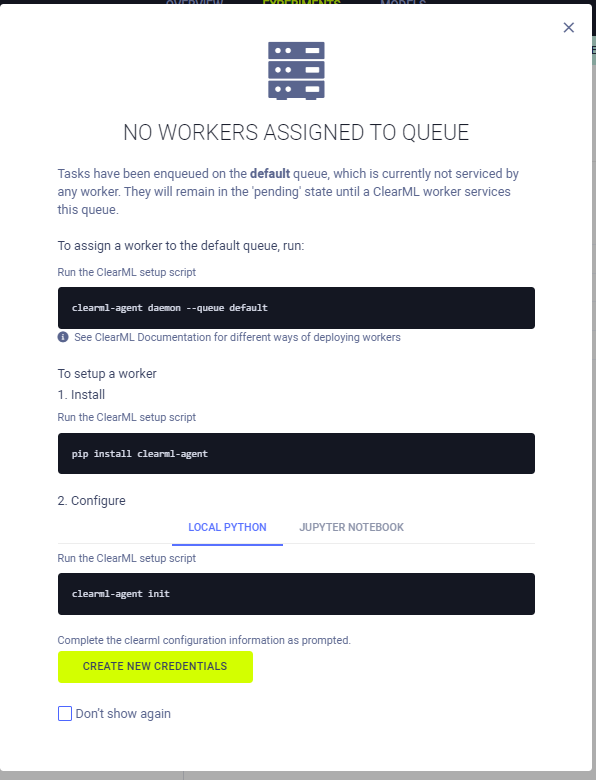
Изображение 19

Дальше вам будет предложено добавить эксперимент в очередь.



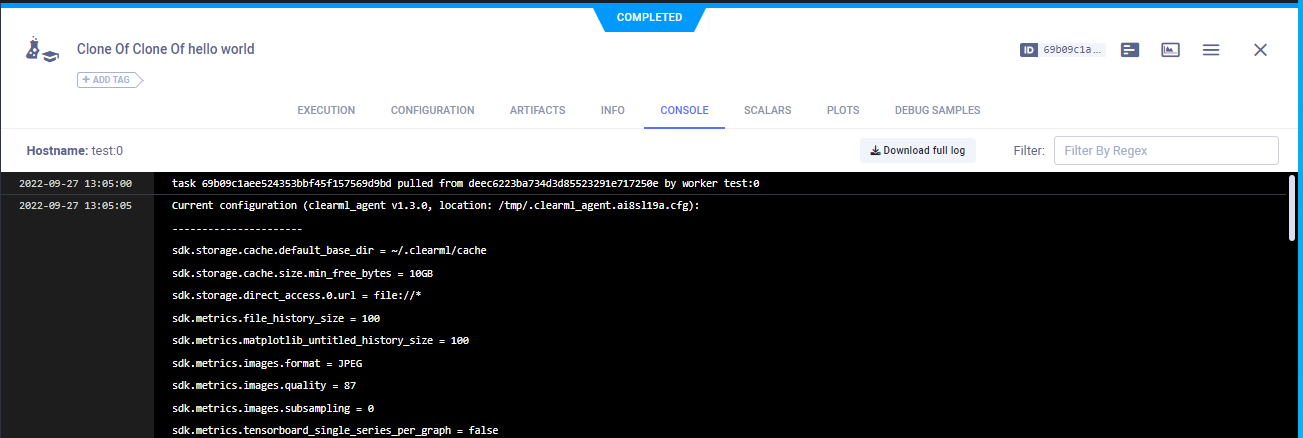
Изображение 20

Если приложение не имеет сервера для обработки очередей, вам придется самим создать его на своем компьютере или сервере. Для этого нужно пошагово выполнить инструкцию и после запуска агента (Agent) отправить эксперимент в очередь и ожидать его обработки.



Изображение 21

За процессом выполнения программы можно следить на вкладке «Console».



Изображение 22

## Тест

1. Что дает управление экспериментами в проектах машинного обучения? (0.25)
   1. **возможность для разных участников проекта обеспечить повторяемость (идентичность) эксперимента**
   2. контроль затрат на проекте
   3. **найти лучшее сочетание признаков и гиперпараметров для качества модели**
   4. уменьшение скорости обучения модели
2. Какая структура используется для выполнения задач в ClearML? (0.25)
   1. стек
   2. дек
   3. **очередь**
   4. двунаправленный список
3. Какая команда запускает выполнение рабочего ClearML скрипта? (0.25)
   1. clearml-task start «имя скрипта»
   2. **clearml-agent daemon --queue «имя очереди»**
   3. clearml-setup
   4. clearml-init-queue
4. На какой вкладке осуществляется контроль выполнения эксперимента? (0.25)
   1. execution
   2. **console**
   3. info
   4. scalars

## Итоги/выводы

В этом юните вы научились управлять экспериментами в проекте машинного обучения, повторять эксперимент и изменять гиперпараметры.

# Модуль 2. Юнит 5. Тюнинг модели. Оптимизация гиперпараметров.

## Введение

В этом юните вы научитесь делать оптимизацию (тюнинг) модели, в том числе выбирая оптимальные значения для гиперпараметров.

## Содержание

Давайте создадим скрипт, который генерирует синтетические данные и обучает модель, имеющую гиперпараметры. Пример кода, который реализует генерацию синтетического набора данных и модель машинного обучения DecisionTreeRegressor из библиотеки sklearn.

from matplotlib import pyplot as plt

import numpy as np

from clearml import Task, Logger

task = Task.init(project\_name='second\_test', task\_name='module use HP')

parameters = {

'n\_train' : 150,

'n\_test' : 1000,

'noise' : 0.1,

'max\_depth': 5,

'random\_state': 17

}

parameters = task.connect(parameters)

def f(x):

    x = x.ravel()

    return np.exp(-x \*\* 2) + 1.5 \* np.exp(-(x - 2) \*\* 2)

def generate(n\_samples, noise):

    X = np.random.rand(n\_samples) \* 10 - 5

    X = np.sort(X).ravel()

    y = np.exp(-X \*\* 2) + 1.5 \* np.exp(-(X - 2) \*\* 2) + \

    np.random.normal(0.0, noise, n\_samples)

    X = X.reshape((n\_samples, 1))

    return X, y

X\_train, y\_train = generate(n\_samples=parameters['n\_train'], noise=parameters['noise'])

X\_test, y\_test = generate(n\_samples=parameters['n\_test'], noise=parameters['noise'])

from sklearn.tree import DecisionTreeRegressor

reg\_tree=DecisionTreeRegressor(max\_depth=parameters['max\_depth'], random\_state=parameters['random\_state'])

reg\_tree.fit(X\_train, y\_train)

reg\_tree\_pred = reg\_tree.predict(X\_test)

plt.figure(figsize=(10, 6))

plt.plot(X\_test, f(X\_test), "b")

plt.scatter(X\_train, y\_train, c="b", s=20)

plt.plot(X\_test, reg\_tree\_pred, "g", lw=2)

plt.xlim([-5, 5])

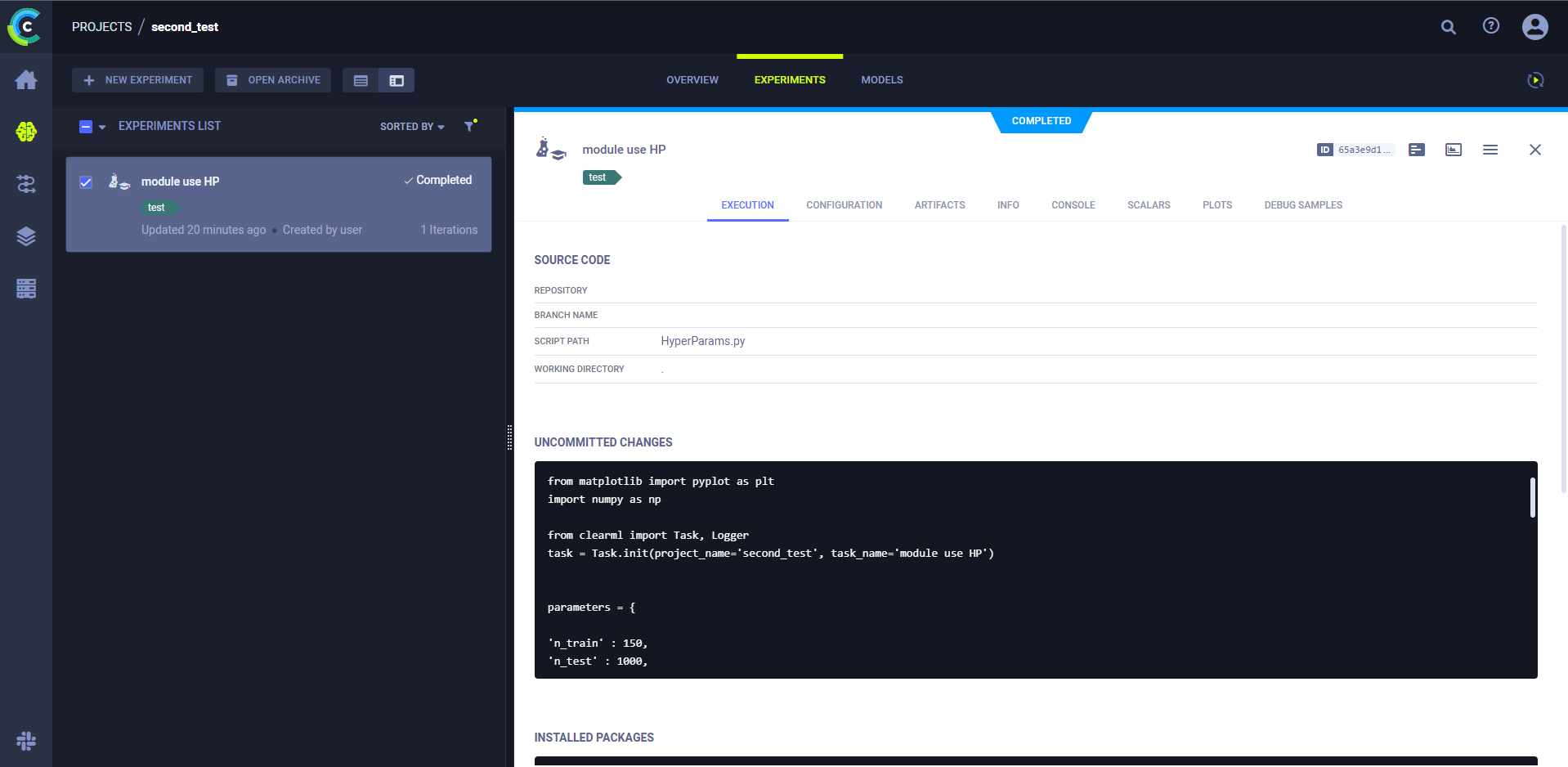
plt.title("Decision tree regressor, MSE = %.2f" % np.sum((y\_test - reg\_tree\_pred) \*\* 2))

plt.show()

MSE = float(np.sum((y\_test - reg\_tree\_pred) \*\* 2))

Logger.current\_logger().report\_scalar(title='accuracy', series='MSE', value=MSE, iteration=1)

Этот код можно запустить в среде ClearML, после чего он должен отобразиться в экспериментах в рабочем пространстве проекта.



Изображение 23

Чтобы оптимизировать гиперпараметры, необходимо разработать программу, которая используется для автоматической подстановки значений в заданных диапазонах.

from clearml import Task, Logger

from clearml.automation import (

    DiscreteParameterRange, HyperParameterOptimizer,

RandomSearch, UniformIntegerParameterRange)

task = Task.init(project\_name='second\_test',

                 task\_name='Automatic Hyper-Parameter Optimization',

                 task\_type=Task.TaskTypes.optimizer,

                 reuse\_last\_task\_id=False)

args = {

    'template\_task\_id': "416395f5297f42478b42849345c2adcd",

    'run\_as\_service': False,

}

an\_optimizer = HyperParameterOptimizer(

    base\_task\_id=args['template\_task\_id'],

    hyper\_parameters=[

UniformIntegerParameterRange('General/n\_train', min\_value=50, max\_value=200, step\_size=50),

   UniformIntegerParameterRange('General/n\_test', min\_value=500, max\_value=2000, step\_size=500),

  UniformIntegerParameterRange('General/max\_depth', min\_value=1, max\_value=10, step\_size=1),

    UniformIntegerParameterRange('General/random\_state', min\_value=17, max\_value=17, step\_size=1),

   ],

   objective\_metric\_title='accuracy',

   objective\_metric\_series='MSE',

   objective\_metric\_sign='min',

)

an\_optimizer.start()

top\_exp = an\_optimizer.get\_top\_experiments(top\_k=3)

print([t.id for t in top\_exp])

task.upload\_artifact('top\_exp', top\_exp)

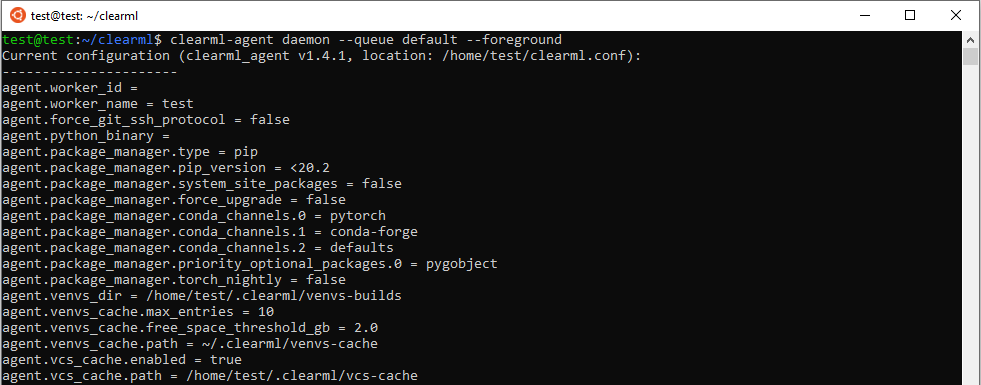
an\_optimizer.wait()

an\_optimizer.stop()

Аргументы и функции кода, необходимые для работы программы:

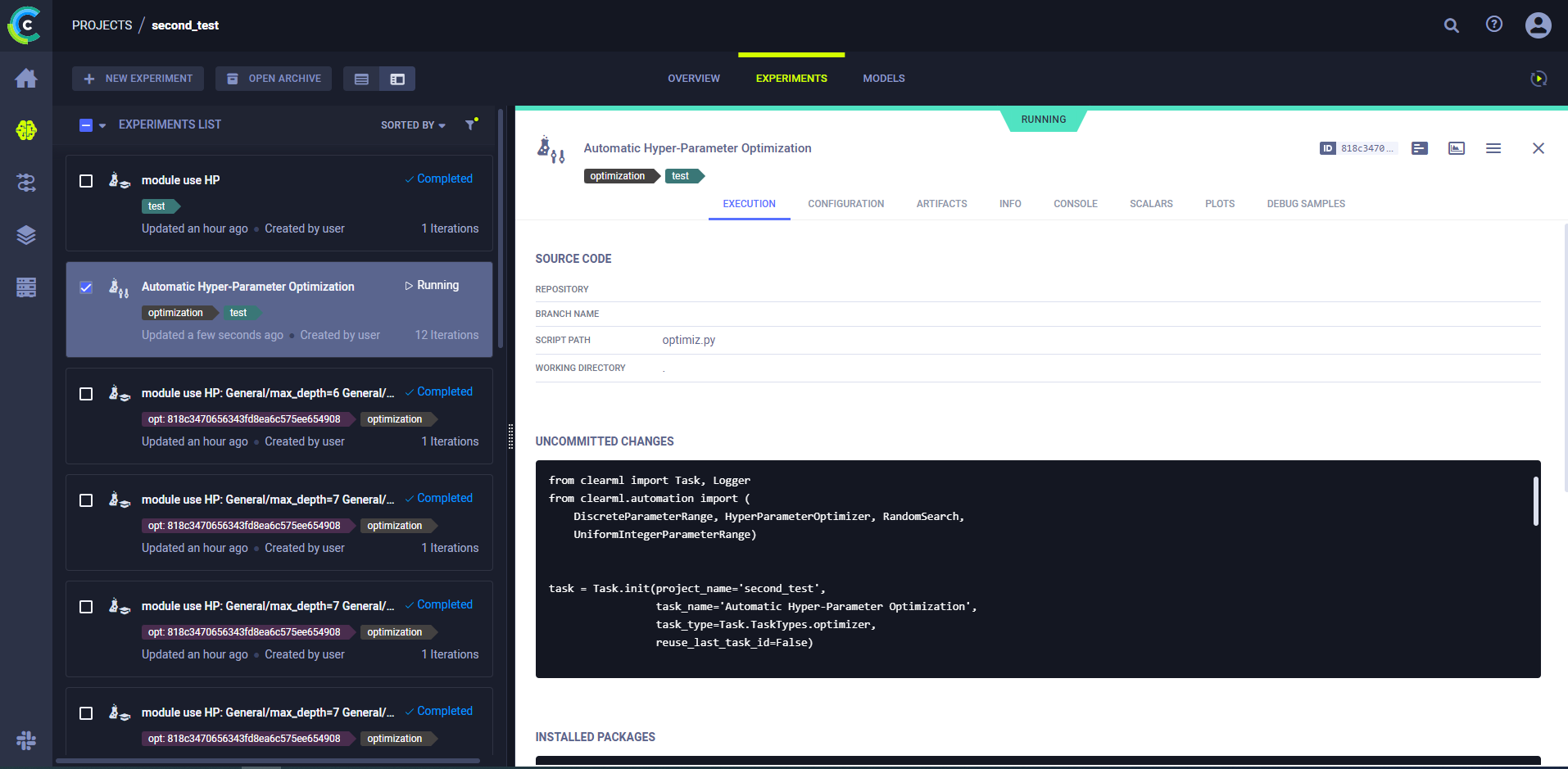
* аргумент «base\_task\_id» — необходим для выбора ID эксперимента который будет использоваться для повторяемости эксперимента с новыми параметрами,
* аргумент «parameters» — используется для передачи гиперпараметров которые необходимо будет изменять в процессе выполнения программы,
* функция «UniformIntegerParameterRange» — используется, чтобы задать диапазоны, в которых будут изменяться гиперпараметры, а также указывается шаг изменения,
* аргумент «objective\_metric\_title» — необходим, чтобы задать, какой график необходимо использовать, чтобы находить наименьшее значение,
* аргумент «objective\_metric\_series» — требуется, чтобы задать, какой из элементов мы будем отслеживать для минимального значения,
* аргумент «objective\_metric\_sign» — указывает, какое значение мы хотим находить (Макс./Мин.).

Перед запуском программного кода необходимо запустить «clearml-agent», так как все вычисления будут производиться через агента.



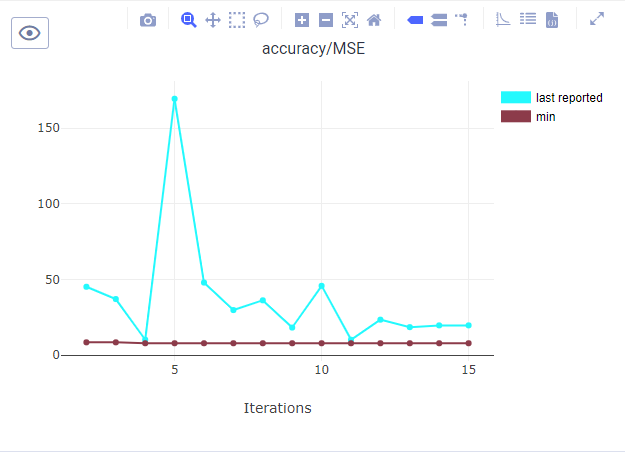
Изображение 24

После указания всех параметров запускаем программу, после чего в интерфейсе рабочего окружения ClearML появятся эксперименты с различными наборами данных.

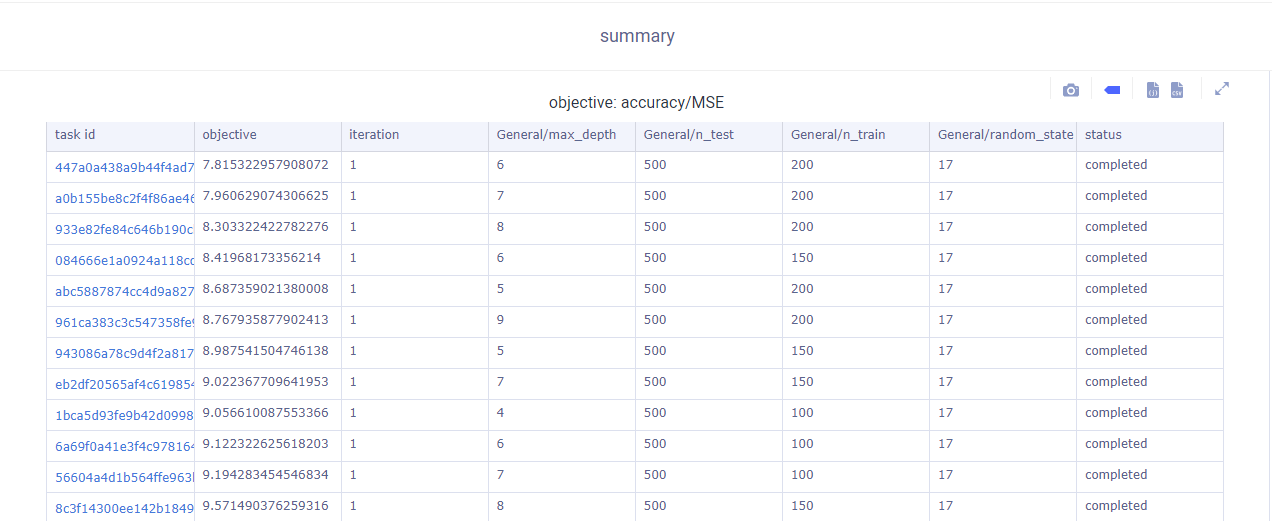


Изображение 25

Когда завершится выполнение программы, можно будет посмотреть итоговые графики и таблицу, отсортированную по возрастанию (зависит от параметра «objective\_metric\_sign»).



Изображение 26



Изображение 27

## Тест

1. Какая функция позволяет найти наилучшее значение гиперпараметра из диапазона? (0.25)
   1. clearml.autimation.HyperParameterRange
   2. clearml.autimation
   3. **clearml.automation.HyperParameterOptimizer**
   4. clearml.autimation
2. Какой параметр используется для задания ID эксперимента в оптимизаторе набора гиперпараметров? (0.25)
   1. experiment\_id
   2. **base\_task\_id**
   3. id
   4. setup\_id
3. Какой метод в классе HyperParameterOptimizer возвращает набор наилучших экспериментов? (0.25)
   1. get\_best\_experiments
   2. **get\_top\_experiments**
   3. get\_sorted\_experiments
   4. get\_fine\_experiments
4. Какой метод запускает работу оптимизатора гиперпараметров? (0.25)
   1. .run()
   2. .get()
   3. **.start()**
   4. .begin()

## Итоги/выводы

В этом юните вы осуществили улучшение (тюнинг) модели машинного обучения за счет подбора оптимальных значений гиперпараметров.

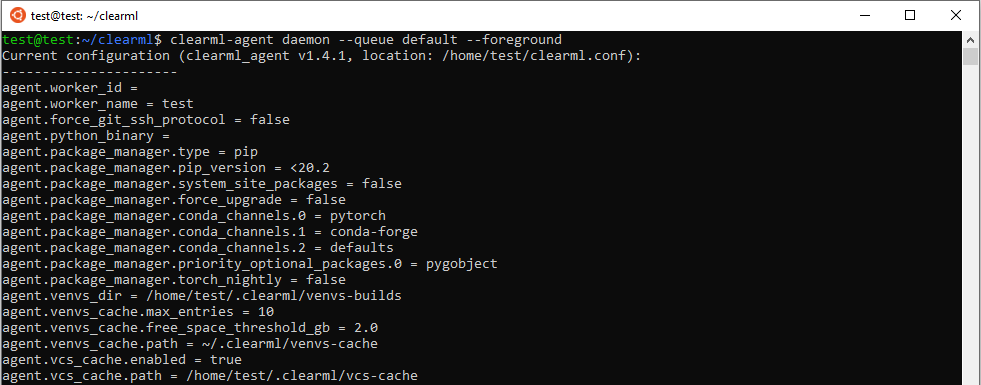
# Модуль 2. Юнит 6. Создание и использование конвейеров

## Введение

Конвейеры (пайплайны, pipelines) — это важная сущность в проекте машинного обучения, объединяя операции и результаты их выполнения в последовательность действий. В этом юните вы научитесь создавать и использовать конвейеры в ClearML.

## Содержание

Для использования конвейера (pipeline) изначально необходимо запустить агента ClearML (clearml-agent).



Изображение 28

Конвейер для ClearML описывается в скрипте python в виде последовательности функций. Для переопределения свойств функций и объединения их в конвейер в ClearML используется специальный декоратор

**@PipelineDecorator.component(cache=True, execution\_queue="default")**

Для создания учебного конвейера в ClearML давайте напишем скрипт python, включающий в себя следующие этапы

* step\_one(): создание синтетического набора данных, массива xs и ys,
* step\_two(): добавление новых признаков, xs\*\*3, xs\*\*3, xs\*\*5,
* step\_three(): разделение набора данных на учебную (train) и тестовую (test) выборки,
* step\_four(): создание и обучение модели линейной регрессии на тренировочных данных,
* step\_five(): скоринг модели линейной регрессии на тестовых данных,
* executing\_pipeline(): объединение всех этапов в единый конвейер и запуск выполнения конвейера

from clearml import PipelineDecorator

from clearml import Task, Logger

@PipelineDecorator.component(cache=True, execution\_queue="default")

def lib():

    from random import randint

    import numpy as np

    from sklearn.model\_selection import train\_test\_split

@PipelineDecorator.component(return\_values=['xs, ys'], execution\_queue="default")

def step\_one():

    from random import randint

    import numpy as np

    xs = np.linspace(0, 10, 50)

    ys = xs\*\*2 + np.random.random(50) \* 10

    return xs, ys

@PipelineDecorator.component(return\_values=['xs1'], execution\_queue="default")

def step\_two(xs: float, t: int = 0):

    import numpy as np

    if t == 0:

        xs1 = np.c\_[xs]

    if t == 1:

        xs1 = np.c\_[xs, xs\*\*2]

    if t == 2:

        xs1 = np.c\_[xs, xs\*\*2, xs\*\*3]

    if t == 3:

        xs1 = np.c\_[xs, xs\*\*2, xs\*\*3, xs\*\*4]

    if t == 4:

        xs1 = np.c\_[xs, xs\*\*2, xs\*\*3, xs\*\*4, xs\*\*5]

    return xs1

@PipelineDecorator.component(return\_values=['X\_train, X\_test, y\_train, y\_test'], execution\_queue="default")

def step\_three(xs1, ys):

    from sklearn.model\_selection import train\_test\_split

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(xs1, ys, test\_size=0.33, random\_state=42)

    return X\_train, X\_test, y\_train, y\_test

@PipelineDecorator.component(return\_values=['model'], execution\_queue="default")

def step\_four(X\_train, y\_train):

    from sklearn.linear\_model import LinearRegression

    model = LinearRegression()

    model.fit(X\_train, y\_train)

    return model

@PipelineDecorator.component(return\_values=['score'], execution\_queue="default")

def step\_five(model, X\_test, y\_test):

    from sklearn.linear\_model import LinearRegression

    score = model.score(X\_test, y\_test)

    return score

@PipelineDecorator.pipeline(name='test\_one', project='test\_one', version='0.0.1')

def executing\_pipeline():

    task = Task.init(project\_name="test\_one", task\_name="test\_one")

    logger = task.get\_logger()

    lib()

    xs, ys = step\_one()

    for i in range(5):

        xs1 = step\_two(xs, i)

        X\_train, X\_test, y\_train, y\_test = step\_three(xs1, ys)

        model = step\_four(X\_train, y\_train)

        logger.report\_scalar(

            "Metric", "series", iteration=i, value=step\_five(model, X\_test, y\_test)

        )

        print(step\_five(model, X\_test, y\_test))

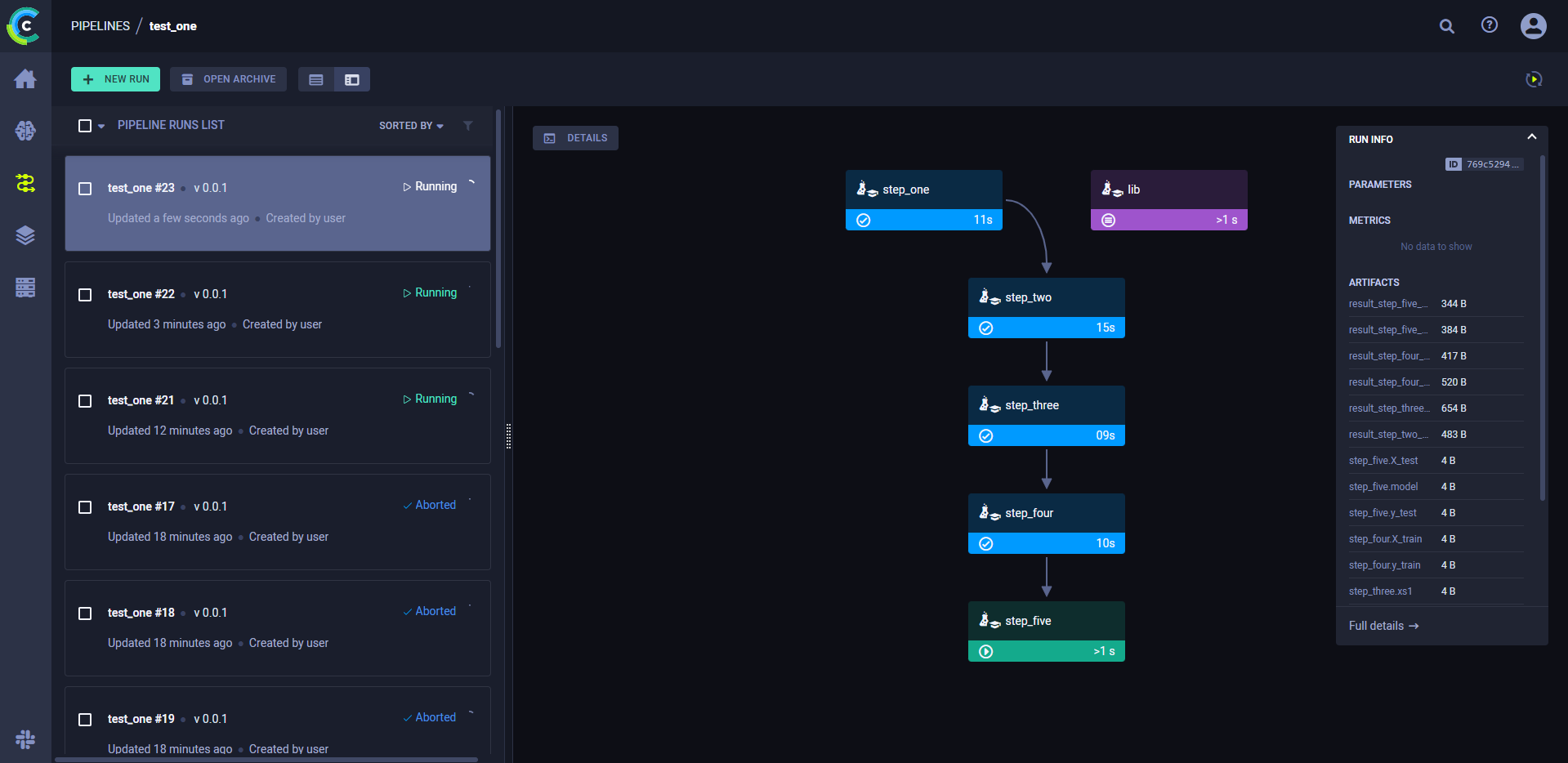
if \_\_name\_\_ == '\_\_main\_\_':

    PipelineDecorator.run\_locally()

    executing\_pipeline()

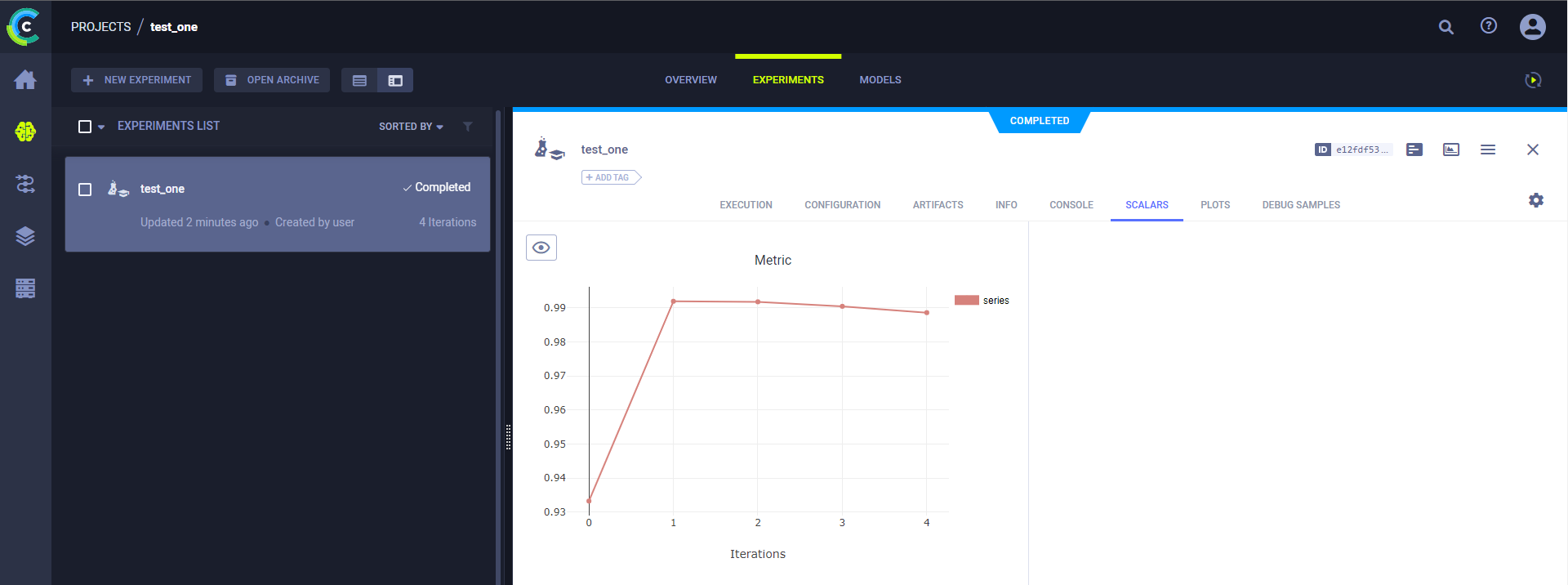
    PipelineDecorator.stop()

После запуска программы в интерфейсе рабочего окружения ClearML в разделе Pipeline появляется схема заданной вами последовательной обработки данных, которая была разделена на несколько шагов.



Изображение 29

После завершения работы кода можно перейти на страницу проекта, где вы увидите построенный по полученным значениям график.



Изображение 30

## Тест

1. Что такое декоратор в python? (0.25)
   1. утилита, улучшающая внешний вид и читаемость функций,
   2. специальный класс для скрытия информации о структуре объекта,
   3. **функция, меняющая свойства другой функции,**
   4. в python нет такой сущности
2. Какой декоратор используется для придания функции python свойств этапа конвейера эксперимента? (0.25)
   1. experiment\_id
   2. **PipelineDecorator()**
   3. id
   4. setup\_id
3. Какой параметр в декораторе конвейера задает возвращаемое значение отдельного этапа? (0.25)
   1. return\_artefacts=
   2. **return\_values=**
   3. return\_results=
   4. return\_data=
4. Какой метод инициализирует конвейер? (0.25)
   1. .run()
   2. .get()
   3. **.run\_locally()**
   4. .begin()

## Итоги/выводы

В этом юните вы научились создавать и использовать конвейеры.

# Модуль 2. Юнит 7. Кэширование.

## Введение

Кэширование необходимо для оптимизации проведения экспериментов, позволяет запоминать результаты отдельных этапов эксперимента и не выполнять их повторно, что существенно сокращает временные и ресурсные расходы. В этом юните вы научитесь осуществлять кэширование в ClearML.

## Содержание

Для кэширования данных необходимо добавить следующие строки в файл clearml.conf:

agent {

venvs\_cache: {

path=~/.clearml/venvs-cache

},

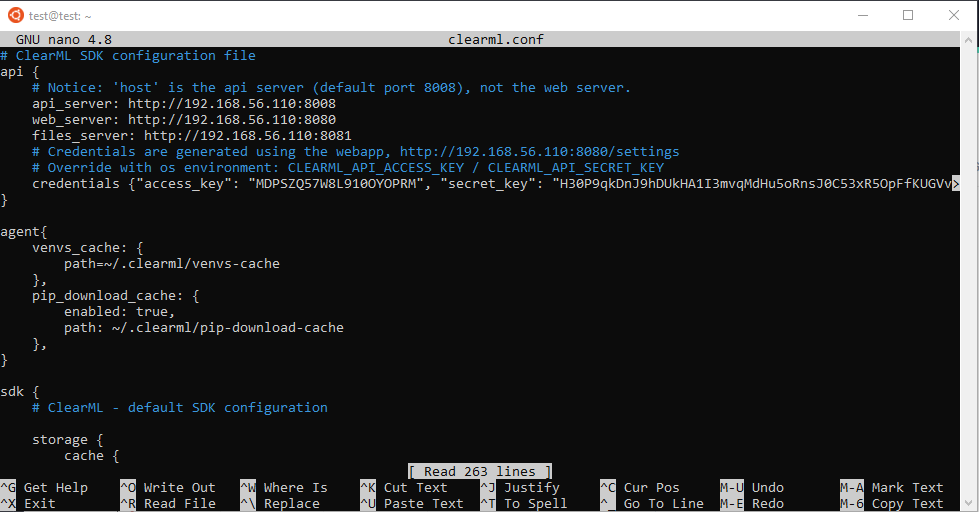
pip\_download\_cache: {

enabled: true,

path: ~/.clearml/pip-download-cache

},

}



После чего при использовании агента ClearML установит нужные библиотеки один раз и для всех последующих действий будет использовать их из кэша, а не скачивать заново.

## Тест

1. Что такое кэширование в контексте ClearML? (0.25)
   1. Сохранение cookies в браузере
   2. Сохранение данных для обучения модели в оперативной памяти
   3. **Сохранение в памяти системы данных по отдельным этапам проекта для последующего переиспользования**
   4. Обеспечение участников команды наличными деньгами для работы над проектом
2. В каком файле осуществляется настройка кэширования? (0.25)
   1. cach.conf
   2. **clearml.conf**
   3. settings.txt
   4. setup.sh
3. Какие преимущества дает кэширование? (0.25)
   1. повышается точность модели
   2. **сокращается время выполнения конвейера**
   3. уменьшается время инференса модели
   4. **снижается входящий трафик**
4. Какой другой инструмент обладает похожей функциональностью? (0.25)
   1. ping
   2. bash
   3. **docker**
   4. windows

## Итоги/выводы

В этом юните вы научились осуществлять кэширование отдельных этапов в проекте машинного обучения в ClearML. Это позволяет оптимизировать время выполнения конвейера, не повторяя при выполнении те этапы, в которых не было изменений.

# Итоги /    выводы по модулю.

В этом модуле вы научились работать с ClearML, который является одним из популярных инструментов в управлении экспериментами в проектах машинного обучения. Для работы с ClearML необходимо установить агента, после чего осуществить подключение к серверу ClearML. Небольшие команды могут использовать облачную версию ClearML сервера, также доступна для использования и локальная инсталляция, например в docker, рассмотренная в этом модуле.

# Список источников

Образовательные и обзорные статьи, справочники, FAQ

1. Учебник по ClearML на habr, <https://habr.com/ru/post/691314/>

Официальные страницы программных продуктов, инструментов, утилит

1. <https://clear.ml/>
2. исходники: <https://github.com/allegroai/clearml>
3. документация: [https://clear.ml/docs/latest/docs](https://clear.ml/docs/latest/docs/)

Репозитории и источники, использованные в модуле для практической работы

1. пример репозитория проекта <https://github.com/CyberLympha>