Трекинг экспериментов в ClearML: краткое объяснение и шаблон кода

Главная идея: чтобы начать хранить эксперименты в ClearML, достаточно обернуть существующий ML-код в объект `Task` и, при необходимости, явно логировать данные, метрики и артефакты через `Task` и его `Logger`. Остальной код (загрузка данных, обучение модели и т.п.) почти не меняется.

1. Что делает ClearML при трекинге эксперимента

При вызове `Task.init(...)` ClearML автоматически сохраняет:

- код и git-коммит (если проект под git)
- версии библиотек
- stdout/stderr (логи)
- метрики и графики от популярных фреймворков (TensorBoard, Matplotlib и др.)
- информацию о ПК, времени запуска и др.

Всё это отображается в веб-интерфейсе как **Task (эксперимент)** в выбранном проекте ClearML.

2. Минимальный шаблон: как «обернуть» ML-скрипт в ClearML

Ниже приведён универсальный пример, который можно вставить в начало ноутбука/скрипта и использовать с любым датасетом и моделью.

```
# ============================
# 1. Импорт ClearML и библиотек
# ============================
from clearml import Task  # основной объект для трекинга экспериментов
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# ============================
# 2. Инициализация эксперимента в ClearML
# ============================
# ВАЖНО:
# - project_name: логическая «папка» для экспериментов в ClearML
# - task_name: понятное имя конкретного запуска (эксперимента)
# - tags: произвольные метки для фильтрации и поиска
task = Task.init(
    project_name="Adult model",          # можно заменить на название курса/проекта
```

```
    task_name="Эксперимент 1",          # осмысленное имя
    tags=["DecisionTreeClassifier"],    # тип модели, версия фичей и т.п.
)

# После этой строки:
# - ClearML начинает отслеживать выполнение кода
# - в веб-интерфейсе появляется новый Task (эксперимент)

# ============================
# 3. Загрузка данных
# ============================
# Обычный код загрузки датасета
fpath = "adults.csv"
df_raw = pd.read_csv(fpath)

# Дополнительно: логирование пути к датасету как артефакта эксперимента
# Артефакт появится во вкладке Artifacts эксперимента в UI ClearML.
task.upload_artifact(
    name="data.raw",        # имя артефакта в ClearML
    artifact_object=fpath   # можно передать путь к файлу или объект (DataFrame и др.)
)

# ============================
# 4. Предобработка и обучение модели
# ============================
# Здесь может быть произвольный код предобработки и обучения.
# ClearML уже пишет логи и информацию об окружении.
X = df_raw.drop(columns=["class"])
y = df_raw["class"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# ============================
# 5. Расчёт метрик и явное логирование
# ============================
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)

# Получение logger, связанного с текущим Task
logger = task.get_logger()

# report_single_value — для «одиночных» финальных метрик (accuracy, AUC и т.п.)
logger.report_single_value(
    name="test_accuracy",   # имя метрики
    value=acc               # значение метрики
)

# В UI: метрика попадёт во вкладку RESULTS → SCALARS

# ============================
# 6. Сохранение и логирование модели как артефакта
# ============================
import joblib

model_path = "adult_dt.pkl"
```

```
joblib.dump(model, model_path)

# Логирование файла модели как артефакта
task.upload_artifact(
    name="model.pkl",        # имя артефакта в ClearML
    artifact_object=model_path
)
# В UI: файл будет доступен во вкладке Artifacts

# ============================
# 7. Завершение эксперимента
# ============================
# Не обязательно, но хорошая практика явного закрытия Task
Task.close()
```

Ключевой момент:

все части кода, не связанные с ClearML (загрузка данных, обучение, оценка модели), могут оставаться без изменений.

Добавляются только:

1. инициализация `Task.init(...)`;
2. (опционально) явное логирование артефактов/метрик через `task.upload_artifact(...)` и `task.get_logger().report_*()`.

3. Настройка подключения к ClearML Server (ключи и хосты – только демо-вариант)

Ниже – **упрощённый и удобный для демонстрации в ноутбуке** способ, когда значения переменных окружения выставляются прямо в коде:

```
import os

os.environ["CLEARML_WEB_HOST"] = "https://app.clear.ml"
os.environ["CLEARML_API_HOST"] = "https://api.clear.ml"
os.environ["CLEARML_FILES_HOST"] = "https://files.clear.ml"
os.environ["CLEARML_API_ACCESS_KEY"] = "ВАШ_ACCESS_KEY"
os.environ["CLEARML_API_SECRET_KEY"] = "ВАШ_SECRET_KEY"
```

После такого блока можно вызывать `Task.init(...)`, и ClearML будет знать, куда отправлять данные.

Важно: хранение ключей доступа прямо в коде (особенно в репозитории) – плохая практика и недопустимо в реальных проектах. В продакшн-среде ключи должны храниться в защищённом конфигурационном файле или в переменных окружения, которые настраиваются вне кода.

4. Логирование данных и артефактов

4.1. Логирование исходных данных (CSV, DataFrame и т.п.)

Пример из ноутбука:

```
fpath = "adults.csv"
df_raw = pd.read_csv(fpath)

# Логирование пути к исходному CSV как артефакта
task.upload_artifact(
    name="data.raw",        # имя в UI ClearML
    artifact_object=fpath   # можно передать строку-путь или объект (например
DataFrame)
)
```

Так можно логировать:

- «сырые» данные (raw data)
- обучающую и тестовую выборки
- предобработанные данные
- вспомогательные файлы (описание фичей, словари и т.п.)

В ClearML артефакты отображаются во вкладке **Artifacts** соответствующего эксперимента.

4.2. Логирование модели

Простейший вариант: сохранить модель в файл и залогировать файл как артефакт:

```
import joblib

model_path = "model.pkl"
joblib.dump(model, model_path)

task.upload_artifact(
    name="model.pkl",
    artifact_object=model_path
)
```

Так логируются модели любых библиотек (sklearn, XGBoost, CatBoost и др.) как бинарные файлы.
Файл можно скачать из UI ClearML и использовать в других скриптах.

5. Логирование метрик и графиков через Logger

5.1. Получение объекта Logger

```
logger = task.get_logger()
```

`logger` – объект класса `Logger`, связанный с текущим `Task`. Через него выполняется явное логирование метрик, графиков, изображений и др..

5.2. Логирование одиночных метрик

```
acc = accuracy_score(y_test, y_pred)

logger.report_single_value(
   name="test_accuracy",  # имя метрики
   value=acc
)
```

Метрика появится в разделе **RESULTS → SCALARS** данного эксперимента.

5.3. Логирование временных рядов (loss по эпохам и т.п.)

Если обучение идёт в цикле (по эпохам/итерациям), можно логировать значения метрики на каждой итерации:

```
for epoch in range(num_epochs):
    # ... обучение одной эпохи ...
    train_loss = ...  # значение loss для текущей эпохи

    logger.report_scalar(
        title="train_loss",     # «имя» графика
        series="loss",          # серия на графике
        value=train_loss,       # значение
        iteration=epoch         # номер итерации (эпохи)
    )
```

В веб-интерфейсе это отобразится как график временного ряда во вкладке **SCALARS**.

6. Использование объекта Dataset для работы с датасетами

ClearML предоставляет отдельный высокоуровневый механизм управления датасетами через класс `Dataset` (Hyper-Datasets). Это не просто артефакт внутри одной задачи, а **версионируемый датасет**, который может переиспользоваться в разных экспериментах и проектах.

6.1. Основные кейсы использования объекта `Dataset`

- создание версионируемого датасета (например, предобработанный датасет);
- загрузка конкретной версии датасета в обучающий скрипт;
- воспроизводимость: модель «знает», с какой версией данных обучалась.

6.2. Создание датасета программно

Пример создания нового датасета из локальной папки с данными:

```
from clearml import Dataset

# Создание нового датасета
dataset = Dataset.create(
    dataset_name="adult_preprocessed",      # имя датасета
    dataset_project="Adult Datasets",       # проект для датасетов
    dataset_version="1.0",                  # опциональная версия
    description="Adult dataset после предобработки"  # описание
)

# Добавление файлов/папок в датасет
# Например, вся папка с подготовленными CSV
dataset.add_files(
    path="data/preprocessed"   # локальный путь к файлам
)

# Загрузка файлов на сервер (обязательно до финализации)
dataset.upload()

# Финализация датасета – после этого версия станет «неизменяемой»
dataset.finalize()
```

После выполнения этих шагов в веб-интерфейсе ClearML появится новый датасет с указанным именем и версией.

Ключевые методы:

- `Dataset.create(...)` — создание нового датасета;

- dataset.add_files(...) / dataset.add_external_files(...) — добавление локальных или удалённых файлов;[14][10]
- dataset.upload() — загрузка файлов в хранилище ClearML;[10]
- dataset.finalize() — завершение редактирования версии датасета (после этого изменить файлы уже нельзя).[10]

6.3. Доступ к уже созданному датасету из обучающего кода

Когда датасет однажды создан и загружен, его можно «подтягивать» в произвольный скрипт обучения:

```
from clearml import Dataset

dataset = Dataset.get(
    dataset_name="adult_preprocessed",
    dataset_project="Adult Datasets",
    dataset_version="1.0",   # можно опустить, тогда возьмётся последняя подходящая версия
)

# Получение локальной read-only копии датасета
dataset_path = dataset.get_local_copy()

print("Данные доступны локально по пути:", dataset_path)
```

Метод get_local_copy():

- скачивает датасет в локальный кэш (если его там ещё нет);
- возвращает путь к папке, где лежат файлы датасета.

После этого можно использовать обычный код работы с файлами:

```
import os
import pandas as pd

csv_path = os.path.join(dataset_path, "adult_preprocessed.csv")
df = pd.read_csv(csv_path)
```

6.4. Связка Task и Dataset в одном эксперименте

Типичный «боевой» шаблон кода:

1. В отдельном скрипте (или задаче) с `Task` создаётся и фиксируется версия датасета через `Dataset.create(...)`, `add_files(...)`, `upload()`, `finalize()`.
2. В скриптах обучения других моделей используется `Dataset.get(...).get_local_copy()`, чтобы работать ровно с этой версией данных.
3. В интерфейсе ClearML можно увидеть, какие задачи обучения используют какие версии датасетов, и тем самым получать воспроизводимость всего процесса обработки данных.

7. Где посмотреть документацию по обсуждаемым объектам

Статья на Хабр: https://habr.com/ru/articles/691314/

Полезные разделы официальной документации ClearML:

- **Fundamentals → Tasks** – концепция Task и работа с экспериментами.[3][1]
- **Fundamentals → Logger** – типы логируемых данных и примеры методов логгера.[4]
- **SDK Reference → Task** – `Task.init`, `Task.close`, `Task.upload_artifact` и др..[9][18][3]
- **SDK Reference → Logger** – методы `report_scalar`, `report_single_value`, `report_image`, `report_plot`, `report_table` и т.д..[7][8][9]
- **ClearML Data SDK / Dataset** – работа с версионируемыми датасетами: `Dataset.create`, `Dataset.get`, `add_files`, `get_local_copy`, `finalize` и др..[11][12][15][14][10]
- **Общий раздел ClearML Documentation** – структура документации, дополнительные примеры и best practices.[5]

Ссылки:

1. https://clear.ml/docs/latest/docs/fundamentals/task/
2. https://clear.ml/docs/latest/docs/getting_started/auto_log_exp/
3. https://clear.ml/docs/latest/docs/references/sdk/task/
4. https://clear.ml/docs/latest/docs/fundamentals/logger/
5. https://clear.ml/docs/latest/docs/
6. https://clear.ml/docs/latest/docs/webapp/webapp_exp_track_visual/
7. https://clear.ml/docs/latest/docs/references/sdk/logger/
8. https://clear.ml/docs/latest/docs/guides/reporting/explicit_reporting/
9. https://clear.ml/docs/latest/docs/clearml_sdk/
10. https://clear.ml/docs/latest/docs/clearml_data/clearml_data_sdk/
11. https://clear.ml/docs/latest/docs/hyperdatasets/dataset/

12. https://clear.ml/docs/latest/docs/clearml_data/data_management_examples/data_man_cifar_classification/
13. https://docs.selectel.ru/en/ml-platform/manage/manage-datasets/
14. https://clear.ml/docs/latest/docs/references/sdk/dataset/
15. https://clear.ml/docs/latest/docs/webapp/datasets/webapp_dataset_viewing/
16. https://clear.ml/docs/latest/docs/clearml_data/data_management_examples/data_man_python/
17. https://clear.ml/docs/latest/docs/getting_started/video_tutorials/hyperdatasets_data_versioning/
18. https://clear.ml/docs/latest/docs/clearml_sdk/task_sdk/