

# Лабораторна робота №1. Організація робочого простору MLOps інженера та автоматизація відстеження експериментів

## 1. Мета роботи

1. Опанувати навички налаштування професійного ізольованого середовища розробки для Data Science проєктів.
2. Засвоїти принципи структурування ML-проєктів згідно зі стандартами індустрії (Cookiecutter Data Science).
3. Навчитися використовувати інструменти контролю версій (Git) з урахуванням специфіки ML-проєктів (ігнорування даних, артефактів).
4. Реалізувати автоматизоване відстеження експериментів (Experiment Tracking) за допомогою платформи MLflow.
5. Провести первинний аналіз даних (EDA) та побудувати базову модель (Baseline) для обраного датасету.

## 2. Завдання для виконання

1. Налаштувати локальне середовище розробки: встановити Python, Git, створити віртуальне середовище.
2. Ініціалізувати Git-репозиторій та налаштувати файл .gitignore.
3. Обрати датасет із запропонованого переліку (див. Додаток А) та завантажити його у відповідну директорію проєкту.
4. Створити Jupyter Notebook для первинного аналізу даних (EDA).
5. Розробити Python-скрипт для тренування моделі, реалізувавши логіку завантаження даних, передобробки та навчання.
6. Інтегрувати у скрипт функціонал MLflow для логування гіперпараметрів, метрик якості та артефактів моделі.
7. Виконати мінімум 5 експериментів з різними налаштуваннями та порівняти їх результати у MLflow UI.

## 3. Теоретичні відомості

### 3.1. Архітектурний контекст

У сучасному MLOps (Machine Learning Operations) пайплайн етап налаштування середовища та експериментування є фундаментом. Від того, наскільки правильно організована структура проєкту та процес фіксації результатів на ранніх етапах, залежить успішність подальшої автоматизації (CI/CD), відтворюваність (Reproducibility) та можливість масштабування рішення. Experiment Tracking виступає сполучною ланкою між дослідницькою діяльністю (Data Science) та інженерною (DevOps), перетворюючи хаотичний пошук моделі на структурований процес.

### 3.2. Ключові поняття

- MLOps (Machine Learning Operations) - методологія, що об'єднує розробку ML-систем (Dev) та їх експлуатацію (Ops) для стандартизації та автоматизації життєвого циклу машинного навчання.
- Experiment Tracking - процес збереження всієї метаданих про запуск навчання моделі. Це включає: конфігурацію (гіперпараметри), версію коду (git commit hash), версію даних, метрики якості та отримані артефакти.
- Artifact (Артефакт) - будь-який файл, створений у результаті виконання скрипту навчання (файл моделі .pkl, графіки, звіти, файли конфігурації).
- Reproducibility (Відтворюваність) - здатність отримати ідентичні результати при повторному запуску експерименту з тими ж вхідними даними та кодом.

### 3.3. Необхідні інструменти

- Python 3.10+: Основна мова програмування.
- Git: Система контролю версій.
- Virtualenv / Conda: Інструменти для ізоляції залежностей.
- MLflow: Open-source платформа для управління життєвим циклом ML (використовується модуль MLflow Tracking).
- JupyterLab / VS Code: Середовище розробки.

## 4. Покрокова інструкція

### 4.1. Підготовка середовища

1. **Створення директорії проєкту та ініціалізація Git:** Відкрийте термінал та виконайте команди:

```
mkdir mlops_lab_1
cd mlops_lab_1
git init
```

2. **Налаштування .gitignore:**

Створіть файл .gitignore у корені проєкту. Це критично важливо для запобігання потраплянню великих файлів даних та системних папок у репозиторій.

Вміст файлу .gitignore:

```
__pycache__/
*.py[cod]
venv/
.env

data/
models/

.ipynb_checkpoints
```

```
mlruns/  
.trash/
```

### 3. Створення віртуального середовища:

Для Windows:

```
python -m venv venv  
.\venv\Scripts\activate
```

Для Linux/macOS:

```
python3 -m venv venv  
source venv/bin/activate
```

### 4. Встановлення залежностей:

Створіть файл requirements.txt:

```
pandas  
numpy  
scikit-learn  
matplotlib  
seaborn  
mlflow  
jupyter
```

Встановіть бібліотеки:

```
pip install -r requirements.txt
```

## 4.2. Основна частина роботи

### *Крок 1: Структурування проєкту*

Створіть базову структуру каталогів, яка відповідає спрощеному стандарту Cookiecutter Data Science (приклад):

```
mkdir data  
mkdir data/raw          # Для сирих даних (read-only)  
mkdir notebooks         # Для Jupyter ноутбуків  
mkdir src               # Для скриптів Python  
mkdir models            # Для збережених моделей
```

### *Крок 2: Завантаження даних*

1. Оберіть один датасет зі списку варіантів (див. Додаток А) або використайте власний датасет.
2. Завантажте файл (наприклад, dataset.csv) у папку data/raw/.

### *Крок 3: Первинний аналіз даних (EDA)*

1. Запустіть Jupyter Notebook: `jupyter notebook`.
2. Створіть файл notebooks/01\_eda.ipynb.

3. Виконайте базовий аналіз:
  - Завантаження даних (pd.read\_csv).
  - Перевірка типів даних та пропусків (df.info(), df.isnull().sum()).
  - Візуалізація розподілу цільової змінної (Target Distribution).
  - Побудова матриці кореляції.

#### *Крок 4: Розробка скрипту навчання з MLflow*

Необхідно самостійно імплементувати файли для тренування та пов'язані модулі, які реалізують повний цикл навчання моделі.

#### **Вимоги до скрипту:**

1. Завантаження даних: Зчитування CSV/Other types з папки data/raw/.
2. Передобробка: Обробка пропущених значень, кодування категоріальних змінних (якщо є), розділення на X (ознаки) та y (цільова змінна).
3. Розділення вибірки: Використання train\_test\_split для створення тренувального та тестового наборів.
4. Ініціалізація MLflow: Встановлення імені експерименту (mlflow.set\_experiment).
5. Логування: У блоці with mlflow.start\_run(): необхідно залогувати:
  - Гіперпараметри моделі (наприклад, n\_estimators, max\_depth, learning\_rate).
  - Метрики якості на тестовій вибірці (accuracy, f1\_score, rmse тощо).
  - Саму навчену модель (mlflow.sklearn.log\_model).
  - Графік (наприклад, Confusion Matrix або Feature Importance) як артефакт.

#### **Приклад синтаксису MLflow (Snippet):**

```
import mlflow
import mlflow.sklearn

# ... ваш код підготовки даних ...

mlflow.set_experiment("My_Experiment_Name")

with mlflow.start_run():
    # Логування параметрів
    mlflow.log_param("param_name", param_value)

    # ... навчання моделі ...

    # Логування метрик
    mlflow.log_metric("metric_name", metric_value)

    # Логування моделі
    mlflow.sklearn.log_model(model, "model_name")

    # Логування файлу (графіку)
    mlflow.log_artifact("plot.png")
```

## Крок 5: Розширення функціоналу

Модифікуйте скрипти для відповідності професійним стандартам:

1. **CLI Аргументи:** Гіперпараметри повинні передаватися як аргументи командного рядка (використовуючи бібліотеку `argparse` або `click`), а не бути "захардкоженими".
2. **Feature Importance:** Додайте логування графіку важливості ознак, який показує, які змінні найбільше вплинули на рішення моделі.
3. **Розширене логування (Tags):**
  - Додайте використання `mlflow.set_tag()` для додавання метаданих (наприклад, автор запуску, версія датасету, тип моделі).
  - Продемонструйте, як в MLflow UI можна фільтрувати запуски, використовуючи Search query (наприклад, `tags.model_type = "RandomForest"`).

### 4.3. Перевірка результатів

1. Запустіть розроблені скрипти навчання, наприклад:

```
python src/train.py
```

2. **Проведіть дослідження впливу гіперпараметрів (Hyperparameter Tuning Analysis):**

- Оберіть один ключовий гіперпараметр (наприклад, `max_depth` для дерев або `n_neighbors` для KNN).
- Виконайте **мінімум 5 запусків**, послідовно змінюючи значення цього параметра (від недостатнього до надлишкового).
- *Важливо:* Логуйте метрики (`accuracy`, `f1`) як для **тестової**, так і для **тренувальної** вибірки. Це дозволить побачити момент, коли модель починає перенавчатися (Overfitting).

3. Запустіть інтерфейс MLflow:

```
mlflow ui
```

4. Відкрийте браузер за адресою `http://127.0.0.1:5000`.

5. Переконайтеся, що:

- Створено експеримент з вашою назвою.
- Відображається список запусків (Runs).
- **Використайте функцію "Compare":** Виділіть всі ваші запуски та натисніть "Compare". Побудуйте графік залежності метрики (Y-axis) від гіперпараметра (X-axis). Проаналізуйте отримані криві.

## 5. Контрольні запитання

1. У чому полягає фундаментальна відмінність між артефактами коду (у DevOps) та артефактами ML-моделей?
2. Чому папку `mlruns` (або іншу папку для зберігання метаданих MLflow) зазвичай додають у `.gitignore` при локальній розробці?

3. Які переваги дає використання віртуальних середовищ (venv/conda) у командній розробці?
4. Назвіть три основні компоненти, які логуює MLflow Tracking, та наведіть приклади для кожного.
5. Як забезпечити повну відтворюваність експерименту (Reproducibility), якщо дані у джерелі постійно оновлюються? Які інструменти для цього потрібні окрім Git?
6. У чому різниця між `mlflow.log_artifact` та `mlflow.log_model`? Яку додаткову інформацію зберігає `log_model`?
7. Яким чином MLflow дозволяє порівнювати ефективність різних моделей, якщо вони були навчені на різних підмножинах даних (наприклад, Cross-Validation)?
8. **Архітектура:** Уявіть, що ви працюєте в команді з 5 Data Scientist-ів. Як потрібно налаштувати MLflow, щоб всі бачили експерименти одне одного? Де в такій архітектурі фізично зберігаються метрики (числа), а де — артефакти (файли моделей)?
9. **Залежності:** Ви зафіксували `random_state`, код та дані, але при запуску на іншому комп'ютері модель видає трохи інші результати (на рівні 4-го знаку після коми). Чому це може статися і як це пов'язано з версіями бібліотек (наприклад, `scikit-learn`, `numpy`) або апаратним забезпеченням (Floating point arithmetic)?
10. **Experiment Design:** Чому логування метрик лише на тестовій вибірці є недостатнім для повноцінного аналізу моделі? Яку роль відіграє валідаційна вибірка у контексті Experiment Tracking і як це відобразити в MLflow?

## 6. Рекомендовані матеріали для поглибленого вивчення

1. MLflow Documentation: Official Docs - основне джерело знань. Рекомендується розділ "Tracking".
2. Databricks Academy: MLflow Fundamentals - курси від творців MLflow.
3. Coursera: "Machine Learning Engineering for Production (MLOps) Specialization" (Andrew Ng) - фундаментальний курс, де розглядається концепція Experiment Tracking.
4. Kaggle Learn: Intro to Machine Learning - вивчення Scikit-learn.
5. YouTube: Канал "DataTalksClub" (курс MLOps Zoomcamp) - практичні відео по MLflow.

## 7. Додатки

### Додаток А. Варіанти індивідуальних завдань (Датасети)

Для виконання лабораторних робіт можна обрати один із наведених нижче датасетів або використати власний (за погодженням з викладачем). Список сформовано для забезпечення різноманітності задач (табличні дані, зображення, NLP) та рівнів складності.

#### Табличні дані (Tabular Data)

1. Credit Card Fraud Detection (Classification, Anomaly Detection)
  - *Задача:* Виявлення шахрайських транзакцій.

- *Особливості:* Критично незбалансований (шахрайство < 1%). Ідеально для метрик Precision/Recall.
  - *Джерело:* [Kaggle Link](#)
2. Telco Customer Churn (Classification)
- *Задача:* Передбачення відтоку клієнтів.
  - *Особливості:* Категоріальні ознаки, помірний дисбаланс.
  - *Джерело:* [Kaggle Link](#)
3. 10 Million House Rent Data of 40 cities (Regression)
- *Задача:* Прогнозування ціни на нерухомість.
  - *Особливості:* 10 мільйонів записів, багато пропусків, feature engineering.
  - *Джерело:* [Kaggle Link](#)
4. Bike Sharing Demand (Regression, Time Series)
- *Задача:* Прогнозування попиту на оренду.
  - *Особливості:* Часові ряди, сезонність, вплив погоди.
  - *Джерело:* [Kaggle Link](#)
5. Heart Diseases and Conditions Data - UCI (Classification)
- *Задача:* Діагностика серцевих захворювань.
  - *Особливості:* Малий датасет, важливість інтерпретації (SHAP).
  - *Джерело:* [Kaggle Link](#)
6. German Credit Risk (Classification)
- *Задача:* Оцінка кредитного ризику (Good/Bad).
  - *Особливості:* "Вартість" помилки (Cost Matrix), етичні питання.
  - *Джерело:* [UCI Link](#)
7. Adult Census Income (Classification)
- *Задача:* Передбачення доходу > \$50K.
  - *Особливості:* Питання Fairness & Bias (стать, раса).
  - *Джерело:* [UCI Link](#)
8. Rain in Australia (Classification)
- *Задача:* Передбачення дощу на завтра.
  - *Особливості:* Незбалансований, часова залежність, пропуски.
  - *Джерело:* [Kaggle Link](#)
9. Walmart Store Sales Forecasting (Regression)
- *Задача:* Прогнозування продажів.
  - *Особливості:* Часові ряди, вплив свят.
  - *Джерело:* [Kaggle Link](#)
10. Fetal Health Classification (Classification)
- *Задача:* Класифікація здоров'я плоду (Normal, Suspect, Pathological).

- *Особливості:* Мультикласова класифікація, сильний дисбаланс.
- *Джерело:* [Kaggle Link](#)

11. Stroke Prediction Dataset (Classification)

- *Задача:* Передбачення інсульту.
- *Особливості:* Дуже сильний дисбаланс (1.8% позитивних), медичні дані.
- *Джерело:* [Kaggle Link](#)

12. Spotify Tracks Attributes and Popularity (Regression/Classification)

- *Задача:* Передбачення популярності пісні або жанру.
- *Особливості:* Реальні дані з API, кореляція ознак.
- *Джерело:* [Kaggle Link](#)

*Комп'ютерний зір (Computer Vision)*

13. Chest X-Ray Images (Pneumonia) (Image Classification)

- *Задача:* Визначення пневмонії за рентгенівськими знімками.
- *Особливості:* Робота з зображеннями, незбалансований, медична відповідальність.
- *Джерело:* [Kaggle Link](#)

14. Face Mask Detection (Object Detection / Classification)

- *Задача:* Визначення наявності маски на обличчі.
- *Особливості:* Можна вирішувати як класифікацію (crop images) або детекцію.
- *Джерело:* [Kaggle Link](#)

15. Traffic Signs Preprocessing (Multi-class Classification)

- *Задача:* Розпізнавання дорожніх знаків (43 класи).
- *Особливості:* Багато класів, реальні умови (освітлення, кут).
- *Джерело:* [Kaggle Link](#)

16. PlantVillage Dataset (Image Classification)

- *Задача:* Визначення хвороб рослин за фото листя.
- *Особливості:* Мультикласова, агро-сектор.
- *Джерело:* [Kaggle Link](#)

17. Skin Cancer MNIST: HAM10000 (Image Classification)

- *Задача:* Класифікація пігментних уражень шкіри.
- *Особливості:* Критично важлива точність, сильний дисбаланс класів.
- *Джерело:* [Kaggle Link](#)

*Обробка природної мови (NLP)*

18. IMDB Movie Reviews (Sentiment Analysis)

- *Задача:* Визначення тональності відгуку (позитивний/негативний).

- *Особливості*: Робота з текстом, токенизація, embeddings.
- *Джерело*: [Kaggle Link](#)

#### 19. Twitter Sentiment Analysis (Hate Speech) (Classification)

- *Задача*: Виявлення расистських/сексистських твітів.
- *Особливості*: Незбалансований, етичні питання, очистка тексту.
- *Джерело*: [Kaggle Link](#)

#### 20. SMS Spam Collection (Classification)

- *Задача*: Класифікація SMS як спам або не спам.
- *Особливості*: Робота з текстом, дисбаланс.
- *Джерело*: [UCI Link](#)

### Структура проєкту (приклад)

```

mlops_lab_1/
├── .gitignore      # Файл виключень Git
├── requirements.txt # Список залежностей
├── README.md       # Опис проєкту
├── venv/           # Віртуальне середовище (не в Git)
├── data/
│   ├── raw/        # Сирі дані (не в Git)
│   └── dataset.csv
├── notebooks/
│   └── 01_eda.ipynb # Ноутбук з аналізом
├── src/
│   └── train.py     # Скрипт навчання
├── mlruns/         # Логи MLflow (не в Git)
└── models/         # Збережені моделі (не в Git)

```