**3. Автоматизація підготовки даних: побудова відтворюваних пайплайнів з Scikit-learn.** ⚙️

* **Завдання:** Продемонструвати, як стандартні кроки (заповнення пропусків, кодування категоріальних ознак, масштабування) об'єднуються в єдиний конвеєр. Пояснити призначення класів Pipeline та ColumnTransformer в Scikit-learn. Створити та продемонструвати повний пайплайн обробки даних, який можна легко інтегрувати з крос-валідацією та підбором гіперпараметрів.

### 2 слайд

### Що таке pipeline машинного навчання?

У штучному інтелекті побудова успішної моделі машинного навчання включає не лише вибір найкращого алгоритму; це також вимагає ефективного управління даними, навчання та розгортання в організованій формі. У такій ситуації важливим стає **pipeline машинного навчання**.

**Pipeline** машинного навчання — це організований підхід, який автоматизує весь процес: від збору сирих даних діо розгортання навченої моделі для практичного використання. У цій презентації розглянемо основні етапи створення конвеєра машинного навчання.

3 слайд

### Зміст

4 слайд

### Вступ до pipeline машинного навчання

### 5 слайд

Pipeline машинного навчання — це систематизований робочий процес, призначений для автоматизації побудови, навчання та розгортання моделей ML. Він включає кілька етапів: збір даних, попередню обробку, інженерію ознак, навчання моделі, оцінку та розгортання.

Замість того, щоб керувати кожним етапом окремо, pipeline допомагає спростити й стандартизувати робочий процес, роблячи розробку швидшою, ефективнішою та масштабованою. Він також покращує управління даними, дозволяючи витягувати, трансформувати й завантажувати їх із різних джерел.

6 слайд

### Переваги pipeline машинного навчання

### 7 слайд

Pipeline машинного навчання надає кілька важливих переваг, автоматизуючи та оптимізуючи процес розробки, навчання й розгортання моделей. Основні з них:

1. **Автоматизація та ефективність**: повторювані завдання (очистка даних, навчання та тестування моделей) автоматизуються, що економить час і дозволяє зосередитися на стратегічних завданнях.
2. **Швидше розгортання моделей**: спрощує перенесення навченої моделі у реальне застосування (наприклад, біржова торгівля, виявлення шахрайства, медицина).
3. **Покращення точності та узгодженості**: дані завжди обробляються однаково, зменшуючи кількість помилок і роблячи передбачення надійнішими.
4. **Робота з великими даними**: добре масштабується на великих наборах даних, особливо у хмарних середовищах.
5. **Економічність**: автоматизація зменшує витрати часу й коштів, скорочує кількість ручної роботи та помилок.
6. **Запобігання data leakage**

### Слайд 8

### Кроки побудови pipeline машинного навчання

### Слайд 9

**Крок 1: Збір та попередня обробка даних**

* Збір даних із баз, API чи CSV.
* Очищення: усунення пропусків, дублікатів, помилок.
* Нормалізація й стандартизація числових значень.
* Перетворення категоріальних змінних у формат, придатний для алгоритмів, тобто в цифри

**Крок 2: Інженерія ознак**

* Вибір найважливіших ознак для покращення моделі.
* Створення нових ознак (feature extraction/feature transformation).

**Крок 3: Розбиття даних**

* Поділ на training, validation і testing набори.
* Використання випадкової вибірки для незбалансованих даних.

**Крок 4: Вибір і навчання моделі**

* Підбір алгоритму (класифікація, регресія, кластеризація тощо).
* Навчання на тренувальному наборі.

**Крок 5: Оцінка та оптимізація моделі**

* Використання метрик: accuracy, precision, recall тощо.
* Тюнінг гіперпараметрів (Grid Search, Random Search), крос-валідація.

**Крок 6: Розгортання моделі**

* Розгортання через Flask, FastAPI, TensorFlow чи хмарні сервіси.
* Збереження моделі для практичного застосування.

**Крок 7: Безперервне навчання та моніторинг**

* Автоматизація з MLOps інструментами (MLflow, Kubeflow).
* Оновлення моделі новими даними для підтримки точності.

Тепер до цікавого

Слайд 10

### Імплементація пайплайну

### Слайд 11

Для початку поговоримо про мінімальні необхідні бібліотеки і потім вже за сам приклад.

import numpy as np

import pandas as pd

це зрозуміло, стандарт

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

потрібна для розділення даних на навчальні і тестові, думайте про них як про флешкарточки, по яким буде навчатись модель

**12 слайд**

from sklearn.preprocessing import StandardScaler, OneHotEncoder

StandardScaler – потрібен для стандартизування даних, що є доволі важливим кроком для Machine learning, бо так комп’ютер краще сприймає дані.

OneHotEncoder – перетворює категорійні(тобто object поля) на бінарні, бо комп’ютер не вміє працювати з стрінгами

Краще звісно зрозуміло як це працює візуально, тому можете звернути увагу на слайд. По простому, він перетворює один стовбчик на декілька різних для кожного унікального значення в стовбчику і задає одиниці на відповідні значення і нулі на інші.

from sklearn.ensemble import RandomForestClassifier

random forest classifier це метаалгоритм (метаоцінювач), який навчає кілька класифікаторів на основі дерев рішень на різних підвибірках набору даних і використовує усереднення, щоб підвищити точність прогнозування та стримувати перенавчання. Це просто один з класифікаторів, можете обирати інший, ми вже їх вивчали на попередньому курсі.

І тепер до головних бібліотек

**13 слайд**

from sklearn.compose import ColumnTransformer

Застосовує перетворювачі до стовпців масиву або pandas DataFrame. Цей естиматор дозволяє різним колонкам чи підмножинам стовпців вхідних даних бути перетвореними окремо, а ознаки, згенеровані кожним трансформером потім об’єднуються (конкатенуються) в єдиний простір ознак. Це корисно для **неоднорідних** або **колонкових** даних, коли потрібно поєднати кілька механізмів вилучення ознак чи перетворень в один перетворювач.

Звісно поки нічого не зрозуміло, але на прикладі стане краще.

**14 слайд**

from sklearn.pipeline import Pipeline

**Pipeline** дає змогу послідовно застосовувати список трансформерів для препроцесингу даних і за потреби завершувати цю послідовність фінальним предиктором для побудови прогнозної моделі.

Знову ж таки, буде зрозуміліше на прикладі.

**15 слайд**

Я додав ще декілька бібліотек, з багатьма з них ви вже знайомі, тому коротко нагадаю:

**seaborn as sns** — зручна візуалізація

**GridSearchCV** (із sklearn.model\_selection) — тюнінг гіперпараметрів

**SimpleImputer** — заповнення пропусків, відповідно до заданою статегією (e.g. mean, median, or most frequent)

**mean\_squared\_error, r2\_score, mean\_absolute\_error** — метрики якості регресії

**joblib** — продакшн-удобняшка, яка буде зберігати і завантажувати пайплайн, але це вже спойлери

**16 слайд**

Перейдемо до PREPROCESSING-у

Першим ділом треба завантажити дані і вивести базову інформацію

Можемо побачити шо make, model, vehicle class i transmission це обжект. Ну точніше не можемо, бо у мене чомусь не працювала дата.інфо коректно, але ми можемо зрозуміти, бо вони стрінги

**17 слайд**

Продовжуємо діяти, як ми і завжди робили на курсі Data mining, дропаємо відповідь, щоб потім на ній вчити модель.

Після чого буде зручно відразу розділити цифрові і object(або категоріальні) колонки. Це нам знадобиться пізніше.

**18 Слайд**

Тепер перейдемо до самого пайплайну, уявляйте його як багато різних труб, які ми в кінці з’єднаємо в одну. Зараз почнемо з маленьких шматочків.

Тут я створюю пайплайн для цифрової частини, використовуючи клас Pipeline в ньому запишемо ліст задач, які має буде виконати цей пайплайн.

Першою таскою буде заповнення пропущених значень, використовуючи Imputer з статегією mean, тобто замінить пропуски на середнє значення.

Друга таска стандартизує значення, щоб середнє було 0 і варіантивність в 1. Як я і казав раніше, це допоможе покращити здібності до передбачення у нашої моделі.

Наступний пайплайн буде для категорікал даних. Тут також використовуємо сімпл імп’ютер, але вже не з меан, бо середнього з стрінгів не існує, а стратегію найбільш повторювальних, бо їх дані важче здвинути не туди, куди треба.

І далі використовуємо наш OneHotEncoder, який перетворить стрінги в бінарні. Оцей handle\_unknown='ignore' у OneHotEncoder каже енкодеру **не падати**, якщо під час transform з’явиться **нова категорія**, якої не було на fit. Це нам в будь-якому разі не важливо для цього прикладу.

Тепер з’єднаємо два пайплайна використовуючи ColumnTransformer,

Він як і пайплайни першим приймає назву степу, потім функцію перетворювач, а потім які дані треба перетворити.

Вписуємо дві таски пайпланів і об’єднання їх в один простір закінчено.

**19 слайд**

Далі збираємо усе в фінальний пайп, тобто наш створений препроцессор, а також модель навчання. Як ви можете побачити, ми вписуємо початково без параметрів модель, але виникає питання чому не відразу застосувати тюнінг вже в пайплайні?

Але якщо так зробити, то кроки препроцесингу (імп’ютер, скейлер і тд) спочатку **фітнуться на всьому X\_train**, а вже потім усередині SearchCV піде CV. Це = **data leakage** (валідаційні фолди бачать статистики з усього train).

**20 слайд**

Тож підготуємось до навчання моделі. Пам’ятаєте як я казав думати про це як про флешкарточки, які ви використовуєте наприклад під час вивчення нової мови, де спереду є картинка курки і позаду карточки слово “Chicken”? Так от, для моделі Х частина відповідає картинці курки, а у частина відповідає слову “Chicken”. Коли модель навчається, вона має право дивитися на X\_train(передню частину картки) i y\_train(на задню).

А потім, коли модель навчилася, то їй показують лише передню частину, X\_test і вона має зробити предікшон. Далі для перевірки на скільки добре відпрацювала модель, можна порівняти з y\_test.

**21 слайд**

**Hyperparameter grid for tuning**

Тут ми задаємо грід для різних параметрів, щоб потім GridSearchCV змогла знайти найкращі параметри для нашого датасету. Усі параметри мають префікс model\_\_, бо ми будемо використовувати їх для кроку з назвою модел всередині пайплайну. Сильно не буду вдаватися в подробиці параметрів, це ми вже проходили на курсі Data mining, але ви можете побачити невеличке пояснення англійською. Єдина на що я б звернув увагу, це на фолди крос-валідації, задаючи значення 5 параметру cv\_folds, ми змушуємо модель розділювати тренувальний набір на 5 рівних частин(фолдів), що бере 4 фолди на навчання і 1 на валідацію. Це дає меншу залежність від одного випадкового розбиття, що надає більш чесне порівняння гіперпараметрів в грідсьорч.

**21 слайд**

Тепер до фінальних кроків, після чого переходимо до цікавого. Використовуємо GridSearchCV до пайплайну. Цей виклик приймає такі аргументи:

**base\_pipeline** — estimator (модель для пошуку). Саме його клонують і тренують у кожному фолді та для кожної комбінації гіперпараметрів

**param\_grid** — сітка гіперпараметрів (dict або список dict-ів).

**cv=5** — 5-фолдова крос-валідація

**n\_jobs=1** — скільки потоків використовувати. 1 = без паралелізму.  
Як правило, для пришвидшення ставлять n\_jobs=-1 (усі ядра), якщо ресурси дозволяють. А також існує помилка з цим на Windows, що при проходженні стикається з референсом якого вже не існує. Я не знайшов як це вирішити і для прикладу не важливо, тож якщо хтось знає, feel free мені написати.

**22 слайд**

Після задання усіх параметрів, навчаємо модель на навчальних даних і отримаємо і отримаємо наступне виведення

Тут ми можемо побачити весь наш пайплайн, на нього можна трошки подивитися, повідкривавши його різні сховані панельки параметрами

**23-24 слайд**

Можемо побачити купу різних параметрів, все це дуже цікаво і частково не зрозуміло звісно, але можна зробити якісь висновки.

**25 слайд**

І останній крок для створення пайплайну це застосування до нього тільки найкращих параметрів. Також для візуалізації виведемо його.

Створимо предікшон.

**26 слайд**

Так ми зробили свій перший пайплайн. Давайте ж спробуємо з ним погратись

**27 слайд**

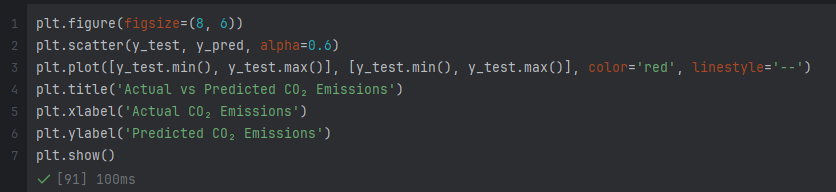
Пайплайн клас має багато різних корисних функцій, от наприклад подивимось на назви усіх енкодед категоріальні колонок.

Тобто ми пишемо пайплайн, посилаємось на відповідний степ, по назві трансформерів, у цьому пайплайні степ і отримаємо вивід. І виводячи це ми бачимо величезний список усіх заендоджених колонок, усі вони були переведені в бінарний формат, як ми зробили раніше. Насправді існує дуже велика кількість вбудованих функцій, звісно усі їх не буду показувати, лише хотів звернути вашу увагу як працює OneHotEncoder.

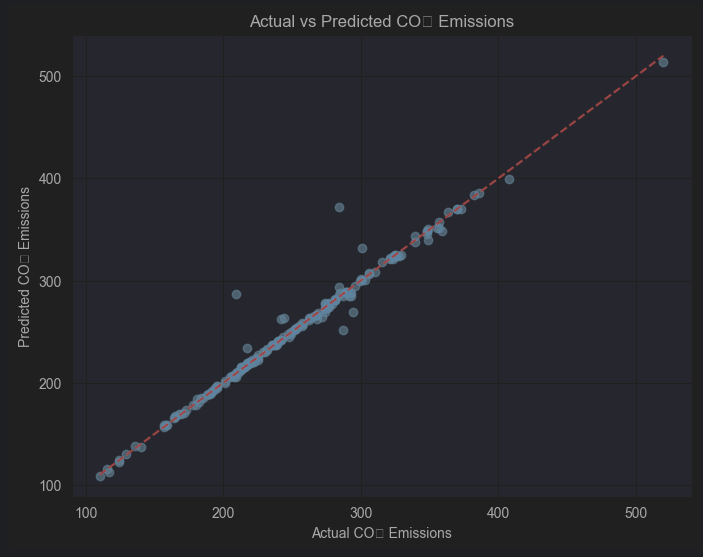
**28 слайд**

Проаналізуємо ж ефективність моделі. Ці метрики деякі з вас можуть згадати з першої лабораторної по цьому предмету. Хто пропустив пари, якщо коротко, то R2 в ідеалі чим ближче до одиниці тим краще, Root Mean Squared Error чим менше ти краще і Mean Absolute Error чим менше тим краще. По даним можемо побачити, що наша модель відпрацьовує дуже гарно, але ніяка модель не буде ідеальною.

Також зробимо графік порівняння реальних даних і предікшених

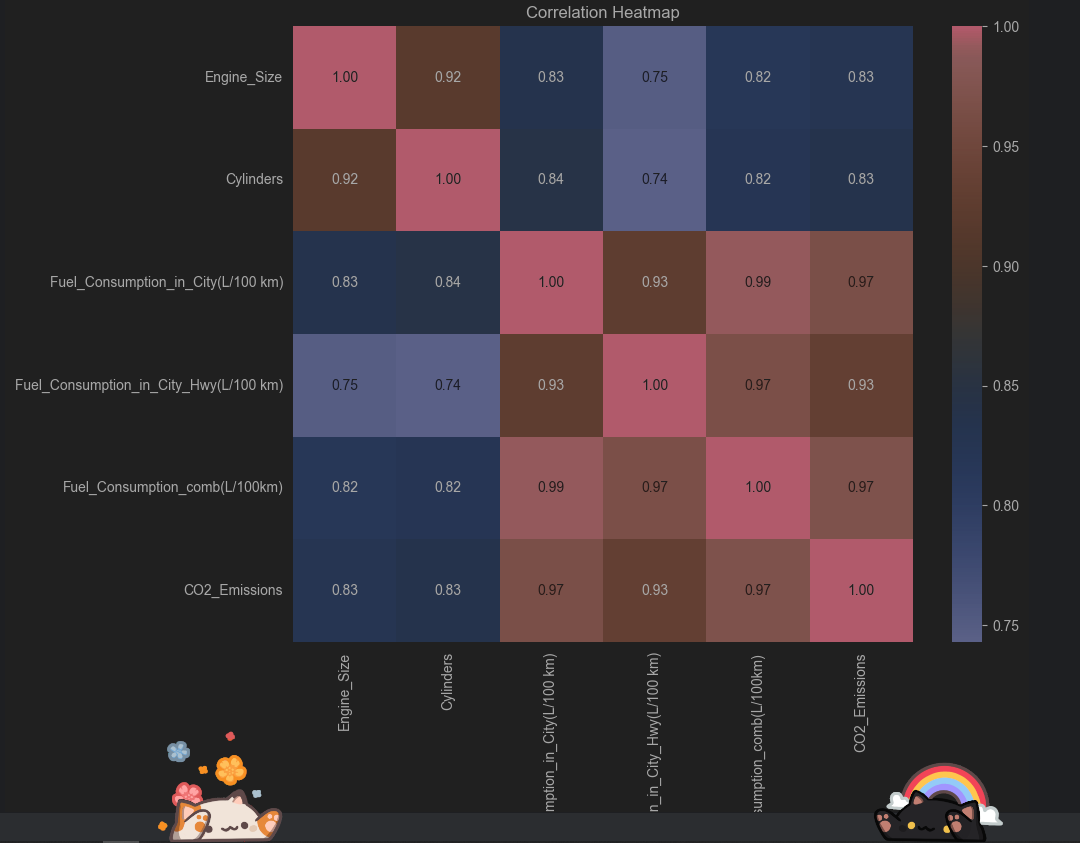


29 слайд



Теж бачимо що майже ідеально працює---

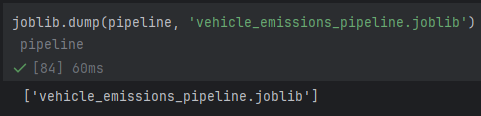
**30 слайд**

Ну і матрицю кореляції в догонку 

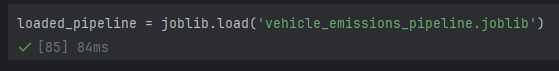
**31 слайд**

Також наостанок ще одна з головних фіч пайплайну це звісно те що його можна зберегти і легко перенести.

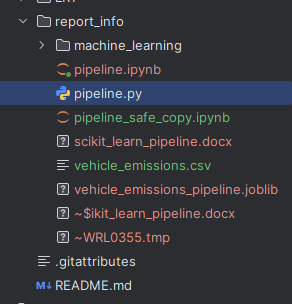
**32 слайд**

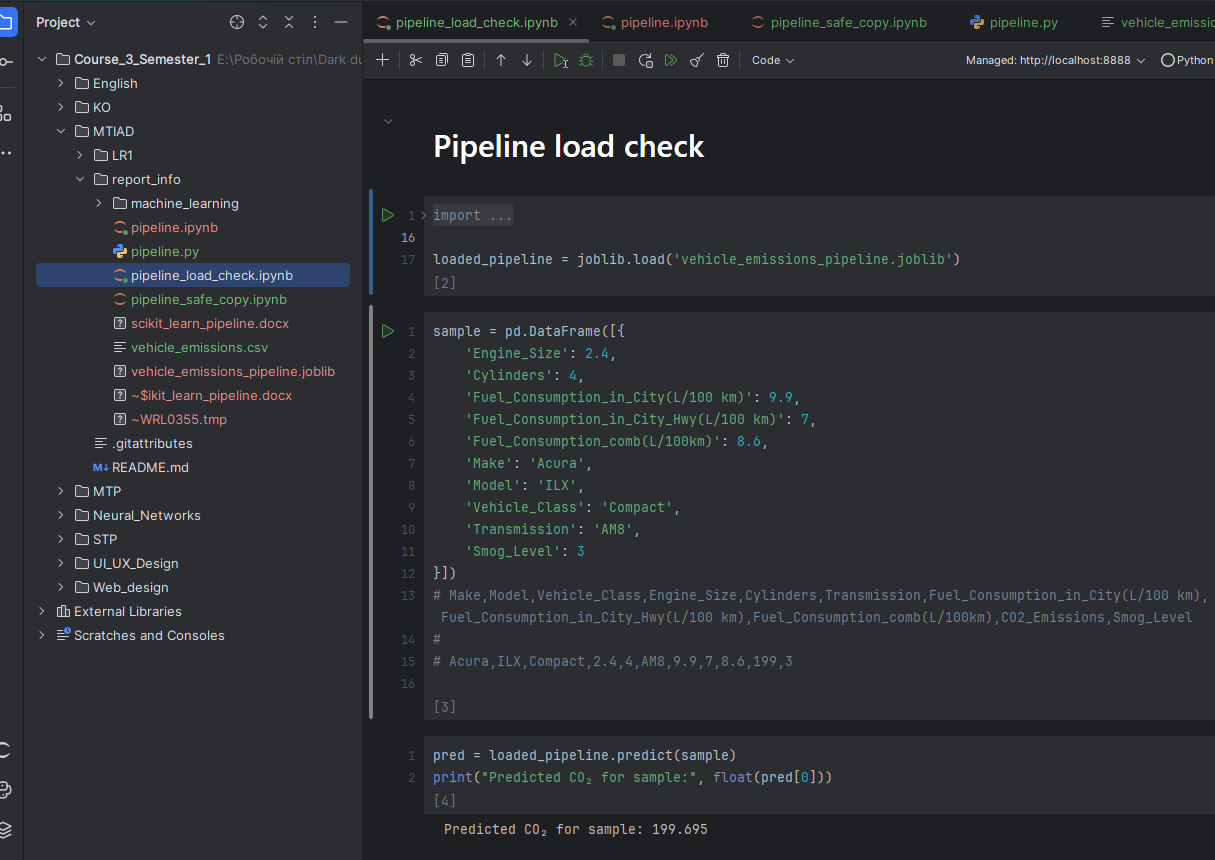


За допомогою цієї команди, задавши ім’я, ми зберігаємо .joblib файл з пайплайном у ньому

Потім для завантаження цього пайплайну використовується така коротенька команда

Яка замінює усі попередні десятки рядків коду одним.



**33 слайд**

Ось тут я тестував в новому файлі по даним з першого рядка і як ми бачимо відхилення лише на ~0.7 більше, що я вважаю ідеальним результом.

**34 слайд**

Підсумовуючи, **machine pipeline** спрощує й автоматизує складний процес розробки моделей ШІ, забезпечуючи ефективність, точність і масштабованість. Інтегруючи структуровані кроки — препроцесинг даних, навчання моделі, оцінювання та розгортання — він упорядковує робочі процеси машинного навчання. Із зростанням попиту на AI-аналітику **ML pipelines** і надалі залишатимуться ключовим рушієм інновацій, роблячи машинне навчання швидшим і більш застосовним до реальних задач.

**35 слайд**

Корисні посилання:

* <https://www.youtube.com/watch?v=T9ETsSD1I0w> --- ну дуууже легкий приклад пайплайну
* <https://www.youtube.com/watch?v=777Qb0gHuJU> – Ускладнений варіант, але без тюнінгу гіперпараметрів
* <https://scikit-learn.org/> - сайт machine learning in python
* https://www.geeksforgeeks.org/blogs/machine-learning-pipeline/ - теж сайт, який гарно пояснює machine learning