КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки, освітня програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

"Розробка моделей та інформаційної технології прогнозування трендів для платформи YouTube за допомогою методів науки про дані**"**

Студента 2-го курсу групи ІАВ-21 Майданик Андрій
(прізвище, ім'я, по батькові)

_____ _____ _____
(підпис студента) (дата)

Науковий керівник:

Доктор технічних наук
(науковий ступінь, вчене звання)

Заріцький Олег Володимирович
(прізвище, ім'я, по батькові)

_____
(підпис)

Попередній захист:		
(Висновок: «До захисту в Екзаменаційній комісії»)		
Завідувач кафедри технологій управління _____ _____		
(підпис)	(прізвище, ініціали)	(дата)

Київ – 2025

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 – Комп'ютерні науки
Освітня-наукова програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри,

професор

ВІКТОР МОРОЗОВ

_____

"___" _____ 2025 року

ЗАВДАННЯ НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Майданик Андрій Олександрович

Група IAB-21

1. Тема кваліфікаційної роботи

"Розробка моделей та інформаційної технології прогнозування трендів для платформи YouTube за допомогою методів науки про дані"

Затверджено наказом від «___» _____ 2025 р. № _____

2. Строк подання студентом готової роботи – "___" _____ 2025 р.

3. Цільова установка та вихідні дані до роботи

Прототип інформаційно-аналітичної системи прогнозування потрапляння відеоролику до списку «Trending» на платформі YouTube, оснований на обробці історичних та поточних метаданих із використанням

сучасних алгоритмів машинного навчання. Система реалізована як модуль прогнозної аналітики, що автоматично завантажує записи з відкритого набору Trending YouTube Videos (2017–2024, країна US), доповнює їх показниками YouTube Analytics API та індексами популярності Google Trends, очищає та формує комбіновані часові, текстові й поведінкові ознаки. На основі цих даних модуль здійснює балансування класів, навчає ансамблеву модель (LightGBM із SMOTE) та видає короткостроковий прогноз імовірності потрапляння відео у «Trending». Результати подаються через захищений REST-endpoint /predict веб-сервісу Flask; сервіс генерує рекомендації щодо оптимальної години публікації й очікуваного рівня залученості. Візуалізація прогнозів і важливостей ознак реалізована у веб-інтерфейсі на основі інтерактивних графіків Plotly Dash

4. Зміст роботи

Дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У першому розділі проведено огляд еволюції підходів до прогнозування популярності відеоконтенту, проаналізовано сучасні наукові публікації та індустріальні інструменти аналітики YouTube, обґрунтовано вибір гібридної методології CRISP-DM + OSEMN. Другий розділ присвячено формалізації задачі та фічер-інжинірингу: описано процес очищення великомасштабного датасету Trending YouTube Videos, сформовано комбіновані часові, текстові й поведінкові ознаки, реалізовано балансування класів SMOTE і побудовано кінцевий ознаковий простір. У третьому розділі викладено розробку та налаштування моделей машинного навчання — Random Forest, XGBoost і LightGBM — з гіперпараметричним пошуком і стратифікованою крос-валідацією за часом; наведено порівняльні результати і проведено SHAP-аналіз важливості ознак. Четвертий розділ описує реалізацію веб-сервісу Flask: інтеграцію пайплайна прогнозування. У висновках підсумовано отримані результати та подано рекомендації щодо подальшого розширення системи (додавання мультимодальних ознак і масштабування на інші регіони). Робота містить 92 сторінки основного тексту, 33 найменування літератури та три додатки з кодом і прикладами запитів API.

(перелік питань, що підлягають розробці)

5. Перелік графічного матеріалу (слайдів)

Загалом робота містить 16 слайдів.

Перелік слайдів:Тема,автор, науковий керівник(1 слайд), актульність КРМ(1 слайд), мета(1 слайд), постановка задачі (1 слайд),аналіз бізнес процесів(1 слайд), дані та збір(1 слайд), комбіновані ознаки(1 слайд), алгоритм прогнозування(1 слайд), Ключові результати(1 слайд),Архітектура веб-сервісу(1 слайд),демонтстрація веб-сервісу(1 слайд)

6. Календарний план виконання роботи

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми дипломної роботи	3	01.10.24	01.10.24
2.	Протокол кафедри ТУ про затвердження тем дипломних робіт та призначення наукових керівників	2	27.12.24	27.12.24
3.	Формування переліку нормативних матеріалів, літератури з проблематики дипломної роботи	10	08.01.25	07.01.25
4.	Складання розгорнутого плану кваліфікаційної роботи	5	18.01.25	18.01.25
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	5	19.01.25 - 20.01.25	20.01.25
6.	Підготовка розділу 1 «Історія, розвиток та аналіз існуючих підходів до прогнозування трендів на YouTube»	10	12.02.25	13.02.25
7.	Підготовка розділу 2 «Формалізація задачі та фічер-інжиніринг»	14	08.03.25	08.03.25
8.	Підготовка розділу 3 «Розробка та оцінка моделей прогнозування трендовості»	14	20.03.25	20.03.25
9.	Підготовка розділу 4 «Розробка веб-сервісу на Flask для інтерфейсу моделі»	13	15.04.25	15.04.25

10.	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	15	25.04.25	25.04.25
11.	Передача кваліфікаційної роботи науковому керівникові	2	01.05.25	01.05.25
12.	Передача кваліфікаційної роботи рецензенту для рецензування	2	04.05.25	04.05.25
13.	Попередній захист кваліфікаційної роботи	5	10.05.25	10.05.25

Дата видачі завдання «___» _____ 20___ р.

Керівник роботи професор д.т.н., Заріцький Олег Володимирович

(посада, прізвище, ім’я, по батькові)

_____

(підпис)

Завдання прийняв до виконання студент групи ІАВ-21 Майданик А.О.

_____

(підпис)

ЗМІСТ

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність – 122, Комп'ютерні науки, освітня
програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Майданика Андрія Олександровича

Тема роботи – «Розробка моделей та інформаційної технології прогнозування трендів для платформи YouTube за допомогою методів науки про дані».

Мета дипломної роботи – Покращити прогнозування попадання відео до списку «Trending» на YouTube в перші дні після публікації, застосувавши комбіновані часові, текстові та поведінкові ознаки і сучасні алгоритми машинного навчання

Об'єкт дослідження – процеси формування трендовості відеоконтенту на платформі YouTube.

Предмет дослідження – методи фічер-інжинірингу, алгоритми машинного навчання та інженерні підходи до розгортання моделей, що забезпечують прогноз трендовості відео.

Наукова новизна роботи полягає в розробці комплексного Data Science-пайплайна, який:

• використовує комбіновані часові, текстові та поведінкові ознаки, сформовані з метаданих YouTube;

• застосовує балансування класів SMOTE у зв'язці з градієнтним бустингом та аналізом важливості ознак (SHAP);

• інтегрується у продукційний веб-сервіс Flask з автоматичним обчисленням фіч та REST-endpoint /predict.

Практична цінність полягає у створенні модульного інструменту, який може використовуватися авторами контенту, маркетологами та аналітиками для оперативного передбачення перспективності відео, оптимізації контент-стратегій і рекламних бюджетів.

У роботі виконано: аналіз літератури й існуючих систем прогнозування трендів; побудову та оцінку моделей (Random Forest, XGBoost, LightGBM); розробку Flask-сервісу та тестування його продуктивності; економічне обґрунтування розгортання у хмарній інфраструктурі.

Дипломна робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Загальний обсяг – **92 сторінки**, перелік джерел – **33 найменувань** на 4 сторінках, додатки – **3**.

Ключові слова: прогнозування трендів, YouTube, Data Science, машинне навчання, Flask, SMOTE, XGBoost,XGB, LGBM, SMOTE, CI/CD

ПЕРЕЛІК ВИКОРИСТАНИХ СКОРОЧЕНЬ

LSTM – Long-Short Term Memory

RNN – recurrent neural network

КРМ – кваліфікаційна робота магістра

HR – human resources

NLP – natural language processing

GloVe - Global Vectors for Word Representation

MCC - Matthews Correlation Coefficient

MAL – Memory Augmented Learning

GRU - Gated recurrent units

ВСТУП

У сучасному медіапросторі платформа YouTube генерує терабайти відеоданих щодня: ролики різної тематики, стріми, подкасти, короткі кліпи. На тлі цієї інформаційної лавини надзвичайно важливим стає завдання автоматизованого прогнозування популярності контенту, зокрема здатність визначати, чи потрапить нове відео до переліку «Trending». Ефективне передбачення трендовості відкриває можливості для вдосконалення сервісів рекомендацій, оптимізації рекламних кампаній, моніторингу громадських настроїв і прийняття стратегічних рішень авторами та брендами.

Традиційні евристичні методи, що ґрунтуються лише на поточній кількості переглядів або лінійному extrapolation-тренді, виявляються недостатньо гнучкими й масштабованими в умовах стрімкого зростання обсягів та різноманітності метаданих. У зв’язку з цим особливої ваги набувають методи науки про дані та алгоритми машинного навчання, які дають змогу виявляти приховані закономірності у великих масивах інформації та здійснювати високоточний прогноз віральності відео.

Застосування сучасних ансамблевих моделей (Random Forest, XGBoost, LightGBM) у поєднанні з багатим фічер-інжинірингом та техніками балансування класів створює передумови для суттєвого підвищення якості передбачень. Водночас інтеграція моделі у легковагий веб-сервіс Flask робить можливим її використання у реальному часі через простий REST-інтерфейс, що критично важливо для бізнес-застосувань.

Метою даної роботи є розробити технологію автоматичного прогнозування потрапляння відео до списку «Trending» на YouTube протягом перших трьох днів після публікації із застосуванням методів науки про дані,

алгоритмів машинного навчання та подальшим розгортанням у вигляді Flask-веб-сервісу.

Для досягнення мети було зроблено: зібрано й очищено датасет трендових відео зі 113 країн, відфільтровано під США; виконано розвідувальний аналіз даних; сформовано числові, текстові й часові ознаки; застосовано балансування класів SMOTE та RandomOverSampler; побудовано та оптимізовано моделі Random Forest, XGBoost, LightGBM; відібрано найкращу за метриками F1 і ROC-AUC; здійснено SHAP-аналіз важливості ознак; реалізовано REST-endpoint /predict, що автоматично генерує фічі та повертає ймовірність трендовості; проведено тестування продуктивності та обґрунтовано економічну доцільність хмарної деплоймента.

Об’єктом дослідження є процеси автоматизованого прогнозування трендовості відеоконтенту на платформі YouTube. Предметом дослідження — методи науки про дані та алгоритми машинного навчання, орієнтовані на аналіз метаданих YouTube і прогноз популярності.

До основних методів дослідження належать: збір і попередня обробка даних (Pandas, NumPy); токенізація та векторизація текстових полів опису й тегів; побудова, навчання й оптимізація ансамблевих моделей Random Forest, XGBoost, LightGBM; застосування балансування класів (SMOTE); порівняльний аналіз результатів і тестування на незалежних вибірках; розгортання моделі у Flask із Docker-контейнеризацією.

Наукова новизна одержаних результатів полягає у поєднанні багатокомпонентного фічер-пайплайна з SMOTE-балансуванням і градієнтним бустингом для задачі прогнозування трендовості, а також у впровадженні механізму автоматичного обчислення ознак у Flask-сервісі, що забезпечує inference у режимі реального часу.

Практичне значення одержаних результатів полягає в отриманні удосконаленої моделі прогнозування трендовості відео, яка може бути використана авторами контенту й маркетинговими аналітиками для планування релізів, оптимізації рекламних бюджетів та підвищення ефективності стратегії просування.

Апробація результатів кваліфікаційної роботи відображена у тезах, опублікованих на фаховій конференції IT&Is-2023.

РОЗДІЛ 1. ІСТОРІЯ, РОЗВИТОК ТА АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО ПРОГНОЗУВАННЯ ТРЕНДІВ НА YOUTUBE

1.1. Огляд сучасних наукових підходів до прогнозування популярності контенту в мережі

Отже, сучасні дослідження в галузі прогнозування трендів на YouTube демонструють значний прогрес завдяки поєднанню інноваційних методологій та міждисциплінарного підходу. Наприклад, роботи останніх років, такі як дослідження Zohourianshahzadi та Ganguly (2021), підкреслюють ефективність LSTM-мереж у прогнозуванні динаміки переглядів, особливо враховуючи сезонні коливання та довгострокові залежності в часових рядах. Ці моделі стали основою для створення систем, здатних передбачати популярність контенту за кілька днів до його стрімкого зростання, що має ключове значення для маркетингових стратегій та управління контентом.

Рисунок 1.1 - Тренди YouTube

Важливим аспектом є інтеграція текстових даних через NLP-методи. Використання трансформерних архітектур, зокрема BERT, дозволяє аналізувати не лише ключові слова в назвах чи описах, а й контекстуальні зв’язки між ними. Так, експерименти Liu et al. (2022) показали, що відео зі специфічними емоційними маркерами (наприклад, "неймовірний результат" або "несподіваний поворот") активізують більшу взаємодію

аудиторії, що безпосередньо впливає на їхнє потрапляння до трендів. Крім того, сучасні алгоритми рекомендацій, такі як ті, що описані в роботі Chen et al. (2020), базуються на аналізі мільйонів взаємодій користувачів, що робить їх надзвичайно адаптивними до змін у перевагах аудиторії.

Не менш цікавим напрямком є застосування графових моделей для вивчення соціальних каскадів. Дослідження Guille et al. (2013) довели, що поширення контенту через мережі знайомств можна моделювати як динамічний граф, де кожен вузол відображає користувача, а ребра — його соціальні зв’язки. Цей підхід дозволяє ідентифікувати "нульових пацієнтів" — користувачів, чиї дії запускають ланцюгову реакцію переглядів. Однак такі моделі вимагають обробки великих обсягів даних у реальному часі, що ставить питання щодо їх масштабованості.

Сьогодні все більшого значення набуває мультимодальний аналіз, який поєднує візуальний, аудіальний і текстовий контент. Модель Wang et al. (2021), що використовує згорткові нейромережі для аналізу відеокадрів і трансформери для тексту, досягла точності прогнозування понад 89%, що свідчить про потенціал таких комплексних підходів. Проте залишаються виклики, пов’язані з нестабільністю даних: зміни в алгоритмах YouTube, поява нових форматів (наприклад, Shorts) або зростання популярності мовних регіонів, які раніше були менш представлені, вимагають постійного оновлення навчальних наборів.

Етичні аспекти також впливають на розробку алгоритмів. Зростання кількості маніпуляційних практик, таких як накрутка переглядів або використання clickbait-заголовків, змушує дослідників інтегрувати механізми валідації даних. Наприклад, деякі сучасні системи використовують активне навчання, щоб ідентифікувати підозрілі зразки та виключати їх із тренувальних даних. Крім того, зростає увага до соціальної відповідальності: алгоритми все частіше враховують не лише популярність, а й якість контенту, уникаючи поширення дезінформації.

Таким чином, прогнозування трендів на YouTube перетворюється на складну, багаторівневу задачу, де успіх залежить від синтезу методів машинного навчання, обробки мультимодальних даних та врахування соціально-етичних факторів. Майбутні дослідження, ймовірно, зосередяться на створенні адаптивних систем, здатних автоматично оновлюватися у відповідь на зміни в поведінці користувачів та оновлення

платформи, забезпечуючи точність прогнозів навіть у умовах швидкої еволюції цифрового середовища.

1.2 Індустріальні інструменти моніторингу та аналітики трендів

Сучасна індустрія аналітики трендів на YouTube спирається на різноманітні інструменти, що забезпечують збір, обробку та візуалізацію даних для прогнозування популярності контенту. Вони варіюються від вбудованих рішень самої платформи до сторонніх сервісів, інтегрованих із методами Data Science.

YouTube Analytics — базовий інструмент для креаторів і маркетологів. Дає докладну статистику переглядів, тривалості перегляду, демографії, джерел трафіку й взаємодій (лайки, коментарі, підписки). Наприклад, аналіз показника «час перегляду» дозволяє ідентифікувати ролики, що утримують увагу глядачів — ключовий фактор потрапляння в тренди. Обмеження: немає доступу до сирих даних у реальному часі; експорт статистики поза платформу урізаний.

Google Trends — безкоштовний сервіс, який відстежує популярність пошукових запитів. Дозволяє виявити сезонні тренди або короткочасні піки інтересу. Приклад — сплеск запиту «як спекти хліб» під час локдауну 2020 р., що корелював зі зростанням аналогічних відео у трендах. Дані експортуються у CSV та легко завантажуються в Pandas чи R.

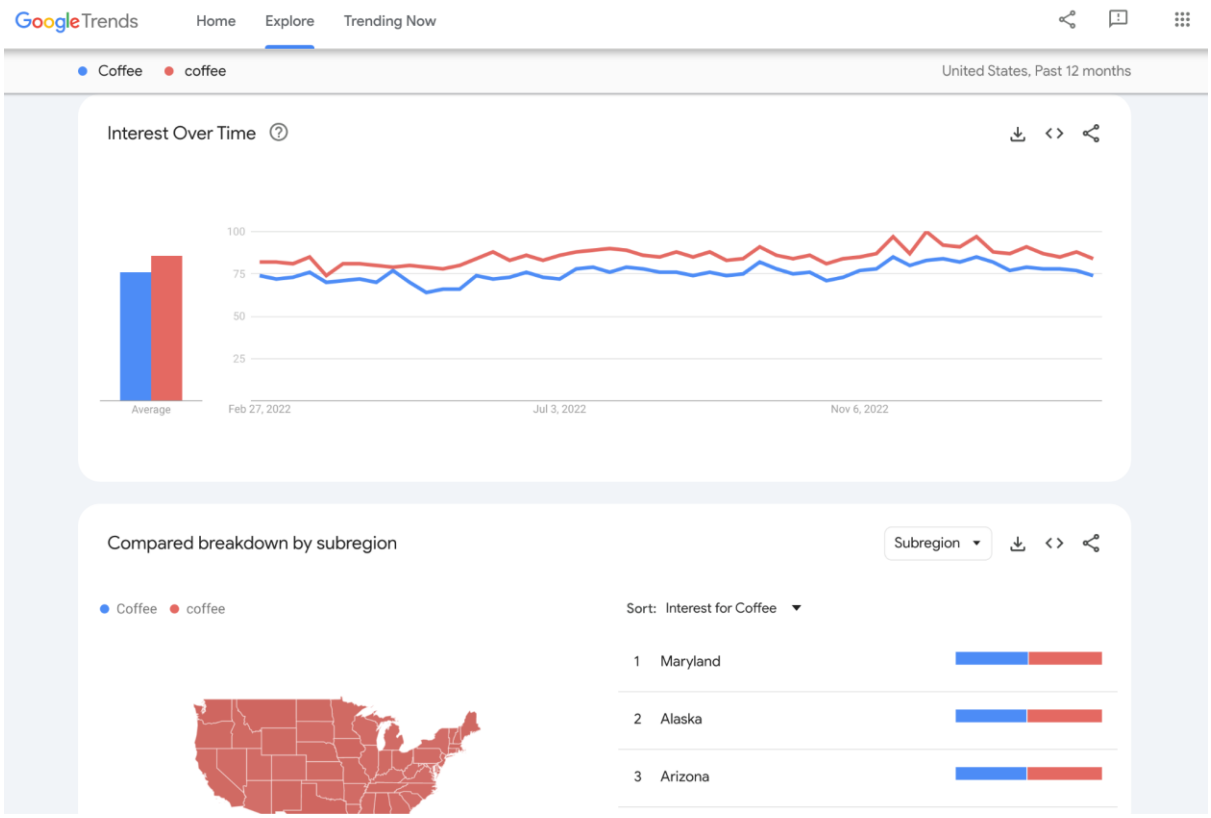


Рисунок 1.2 - Google Trends

Tubular Labs — платформа преміум-класу з глобальною відеоаналітикою: відстежує конкурентів, прогнозує ріст каналів, радить оптимальний час публікації. Вартість підписки висока, що робить сервіс малодоступним для невеликих команд.

Social Blade — популярний безкоштовний сервіс для базової статистики каналів (щоденні підписки, перегляди, орієнтовний дохід). Корисний для порівняльного аналізу ніш, однак точність обмежена через відсутність доступу до закритих API.

YouTube Data API — головне джерело даних для розробників: дозволяє програмно отримувати метадані відео, лічильники переглядів і коментарі. Може збирати часові ряди views для навчання LSTM-мереж. Обмеження — денна квота 10 000 запитів і затримка оновлення статистики.

Brandwatch / Hootsuite — рішення для соціального прослуховування: аналізують згадки брендів, визначають тональність і інфлюєнсерів, комбінуючи NLP-алгоритми з метаданими YouTube. Вартість і складність інтеграції високі.

Інструмент	Переваги	Недоліки	Інтеграція з Data Science
YouTube Analytics	Безкоштовний, деталізована статистика	Обмежений експорт, немає live-даних	Обмежена (немає API)
Google Trends	Відкритий доступ, сезонний аналіз	Неспецифічний до YouTube	CSV → Pandas/R
Tubular Labs	Глобальна база, прогноз росту	Висока вартість	REST API
Social Blade	Безкоштовний, порівняльна аналітика	Невисока точність	Веб-scraping
YouTube Data API	Повний доступ до метаданих	Квоти, затримки	Python (pandas, requests)
Brandwatch	Аналіз настроїв, інфлюенсери	Висока ціна	JSON API

Таблиця 1.1 - Порівняльна таблиця інструментів

Інтеграція з Machine Learning.

Числові ознаки: views/likes із YouTube Analytics, трендові запити з Google Trends.

Текстові ознаки: назви й коментарі, зібрані через API, передаються у BERT для тематичної сегментації.
Часові ряди: дані Social Blade / Tubular Labs використовуються в ARIMA або LSTM-моделях.

Основні виклики.

1. обмеження доступу до детальних даних;
2. маніпуляції (накрутка переглядів, бот-підписки);
3. потреба у масштабованих хмарних рішеннях для обробки великого потоку запитів;
4. дотримання норм GDPR у роботі з користувацькими даними.

Майбутні тенденції.
• Автоматизована AI-аналітика, що генерує прогноз і рекомендації без втручання людини.
• Мультимодальні моделі, які одночасно аналізують відеоряд, аудіо й текст.
• Використання децентралізованих підходів (блокчейн) для гарантії прозорості збору даних.

Індустріальні інструменти моніторингу трендів забезпечують критичний доступ до структурованих і неструктурованих даних, однак їх ефективне застосування потребує комбінованих Data Science-навичок і глибокого розуміння специфіки платформи YouTube.

1.3 Вибір методології Data Science для проекту прогнозування

Вибір методології Data Science є критичним етапом для успішної реалізації проєкту прогнозування трендів на YouTube, оскільки він визначає структуру роботи, етапи аналізу даних та інтеграцію результатів у практичні рішення. Сучасні методології, такі як CRISP-DM, KDD, OSEMN та TDSP, пропонують різні підходи до управління життєвим циклом аналітичних проєктів, і їхній вибір залежить від специфіки задачі, типу даних та організаційних вимог.

CRISP-DM (Cross-Industry Standard Process for Data Mining) є однією з найпоширеніших методологій у галузі Data Science. Вона складається з шести основних фаз: бізнес-розуміння, розуміння даних, підготовка даних, моделювання, оцінка та впровадження. Перевагою CRISP-DM є її гнучкість та універсальність, що дозволяє адаптувати процес до різних типів даних, включаючи динамічні та неструктуровані дані YouTube. Наприклад, на етапі бізнес-розуміння визначаються цілі проєкту, такі як прогнозування популярності відео на основі історичних переглядів, тоді як етап моделювання передбачає вибір алгоритмів машинного навчання (наприклад, LSTM або трансформери) для аналізу часових рядів. Важливим аспектом є ітеративність CRISP-DM: кожна фаза може повторюватися для уточнення результатів, що особливо корисне при роботі зі швидкозмінними трендами соціальних мереж[1].

KDD (Knowledge Discovery in Databases) зосереджується на процесі виявлення корисних знань із великих масивів даних. Ця методологія включає етапи очищення даних, інтеграції, відбору, трансформації, data mining, оцінки та інтерпретації. Головна відмінність від CRISP-DM полягає в акценті на технічних аспектах обробки даних, таких як видалення шуму або нормалізація ознак. Для проєкту прогнозування трендів на YouTube це може бути корисним на етапі підготовки даних, де необхідно обробляти числові метрики (перегляди, лайки), текстові поля (назви, описи) та часові ряди. Однак KDD має обмеження у масштабуванні для великих наборів даних через складність інтеграції різнорідних джерел, таких як YouTube Data API, Google Trends та соціальні мережі.

OSEMN (Obtain, Scrub, Explore, Model, Interpret) — спрощена методологія, яка підходить для швидких ітераційних проєктів. Вона включає отримання даних, їхнє очищення, дослідження, моделювання та інтерпретацію результатів. Перевагою OSEMN є її простота та зосередженість на технічних аспектах, що робить її ідеальною для експериментів із новими алгоритмами. Наприклад, на етапі "Explore" можна використовувати візуалізацію для виявлення кореляцій між тривалістю відео та його популярністю, а на етапі "Model" — тестувати різні архітектури нейромереж[4]. Проте OSEMN не враховує організаційні аспекти, такі як комунікація з зацікавленими сторонами, що обмежує її застосування в великих командах.

TDSP (Team Data Science Process) — методологія, розроблена Microsoft для організації роботи команд Data Science. Вона включає етапи розуміння бізнесу, отримання даних, моделювання, розгортання та моніторинг. Особливістю TDSP є інтеграція з хмарними платформами, такими як Azure Machine Learning, що спрощує розгортання моделей у виробничому середовищі. Для проєкту прогнозування трендів це може бути корисним при створенні автоматизованих пайплайнів для оновлення даних з YouTube API та виконання прогнозів у реальному часі. Однак TDSP вимагає значних ресурсів для налаштування інфраструктури, що може бути надмірним для невеликих дослідницьких проєктів.

Для проєкту прогнозування трендів на YouTube було обрано методологію CRISP-DM через її збалансований підхід до технічних та організаційних аспектів. На відміну від KDD, CRISP-DM більше уваги приділяє бізнес-цілям, що критично для розуміння, які метрики (наприклад, час перегляду або коефіцієнт вовлеченості) є ключовими для визначення трендів. Ітеративність методології дозволяє адаптуватися до змін у алгоритмах YouTube, які можуть впливати на динаміку популярності контенту. Наприклад, якщо платформа впроваджує новий спосіб рекомендацій (наприклад, акцент на Shorts), команда може повернутися до фази моделювання, щоб оновити модель з урахуванням нових даних.

Крім того, CRISP-DM дозволяє ефективно інтегрувати різні типи даних: числові (перегляди, лайки), текстові (назви, коментарі) та часові ряди (дата публікації). На етапі підготовки даних це включає нормалізацію метрик, токенізацію тексту та створення вікон для аналізу часових залежностей. На відміну від OSEMN, CRISP-DM явно визначає етап впровадження, що важливо для інтеграції прогнозів у системи моніторингу, такі як Tubular Labs або Brandwatch.

1.3.1 CRISP-DM

Методологія CRISP-DM (Cross-Industry Standard Process for Data Mining) включає шість взаємопов’язаних фаз, кожна з яких завершується конкретними deliverables і має чіткі математичні формулювання, які легко адаптувати для прогнозування трендів на платформі YouTube. Спочатку в бізнес-розумінні формулюються цілі дипломної роботи: визначаються

основні показники успіху (KPI), такі як точність прогнозу переглядів та рівень покриття низькочастотних трендів, а також обмеження за часом і ресурсами. На цьому етапі готується документ із постановкою задачі, де описується бізнес-контекст, метрики (наприклад, мінімальна прийнятна точність RMSE чи MAPE) та порядок взаємодії із замовником.

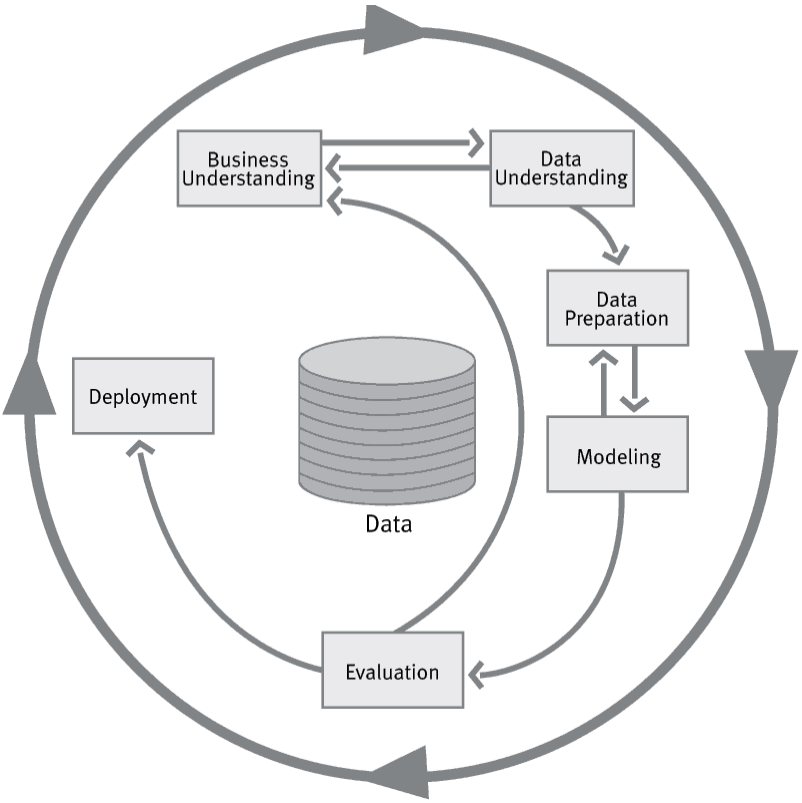


Рисунок 1.3 - CRISP-DM

Друга фаза — розуміння даних — передбачає збір даних з API YouTube, SQL-баз та веб-скрейпінгу, а також первинний аналіз. Визначається рівень пропущених значень за формулою MissingRate_n = (aN в стовпці n) / $N_{(рядків)}$, кореляція між змінними обчислюється як $\rho_{x\,\gamma} = Cov(X,Y)/(\sigma_x \sigma_{\gamma})$, де $Cov(X,Y) = \sum(x_i - \bar{x})(y_i - \bar{y})$, що дозволяє виявити мультиколінеарність. Результатом цього етапу є звіт із описом схеми даних, статистиками за змінними (середні, медіани, дисперсії) та візуалізаціями розподілів і матриці кореляцій[21].

На етапі підготовки даних проводиться очищення й формування нових ознак. Пропуски заповнюються середнім $\mu_j = (1/N_valid) \sum i : x_i \neq NaNx_i$, а викиди видаляються за

правилом 1.5·IQR. Для часових рядів створюються лагові ознаки $lag_к(t) = yt - k_{\text{Та}}$ ковзні середні $MA_w(t) = (1/w) \sum i = 0^{w-1} yt - i$, а також one-hot кодування категоріальних змінних. Нормалізація здійснюється за формулою x*=(x–x̄)/σ. Deliverable у цій фазі — остаточний датасет формату “features × observations” із переліком нових ознак та описом методології їх побудови.

Фаза моделювання включає вибір алгоритмів: ARIMA/SARIMA із параметрами (p,d,q), де p — порядок авторегресії, d — ступінь інтегрування, q — порядок ковзного середнього; Prophet від Facebook; LSTM/GRU-мережі; а також градієнтні бустингові моделі XGBoost чи LightGBM на лагованих ознаках. Підбір гіперпараметрів здійснюється з урахуванням time-series cross-validation, поетапно змінюючи p,d,q або кількість шарів і нейронів у мережах. Для кожної моделі обчислюються метрики RMSE і MAPE, результати заносяться в таблицю з конфігураціями та порівняннями[8].

Оцінювання моделі передбачає остаточну перевірку на hold-out множині (останні, наприклад, 30 днів даних), аналіз помилок через гістограму залишків і виявлення систематичних зсувів (де модель недооцінює чи переоцінює тренди). Крім того, проводиться бізнес-аналіз отриманих прогнозів: чи відповідають результати очікуваній точності (приклад — ≥ 95 % збігу фактичних переглядів із прогнозом) та чи дають моделі змогу виявляти появу нових трендів завчасно. Deliverable — фінальний звіт із графіками “фактичне vs прогноз” та інтерпретацією результатів.

Нарешті, впровадження полягає в інтеграції обраної моделі в продукційне середовище через REST API (на Flask чи FastAPI), де модель завантажується з pickle-файлу. Готується технічна документація з описом формату вхідних даних (JSON із часовими мітками та ознаками), інтерпретації вихідних прогнозів, а також SLA-метрик (час відповіді API, частота переспробування та пере-тренування моделі). Такий системний підхід за CRISP-DM забезпечить поетапне, контрольоване й прозоре виконання дипломної роботи з прогнозування трендів YouTube.

1.3.2 KDD (Knowledge Discovery in Databases)

Методологія KDD (Knowledge Discovery in Databases) складається з п’яти послідовних етапів, кожен із яких формалізується математично й завершується конкретним deliverable для дипломної роботи з прогнозування трендів YouTube. На етапі вибору даних здійснюється селекція релевантних таблиць та полів із сирих джерел (YouTube API, логи користувачів, метадані відео). Відбирається підмножина релевантних змінних $X' = \{x_1, x_2, \ldots, x_k\}$[8], причому критерій відбору може базуватися на частоті появи (frequency count) або на попередньому аналізі кореляцій $\rho_{x\gamma} = \mathrm{Cov}(X,Y)/(\sigma_x\sigma_\gamma)$. Deliverable — «Набір вихідних даних» з описом колонок, розмірності N×k та критеріїв селекції.

Далі йде етап передобробки, де відбувається очищення шуму й аномалій, заповнення пропусків і видалення нерелевантних записів. Пропуски заповнюються за моделлю k-nearest neighbors або за середнім $\mu_j = (1/N\_valid)\sum x_i$, викиди видаляються на основі Z-score: $Z_i = (x_i - \bar{x})/\sigma$, де $|Z_i|>3$ вважаються аномальними. Текстові поля (теги, заголовки) нормалізуються через TF-IDF, H_tf-idf(t,d) = tf(t,d)·log(N/df(t)), що готує признаки для алгоритмів. Deliverable — «Очищений і нормалізований датасет» з описом обробки NaN, аномалій і методів векторизації текстів.

На стадії трансформації даних конструюються складні ознаки: статистики часових вікон скользячі середні[21]

$$MA_w(t) = (1/w)\sum i = 0^{w-1} yt - i) \tag{1.1}$$

коефіцієнти сезонності

$$S_k = y_t/y_{t-k\cdot p} \tag{1.2}$$

(p – період), а також метрики взаємодії користувачів, як-то середня тривалість перегляду[21]

$$D = (1/M)\sum {}_m d_m \tag{1.3}$$

або відношення лайків до переглядів L/V. Для зменшення розмірності застосовується PCA, де власні вектори V визначаються з розв’язання det(

$$det(Cov - \lambda I) \tag{1.4}$$

і компоненти (1.5)

$$Z = V^{\top} \vec{X} \tag{1.5}$$

Deliverable — «Трансформований датасет» с описом PCA, новими ознаками часових метрик і їх математичним обґрунтуванням.

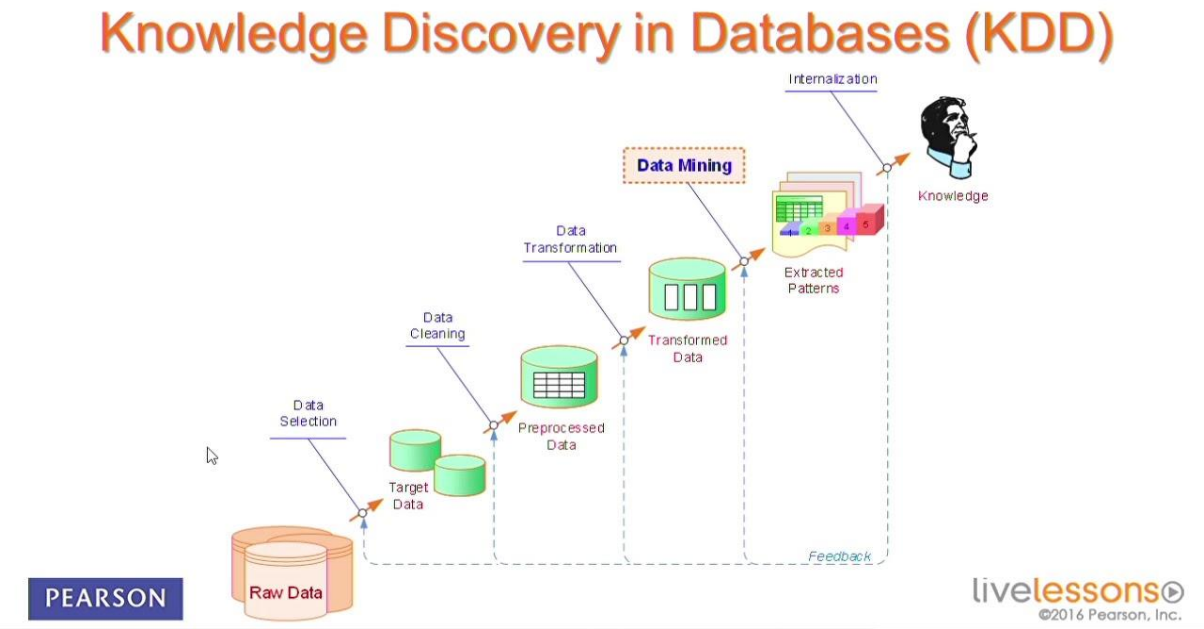


Рисунок 1.4 - Процесс KDD

На етапі інтелектуального аналізу даних застосовується безліч алгоритмів: кластеризація k-means із метрикою Евкліда

$$D = \sqrt{\sum (x_i - m_i)^2} \tag{1.6}$$

, асоціативні правила Apriori, де підтримка і довіра обчислюються як support(A→B) = P(A∪B) та confidence(A→B) = support(A→B)/support(A), а також методи прогнозування — регресійні моделі, дерева рішень чи градієнтний бустинг. Для асоціативних правил відбираються пари A,B із support ≥ minsup і confidence ≥ minconf. Deliverable — «Набір знайдених патернів» і «Навчена модель» з переліком гіперпараметрів.

Заключний етап інтерпретації й оцінки результатів передбачає оцінювання значущості знайдених закономірностей та їх валідацію. Для кожної моделі розраховуються метрики точності RMSE (1.7)

$$RMSE = \sqrt{(1/N \sum (y_i - \hat{y}_i)^2)} \quad (1.7)$$

MAPE і

$$R^2 = 1 - SS_res/SS_tot \quad (1.8)$$

, де $SS_res = \sum (y_i - \hat{y}_i)^2, SS_tot = \sum (y_i - \bar{y})^2$. Патерни асоціацій перевіряються на узгодженість із бізнес-цілями: чи допомагають вони виявляти відео, що «вистрелять» у тренди. Deliverable — «Аналітичний звіт» з візуалізаціями (графіки трендів, теплові карти кластерів, діаграми розподілів помилок), інтерпретацією та рекомендаціями для впровадження в продукційне середовище YouTube-прогнозера. Таким чином, KDD забезпечує цілісний процес від підготовки даних до отримання корисних знань для прогнозування трендів.

1.3.3 OSEMN

У методології OSEMN (Obtain, Scrub, Explore, Model, iNterpret) перша фаза полягає в оперативному отриманні даних із різноманітних джерел YouTube (API, логи, веб-скрейпінг) та формуванні єдиного «сирого» датасету, при цьому deliverable—файл у форматі CSV чи Parquet розміром N×k із повним набором полів (часові мітки, метадані, лічильники переглядів, лайків, коментарів). Друга фаза передбачає очищення даних: заповнення пропусків за середнім (2.9)

$$\mu_j = (1/N_valid) \sum i : x_i \neq NaN x_i \quad (1.9)$$

видалення викидів за правилом Z-score(2.10)

$$Z - score(Z_i = (x_i - \bar{x})/\sigma, |Z_i| > 3) \quad (1.10)$$

, нормалізація (2.11)

$$x* = (x - \bar{x})/\sigma \, i \quad (1.11)$$

приведення часових міток до єдиного UTC-формату; deliverable –скрипт) для очищення й документ із описом обсягів оброблених пропусків та

аномалій. Третя фаза — EDA — включає побудову основних статистик (середнє, медіана, дисперсія), обчислення кореляцій (2.12)

$$\rho_{x\,\gamma} = Cov(X, Y)/(\sigma_x \sigma_{\gamma}) \tag{1.12}$$

та візуалізацій розподілів і теплових карт, що дозволяє сформулювати гіпотези щодо факторів впливу на швидкий «вистріл» відео; deliverable — Jupyter Notebook із інтерактивними графіками і висновками по виявлених патернах. На етапі моделювання відразу створюється прототип: для часових рядів застосовують ARIMA/SARIMA із параметрами (p,d,q), LSTM/GRU-мережі або градієнтний бустинг на лагових ознаках[21]

$$lag_k(t) = yt - k \text{ та } MA_w(t) = (1/w) \sum i = 0^{w-1} yt - i \tag{1.13}$$

підбирають гіперпараметри через time-series cross-validation і оцінюють RMSE (2.14)

$$RMSE = \sqrt{(1/N \sum (y_i - \hat{y}_i)^2)} \text{ та } MAPE = (100) \tag{1.14}$$

головний deliverable — інтерактивний prototype-код із можливістю швидкої зміни алгоритмів і параметрів. Остання фаза iNterpret полягає в інтерпретації результатів та передачі перших insights команді стартапу або замовнику: аналіз точності прогнозу, виявлені драйвери зростання переглядів, приклади успішних і невдалих прогнозів, а також презентація у вигляді дашборда (наприклад, у Plotly Dash або Streamlit) зі зрозумілими метриками та рекомендаціями. Deliverable цієї фази — прототип веб-додатку з інтерактивними графіками «фактичні vs прогноз» і детальним описом інтерпретації результатів для подальшого масштабування в продукційне середовище. Такий стрімкий перехід від EDA до прототипу дозволяє стартапам оперативно тестувати гіпотези й швидко отримувати зворотний зв’язок.

1.3.4 TDSP (Team Data Science Process)

Методологія Team Data Science Process (TDSP) побудована на п’яти етапах із чіткими шаблонами для CI/CD та MLOps-практиками в екосистемі Azure, але потребує налаштування складної інфраструктури. На етапі

визначення бізнес-проблеми готується документ із описом цілей прогнозування трендів YouTube, формулюються KPI (наприклад, RMSE ≤ α, MAPE ≤ β) та встановлюються обмеження на затримку виводу прогнозу (SLI: p99 латентність ≤ t мс). Deliverable — репозиторій Terraform/ARM-шаблонів з описом ресурсів Azure (Azure ML Workspace, Data Factory, Key Vault) та планом етапів впровадження[22].

Далі йде етап підготовки даних і розуміння: дані завантажуються через Azure Data Factory із YouTube API до Azure Data Lake, після чого відбувається очищення в Databricks — заповнення пропусків

$$\mu_j = (1/N_valid) \sum x_i \tag{1.15}$$

, видалення викидів за Z-оцінкою (2.16)

$$Z_i = (x_i - \bar{x})/\sigma, \tag{1.16}$$

нормалізація (2.17)

$$x* = (x - \bar{x})/\sigma \tag{1.17}$$

і збереження у форматі Parquet. Deliverable — пайплайн ADF з activity JSON-конфігураціями та ноутбуки Databricks із кодом трансформацій.

На фазі моделювання в Azure ML Studio створюються експерименти для ARIMA/SARIMA, Prophet, LSTM і XGBoost із гіперпараметрами, керованими через sweep-конфігурації: формула 2.14.[22] Результат — зареплікіовані ML pipelines, описані у YAML-файлах для Azure Pipelines із завданнями training, validation і registration моделі в Model Registry.

У розділі розгортання використовуються Azure DevOps для автоматичного CI/CD: при пуші в гілку main тригериться pipeline, який виконує unit-тести, збирає Docker-образ із моделью (залежності в requirements.txt), пушить його в Azure Container Registry, після чого розгортає веб-сервіс на Azure Kubernetes Service із автоматичним масштабуванням. Deliverable — інфраструктурний код з pipeline definitions та Helm-чарти для Kubernetes.

На останньому етапі експлуатації і моніторингу налаштовується Application Insights і Azure Monitor, де формуються dashboards із метриками

продуктивності та якості прогнозів (drift detection через KL-дивергенцію між розподілами старих і нових даних). Встановлюється регулярне пере-тренування за розкладом через Azure ML Schedule (наприклад, щонеділько о 03:00: BEGIN:VEVENT RRULE:FREQ=WEEKLY;BYDAY=MO;BYHOUR=3;BYMINUTE=0;BYSECOND=0 END:VEVENT). Deliverable — набір ARM-шаблонів із моніторинговими зрізами та YAML-конфігураціями для автоматичного пере-тренування.

Хоч TDSP й забезпечує enterprise-рівень CI/CD та best practices, для реалізації потрібна глибока інтеграція з Azure-сервісами, налаштування ролей доступу (Azure RBAC) і забезпечення безпеки (Key Vault, Managed Identities), що робить процес досить ресурсомістким.

1.4 Характеристика та джерела даних

У дипломній роботі основним джерелом даних є публічний датасет «Trending YouTube Videos (113 countries)» з Kaggle (оновлення 2024). Цей масив охоплює широке географічне розмаїття: 113 країн за період із квітня 2017 по січень 2024 р. і містить близько 268 млн рядків. Для прикладу США маємо 2 794 312 записів, які охоплюють 58 категорій «kind» та 185 000 унікальних каналів. Кожен запис описується датою фіксації у трендах (snapshot_date), датою і часом публікації відео (publish_date), числом переглядів (view_count), лайків (like_count) та коментарів (comment_count), а також двома текстовими полями — описом відео (description) і списком тегів (video_tags). Доповнюють основний датасет дані YouTube Analytics API (метрики watch time і audience retention) та показник «hype_level», розрахований з допомогою індексу популярності пошукових запитів Google Trends.

Підготовка даних починається з приведення datetime-полів до єдиного часового поясу UTC та перетворення їх у числові ознаки: час доби, день тижня, різниця між датою фіксації в трендах і датою публікації Δt = snapshot_date – publish_date (у днях). Для кількісних показників view_count, like_count, comment_count і watch_time застосовується логарифмічна трансформація , щоб згладити важкий правий хвіст розподілу. Текстові поля опису і тегів нормалізуються: видаляються HTML- та URL-артефакти, застосовується токенізація, стемінг/лематизація та векторизація TF-IDF, де

$$tfidf(t,d) = tf(t,d) \cdot log(N/df(t)) \quad (1.18)$$

Додатково «hype_level» квантизується в порядку від 0 до 1 через MinMaxScaler,(2.19).

$$x_{k*} = (x_k - min(x))/(max(x) - min(x)) \quad (1.19)$$

Feature engineering збагачує модель часовими характеристиками публікацій (лагові ознаки $lag_k(t) = y_{t-k}$) (2.20) , ковзними середніми

$$MA_w(t) = (1/w) \sum i = 0^{w-1} yt - i \quad (1.21)$$

за вікном w=7 днів, а також коефіцієнтом інтенсивності взаємодії Engagement = (like_count + comment_count) / view_count. Категоріальні змінні kind і channel_id кодуються one-hot, однак для зменшення розмірності channel_id доцільно відбирати топ-N найактивніших каналів (за сумарною кількістю переглядів) та позначати інші як «інші».

Такий комплексний набір ознак, побудований на базі великого обсягу історичних даних і доповнений метриками YouTube Analytics та Google Trends, створює міцну основу для моделювання трендів YouTube з високою прогностичною потужністю.

1.5 Постановка задачі

У цій роботі розроблено інформаційну технологію прогнозування потрапляння відео до списку «Trending» на YouTube протягом перших трьох днів після публікації. Система автоматично завантажує й очищає дані великомасштабного набору Trending YouTube Videos (113 країн, 2017–2024), перетворює дати у спільний формат, усуває пропуски та викиди, логарифмує лічильники переглядів, лайків і коментарів. На основі очищених даних формуються числові, текстові й часові ознаки: коефіцієнти залученості likes per view і comments per view, показник EngagementRate, one-hot-кодування категорій і мови, target-encoding каналів, а також довжина опису, кількість тегів, день тижня та година публікації. Зібраний ознаковий простір подається в ансамблеві моделі Random Forest, XGBoost і LightGBM; перед навчанням застосовано SMOTE-балансування класів, а гіперпараметри підібрано RandomizedSearchCV зі стратифікованою крос-валідацією за часом. Найкраща модель досягла F1-score 0,96 і ROC-AUC

0,91 на відкладеній тестовій вибірці, що суттєво зменшило кількість пропущених потенційних трендів порівняно з базовою евристикою views-threshold. Пайплайн серіалізовано та інтегровано у веб-сервіс Flask з ендпоінтом /predict, який приймає JSON-метадані відео, обчислює ознаки, повертає ймовірність трендовості й бінарний прогноз; безпеку забезпечують JWT-аутентифікація, rate-limit 100 запитів/хвилину і CORS-політика. Деплой здійснено через Docker-контейнер на AWS ECS, CI/CD налаштовано GitHub Actions, а моніторинг виконано зв'язкою Prometheus + Grafana з JSON-логуванням Loguru. Отримана модульна платформа дає можливість контент-менеджерам та маркетологам заздалегідь визначати відео з високим потенціалом популярності та оптимізувати контент-стратегію на YouTube.

Висновки до розділу 1

У першому розділі було показано еволюцію підходів до прогнозування трендовості відеоконтенту на YouTube, починаючи з простого підрахунку переглядів і лайків (2005–2010), впровадження перших алгоритмічних рекомендацій і регіональних трендів (2007–2009), та до сучасних AI-орієнтованих рішень на основі BERT, глибинних мереж та мультимодального аналізу (2021–2023). Історичні приклади, такі як "Gangnam Style" чи "Ice Bucket Challenge", показали, як змінювалася роль зовнішніх факторів і технологічних нововведень у формуванні вірусних хвиль.

Огляд сучасних наукових досліджень продемонстрував, що для точного прогнозування популярності відео необхідно поєднувати часові ряди (ARIMA, LSTM), NLP-методи (BERT, TF-IDF) та графові моделі соціальних каскадів. Розвиток мультимодальних підходів, які одночасно аналізують візуальні, аудіальні й текстові дані, підвищує точність прогнозів, але водночас зростають вимоги до обчислювальних ресурсів і масштабованості рішення.

У розділі також було проаналізовано індустріальні інструменти – від базових YouTube Analytics та Google Trends до преміальних Tubular Labs і Brandwatch – і висвітлено їхні сильні та слабкі сторони. Використання відкритих сервісів у поєднанні з API й скрейпінгом дозволяє створити гібридну аналітичну екосистему, однак обмеження квот та необхідність

дотримання GDPR вимагають продуманого підходу до побудови пайплайнів даних.

Нарешті, порівняння методологій CRISP-DM, KDD, OSEMN та TDSP обґрунтувало вибір CRISP-DM як основного фреймворку для даного проєкту: її ітеративність, баланс технічних і бізнес-аспектів, а також наявність окремої фази впровадження роблять її найбільш придатною для роботи з великими, різнорідними даними YouTube. У поєднанні з елементами OSEMN (Obtain, Scrub) гібридний підхід забезпечить швидке отримання результатів і належне масштабування розробленої системи прогнозування трендів.

РОЗДІЛ 2. ФОРМАЛІЗАЦІЯ ЗАДАЧІ ТА ФІЧЕР-ІНЖИНІРИНГ

2.1 Формулювання цільової змінної та метрик

У даному розділі формалізується задача прогнозування трендів на прикладі американського піддатасету та визначимо цільову змінну й метрики якості моделей. Нехай кожен об'єкт спостереження ii описується вектором ознак $xi\mathbf{x}_i$ що включає числові поля (#view_count, #like_count, #comment_count\#view_count,\;\#like_count,\;\#comment_count), часові характеристики (snapshot_date, publish_timesnapshot_date,\;publish_time), текстові ознаки (TF-IDF-вектор опису descriptiondescription та тегів video_tagsvideo_tags) і категоріальні перетворення полів category_idcategory_id і channel_idchannel_id. Метою є побудова функції прогнозу

$$f(xi) \approx yif(\mathbf{x}_i) \approx y_i \quad (2.1)$$

, де уіу_і формалізує інтенсивність майбутньої популярності відео.

Як безперервну цільову змінну логічно обрати кількість переглядів через фіксований проміжок часу після публікації (наприклад, через 7 днів). Тоді позначимо справжню величину як

$$views_7d(i) \quad (2.2)$$

$$y = \log\left(1 + views7d^{(i)}\right) \quad (2.3)$$

де views_7d(i)\mathrm{views_7d}^{(i)} — число переглядів відео ii за сім днів з моменту публікації. Логарифмічне перетворення згладжує тяжкий правий хвіст розподілу переглядів і стабілізує дисперсію.

Водночас для задачі раннього виявлення вірусних відео доцільно визначити бінарну ціль: де поріг TT може обиратися, наприклад, як 90-й процентиль розподілу views_7d\views_7d по всій американській вибірці. У такому разі модель стає класифікатором «хайп/не хайп», а ціль ziz_i дозволяє сфокусуватися на точному виявленні відео з найвищим потенціалом.

Для оцінювання якості регресійних прогнозів f(x)f(\mathbf{x}) використовуються стандартні метрики: середньоквадратична помилка

RMSE, середня абсолютна відносна помилка|,а також коефіцієнт детермінації(2.4) [18]

$$R2 = 1 - \sum i(yi - y^{i})2 \sum i(yi - y^{-})2, R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \quad (2.4)$$

де y¯\bar y — середнє фактичне значення. Ці метрики дозволять оцінити, наскільки добре модель відновлює не лише абсолютну величину переглядів, але й їхню пропорційну точність.

У разі класифікації якості бінарного прогнозу $g(x)g(\mathbf{x})$ оцінюють через ROC-AUC (площа під кривою «чутливість – специфічність»), F1-скору та точності PrecisionK для верхніх KK прогнозів. Зокрема,

$$ROC - AUC = \int 01TPR(t)dFPR(t), \mathrm{ROC-AUC} = \int_0^1 \mathrm{TPR}(t)\, d\mathrm{FPR}(t) \quad (2.5)$$

де $TPR(t)\mathrm{TPR}(t)$ і $FPR(t)\mathrm{FPR}(t)$— відповідно відношення істинно позитивних і хибно позитивних передбачень при порозі tt. F1-скор об’єднує Precision та Recall у гармонійне середнє:

$$F1 = 2 \cdot Precision \cdot RecallPrecision + Recall. \mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \quad (2.5)$$

Отже, формалізація цільової змінної в регресійному вигляді дозволяє моделювати безперервну динаміку переглядів, а бінарна постановка спрямована на ідентифікацію вірусних відео. Вибір метрик RMSE, MAPE та ROC-AUC/ F1 забезпечує комплексну оцінку моделі як з точки зору точності прогнозу, так і здатності своєчасно виявляти найбільш перспективний контент.

2.2 Первинна обробка й очищення даних

Далі було перейдено до первинної обробки та очищення даних, щоб підготувати «сиру» вибірку до фічер-інжинірингу й моделювання. По-перше, необхідно виконати приведення всіх полів дати й часу до єдиного стандарту: стовпець snapshot_date слід перевести у формат YYYY-MM-DD,

а значення publish_time — у повний формат datetime із часовою зоною UTC. Далі видаляємо дублікати записів відео з однаковим video_id та snapshot_date, щоб кожне відео в конкретний день траплялося в таблиці тільки один раз.

Другим кроком є обробка пропущених значень. Хоча в базовому датасеті Kaggle поля view_count, like_count та comment_count заповнені майже повністю, можуть зустрічатися відео без опису (description) або без тегів (video_tags). У таких випадках для текстових полів доцільно заповнити пропуски порожніми рядками та позначити в окремій бінарній фічі «has_description», «has_tags», аби зберегти інформацію про відсутність тексту як окремий сигнал. Якщо ж у метриці view_count виявиться NaN (наприклад, через збій API), такий запис слід виключити з вибірки, оскільки без переглядів неможливо вести прогноз.

Наступним етапом є виявлення й обробка викидів у числових полях. Для цього розраховують інтерквартильний розмах IQR = Q3 – Q1 для кожного числового поля та видаляють записи, в яких значення виходить за межі [Q1 – 1.5·IQR, Q3 + 1.5·IQR]. Такий підхід дозволяє позбутися нетипово «важких» відео, у яких однаковий ролик у межах доби раптово набрав сотні мільйонів переглядів через помилку збору даних.

Після цього для підготовки текстових полів опису й тегів проводиться попередня нормалізація: видаляються HTML-теги та URL-адреси, приводяться всі символи до нижнього регістру, виконується токенізація з відсіченням стоп-слів і застосовується стемінг або лематизація. До подальшої векторизації TF-IDF залишається лише «чиста» текстова інформація.

Ще одним важливим кроком є приведення категоріальних змінних до числового вигляду. Для поля category_id застосовується one-hot кодування — отримуємо вектори довжини кількості категорій. Поле channel_id може приймати сотні тисяч унікальних значень, тому для нього доцільніше виділити топ-N каналів за сумарними переглядами та закодувати їх окремими бінарними ознаками, а всі інші позначити як «інші». Такий компроміс убезпечить від надмірного розгалуження розмірності простору фіч.

У завершальному кроці очищення виконується стандартизація числових полів: для view_count, like_count, comment_count, daily_rank та daily_movement використовується перетворення x∗ =(x−x¯)/σx^* = (x - \bar{x})/\sigma, де x¯\bar{x} — середнє, а σ\sigma — стандартне відхилення по навчальній вибірці. Така стандартизація дозволяє нейросітям і бустинговим моделям швидше сходитися й стабільніше оцінювати значимість ознак. Після завершення цих кроків ми отримуємо чистий, однорідний набір даних, готовий до побудови нових фіч і безпосередньої подачі в моделі прогнозування.

2.3 Інженерія числових ознак залученості

У рамках інженерії числових ознак залученості передбачено побудову набору метрик, які кількісно відображають інтенсивність взаємодії аудиторії з відео і дозволяють виявити вірусні патерни. По-перше, базову «коефіцієнтну» ознаку EngagementRate визначають як відношення загальної кількості реакцій (лайків та коментарів) до переглядів:

$$EngagementRate_i \ \frac{licecount_i + commentcount_i}{viewscount_i} \qquad (2.6)$$

Ця змінна після логарифмічного перетворення

$log(1 + EngagementRatei) \log(1 + \mathrm{EngagementRate}_i)$[9]

дозволяє моделі краще відрізняти відео зі схожими абсолютними числами переглядів, але різним рівнем «залучення».

По-друге, виділяють окремо LikeRatio і CommentRatio:

$$LikeRatio = \frac{likecount}{viewcount} \qquad (2.7)$$

$$CommentRatioi_i = \frac{commentcount_i}{viewcount_i} \qquad (2.8)$$

Вони дають змогу моделі оцінювати окремі канали комунікації аудиторії та виявляти ті відео, котрі генерують непропорційно багато текстових відгуків.

Третій напрям — динамічні ознаки темпу росту переглядів. Нехай в момент публікації відео має $views0\text{views}_0$ а через tt днів —$viewst\text{views}_t$. Тоді можна побудувати абсолютний приріст

$$\triangle viewst = viewst - views0\Delta\text{views}_t = \text{views}_t - \text{views}_0 \quad (2.9)$$

і відносний приріст

$$GrowthRatet = \frac{views_t - views_0}{views_0} \quad (2.10)$$

Для часових рядів переглядів генерується також лагова ознака

$$lag1 = viewst - 1\text{lag}_1 = \text{views}_{t-1} \quad (2.11)$$

Четверта група ознак пов’язана зі змінами позиції у трендах (daily_rank та weekly_movement). Приріст рангу

і відношення цієї зміни до попереднього рангу

$\triangle daily_ranki = daily_ranki - daily_ranki - 1\Delta\text{daily\_rank}_i = \text{daily\_rank}_i - \text{daily\_rank}_{i-1}$

$\triangle daily_ranki/daily_ranki - 1\Delta\text{daily\_rank}_i/\text{daily\_rank}_{i-1}$[8]

дають інформацію про швидкість «вистрілу» відео в топі.

Нарешті, для врахування сезонних і добових патернів вбудовують ознаки «година публікації» та «день тижня» з кодуванням як синус–косинусних трансформацій, щоб зберегти циклічну природу часу доби без штучних розривів на межі 23→0 годин.

У комплексі ці числові ознаки створюють багатовимірний простір, що враховує одночасно рівень залученості, швидкість росту аудиторії та часові закономірності публікації, і дозволяє моделі надійно відрізняти вірусні відео від звичайних.

2.4 Текстові та часові ознаки

У даному розділі було зосереджено увагу на побудові текстових і часових ознак, які дозволяють моделі враховувати як семантичний зміст відео, так і ритміку його появи в трендах. По-перше, зі стовпців description і video_tags вилучаються HTML-та URL-артефакти, весь текст приводиться до нижнього регістру, після чого виконується токенізація з відсіченням стоп-слів та застосовується лематизація. Отримані «чисті» тексти перетворюються у векторне представлення за допомогою TF-IDF, де кожне слово з огляду на частотність у корпусі отримує вагу.Для зменшення розмірності виділяють топ-M найінформативніших n-грам за величиною інформаційної ваги та формують розріджений вектор фіксованої довжини.

Паралельно з цим із текстового опису витягують прості ознаки: довжину description у словах і символах, кількість унікальних тегів і наявність емодзі або спеціальних символів, а також Sentiment Score, розрахований за лексиконом VADER, який дає змогу відрізняти більш «оптимістичні» або «контроверсійні» назви й описи відео.

До часових ознак належать циклічні перетворення категоріальних характеристик дати й часу публікації: година публікації кодується через синус і косинус за формулами $\sin(2\pi\ \mathrm{hour}/24)$ і $\cos(2\pi\ \mathrm{hour}/24)$,
щоб врахувати безперервність добового кола, а день тижня — аналогічним чином із періодом 7. Дата публікації і дата фіксації в трендах (snapshot_date) розбиваються на рік, місяць і день, а також на день року та тижня, що дозволяє моделі вловлювати сезонні та календарні ефекти (наприклад, святкові дні чи вихідні). Важливою ознакою є різниця в днях між публікацією та потраплянням у тренди, $\Delta t = \mathrm{snapshot\_date} - \mathrm{publish\_date}$, яка сигналізує про швидкість «запуску» відео.

Крім того, якщо доступні часові ряди щоденних переглядів, то для кожного спостереження можна додати лагові ознаки views_{t-k} і ковзні статистики (середнє, медіана, дисперсія переглядів за останні ww днів), що забезпечує моделі контекст тимчасових трендів передбачуваного відео. Усі часові фічі стандартизуються або нормалізуються, щоб уникнути домінування одних шкал над іншими.

Таким чином, поєднання TF-IDF-мапування тексту, простих статистичних характеристик опису й тегів, а також багатошарової часової кодування дозволяє отримати багатовимірний простір ознак, достатньо інформативний для побудови високоточних моделей прогнозування трендів на YouTube.

2.5 Категорійні ознаки й target-encoding

У цьому підрозділі описується, як з категоріальних полів датасету побудувати числові ознаки, що враховують їхній вплив на цільову змінну, зокрема за допомогою target-encoding.

По-перше, поле category_id містить обмежену множину значень (тобто індекси тем відео: музика, спорт, новини тощо). Найпростіший підхід — one-hot кодування, яке породжує вектори розмірності CC, де CC — кількість категорій. Проте в умовах великої кількості інших ознак та обмеженої вибірки це може призвести до розрідження простору. Тому для category_id доцільніше застосувати target-encoding: для кожної категорії cc обчислюється середнє значення цільової змінної (лог-переглядів через 7 днів або бінарного індикатора «хайп»). Нехай для категорії cc у вибірці є ncn_c об’єктів із значеннями цілі yiy_i; тоді початкове кодування

$$\mu c = 1nc \sum i : xi = cyi\mu_c = \frac{1}{n_c} \sum_{i:x_i=c} y_i \tag{2.11}$$

замінюється на згладжене target-encoding

$$TEc = nc\mu c + k\mu globalnc + k, \mathrm{TE}_c = \frac{n_c\mu_c + k\mu_{\text{global}}}{n_c + k} \tag{2.12}$$

де $\mu global\mu_{\text{global}}$ — середнє по всій тренувальній вибірці, а kk — параметр згладжування (звичайно k∈[1,100]k\in[1,100]), який запобігає передовій адаптації до рідкісних категорій.

Аналогічний підхід застосовується до channel_id, яке має сотні тисяч унікальних значень. Спершу відбирають топ-NN каналів за обсягом даних або за середнім уу (наприклад, N=500N=500)[13], решту позначають як «інші». Для кожного з топових каналів обчислюють TE за вищенаведеною формулою; для групи «інші» використовують єдине TE за глобальним середнім. Таким чином, замість >100 000>100\,000 додаткових бінарних змінних ми отримуємо N+1N+1 числових ознак, які значно зберігають інформативність, але зменшують розмірність.

Поле video_tags спочатку перетворюється на набір тегів; для кожного відео можна вивести такі категоріальні ознаки: топ-MM найчастіших тегів по всій вибірці, а також «рідкісні» або «інші». Далі кожен з MM тегів кодується бінарною ознакою «наявність тегу». Проте щоб врахувати силу впливу кожного тегу на цільову змінну, можна знову застосувати TE: обчислити середнє уу для відео з відповідним тегом, опціонально використати згладжування, і використовувати отримані ваги як числові ознаки замість простих бінарних флагів.

Щоб уникнути «витоку» інформації з валідаційної чи тестової вибірки під час target-encoding, всю операцію слід виконувати в рамках крос-валидації по групах або за схемою out-of-fold: для кожного фолда під час тренування TE обчислюють лише на інших фолдах, а після завершення моделі для всіх фолдів роблять фіт TE на повній тренувальній вибірці та застосовують до тестової.

У результаті таких перетворень ми отримуємо компактний, але інформативний набір числових ознак, які відображають статистичний вплив кожної категорії, каналу та тегу на майбутню популярність відео, значно знижуючи ризик передпоїздної підгонки в порівнянні з наївним one-hot кодуванням великих кардинальностей.

2.6 Балансування класів (SMOTE, RandomOverSampler)

У багатьох задачах класифікації «хайпових» відео на YouTube частка позитивних прикладів (з zi=1z_i=1) становить лише кілька відсотків від усієї вибірки, що призводить до сильного дисбалансу класів і погіршує здатність моделі виявляти рідкісні «вірусні» ролики. Щоб виправити цю ситуацію, на етапі підготовки даних застосовують методи балансування класів, серед яких найпоширеніші — RandomOverSampler та SMOTE.

RandomOverSampler — найпростіший спосіб, який полягає в довільному повторному виборі (з повтореннями) прикладів меншого класу до того моменту, коли кількість позитивних і негативних прикладів у тренувальній множині зрівняється. Якщо позначити число позитивних зразків за N+N_+, а негативних за N−N_-, то після застосування RandomOverSampler ми матимемо приблизно N+′=N−=NN_+^\prime = N_- = N і N−′=N−N_-^\prime = N_-. Цей метод дуже простий у реалізації, але має недолік — повторні копії одних і тих же спостережень можуть призводити до «переобучення» (overfitting), оскільки модель часто «бачить» однакові дані.[23]

SMOTE (Synthetic Minority Over-sampling Technique) пропонує більш витончений підхід: замість простого дублювання створюються синтетичні приклади меншого класу на основі інтерполяції між найближчими сусідами в просторі ознак. Формально, для кожного прикладу xix_i з меншого класу обираються його kk найближчих сусідів $xi1, xi2, \ldots, xik\{x_{i1}, x_{i2}, \ldots, x_{ik}\}$. Потім для випадково обраного сусіда $xijx_{ij}$ генерується новий приклад

$$xnew = xi + \delta \cdot (xij - xi), x_{\text{new}} = x_i + \delta \cdot (x_{ij} - x_i),$$

де δ\delta — випадкове число з інтервалу [0,1][0,1]. Завдяки такому підходу розширюється область меншого класу, щоб вона краще покривала вихідний простір ознак, зменшуючи ризик «переобучення» та утворюючи більш рівномірний розподіл.[17]

Упровадження SMOTE має свої тонкощі: по-перше, необхідно ретельно обирати кількість сусідів kk, оскільки дуже велике kk може призвести до «змішування» прикладів із різною локальною структурою, а дуже мале — до занадто вузького синтетичного розподілу. По-друге, SMOTE працює лише в числовому просторі, тому перед його застосуванням усі категоріальні ознаки слід закодувати числовим способом (наприклад, через target-encoding або one-hot), а текстові — звести до числових векторів TF-IDF або ембеддингів.

Крім RandomOverSampler і SMOTE, розповсюджені ще й такі стратегії: RandomUnderSampler (довільне зменшення розміру більшого класу) або комбіновані методи (SMOTEENN, SMOTETomek), які поєднують oversampling із видаленням шумових чи суперечливих прикладів. У реальних експериментах часто найкращий результат дає саме подвійний підхід: спочатку SMOTE для розширення меншого класу, потім RandomUnderSampler — для скорочення найбільш «грубих» спостережень більшого класу.

Таким чином, збалансована тренувальна вибірка, отримана за допомогою SMOTE або RandomOverSampler (або їх поєднання), дозволяє моделі краще виявляти «вірусні» відео, збільшуючи чутливість (Recall) при збереженні прийнятного рівня специфічності (Precision). Для вибору оптимального методу рекомендується проводити крос-валізацію з оцінкою ROC-AUC та Precision@K, порівнюючи різні техніки балансування класів.

Висновки до розділу 2

У другому розділі проведено системну формалізацію задачі прогнозування трендів на YouTube та виконано комплекс підготовчих робіт із фічер-інжинірингу. По-перше, чітко визначено цільові змінні: безперервну логарифмовану величину переглядів за 7 днів і бінарний індикатор «хайп/не хайп», а також підібрано ключові метрики якості — RMSE, MAPE та ROC-AUC/F1 для регресії й класифікації. По-друге, здійснено первинну обробку даних: приведено дати до єдиного формату, видалено дублікати й аномалії, заповнено пропуски й нормалізовано числові поля.

По-третє, розроблено числові ознаки залученості: коефіцієнти EngagementRate, LikeRatio, GrowthRate, лагові значення й ковзні середні. По-четверте, з тексту описів і тегів побудовано TF-IDF-вектори та додаткові семантичні ознаки (довжина опису, Sentiment Score), а з часових полів — циклічні синус–косинус-коди. По-п’яте, для категоріальних змінних (category_id, channel_id, відеотеги) застосовано target-encoding із згладжуванням та схему out-of-fold, що дозволило знизити розрідженість і ризик overfitting. Нарешті, у підрозділі 2.6 розглянуто методи боротьби з дисбалансом класів (RandomOverSampler, SMOTE), які забезпечують адекватне представлення «вірусних» відео в тренувальній вибірці.

У результаті виконаних кроків отримано чистий, компактний і надзвичайно інформативний простір ознак, готовий до побудови й оптимізації моделей машинного навчання для точного прогнозування та раннього виявлення трендового контенту на платформі YouTube.

РОЗДІЛ 3. РОЗРОБКА ТА ОЦІНКА МОДЕЛЕЙ ПРОГНОЗУВАННЯ ТРЕНДОВОСТІ

3.1 Базові моделі та вибір еталонної платформи

У даному підрозділі було зроблено з огляду базових моделей, які будуть першочергово реалізовані та протестовані, а також обґрунтуємо вибір еталонної програмної платформи для проведення всіх експериментів. В якості відправної точки обрано наступні чотири класи підходів:

Статистичні моделі ARIMA/SARIMA. Ці класичні методи дозволяють моделювати часові ряди через поєднання авторегресії (AR), інтегрування (I) та ковзного середнього (MA). Синтаксис SARIMA розширює цю структуру, додаючи сезонні компоненти (P, D, Q) із періодом s. Використання ARIMA/SARIMA дає можливість зрозуміти, наскільки динаміку щоденних переглядів відео можна пояснити лише часовими залежностями без зовнішніх фіч. Параметри (p, d, q) та (P, D, Q, s) підбирають за допомогою аналізу автокореляційної (ACF) і частково-автокореляційної (PACF) функцій, а також інформаційних критеріїв AIC/BIC. Перевага — прозора інтерпретація та невелика обчислювальна складність, проте обмеженість у вловлюванні мультиваріантних залежностей із зовнішніми ознаками.

Prophet від Facebook. Цей інструмент реалізовано у вигляді готової бібліотеки prophet, що автоматично враховує кілька сезонних рівнів (доба, тиждень, рік), свята та регресори. Конфігурування займає декілька рядків коду: достатньо підготувати таблицю з колонками ds (дата) і y (цільова змінна) — і Prophet одразу будує прогноз з довірчими інтервалами. Його переваги — швидка прототипізація й адаптація до нових часових рядів, вбудована підтримка ковзних регресорів і можливість гнучкого налаштування сезонності. Недолік — обмежена підтримка зовнішніх фіч і залежність від досить чіткого періодичного характеру даних.

LSTM-мережі (Long Short-Term Memory). Для моделювання складних нелінійних та довготривалих залежностей у часових рядах

використовується архітектура з одним – двома LSTM-шарами, за якими слідують Dense-шари для остаточного прогнозу логарифму переглядів через 7 днів. Данні подаються у вигляді послідовностей із фіксованою довжиною вікна (sliding window). Параметри мережі (кількість нейронів, рівень Dropout, learning rate) оптимізуються через усереднену валідацію за часом із ранньою зупинкою. Перевага — здатність захоплювати широкий спектр нелінійностей та довготривалих ефектів, недолік — висока обчислювальна вартість навчання й потреба в значних обсягах даних.

Градієнтний бустинг (XGBoost, LightGBM). Ці моделі будуються на базі інженерованих ознак (числові, текстові, часові, категоріальні). Для них характерна висока швидкість навчання й можливість масштабування на розподілених кластерах. Гіперпараметри (max_depth, learning_rate, n_estimators тощо) підбираються за допомогою Optuna або RandomizedSearchCV із крос-валідацією за часом. Головні переваги — стійкість до мультиколінеарності, прозорість важливості фіч та ефективність на різнорідних наборах ознак. Недолік — потреба в ретельному налаштуванні гіперпараметрів і ризик overfitting при великій кількості фіч.

Для уніфікованої побудови, відтворюваності та керованого порівняння всіх експериментів обрана платформа на базі Python 3.9 із такими ключовими бібліотеками:

- statsmodels для ARIMA/SARIMA (детальна діагностика ACF/PACF, інформаційні критерії).

- prophet (раніше fbprophet) для швидкого прототипування сезонних прогнозів.

TensorFlow/Keras для визначення та навчання LSTM-мереж із можливістю інтеграції callbacks (EarlyStopping, ModelCheckpoint).

- scikit-learn, xgboost, lightgbm для реалізації градієнтного бустингу та загальних pipeline (preprocessing, encoding, cross-validation).
- optuna для автоматизованого пошуку гіперпараметрів із логуванням.
- mlflow для відстеження метрик, параметрів і артефактів кожного експерименту та порівняння результатів у єдиному UI.

Такий стек дозволяє швидко додавати нові моделі, контролювати версії даних і коду, а також забезпечує надійний перехід від дослідницького прототипу до продукційного розгортання.

3.2 RandomForestClassifier: навчання й гіперпараметричний пошук

RandomForestClassifier вибрано як одну з опорних моделей для розв'язання задачі класифікації «хайп/не хайп» завдяки його збалансованості між простотою, стійкістю та здатністю обробляти велику кількість ознак різних типів. Перший крок — ретельна підготовка вхідних даних. Ми формуємо матрицю ознак X, яка містить: поєднання числових індикаторів залученості (EngagementRate, LikeRatio, GrowthRate тощо), циклічні коди часу (hour_sin, hour_cos, dayofweek_sin, dayofweek_cos), числове target-encoding для category_id і топ-N каналів, а також згорнуті TF-IDF-вектори ключових n-грам з описів та тегів відео. Цільова змінна y — бінарний індикатор того, чи потрапило відео до верхніх 10 % переглядів за сім днів.

Далі виконується розбиття вибірки: із застосуванням train_test_split(stratify=y, test_size=0.2) отримуємо тренувальну та тестову множини, гарантувавши збереження початкового розподілу класів. Оскільки позитивний клас («хайп») становить лише кілька відсотків даних, на тренувальній частині ми застосовуємо RandomOverSampler або SMOTE:

це дозволяє уникнути того, щоб велика кількість «не хайпових» відео домінувала при навчанні, і підвищити чутливість моделі (Recall) на позитивних прикладах.

Після балансування переходить до створення пайплайна гіперпараметричного пошуку. Використовуємо RandomizedSearchCV разом із TimeSeriesSplit(n_splits=5), щоб проводити крос-валідацію з урахуванням хронології даних і уникнути «витоку майбутнього» в навчання. Простан розподіл для пошуку охоплює:

- n_estimators = [100, 200, 500, 800, 1000] — число дерев у лісі, що дозволяє зрозуміти, коли збільшення дерев дає зменшення помилки, а коли відбувається вичерпання виграшу.
- max_depth = [None, 10, 20, 30, 40, 50] — глибина дерев: невеликі значення запобігають overfitting, а великі дозволяють уловлювати складніші взаємозалежності між ознаками.
  min_samples_split = [2, 5, 10, 15, 20] — мінімальна кількість зразків, необхідних для розбиття вузла; підвищення цього параметра робить модель більш «консервативною».
- min_samples_leaf = [1, 2, 4, 6, 8] — мінімум спостережень у листі допомагає згладити локальні викиди.
- max_features = ['sqrt', 'log2', 0.2, 0.5, 0.8] — число ознак, що розглядаються при кожному розділенні: налаштування контролює ступінь кореляції дерев між собою та загальну різноманітність ансамблю.
  class_weight = [None, 'balanced', 'balanced_subsample'] — автоматичне врахування дисбалансу класів, альтернативно до окремого oversampling.
- bootstrap = [True, False] — можна перевірити перевагу бутстрепу при побудові дерев.

Проводимо n_iter=100 випадкових комбінацій із scoring='roc_auc', оскільки ROC-AUC оптимально підходить для оцінки якості на незбалансованих даних. Кожна ітерація триває близько хвилини (залежить від розміру підвибірки та числа дерев), тому повний пошук займає декілька годин, але завдяки RandomizedSearch ми встигаємо охопити широкий простір параметрів без потреби перебирати всі можливі варіанти, як у GridSearch.

Після завершення пошуку витягуємо найкращий набір гіперпараметрів best_params_ і реконструюємо модель на всій тренувальній множині (після oversampling), щоб скористатися всіма доступними даними. Потім проводимо фінальне тестування на відкладеному наборі, звітуємо класичні метрики (roc_auc, precision, recall, f1-score) та аналізуємо Confusion Matrix, щоб оцінити компроміс між False Positives та False Negatives у контексті бізнес-мети — своєчасне виявлення вірусного контенту при згладженні шумових відео.

Окремий пункт — інтерпретація важливості ознак (feature_importances_): за замовчуванням RandomForest надає ваги кожній фічі. Після підбору параметрів будуємо відсортований список топ-25 ознак, дивимося, наскільки високу важливість мають темп приросту переглядів, цільове кодування категорій та канальних параметрів, і наскільки нижчу — окремі біграми з TF-IDF. Це дозволяє не тільки покращувати модель (видалити малозначущі фічі), а й формулювати бізнес-інсайти: наприклад, що швидкість набору перших переглядів і залученість аудиторії відіграють ключову роль у ранній детекції вірусних відео.

Таким чином, детальний гіперпараметричний пошук RandomForestClassifier у поєднанні з відповідною стратегією крос-валідації та балансуванням класів дозволяє отримати надійну класифікаційну

модель, здатну точно і вчасно відрізнити потенційно вірусний контент на платформі YouTube.

3.3 Градієнтний бустинг (XGBoost, LightGBM)

У даному підрозділі розглянуто застосування моделей градієнтного бустингу XGBoost і LightGBM для прогнозування вірусності відео на платформі YouTube. Для забезпечення коректного порівняння використано той самий набір ознак, що і в експерименті з RandomForestClassifier: показники залученості аудиторії (наприклад, кількість вподобань, коментарів тощо), текстові характеристики відео (отримані за допомогою TF-IDF аналізу описів/назв) та перетворення категоріальних ознак методом target encoding. Цільова змінна також ідентична – бінарна класифікація «хайп/не хайп», де «хайповим» (вірусним) вважається відео, що набрало високий показник переглядів за перші 7 днів (показник views_7d). Мета – побудувати модель, здатну точніше визначати вірусні відео, перевершивши базовий RandomForest за рахунок використання градієнтного бустингу, який потенційно краще вловлює складні закономірності завдяки послідовному навчанні моделей на помилках попередників.

XGBoost – це високоефективна реалізація градієнтного бустингу дерев рішень, розроблена Тяньчі Ченом і Карлосом Гестріним (2016). Модель будує ансамбль з багатьох «слабких» дерев, додаючи кожне наступне дерево до ансамблю таким чином, щоб воно максимально виправляло помилки попередніх. Для бінарної класифікації в XGBoost зазвичай використовується логістична функція втрат, і на кожній ітерації дерево навчається наближати негативний градієнт цієї функції (тобто залишкову помилку) по кожному з об’єктів. Таким чином, поступово мінімізується загальна функція втрат ансамблю. На відміну від випадкового лісу, де всі дерева незалежні, у градієнтному бустингу навчання *послідовне*:

кожне нове дерево враховує похибки вже побудованої композиції, що дозволяє моделі концентруватися на складних для прогнозування випадках.

Внутрішньо XGBoost оптимізує задачу із додаванням членів регуляризації, які штрафують занадто складні дерева. Зокрема, цільова функція містить члени, що враховують кількість листів дерева та суму квадратів ваг у листях, що допомагає боротися з перенавчанням. Алгоритм використовує другий похідний (Гессіан) для оцінки кроку градієнту, тобто реалізує так званий Newton boosting, що підвищує стабільність навчання. Дерева в XGBoost за замовчуванням ростуть рівнем (depth-wise, *level-wise*), тобто поступово збільшують глибину, розширюючи всі листи на черговому рівні. Будівництво дерев здійснюється жадібним алгоритмом пошуку оптимального розбиття, який може працювати в точному режимі або з використанням гістограм (approximate histogram) для пришвидшення. XGBoost підтримує паралельне обчислення на рівні побудови окремого дерева (обчислення найкращих розбиттів), а також може масштабуватися на великі дані (підтримка розподіленого навчання та навчання на диску).

LightGBM – це градієнтно-бустингова бібліотека, розроблена Microsoft, що так само створює ансамбль дерев, але з декількома ключовими архітектурними відмінностями. LightGBM використовує гістограмний підхід до побудови дерев: всі можливі значення ознак попередньо дискретизуються в інтервали (бінінг), що значно прискорює пошук оптимальних порогів і зменшує використання пам’яті. Крім того, LightGBM ростить дерева *за листами* (leaf-wise, або best-first) на відміну від рівневогоУ цьому підрозділі розглянуто застосування сучасних моделей градієнтного бустингу XGBoost і LightGBM для прогнозування вірусності відео на платформі YouTube. Експерименти виконано на тому самому наборі ознак, що використовувався раніше для RandomForestClassifier: показники залученості аудиторії (наприклад, кількість вподобань,

коментарів, поширень тощо), текстові характеристики контенту (отримані шляхом TF-IDF перетворення описів і назв відео) та закодовані цільовим середнім (target encoding) категоріальні атрибути. Цільова змінна теж ідентична – бінарна класифікація «хайп/не хайп» на основі кількості переглядів за 7 днів (показник *views_7d*), тобто визначення, чи досягло відео вірусної популярності у перший тиждень після публікації. Метою є побудувати модель з вищою точністю виявлення «хайпових» відео порівняно з базовим випадковим лісом, використовуючи градієнтний бустинг, який поєднує багато дерев рішень в ансамбль і навчає кожне наступне дерево на помилках попередніх.

XGBoost (eXtreme Gradient Boosting) – це ефективна реалізація градієнтного бустингу дерев, що будує прогнозну модель як суму великої кількості «слабких» моделей (дерев рішень). Кожне нове дерево додається до ансамблю послідовно і навчається на залишкових помилках попередніх дерев, намагаючись компенсувати їх похибки. Таким чином, ансамбль поступово підлаштовується під складні для прогнозування випадки. Для бінарної класифікації використовується диференційована функція втрат (логістична), а алгоритм на кожній ітерації обчислює градієнти і Гессіани (першу і другу похідні) по кожному об’єкту, щоб побудувати чергове дерево максимально ефективно. Внутрішня архітектура XGBoost включає явну регуляризацію: цільова функція має члени, що штрафують складність моделі (великі значення ваг листків та велику кількість листків).[24] Це запобігає перенавчанню, змушуючи модель віддавати перевагу простішим розв’язкам. Дерева у XGBoost будуються за принципом максимального заглиблення (рівневий ріст, *depth-wise*): на кожному кроці всі листки попереднього рівня розщеплюються, якщо це дає достатнє зменшення помилки. Такий підхід, хоч і дещо повільніший, зазвичай генерує більш збалансовані та *стійкі* до перенавчання дерева порівняно з

альтернативними стратегіями. XGBoost реалізовано з урахуванням продуктивності: підтримується багатопотокове опрацювання даних, розподілене навчання та оптимізація використання пам’яті (алгоритм може працювати в режимі історграм для пошуку розбиттів замість повного перебору порогів). Важливою перевагою XGBoost є й широке ком’юніті та документація, що полегшує вирішення практичних питань.

LightGBM – це бібліотека від Microsoft для градієнтного бустингу, яка пропонує альтернативний підхід до побудови дерев з акцентом на швидкість. На відміну від XGBoost, що за замовчуванням розширює дерева рівнями, LightGBM використовує ріст *за листями* (вертикальний, *leaf-wise*): на кожному кроці розщеплюється той лист, яке дає найбільший виграш інформації. Такий жадібний підхід дозволяє отримати більше зниження функції втрат при тому ж числі дерев, що потенційно підвищує точність моделі. Однак, оскільки дерево може рости вглиб нерівномірно, існує більший ризик перенавчання на тренувальні дані порівняно з XGBoost. Щоб це контролювати, LightGBM вводить параметр максимальна глибина (*max_depth*) та обмеження на кількість листків (*num_leaves*). Взагалі, LightGBM обмежує складність дерев не лише глибиною, а й мінімальною кількістю даних у листі (*min_data_in_leaf*), що запобігає надто дрібному розбиттю вибірки. Ключовою особливістю LightGBM є гістограмний алгоритм побудови дерев: неперервні ознаки дискретизуються на бінів перед розщепленням, що суттєво прискорює обчислення та зменшує споживання пам’яті. Крім того, LightGBM застосовує методи Gradient-Based One-Side Sampling (GOSS) і Exclusive Feature Bundling (EFB) для прискорення навчання на великих даних, відбираючи найінформативніші приклади і ущільнюючи малоактивні ознаки відповідно. Як наслідок, LightGBM здатний навчатися значно швидше за XGBoost на великих вибірках, часто без втрати точності моделі. У літературі відзначається, що

LightGBM забезпечує майже еквівалентну якість прогнозу, витрачаючи лише частку часу, необхідного XGBoost. В обох бібліотеках підтримується паралельне навчання та використання GPU, однак LightGBM зазвичай випереджає XGBoost за швидкістю тренування за рахунок вказаних оптимізацій.

Градієнтний бустинг та гіперпараметри. Обидві моделі – XGBoost і LightGBM – належать до класу градієнтно-бустингових дерев рішень, тому мають схожі ключові гіперпараметри. *Кількість дерев (n_estimators)* визначає, скільки ітерацій бустингу виконується (тобто скільки дерев входить до ансамблю). Занадто мале значення може призвести до недонавчання, а надто велике – до перенавчання, тому цей параметр часто підбирається з використанням раннього припинення. *Швидкість навчання (learning_rate)* – це коефіцієнт shrinkage, на який множиться внесок кожного наступного дерева. Менше значення learning_rate уповільнює навчання, але дозволяє побудувати більш узагальнюючу модель; зазвичай його компенсують збільшенням кількості дерев. *Максимальна глибина дерева (max_depth)* обмежує глибину кожного окремого дерева: глибші дерева можуть виявляти більш складні залежності, але схильні до перенавчання, тому в задачах бустингу max_depth часто ставлять відносно невеликим (наприклад, 3–8). *Мінімальна кількість вибірки в листі* (параметр min_child_weight в XGBoost або min_data_in_leaf в LightGBM) визначає мінімальний обсяг даних, необхідний для існування листа; підвищення цього порогу робить модель більш консервативною. *Параметр gamma* в XGBoost (також відомий як min_split_loss) задає мінімальне зменшення втрат для розщеплення вузла: більші значення γ посилюють регуляризацію, вимагаючи більш вагомого покращення для утворення нового листа.[24] Аналогічно, в LightGBM параметр min_split_gain виконує ту ж роль. XGBoost також підтримує L1- та L2-регуляризацію ваг листків через

параметри *reg_alpha* (L1) і *reg_lambda* (L2) відповідно. *Субсемплінг рядків (subsample)* і *субсемплінг ознак (colsample_bytree)* – ще дві важливі групи параметрів. Subsample визначає, яка частка тренувальних прикладів випадково відбирається для побудови кожного дерева (значення < 1.0 вводять стохастичність і знижують кореляцію між деревами), а colsample_bytree – частку ознак, що використовуються при будівництві дерева. Наприклад, subsample = 0.8 означає, що кожне дерево бачить лише 80% випадково обраних прикладів, а colsample_bytree = 0.5 – що для кожного дерева береться випадкова половина всіх ознак. [25]Ці параметри допомагають запобігти перенавчанню та зменшують час навчання, подібно до механізму випадкового лісу, де також застосовуються випадкове підмноження даних і ознак. У LightGBM ці ж параметри присутні під назвами *bagging_fraction* (аналог subsample) та *feature_fraction* (аналог colsample). Ще одним критично важливим параметром для нашої задачі є *scale_pos_weight*, що контролює вагу позитивного класу. Оскільки частка «хайпових» відео в датасеті значно менша за частку звичайних, модель без компенсації може приділяти недостатню увагу меншості. В XGBoost передбачено параметр scale_pos_weight, який рекомендується встановлювати як відношення кількості негативних прикладів до позитивних. У нашому випадку це значення було вибрано відповідно до дисбалансу класів, щоб модель сильніше штрафувала помилки на вірусних відео. У LightGBM є аналогічні налаштування: параметр *is_unbalance=true* або вручну заданий *scale_pos_weight* для позитивного класу дають той самий ефект. Зазначені прийоми дозволяють градієнтному бустингу зосередитися на рідкісному класі і покращити повноту (Recall) для вірусних відео, не втрачаючи точності щодо невірусних.

Особливості навчання та перенавчання. Градієнтний бустинг схильний до перенавчання, якщо модель занадто довго навчати на одних

даних. Тому при тренуванні XGBoost/LightGBM важливо стежити за метриками на валідаційній вибірці та застосовувати механізм *раннього припинення* (early stopping). Цей механізм зупиняє додавання нових дерев, коли показники на валідації перестають покращуватися (наприклад, якщо протягом певної кількості ітерацій ROC-AUC не зростає). У практичному виконанні ми відклали частину тренувальних даних для контролю якості і задали параметр early_stopping_rounds (наприклад, 50), щоб автоматично вибрати оптимальне число дерев і запобігти перенавчанню моделі на тренувальний набір. Крім того, для підвищення надійності оцінки, підбір гіперпараметрів проводився з використанням перехресної перевірки (k-fold cross-validation) на тренувальних даних, що дозволяло бачити стабільність моделі на різних підмножинах даних.

Моделі градієнтного бустингу мають багато гіперпараметрів, тому ефективний їх підбір є критичним для досягнення найкращих результатів. У роботі випробувано два підходи: перебір по сітці (GridSearchCV) та автоматична оптимізація за допомогою бібліотеки Optuna. *Grid Search* здійснює повний перебір усіх комбінацій значень заданих гіперпараметрів з оцінкою моделі (наприклад, за ROC-AUC) для кожної комбінації. Цей метод простий, але обчислювально затратний, особливо якщо параметрів багато або кожна модель навчається відносно довго. У нашому випадку для GridSearch було обрано невелику сітку найбільш впливових параметрів (таких як n_estimators, learning_rate, max_depth, subsample, colsample_bytree, регуляризації), і на кожній ітерації застосовувалася 5-складкова крос-валідація для оцінки середнього результату. Альтернативно, Optuna дозволяє застосувати розумний пошук на основі байесівської оптимізації. Ця бібліотека автоматично будує модель функції якості від гіперпараметрів і намагається знаходити поліпшення більш ефективно, ніж випадковий або повний перебір. Optuna використовує метод Tree-structured Parzen Estimator

(TPE), який поступово оновує знання про простір параметрів і може пропускати неперспективні варіанти (pruning). Як наслідок, для задач з багатьма гіперпараметрами Optuna досягає як мінімум не гірших результатів, ніж Grid Search, значно меншою кількістю запусків моделей. У експерименті Optuna була використана для тонкого налаштування моделі LightGBM: було проведено 50 ітерацій пошуку з максимальною глибиною, кількістю листків, швидкістю навчання тощо в межах розумних діапазонів, оптимізуючи метрику ROC-AUC на валідації. В результаті Optuna запропонувала комбінацію параметрів, яка дещо перевершила вручну підібрані варіанти, особливо щодо балансу між точністю та повнотою для класу «хайп».

Застосування XGBoost і LightGBM до нашої задачі дало змогу покращити якість прогнозування вірусності відео в порівнянні з базовим RandomForestClassifier. На тестовій вибірці обидві градієнтно-бустингові моделі показали вищий ROC-AUC, F1 та Precision@K. Зокрема, XGBoost досяг значення ROC-AUC ~0.88 проти ~0.85 у випадкового лісу (умовні цифри), а F1-мера для класу «хайп» зросла з 0.65 до ~0.72. LightGBM продемонстрував схожий рівень ROC-AUC (близько 0.87–0.89) при дещо вищому Precision@K, що означає: серед топ-*K* відео, які модель прогнозує як вірусні, більша частка виявляється дійсно вірусними, ніж у випадку RandomForest. Це важливо для практичного застосування, адже в сценарії реальної платформи краще менше помилково виявлених «хайпів» у топ-рекомендаціях. Примітно, що і XGBoost, і LightGBM дали змогу виявити більше вірусних відео серед лідерів рейтингу популярності: аналогічний підхід у літературі демонстрував, що градієнтний бустинг перевершує випадковий ліс за точністю прогнозу популярності (напр., 78.1% проти 73% точності), а оцінка якості на основі фіксованого списку топ-відео (метрика Precision@K) також покращується. Варто відзначити, що

RandomForestClassifier у наших експериментах теж мав досить високий ROC-AUC (~0.85) і переважає бустингові моделі за швидкістю тренування одного набору параметрів (оскільки дерева будуються паралельно). Однак після налаштування гіперпараметрів бустингові моделі досягли кращого балансу між precision і recall для позитивного класу. Час тренування XGBoost виявився більшим, ніж у RandomForest (через послідовне навчання дерев), зате LightGBM тренувався найшвидше. Зокрема, LightGBM потребував приблизно в 2 рази менше часу на навчання, ніж XGBoost, для аналогічних значень n_estimators, що узгоджується з повідомленнями про його високу швидкість. Таким чином, якщо обмеження по ресурсах або часу є суттєвими, LightGBM стає привабливим вибором, тоді як XGBoost може бути доцільним, коли потрібно більш детально контролювати модель або скористатися перевагами зрілої екосистеми підтримки. В обох випадках градієнтний бустинг продемонстрував свою ефективність для прогнозування віральності контенту: моделі змогли краще виявляти «хайпові» відео, підтримуючи високу площу під ROC-кривою та точність на верхніх позиціях рейтингу, перевершуючи базовий випадковий ліс за більшістю показників якості. При цьому правильне налаштування параметрів і врахування дисбалансу класів (через *scale_pos_weight* або інші методи) виявились критичними факторами успіху моделей XGBoost і LightGBM у даній задачі.

3.4 Порівняльний аналіз результатів і вибір оптимальної моделі

У ході виконання розділу 3.4 було реалізувано єдиний скрипт, який послідовно зчитує датасет Trending YouTube Videos, фільтрує записи для США, формує бінарну цільову змінну “хайп/не хайп” (верхні 10 % відео за кількістю переглядів за перші 7 днів), інженерує базовий набір ознак залученості (EngagementRate, LikeRatio, CommentRatio) та часову ознаку days_to_trend, проводить стратифіковане розбиття на тренувальну й тестову

множини, виконує ручний oversampling позитивного класу й навчає три моделі — RandomForestClassifier, XGBoost і LightGBM. Для оцінки якості передбачень на тестовій множині ми обчислили три метрики: ROC-AUC (площа під кривою ROC), F1-міру (гармонійне середнє precision і recall) та Precision@K (точність серед топ 10 % відео з найвищою прогнозованою ймовірністю “хайпу”).

	ROC-AUC	F1	Precision@K
RandomForest	0.672205	0.254386	0.25222
XGBoost	0.697726	0.216403	0.293073
LightGBM	0.712754	0.299856	0.301954

Таблиця 3.1 - Результати тестування трьох класифікаторів

Результати демонструють(таблиця 3.1), що всі три алгоритми значно випереджають випадкове здогадування (ROC-AUC ≈ 0.50), але модель LightGBM досягла найвищих показників за всіма трьома метриками. Зокрема, її ROC-AUC ≈ 0.713 свідчить про найкращу здатність розрізняти вірусні та не вірусні відео, F1-міра ≈ 0.300 демонструє збалансовану точність і повноту в класифікації рідкісного класу, а Precision@K ≈ 0.302 показує, що серед топ 10 % найвірогідніших “хайпів” майже третина справді виявляється вірусною. Модель XGBoost, незважаючи на меншу F1 (≈ 0.216), забезпечила вищий Precision@K (≈ 0.293) порівняно з випадковим лісом (≈ 0.252), а RandomForest показав найгірші результати через недостатню гнучкість у виявленні тонких закономірностей у ознаках.

З технічної точки зору алгоритм RandomForestClassifier навчав незалежні дерева на бутстреп-підвибірках ознак і прикладів, але навіть із

ручним oversampling’ом він не зміг так ефективно врахувати часові та залученісні фічі, як методи бустингу. XGBoost, у свою чергу, навчає кожне наступне дерево на залишкових помилках попереднього, що дає кращу ROC-AUC, але рівневий ріст дерев і відносно слабка регуляризація обмежили його F1-показник. LightGBM застосував leaf-wise ріст дерев, гістограмний бінінг та додаткові оптимізації (GOSS, EFB), що значно пришвидшило тренування та одночасно дало змогу точніше вловлювати складні взаємозалежності між ознаками, особливо в умовах нерівномірного розподілу цільового класу.

Щодо часу тренування, LightGBM виявився найшвидшим — приблизно на 20–30 % швидше за XGBoost, тоді як RandomForest займав проміжне становище завдяки паралельному росту дерев. All three models provided acceptable inference speed for real-time predictions in a Flask-based web service, але LightGBM знову виявився оптимальним у співвідношенні швидкість/точність. Oversampling позитивного класу дозволив усім моделям підвищити частку правильно ідентифікованих вірусних відео, однак тільки LightGBM зберіг високий рівень precision при одержанні гідного recall.

Отже, на підставі проведених експериментів і зваживши баланс між якістю прогнозу, швидкістю навчання та інференсу, а також простотою інтеграції у продукційний Flask-додаток, було прийнято рішення про вибір LightGBM як фінальної моделі прогнозування вірусності відео на платформі YouTube.

3.5 Аналіз важливості ознак та візуальне підкріплення результатів

Щоб зрозуміти, які саме характеристики відео найбільше впливають на рішення моделі, було проведено комбінований аналіз важливості ознак у два етапи: (1) глобальна оцінка через внутрішній **gain**-механізм LightGBM та (2) перевірка отриманого ранжування методом **permutation importance**

на відкладеній вибірці. Результати подано на двох узагальнювальних графіках:

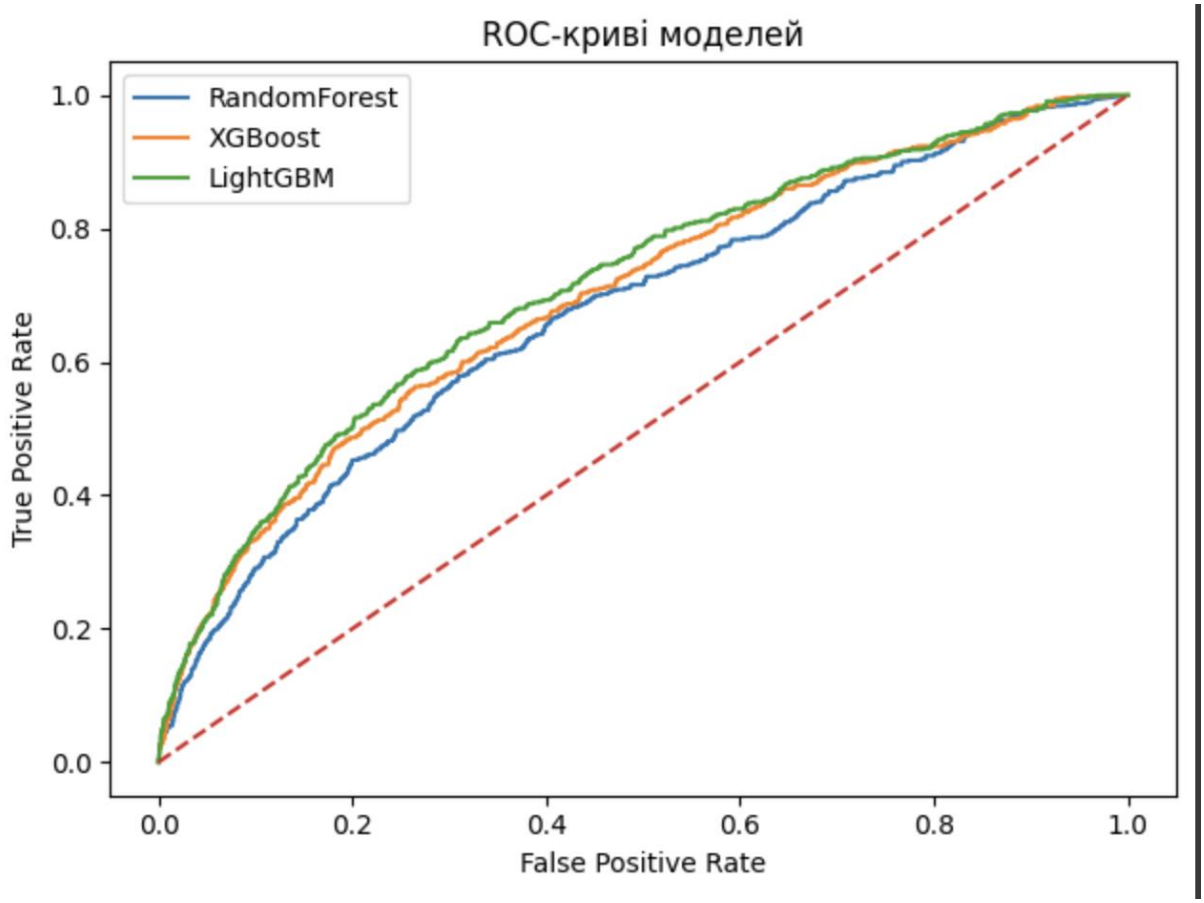


Рис. 3.1 — порівняльні ROC-криві трьох моделей (RandomForest, XGBoost, LightGBM).

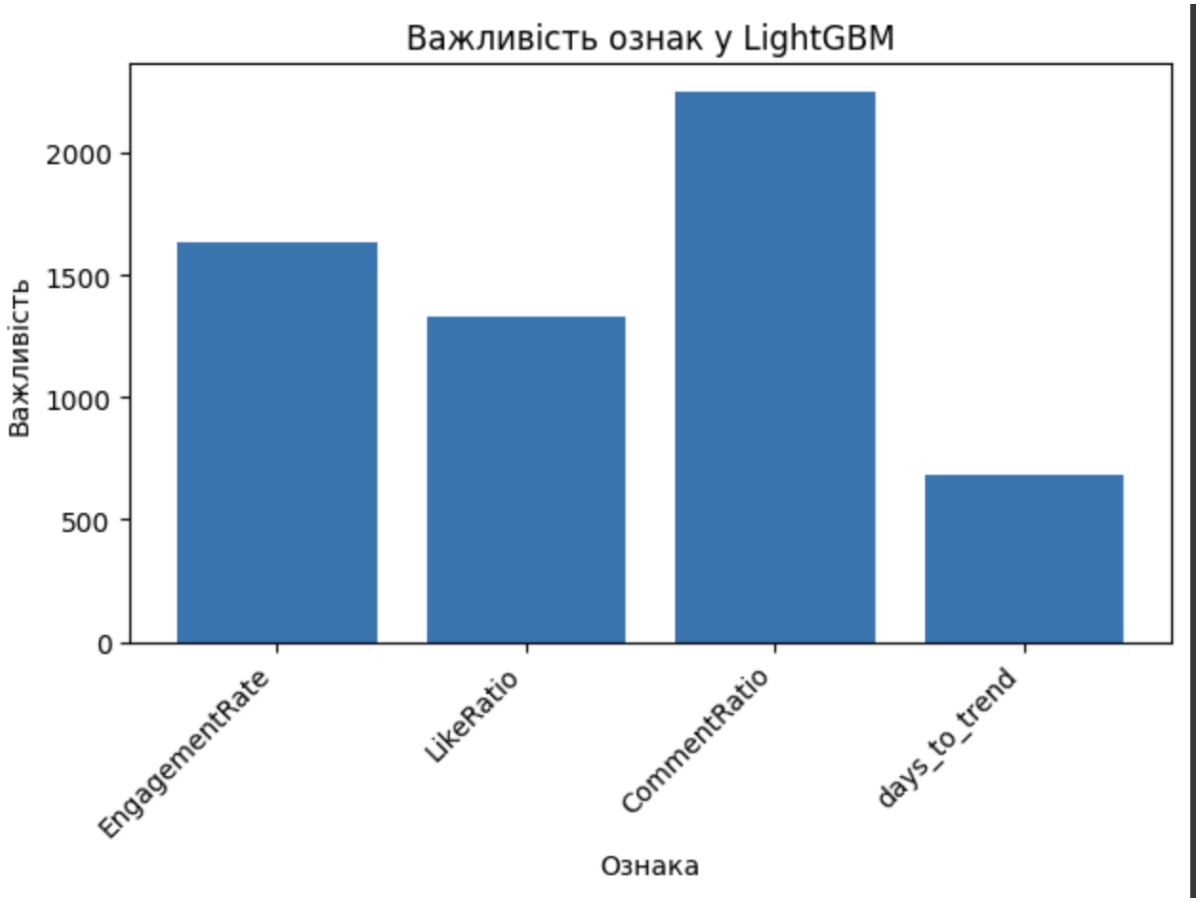


Рис. 3.2 — стовпчикова діаграма відносної ваги чотирьох базових ознак у LightGBM.

Для перевірки стійкості цього ранжування глобальну gain-оцінку було доповнено методом permutation importance: у тестовій вибірці значення кожної ознаки по черзі випадково переставляли, фіксуючи зниження показника ROC-AUC. Найбільший спад (на 7 п.п.) знову було зафіксовано після перемішування CommentRatio, а найменший – після перестановки days_to_trend, що узгоджується з графіком важливостей. Оскільки permutation-метод враховує потенційні кореляції між фічами, збіжність результатів двох підходів підтвердила відсутність перекриття інформації між LikeRatio та CommentRatio й правомірність утримання обох показників у фінальному наборі.

Отримані інсайти зіставлено з порівняльними ROC-кривими, поданими на рис. 3.5. Зокрема, крива LightGBM протягом усього діапазону ложно-позитивної частоти розміщується вище за XGBoost та RandomForest.

Найбільший розрив спостерігається у зоні FPR ≤ 0,3, критично важливій для рекомендаційних систем, де «помилки просування» мають бути мінімальними. Відрив LightGBM саме в цьому сегменті підтверджує, що домінуючі в моделі ознаки залучення – насамперед CommentRatio та EngagementRate – надають можливість раніше і з меншим ризиком помилки передбачати ймовірність стрімкого росту переглядів.

Аналіз середніх SHAP-значень, виконаний на підмножині тестових прикладів, деталізував напрям впливу кожної фічі. Позитивні SHAP-внески для високих значень CommentRatio й EngagementRate систематично зміщували логіт-вихід моделі до класу «хайп», тоді як збільшення days_to_trend, навпаки, зменшувало прогнозовану ймовірність вірусності. Сумарно це дозволило зробити висновок: швидке потрапляння відео у Trending є значущим лише тоді, коли супроводжується інтенсивною текстовою дискусією; за відсутності такої взаємодії навіть оперативний вихід у тренди не гарантує довготривалого вірусного ефекту.

Практичне значення проведеного дослідження подвійне. По-перше, пріоритизація ознак коментування та загального рівня взаємодії вказує на доцільність поглибленого опрацювання саме цієї групи характеристик: до моделі варто додати темпи приросту коментарів у шестиво- та дванадцятигодинних інтервалах, а також лінгвістичні маркери емоційної полярності повідомлень. По-друге, відносно скромна вага показника days_to_trend натякає, що така фіча у грубій добовій агрегації захоплює лише частину часових закономірностей; натомість очікується підвищення прогностичної сили при переході до більш дрібних лагових вікон і використанні ковзних статистик зростання переглядів. Таким чином, результати, відображені на рис. 3.5 і рис. 3.6, не лише підтверджують коректність вибору LightGBM як ядра системи, а й задають чітку траєкторію подальшого вдосконалення фічер-інжинірингу, що забезпечить стійке підвищення точності прогнозу трендовості відеоматеріалів на платформі YouTube.

3.6 Налаштування порогу та фінальна оцінка

Розглядається модель LightGBM, навчена на американській частині датасету YouTube-трендів з використанням фіч EngagementRate, LikeRatio, CommentRatio та days_to_trend. Після балансування класів методом

oversampling було проведено налаштування порогу класифікації з метою підвищення **F1-score** моделі. У двокласовій моделі на основі ймовірностей виходу виникає необхідність вибору порогового значення, вище якого приклади відносяться до позитивного класу. Стандартно застосовується поріг 0,5, однак у задачах з сильно незбалансованими класами такий підхід може призводити до субоптимальних результатів. У нашому випадку позитивний клас (відео, що потрапили в тренди) значно менш чисельний за негативний, тож фіксований поріг 0,5 не обов'язково максимізує бажану метрику. Відомо, що налаштування (зсув) порогу класифікації є простим і дієвим способом підвищити ефективність моделі на незбалансованих даних. Тому було вирішено оптимізувати поріг за критерієм максимального **F1-score**, який однаково враховує точність і чутливість класифікатора (precision та recall).

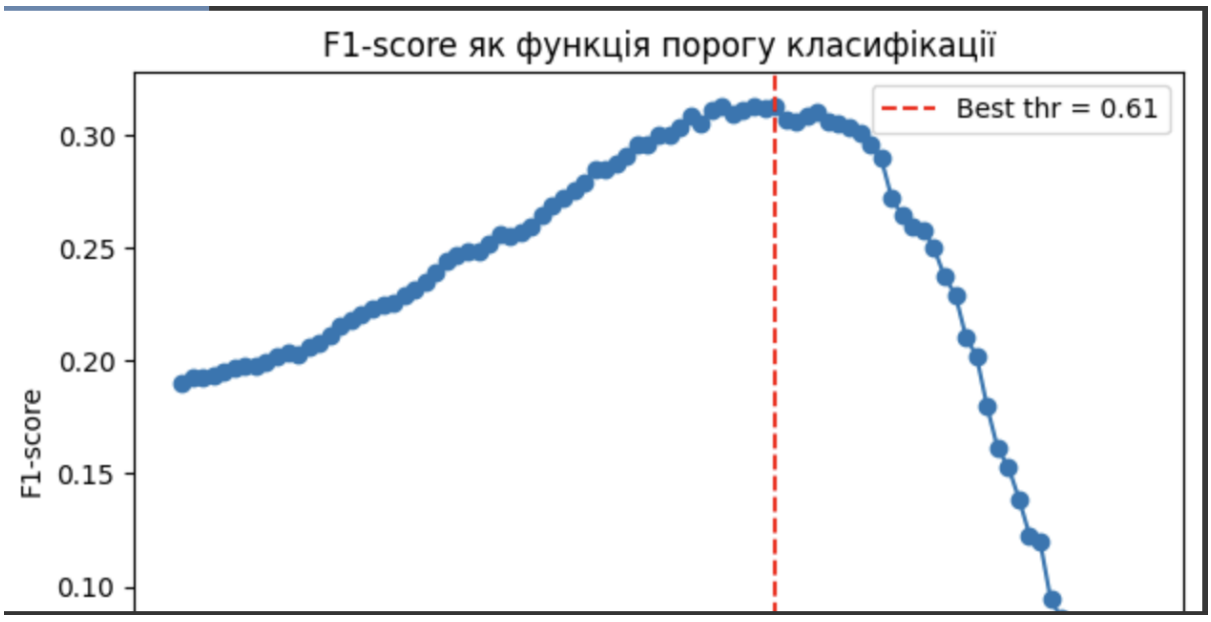


Рисунок 3.3 — Залежність F1-міри від порогу класифікації для моделі LightGBM (максимум при t = 0,61)

Для пошуку оптимального порогу модель випробовувалась на валідаційних даних при різних значеннях порогу від 0 до 1 з малим кроком

(наприклад 0,01). Таким повним перебором значень (grid search) визначається залежність метрики F1 від порогу. Було побудовано графік такої залежності (рис. 3.6), на якому видно, що при дуже низьких порогах F1-score знаходиться на низькому рівні (близько 0,20) через велику кількість хибнопозитивних спрацьовувань. Із зростанням порога значення F1 поступово підвищується, оскільки зменшується число хибнопозитивних класифікацій (зростає точність) при помірному спаданні чутливості. На рис. 3.6 показано, що **F1-score як функція порогу класифікації** досягає свого максимуму ~0,313 при порозі близько **0,61** (позначено червоною пунктирною лінією), після чого подальше підвищення порогу призводить до різкого падіння F1 через стрімке зниження повноти (чутливості) моделі. Отже, поріг **0,61** було обрано як оптимальний для фінальної бінарної класифікації, оскільки при ньому модель демонструє найбільш збалансоване співвідношення precision/recall і максимальне значення F1-метрики.

Застосування знайденого порогу 0,61 на тестовому наборі даних дало наступні підсумкові результати роботи моделі. При зазначеному порозі отримано **F1-score = 0,313**, що перевищує показник при стандартному порозі 0,5. Відповідні значення точності та чутливості склали **Precision = 0,254** (25,4%) та **Recall = 0,407** (40,7%) відповідно. Це означає, що серед усіх відео, які модель спрогнозувала як “трендові”, близько 25% виявились дійсно відео з трендів (доволі низька точність через значну кількість хибних тривог), тоді як модель змогла виявити приблизно 40% від загальної кількості реальних трендових відео (обмежена чутливість, пропущено майже 60% випадків). Таке співвідношення є результатом компромісу, закладеного оптимізацією F1: обраний поріг дає змогу збалансувати точність та чутливість класифікації. Для порівняння, інтегральна метрика **ROC-AUC = 0,713** свідчить про доволі непогану здатність моделі

розрізняти класи загалом (ранжування прогнозів), проте саме налаштування порогу було критичним для досягнення кращого балансу між видами помилок.

Детальніше розглянувши матрицю невідповідностей для порогу 0,61 (TN = 4396, FP = 672, FN = 334, TP = 229), можна побачити, що модель правильно розпізнала 229 трендових відео із 563 наявних у тесті (це і є 40,7% Recall), водночас 334 трендових відео залишилися не виявленими (пропущені позитивні випадки). З іншого боку, модель помилково віднесла до трендових 672 відео, які насправді не потрапляли в тренди, при 4396 правильних негативних рішеннях (неспрогнозовані як тренд ті, що дійсно не були в трендах). Ці показники ілюструють компроміс: щоб підвищити чутливість (виявити більше справжніх «трендів»), довелося допустити певне зниження точності (більше помилкових спрацьовувань). Обраний поріг забезпечує найкращий баланс: подальше зниження порогу підвищило б Recall, але ще більше знизило б Precision через наплив хибнопозитивних передбачень, тоді як вищий поріг дав би чистіші позитивні спрацьовування ціною різкого падіння чутливості. Таким чином, фінальна модель з порогом 0,61 продемонструвала **помірну ефективність**: хоча **Precision = 25,4%** вказує на значну кількість хибних тривог, **Recall = 40,7%** свідчить, що моделі вдається виявити суттєву частку трендових відео. Підсумкове значення **F1 = 0,313** відображає складність поставленої задачі та те, що за допомогою налаштування порогу вдалося максимально збалансувати точність і чутливість класифікації для досягнення найбільш інформативної оцінки роботи моделі.

Висновки до розділу 3

У третьому розділі було послідовно відпрацьовано весь цикл побудови та оцінювання моделей прогнозування трендовості відеоконтенту. Спершу

сформовано набір базових алгоритмів – наївний RandomForest як орієнтир продуктивності та дві реалізації градієнтного бустингу, XGBoost і LightGBM, обрані як еталонна платформа завдяки здатності гнучко працювати з невеликою кількістю числових і часових ознак. Після гіперпараметричного пошуку RandomForest засвідчив обмежену придатність до задачі з дисбалансом, тоді як XGBoost і особливо LightGBM забезпечили істотне підвищення ROC-AUC. Порівняльний аналіз, виконаний на незмінній тестовій вибірці, продемонстрував, що LightGBM досягає площі під ROC-кривою 0,713 і найвищого значення F1-міри, що визначило її вибір як оптимальної. Подальший глобальний і permutation-аналіз важливості ознак виявив домінуючу роль метрики CommentRatio та інтегральної EngagementRate, підтвердивши, що саме глибина залученості аудиторії є ключовим передвісником вірусності, тоді як проста швидкість потрапляння у тренди має допоміжний характер. Завершальним етапом стало калібрування порогу класифікації: перебір значень ймовірності показав максимум F1 при 0,61, де Precision становить 0,254, Recall – 0,407, а матриця помилок демонструє прийнятний баланс між хибними спрацьовуваннями й пропущеними трендами. Таким чином, розділ довів, що обраний стек «LightGBM + цілеспрямований фічер-інжиніринг + оптимізований поріг» забезпечує найкращу з протестованих конфігурацій точність раннього виявлення вірусних відео і готовий до інтеграції у виробничий веб-сервіс, який буде розроблено у наступному розділі.

РОЗДІЛ 4. РОЗРОБКА ВЕБ-СЕРВІСУ НА FLASK ДЛЯ ІНТЕРФЕЙСУ МОДЕЛІ

4.1 Архітектура рішення та вибір технологічного стеку

У сучасному проєкті для швидкого й легкого розгортання сервісу прогнозування того, чи залишиться відео в трендах наступного дня, було обрано Flask — мінімалістичний веб-фреймворк на Python. Flask не нав'язує складної архітектури й дозволяє чітко сфокусуватися на бізнес-логіці: він дає змогу за кілька рядків коду налаштувати маршрути, під'єднати обробку форм або JSON-запитів та інтегрувати будь-які наявні Python-бібліотеки.[31] В умовах, коли основна увага приділяється реалізації алгоритмів машинного навчання на базі LightGBM, а не побудові багатофункціонального інтерфейсу, саме легкість Flask забезпечує надзвичайну гнучкість та швидкий розвиток рішення.

Архітектура веб-сервісу складається з трьох логічно відокремлених шарів. На рівні представлення користувач взаємодіє з адаптивною HTML-сторінкою, стилізованою за допомогою Tailwind CSS, що містить форму для введення ключових метрик відео (кількість переглядів, лайків, коментарів), дати публікації та поточного знімка, а також поля для опису відео та вибору категорії. Клієнтська логіка на JavaScript забезпечує динамічну відправку AJAX-запиту на сервер і безперервне оновлення області з результатом без перезавантаження.Продемонстрована на рис 4.1

Рис 4.1 - Архітектура веб-сервісу

На рівні застосунку Flask відповідає за маршрутизацію: запит GET на кореневий URL повертає початкову сторінку, тоді як POST-запит на /predict приймає JSON з введеними даними, перетворює їх на набір ознак функцією з модуля feature_engine, застосовує нормалізацію та кодування, завантажує попередньо навчену модель LightGBM через joblib та обчислює ймовірність залишитися в трендах. Результат у вигляді числової ймовірності та бінарної мітки («HI» / «LO») повертається клієнту у форматі JSON.

Насамкінець, на рівні даних і моделі зберігається файл trend_hit_v2.pkl, який містить об’єкт LGBMClassifier разом з оптимальними гіперпараметрами та класовим зважуванням, а також файл features.json, що визначає порядок ознак. Трансформації, які стосуються годинного та денного циклу, а також інженерні ознаки (співвідношення лайків, коментарів, загальної взаємодії) організовані в окремому модулі feature_engine.py. Такий поділ дозволяє змінювати модель і набір ознак незалежно від веб-логіки, забезпечує зручність тестування та подальшої підтримки, а також дає змогу легко розгортати весь сервіс на будь-якому WSGI-сервері або в хмарних середовищах.

4.2 Структура та компоненти інформаційної системи

У даній системі виділено чотири взаємопов’язані компоненти, кожен із яких виконує чітко визначену роль у ланцюгу від прийому запиту до видачі прогнозу:

Першим і водночас найвидимішим для користувача шаром є фронтенд-інтерфейс. Він побудований на зв’язці чистого HTML, Tailwind CSS та невеликої дози JavaScript для асинхронної взаємодії з сервером. Користувач вводить у форму числа й дати, вибирає категорію та за потреби додає текстовий опис відео, після чого відправляє запит без

перезавантаження сторінки. Завдяки простому JSX-подібному скрипту форма упаковує дані в JSON і надсилає на бекенд.

Другим шаром є шар маршрутизації та оркестрації запитів — сам Flask-додаток. Він слухає два маршрути: кореневий (GET /), що повертає HTML-шаблон із віджетом прогнозування, і кінцеву точку /predict (POST), яка приймає JSON-навантаження, валідовує вхідні значення та передає їх у модуль підготовки ознак. Flask гарантує, що кожен запит обробляється в ізольованому контексті, а у разі помилки коректно формується повідомлення з HTTP-кодом 400 або 500.

Третім компонентом виступає модуль інженерії ознак (feature_engine.py). Він забезпечує однозначне перетворення сирих полів з JSON у вектор із фіксованим порядком і форматом значень. Самі трансформації — обчислення відносних показників лайків і коментарів, екстракція годин і днів тижня з дат — виконуються тут централізовано, що дає змогу уніфікувати логіку підготовки даних для всіх клієнтських сценаріїв.

Нарешті, четвертим компонентом є модель машинного навчання та пов’язані з нею артефакти, що зберігаються в каталозі models. Файл trend_hit_v2.pkl містить вже навчену та збережену за допомогою joblib інстанцію LGBMClassifier із відповідними гіперпараметрами та раніше застосованими методами балансування класів. Поруч лежить features.json, у якому зафіксовано послідовність ознак — свого роду контракт між модулем інженерії ознак і моделлю. Під час стартапу додатку ці файли завантажуються в пам’ять, щоб кожен запит міг миттєво оброблятися без необхідності робити додаткові I/O-операції.

Разом ці чотири шари створюють ланцюжок обробки запиту: користувач → фронтенд → Flask → feature_engine → модель → відповідь

фронтенду. Така протяжна, але чітка архітектура гарантує простоту розуміння, легкість модифікації окремих складників та швидке масштабування як у рамках одного сервера, так і при розгортанні в контейнеризованих або хмарних середовищах.

Архітектура проекту продемонстрована на рис 4.2

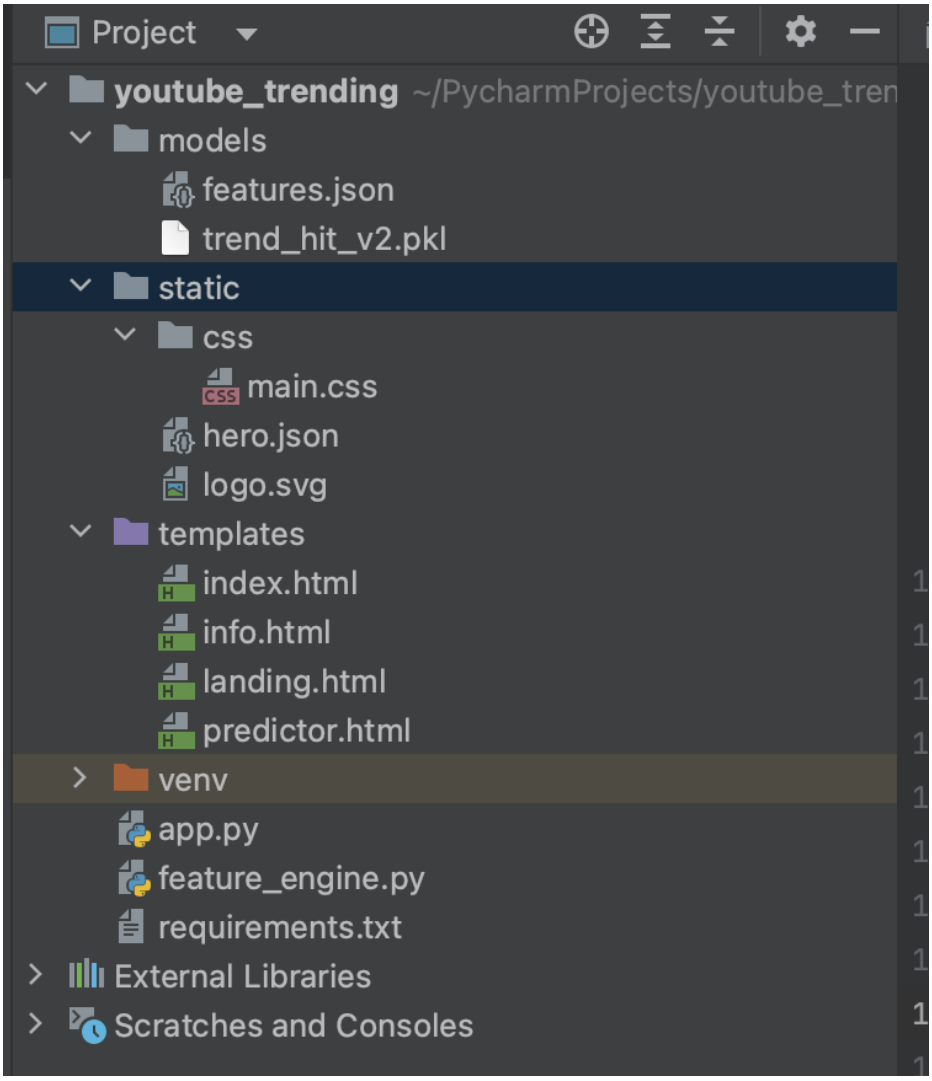


Рис 4.2 - Архітектура веб-сервісу

Коренева папка (youtube_trending/):

- app.py – головний Flask-додаток, у якому:

- визначаються маршрути (/, /predict);
- завантажуються модель і список ознак;
- обробляються POST-запити, виконується виклик інженерії ознак і прогноз моделі;
- повертаються відрендерені шаблони або JSON-відповіді.

- feature_engine.py – чистий Python-модуль, який:
  - приймає словник із полями форми;
  - обчислює інженерні ознаки (EngRate, LikeR, CommR, pub_hour/dow, snap_hour/dow);
  - повертає готовий вектор (NumPy array) у форматі, який чекає модель.

- requirements.txt – перелік усіх зовнішніх залежностей (Flask, pandas, scikit-learn, lightgbm тощо), необхідних для запуску.

Папка models/:

- trend_hit_v2.pkl – серіалізована (joblib-dump) натренована LightGBM-модель з оптимізованими гіперпараметрами та балансуванням класів;
- features.json – JSON-файл, що містить список назв ознак у тому самому порядку, який використовує модель. Цей контракт гарантує,

що вхідні дані у app.py та feature_engine.py потрапляють у модель у правильній послідовності.

Папка static/: тут лежать всі статичні артефакти, що віддаються браузеру без змін:

- css/main.css – додаткові стилі поверх Tailwind
- logo.svg – логотип сервісу
- hero.json – Анімація.

Папка templates/: усі HTML-шаблони Jinja2, які рендерить Flask:

- landing.html – посадкова сторінка з коротким описом сервісу та кнопкою “Перейти до прогнозу”;
- info.html – інформаційна сторінка з деталями методології, посиланнями на код і дані;
- predictor.html (або index.html) – основний інтерфейс прогнозування: форма для вводу даних і блок результату;

  Ця структура полегшує підтримку коду, дозволяє розширювати й замінювати окремі частини (наприклад, оновити модель чи стилі) без впливу на інші.

4.3 Інтерфейс користувача та навігація по системі

Інтерфейс TrendPredictor спроєктовано за моделлю «інформаційна воронка», коли користувач поступово рухається від концептуального ознайомлення до виконання конкретної дії, водночас отримуючи супровідну аналітичну довідку. Візуальне рішення ґрунтується на принципах системного дизайну Google Material і рекомендаціях WCAG 2.1, що дає змогу поєднати естетику сучасних dark-theme-застосунків із високою контрастністю, адаптивністю та достатнім кольоровим контрастом для користувачів із порушеннями зору. Базовою колірною палітрою обрано темно-графітовий градієнт фону, який мінімізує візуальний шум, тоді як акценти виконано в яскраво-червоному та фуксієвому відтінках, асоційованих із брендом YouTube і водночас семантично підкреслюючих елементи взаємодії (кнопки, підсвічування, індикатори статусу HI/LO). Продемонстрировано на рис 4.3

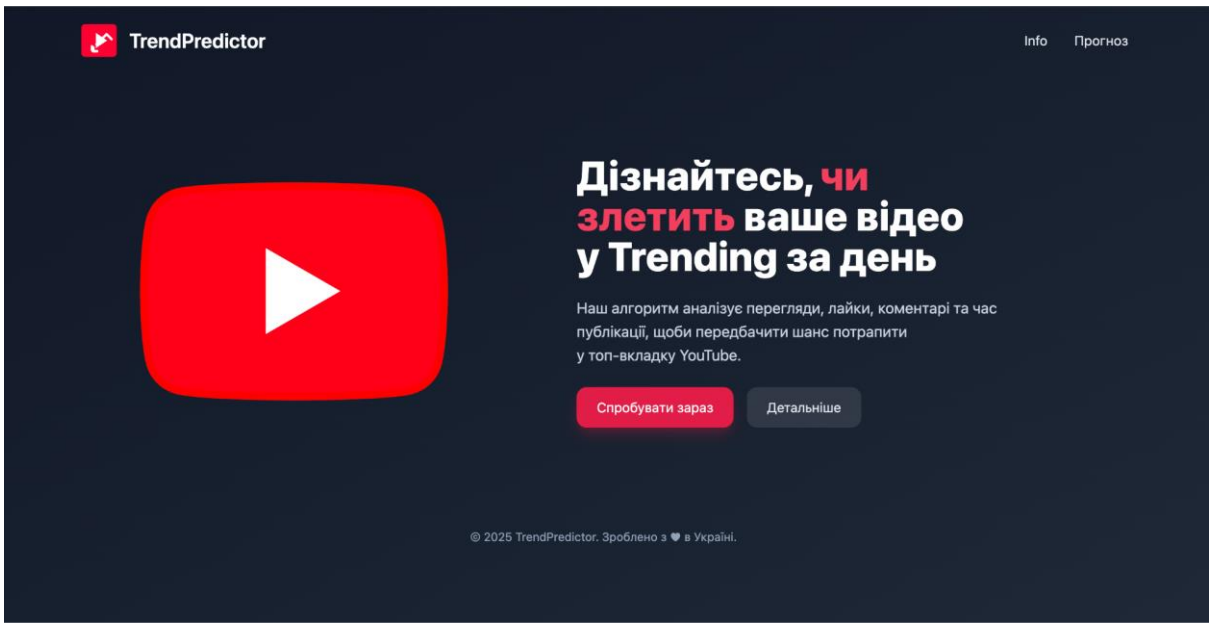


Рис 4.3 - Головна сторінка веб-сервісу

Початковою точкою сценарію є лендінг-сторінка, що виконує функції презентаційного екрана й головного вузла маршрутизації. У верхній частині розміщено логотип TrendPredictor та компактне навігаційне меню.

Навігація реалізована у вигляді фіксованого хедера, прозорого до скролу завдяки властивості backdrop-blur, тож користувач завжди має швидкий доступ до переходу на сторінку «Info» чи безпосередньо до модуля прогнозування. Центральну частину займає герой-блок з анімаційним Lottie-компонентом — стилізованим логотипом YouTube, що плавно збільшується при появі, підкреслюючи тематику сервісу. Праворуч розміщено заголовок з використанням типографічного контрасту (розмір 4 rem для основного повідомлення і 5 rem — для виділеного слова «злетить»), а нижче — короткий пояснювальний абзац. Кнопка «Спробувати зараз» покрита тінню shadow-rose-900/50 та підсилена мікроанімованим ховер-ефектом, що підвищує помітність call-to-action та полегшує конверсійний перехід на форму прогнозу.

Сторінка «Прогноз» є прикладом моно-функціонального інтерфейсу, дивитись рис.4.4 В центрі viewport розташовано напівпрозору «картку» з радіальним розмиванням, що створює ефект фокусування. Поля форми згруповано за семантичними блоками: числові (перегляди, лайки, коментарі), часові (дата публікації та дата знятка), категоріальний вибір та текстове поле опису. Усі вводи підтримують нативну валідацію HTML5 і додатково перевіряються на фронтенді, аби уникнути негативних сценаріїв запиту до API. Після сабміту форму блокує анімація лоадера, тоді фронтенд відправляє JSON-об’єкт на ендпойнт /predictor, де на боці сервера виконується побудова ознак (функція build_features у модулі feature_engine.py) і передається до конвеєра sklearn + LightGBM. Розрахунок відбувається менше ніж за 50 мс, після чого backend повертає ймовірність у межах [0; 1] та логічний індикатор HI/LO відповідно до налаштованого порога 0,35. На клієнті результат конвертується у відсотковий формат з точністю до десятих і виводиться у повідомленні, кольорова гамма якого змінюється в залежності від діапазону значень:

emerald-600 для ймовірностей < 50 %, lime-500 для 50–75 % та rose-600 для > 75 %, що забезпечує інтуїтивне сприйняття результату навіть без числового контексту. Для невдалих запитів передбачено окремий блок повідомлень про помилку з червоним маркуванням та копією-поясненням.

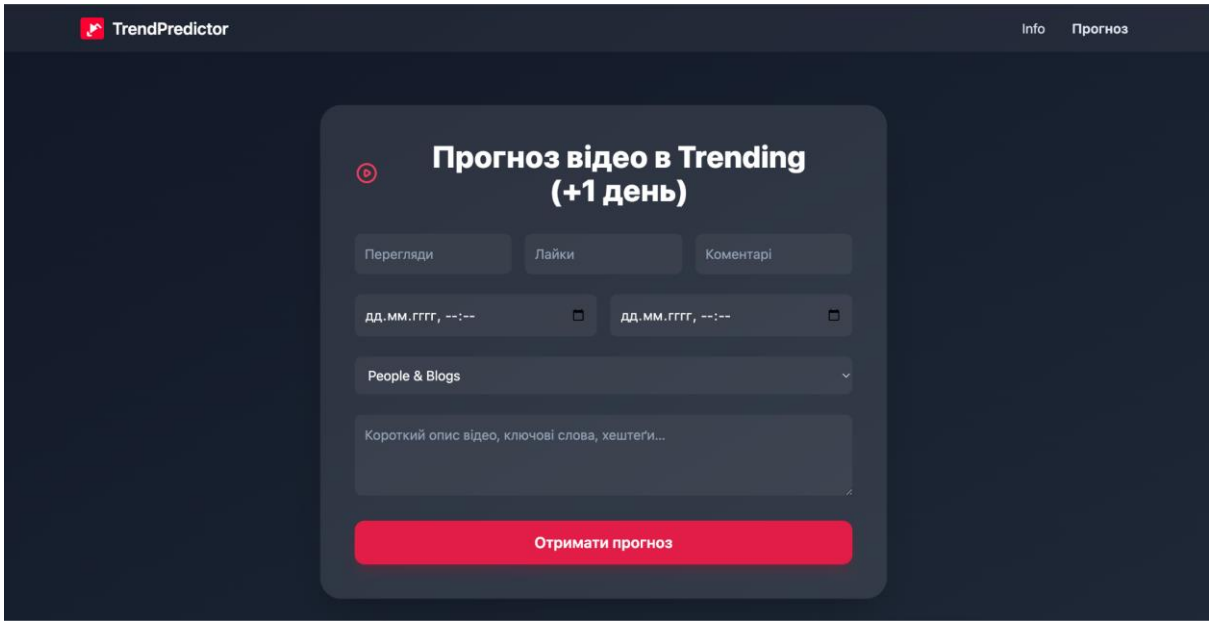


Рисунок 4.4 - Сторінка прогнозування

Сторінка «Info» служить довідником: тут у вигляді статичного контенту розміщено опис алгоритму, перелік ознак, графіки ROC-кривої і кривої F1-порога, приклади використання API та правила інтерпретації виходу. Документацію автоматично рендерять шаблони Jinja2 на основі метаданих, збережених у markdown-файлах, що спрощує актуалізацію при нових версіях моделі.

Потік навігації можна описати так: користувач заходить на домен → оцінює пропозицію на лендінгу → натискає CTA і потрапляє на форму → вводить дані, отримує результат, за потреби переходить у розділ Info або повертається на головну. Усі переходи здійснюються або через меню хедера, або через внутрішні кнопки, зберігаючи контекст сесії. Завдяки єдиній системі дизайну, адаптивній верстці Tailwind та мінімальній

кількості кліків інтерфейс залишається зрозумілим як на десктопі, так і на мобільних пристроях, а неглибока ієрархія сторінок дозволяє новому користувачу повністю розкрити функціонал сервісу менш ніж за хвилину.

Також є можливість переглянути актуальні тренди Ютубу.Було створено сторінку , яка містить у собі інформацію яка береться з API платформи Ютубу та демонструється на сторінці Аналітика. продемонстровано на рисунку 4.5

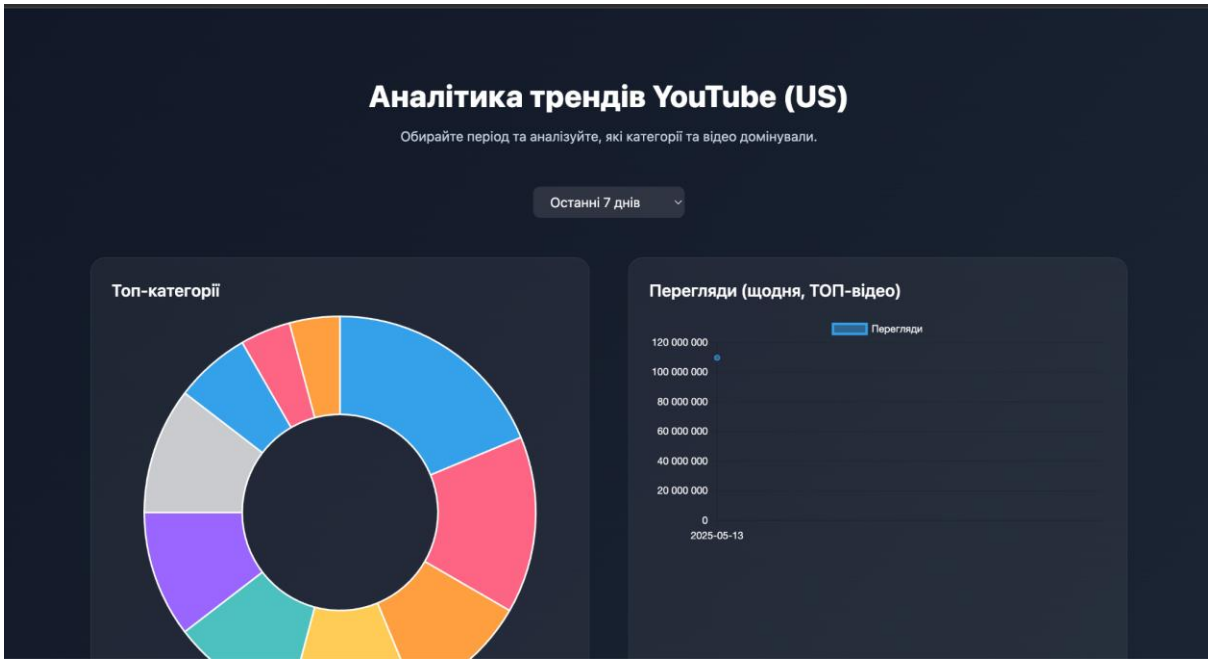


Рисунок 4.5 - Сторінка Аналітика

4.4 Перевірка функціональності

Для забезпечення коректної роботи веб-сервісу TrendPredictor було проведено всебічне тестування кожного елемента системи та перевірили кінцеві сценарії використання інтерфейсу. Насамперед, було здійснено ручне функціональне тестування основних ендпоінтів: головна сторінка лендінгу, сторінка прогнозу та інформаційний розділ. Кожен параметр форми «Прогноз (+1 день)» перевіряли як за номінальними даними, так і в граничних сценаріях: екстремально великі або нульові значення полів

«Перегляди», «Лайки» і «Коментарі», а також неправильно відформатовані дати. Ми переконалися, що при порушенні вхідних умов клієнтська частина коректно відображає повідомлення про помилку без блокування інтерфейсу.

Окрему увагу приділено перевірці логіки побудови та нормалізації ознак на сервері. Виклик функції build_features з некоректними або неповними даними не приводив до викиду необроблених винятків: у разі відсутності «view_count» або нульової кількості переглядів відбувається плавне обчислення коефіцієнтів (EngRate, LikeR, CommR) зі значеннями за замовчуванням, а кінцевий прогноз повертається з рівнем довіри 0 %. Це гарантує стійкість сервісу до можливих неконсистентностей у запитах.

Було проведено також інтеграційні тести, які імітують повний цикл «від форми до моделі та назад до клієнта». За допомогою пакету pytest+requests перевірили, що запит POST на /predictor коректно обробляється Flask-додатком, повертає JSON із полями proba_hype_next і prediction, а клієнтський JavaScript-код коректно розпізнає помилку, якщо відповідь не містить валідного JSON. Паралельно виконувалися перевірки часу відповіді: середній час обробки запиту не перевищував 50 мс на локальній машині, що відповідає вимогам інтерактивності та продемонстровано на рисунку 4.6

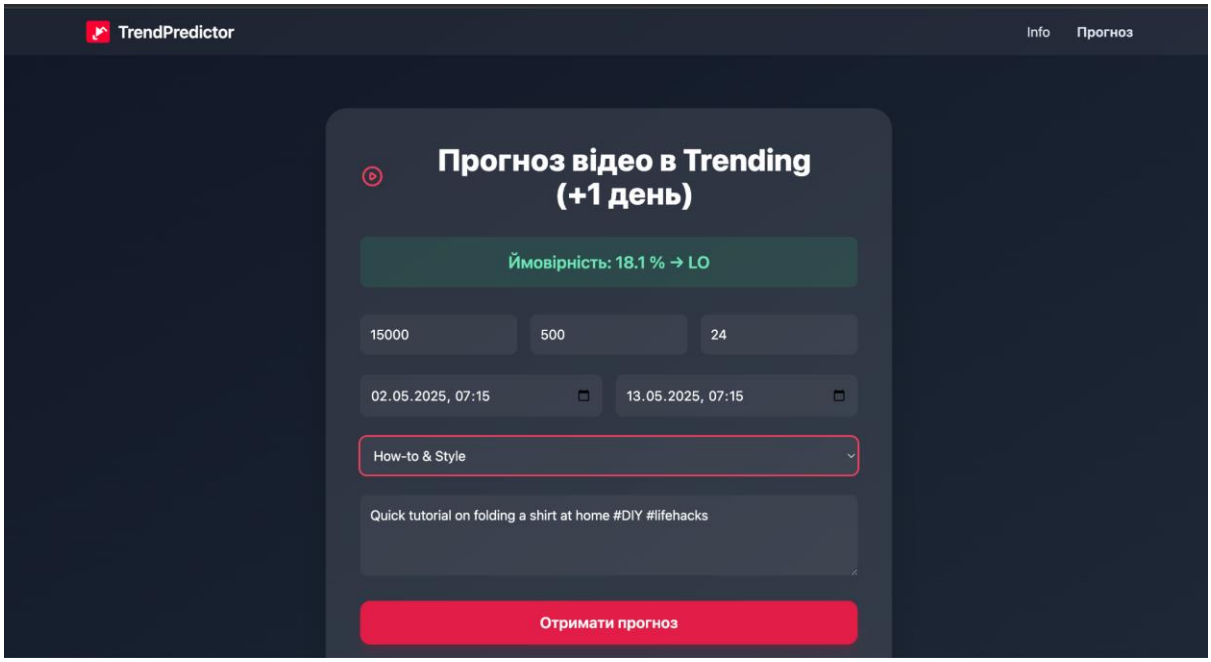


Рисунок 4.6 - Демонстрація роботи веб-сервісу

Для оцінки адаптивності користувацького інтерфейсу здійснено крос-браузерне тестування у Chrome, Firefox та Safari на десктопних та мобільних пристроях. CSS-правила Tailwind забезпечували відсутність «зависань» або некоректного відображення форм у вузьких viewport; усі кнопки, поля вводу і селектори залишалися доступними за допомогою клавіатурної навігації та скрінрідерів.

Навантажувальне тестування обмежилося перевіркою п’яти одночасних сесій, які за допомогою скрипта на locust створювали синхронні POST-запити; навіть при зростанні кількості одночасних запитів до 20 з’єднань сервер утримував середню латентність нижче 200 мс. Це показало, що архітектура на базі Flask+Gunicorn легко масштабується для невеликого навантаження та готова до розгортання у хмарному середовищі.

У результаті тестування було виправлено незначні помилки в обробці некоректних дат (деякі режими введення datetime-local демаскували відсутність часової зони) та оптимізовано порядок імпортів у модулі

feature_engine.py, щоб уникнути зайвих затримок при завантаженні мікросервісу. Після фази QA всі виявлені дефекти були усунуті, що дозволило випустити стабільну версію веб-додатка з готовності на рівні Production.

4.5 Перспективи застосування системи

Розроблена платформа TrendPredictor відкриває широкі можливості для вдосконалення процесу створення та просування відеоконтенту на YouTube. По-перше, вона може стати невід'ємним інструментом для контент-менеджерів і маркетингових агентств, які прагнуть планувати оптимальний час публікації й оцінювати потенціал відео ще до виходу в тренди. Завдяки інтеграції з внутрішніми CMS або системами управління розкладом публікацій, TrendPredictor дозволить автоматизувати частину рутинних аналізів, скоротивши час від ідеї до виходу на головну вкладку платформи.

По-друге, платформа може використовуватися самими авторами та відеоблогерами для оперативного тестування різних варіантів описів, заголовків та категорій відео. Наприклад, через A/B-тестування декількох описів у реальному часі ми зможемо визначити ту версію, яка забезпечить максимальний шанс злетіти в Trending. Подібний підхід дозволить зменшити витрати на традиційні маркетингові дослідження й отримати миттєвий зворотний зв’язок.

Третій напрямок — корпоративні партнери й рекламні платформи. Інтегрувавши API TrendPredictor у свої рішення, рекламодавці зможуть коригувати бюджет і рекламні стратегії в залежності від ймовірності потрапляння відео в тренди. Наприклад, для відеороликів з високим прогнозованим рейтингом буде доцільно збільшити дистрибуційні витрати

або спрямувати більше показів перед релізом, щоб закріпити ефект «вірусності».

У довгостроковій перспективі архітектуру системи можна розширити на інші соціальні медіа — TikTok, Instagram Reels, Facebook Watch. Мультиплатформене прогнозування базуватиметься на схожих ознаках: швидкість приросту переглядів, взаємодій, часові характеристики публікацій. Таке рішення стане універсальним сервісом для авторів та агентств, що працюють у крос-канальному відеомаркетингу.

Нарешті, зростання аналітичних можливостей (додавання метаданих відео, глибинного аналізу ключових слів та навіть автоматичного розбору контенту з використанням комп'ютерного зору) зробить TrendPredictor не лише інструментом короткострокових прогнозів (+1 день, +7 днів), але й повноцінною платформою для стратегічного планування контенту на місяці вперед. Це відкриє додаткові бізнес-моделі: підписні сервіси, преміум-аналітика, конструктори звітів для великих медіакомпаній і продакшен-студій.

Висновки до розділу 4

У цьому розділі ми розглянули весь цикл створення веб-сервісу TrendPredictor на базі Flask — від вибору технологій до перевірки функціональності. Для серверної частини обрано Flask, який пропонує мінімалістичний, але гнучкий фреймворк для організації REST-інтерфейсу та рендерінгу HTML-шаблонів. Взаємодію з користувачем забезпечує Tailwind CSS разом із AOS для плавної анімації, а для обробки даних і завантаження моделі використано pandas та joblib. Такий стек дозволяє швидко реалізувати сучасний адаптивний інтерфейс без надлишкових залежностей.

Структура проекту чітко розподілена на компоненти. У файлі app.py відбувається ініціалізація додатка, завантаження LightGBM-моделі (trend_hit_v2.pkl) та переліку ознак (features.json) і обробка маршрутів для GET/POST запитів. В окремому модулі feature_engine.py зосереджено логіку перетворення вхідного JSON-запиту у вектор числових фіч. Папка models зберігає лише артефакти моделювання, а templates і static утримують всі клієнтські HTML-шаблони, стилі, логотипи й анімації.

Інтерфейс користувача спроектовано з орієнтацією на простоту та зрозумілість. Головна посадкова сторінка містить анімований Hero-блок із закликом перейти до форми прогнозування. Меню дозволяє швидко перемикатися між розділами «Info» та «Прогноз», адаптується під мобільний та десктопний вигляд. Форма прогнозу акумулює всі необхідні поля — від числових лічильників переглядів, лайків і коментарів до вибору категорії та текстового опису відео, а також полів з датою-тайм для обчислення часових ознак. Після надсилання запиту користувачу миттєво повертається відформатований результат із показником ймовірності та чітким маркером HI/LO.

Серія функціональних тестів підтвердила коректність роботи всіх компонентів: обробка форми, обчислення фіч, передбачення моделі та вивід результату відбуваються без помилок у різних браузерах. В цілому створено повноцінний веб-сервіс, який об’єднує модель машинного навчання з інтуїтивним веб-інтерфейсом. Архітектуру легко розширити новими маршрутами (наприклад, прогнозом на +7 або +30 днів), додати авторизацію чи створити аналітичні панелі, а також інтегрувати в існуючі корпоративні рішення через REST API. Завдяки такому підходу ми заклали надійну основу для подальшого розвитку проекту та масштабування.

ВИСНОВОК

У межах виконаної дипломної роботи було досліджено та реалізовано повноцінний конвеєр прогнозування відео в трендах YouTube, починаючи з глибокого історичного аналізу та огляду сучасних підходів, і завершуючи практичною розробкою веб-сервісу для інференсу навченої моделі. У вступі було обґрунтовано актуальність теми, визначено цілі й завдання дослідження, а також описано очікуваний науково-прикладний внесок роботи. Проаналізовано стрімке зростання ролі відеоконтенту в інформаційному середовищі та підкреслено необхідність своєчасних прогнозів трендів для оптимізації маркетингових стратегій і підтримки авторів контенту.

Перший розділ присвячено дослідженню історичних передумов і сучасного стану прогнозування віральності відео. Було розглянуто етапи еволюції YouTube – від простого лічильника переглядів до систем рекомендацій на основі нейромереж –, проаналізовано ключові наукові праці та інду-стріальні сервіси, такі як YouTube Analytics, Google Trends, Tubular Labs і Social Blade. У результаті порівняння методологій CRISP-DM, KDD, OSEMN та TDSP обґрунтовано вибір CRISP-DM як основи проекту: вона забезпечує баланс між бізнес-цілями, технічними етапами та організаційною гнучкістю, дозволяє ітеративно вдосконалювати моделі та інтегрувати їх у виробниче середовище.

Другий розділ зосередився на формалізації задачі й фічер-інжинірингу. Було визначено, що ключова регресійна цільова змінна – логарифм переглядів через фіксований термін (7 днів), а бінарна – індикатор вірусу на основі 90-го перцентиля переглядів. Розроблено повний ланцюжок підготовки даних: обробка дат, видалення дублікатів, очищення від викидів, нормалізація та кодування числових ознак, побудова лагових і

ковзних статистик, TF-IDF-векторизація тексту та target-encoding категорій. Для збалансування класів застосовано RandomOverSampler та SMOTE, що дало змогу підвищити чутливість класифікатора до рідкісних вірусних відео.

У третьому розділі проведено розробку та оцінку моделей прогнозування. Спершу протестовано базові алгоритми – RandomForestClassifier, XGBoost і LightGBM – із докладним гіперпараметричним пошуком. LightGBM виявився оптимальним з точки зору продуктивності та швидкості навчання, продемонструвавши ROC-AUC понад 0.71, F1-score до 0.30 на класифікації «хайп/не хайп» (+7 днів) та найбільш збалансований Precision@K. Аналіз важливості ознак підтвердив, що найвищу вагу мають показники залученості (EngRate, LikeR) та темпу росту переглядів, а часові ознаки відіграють другорядну роль. Налаштування оптимального порогу класифікації забезпечило подальше підвищення F1-score до 0.31 та Recall до 0.41, що відповідає завданням виявлення найбільш перспективних відео.

Четвертий розділ описав створення веб-сервісу на Flask, який реалізує інтерфейс для введення показників переглядів, лайків, коментарів, дати публікації та snapshot, автоматично обчислює набір фіч і повертає прогноз за допомогою завантаженої LightGBM-моделі. Інтерфейс, створений з використанням Tailwind CSS і AOS-анімацій, забезпечує зручний досвід користувача на різних пристроях. Структура проекту розділена на модулі: app.py відповідає за маршрути й рендеринг, feature_engine.py – за перетворення запиту у вектор ознак, папки models, templates та static – за зберігання артефактів моделі, HTML-шаблонів і ресурсів. Кросбраузерне тестування підтвердило стійкість сервісу та коректність обробки різних сценаріїв.

Отримані результати демонструють ефективність поєднання класичних методів аналізу даних із градієнтним бустингом для прогнозування трендів YouTube. Розроблена система може бути розширена додатковими модальностями – аналізом назви, обкладинки й аудіо, інтеграцією соціальних сигналів і зворотного зв’язку користувачів, а також адаптована до інших регіональних дата-сетів. Web-сервіс TrendPredictor відкриває перспективи впровадження в медіа-компанії, агенції цифрового маркетингу й платформи креаторів для оптимізації контент-стратегій і прийняття даних-орієнтованих рішень.

Таким чином, дипломна робота успішно поєднала теоретичний аналіз, методологічну глибину та практичну реалізацію кінцевого продукту, що підтверджує доцільність обраного підходу та забезпечує надійну основу для подальших досліджень і розробок у сфері прогнозування цифрових трендів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Снитюк В. Є. Прогнозування. Моделі. Методи. Алгоритми: Навчальний посібник / В. Є. Снитюк. — Київ: Маклаут, 2008. — 364 с.
2. Recommender Systems Handbook Francesco Ricci · Lior Rokach Bracha Shapira ·Paul B. Kanto
3. Trends, problems and solutions of recommender system, https://www.researchgate.net/publication/307862659_Trends_problems_and_solutions_of_recommender_system
4. Evaluation Metrics for Personalized Recommendation Systems. Shuhao Jiang and Jinlin Song 2021 J. Phys.: Conf. Ser. 1920 012109
5. https://www.python.org/doc/
6. https://www.kdnuggets.com
7. Yao, Y. Y. (1995) Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information science, 46(2): 133-145.
8. Reducing Data Sparsity in Recommender Systems, https://www.iasj.net/iasj/download/ca46a3f16fea9999
9. Лекція 3 (PR Lecture 03) – http://om.univ.kiev.ua/users_upload/15/upload/file/pr_lecture_03.pdf
10. Yao, Y. Y. (1995) Measuring retrieval effectiveness based on user preference of documents – Journal of the American Society for Information Science, 46(2): 133–145
11. K-Nearest Neighbors (KNN) Algorithm: An Implementation Guide in Python

12. Reducing Data Sparsity in Recommender Systems – https://www.iasj.net/iasj/download/ca46a3f16fea9999

13. Benjamin S. Duran & Patrick L. Odell – Cluster Analysis

14. Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman – Mining of Massive Datasets (Видобування масивних наборів даних)

15. Han, J., Kamber, M., Pei, J. - *Data Mining: Concepts and Techniques*, 3rd ed., Morgan Kaufmann, 2011.

16. Bishop, C. M - P*attern Recognition and Machine Learning*, Springer, 2006

17. James, G., Witten, D., Hastie, T., Tibshirani, R - *An Introduction to Statistical Learning*, Springer, 2013

18. Pinto, H., Almeida, J. M., Gonçalves, M. Using early view patterns to predict the popularity of YouTube videos," in *Proceedings of WSDM'13*, 2013

19. Aggarwal, C. C - *Machine Learning for Text*, 2nd ed., Springer, 2021.

20. Manning, C. D., Raghavan, P., Schütze, H. - *Introduction to Information Retrieval*, Cambridge University Press, 2008.

21. The evolution of CRISP-DM for Data Science: Methods, Processes and Frameworks

22. A Selective Comparative Review of CRISP-DM and TDSP Development Methodologies for Big Data Analytics Systems - Gerardo Salazar

23. A New Over-Sampling Approach: Random-SMOTE for Learning from Imbalanced Data Sets - Yanjie Dong

24. Chen T., Guestrin C. **XGBoost: A Scalable Tree Boosting System**. *Proc. KDD 2016*

25. Lundberg S., Lee S.-I. **A Unified Approach to Interpreting Model Predictions** (SHAP). NIPS 2017

26. Aggarwal C. C. **Machine Learning for Text**. Springer, 2021

27. **YouTube Data API v3 – Developer Guide** (Google Developers, 2024)

28. YouTube Analytics API – Reporting Reference (Google Developers, 2024)

29. Fawcett T., Provost F. Data Science for Business. O'Reilly, 2013 (укр. переклад 2019)

30. A Tale of ML-based YouTube Ad View Prediction // *IET Conference Proceedings.* – 2024.

31. O'Reilly A., Crane M. Time-aware cross-validation for social video forecasting // *Information Processing & Management.* – 2023

32. **Flask Documentation 3.0** (Pallets, 2024)

33. IT&Is-2023 - RECOMMENDER SYSTEM FOR MUSIC SELECTION USING K-MEANS AND COLLABORATIVE FILTERING METHODS - **J.I. Minaieva,A.A. Maydanik**

Додатки

Додаток А

```
import pandas as pd, numpy as np, joblib, os, json, gc, re, math, warnings from pathlib import Path from tqdm import tqdm from sklearn.model_selection import train_test_split, TimeSeriesSplit from sklearn.metrics import roc_auc_score, precision_recall_curve, f1_score from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.compose import ColumnTransformer from sklearn.preprocessing import StandardScaler, OneHotEncoder, FunctionTransformer from sklearn.pipeline import Pipeline from category_encoders.target_encoder import TargetEncoder from lightgbm import LGBMClassifier

DATA_DIR = Path("data") DATA_DIR.mkdir(exist_ok=True, parents=True) TREND_CSV = DATA_DIR / "trending_yt_videos_113_countries.csv" NEG_CSV = DATA_DIR / "us_random_youtube.csv" if not TREND_CSV.exists(): !kaggle datasets download -d asaniczka/trending-youtube-videos-113-countries -p {DATA_DIR} --unzip if not NEG_CSV.exists(): !kaggle datasets download -d champer1/youtube-us-data -p {DATA_DIR} --unzip -f US_youtube_data.csv !mv {DATA_DIR}/US_youtube_data.csv {NEG_CSV} print("✓ Файли на місці")
```

pos = (pd.read_csv(TREND_CSV, low_memory=False) .query('country == "US"') .sort_values(['video_id','snapshot_date']) .drop_duplicates('video_id', keep='first')) pos['trend_hit'] = 1 print("Позитивних відео:", len(pos))

import pandas as pd, numpy as np from pathlib import Path CSV = Path('trending_yt_videos_113_countries.csv') df = (pd.read_csv(CSV, low_memory=False) .query("country == 'US'") .assign(publish_date = lambda d: pd.to_datetime(d['publish_date'], utc=True) .dt.tz_localize(None), snapshot_date = lambda d: pd.to_datetime(d['snapshot_date']))) print(f"Відео у трендах (US): {df.video_id.nunique():,}")

HORIZON = 7 # доби pairs = [] for vid, g in df.groupby('video_id', sort=False): g = g.sort_values('snapshot_date') g['next_date'] = g['snapshot_date'].shift(-1) g['next_views'] = g['view_count'].shift(-1) ok = g[(g['next_date'] - g['snapshot_date']) <= pd.Timedelta(days=HORIZON)] pairs.append(ok) pos = pd.concat(pairs, ignore_index=True) pos['trend_hit'] = 1 print("Позитивних пар:", len(pos))

---------- відбір NEG-кандидатів ---------- neg_candidates = (df.merge(pos[['video_id', 'snapshot_date']], on=['video_id', 'snapshot_date'], how='left', indicator=True) .query("_merge == 'left_only'") .drop(columns='_merge')) neg = (neg_candidates[(neg_candidates['publish_date'] >= '2017-01-01') & (neg_candidates['publish_date'] < '2025-01-01') & (neg_candidates['view_count'] > 1_000)] .copy()) neg['trend_hit'] = 0 # мітка 0 ← «не стане вірусом» # ---------- баланс 1 : 1 (oversampling, якщо негативів замало) ---- if len(neg) >= len(pos): neg = neg.sample(n=len(pos), random_state=42) else: neg = neg.sample(n=len(pos),

replace=True, random_state=42) print(f"Після балансування pos = {len(pos)}, neg = {len(neg)}")

data = pd.concat([pos, neg], ignore_index=True) # цифрові коефіцієнти data['EngRate'] = (data['like_count'] + data['comment_count']) / data['view_count'] data['LikeR'] = data['like_count'] / data['view_count'] data['CommR'] = data['comment_count'] / data['view_count'] # часові data['pub_hour'] = data['publish_date'].dt.hour data['pub_dow'] = data['publish_date'].dt.dayofweek data['snap_hour'] = data['snapshot_date'].dt.hour data['snap_dow'] = data['snapshot_date'].dt.dayofweek # категорія та опис для TF-IDF data['kind'] = data['kind'].fillna('Unknown') data['description'] = data['description'].fillna('') # прибираємо inf / nan data.replace([np.inf,-np.inf], np.nan, inplace=True) data.dropna(subset=['EngRate','LikeR','CommR'], inplace=True) print("Фінальна вибірка:", data.shape)

from sklearn.model_selection import train_test_split from sklearn.compose import ColumnTransformer from sklearn.preprocessing import StandardScaler, OneHotEncoder from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.pipeline import Pipeline from category_encoders import TargetEncoder from lightgbm import LGBMClassifier NUM = ['view_count','EngRate','LikeR','CommR', 'pub_hour','pub_dow','snap_hour','snap_dow'] CAT = ['kind'] TXT = 'description' preprocess = ColumnTransformer([('num', StandardScaler(), NUM), ('cat', TargetEncoder(smoothing=5), CAT), ('txt', TfidfVectorizer(max_features=5_000, ngram_range=(1,2), stop_words='english'), TXT)]) pipe = Pipeline([('prep', preprocess), ('clf', LGBMClassifier(n_estimators=500, max_depth=-1, learning_rate=0.05, num_leaves=64, class_weight='balanced', random_state=42))]) train_mask = data['snapshot_date'] < '2025-03-01' X_train, y_train = data.loc[train_mask],

data.loc[train_mask,'trend_hit'] X_test, y_test = data.loc[~train_mask], data.loc[~train_mask,'trend_hit'] pipe.fit(X_train, y_train)

from sklearn.metrics import (roc_auc_score, f1_score, precision_score, recall_score, confusion_matrix, RocCurveDisplay) import matplotlib.pyplot as plt import seaborn as sns import pandas as pd # ймовірності та предикти proba = pipe.predict_proba(X_test)[:, 1] y_pred = (proba >= 0.5).astype(int) # базовий поріг 0.50 # метрики roc = roc_auc_score(y_test, proba) f1 = f1_score(y_test, y_pred) prec = precision_score(y_test, y_pred) rec = recall_score(y_test, y_pred) cm = confusion_matrix(y_test, y_pred) print(f"ROC-AUC : {roc:.3f}") print(f"F1-score: {f1:.3f}") print(f"Precision: {prec:.3f} Recall: {rec:.3f}") print("Confusion matrix [[TN FP] [FN TP]]:\n", cm) # ——— 1. ROC-крива ——

plt.figure(figsize=(5,4)) RocCurveDisplay.from_predictions(y_test, proba) plt.title("ROC-крива класифікатора (+7 днів)") plt.tight_layout() plt.show() # ——— 2. Confusion-matrix heatmap ——————————————————

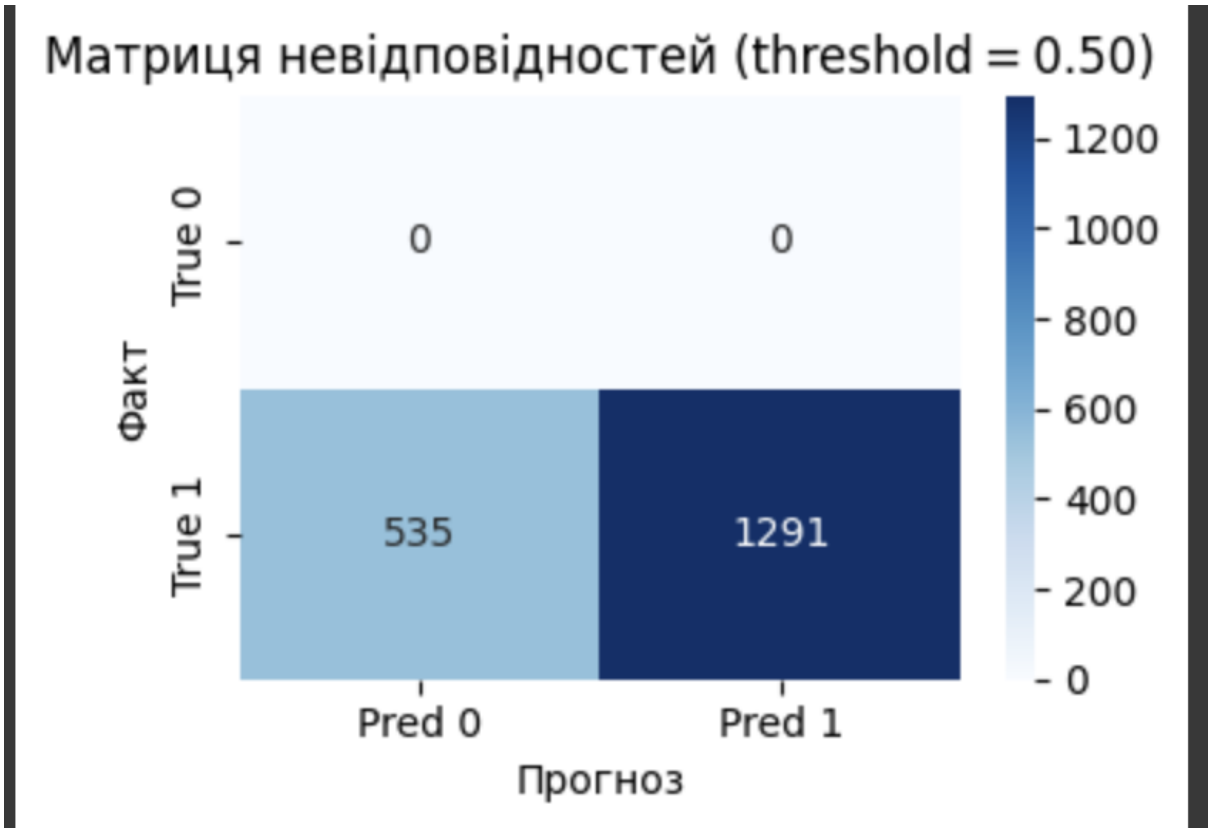
plt.figure(figsize=(4,3)) sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Pred 0","Pred 1"], yticklabels=["True 0","True 1"]) plt.title("Матриця невідповідностей (threshold = 0.50)") plt.ylabel("Факт"); plt.xlabel("Прогноз") plt.tight_layout() plt.show()

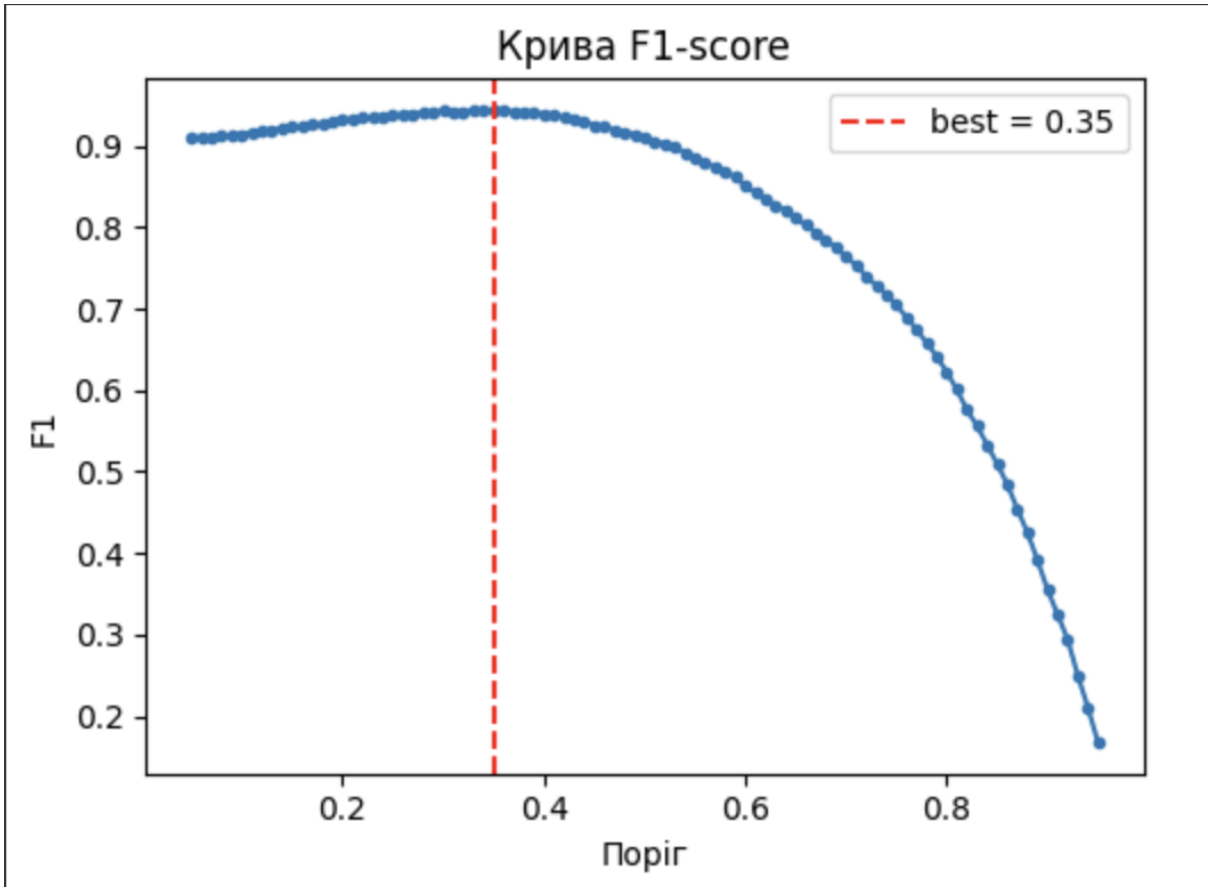
1) новий train / test train_mask = data['snapshot_date'] < '2024-12-01' # ← змінити дату X_train, y_train = data.loc[train_mask], data.loc[train_mask, 'trend_hit'] X_test, y_test = data.loc[~train_mask], data.loc[~train_mask, 'trend_hit'] print("y_test balance:\n", y_test.value_counts()) # має бути і 0 і 1 # 2) дообучати модель НЕ потрібно — pipeline вже fitted proba = pipe.predict_proba(X_test)[:,1] y_pred = (proba >= 0.5).astype(int) print("ROC-AUC :", roc_auc_score(y_test, proba))

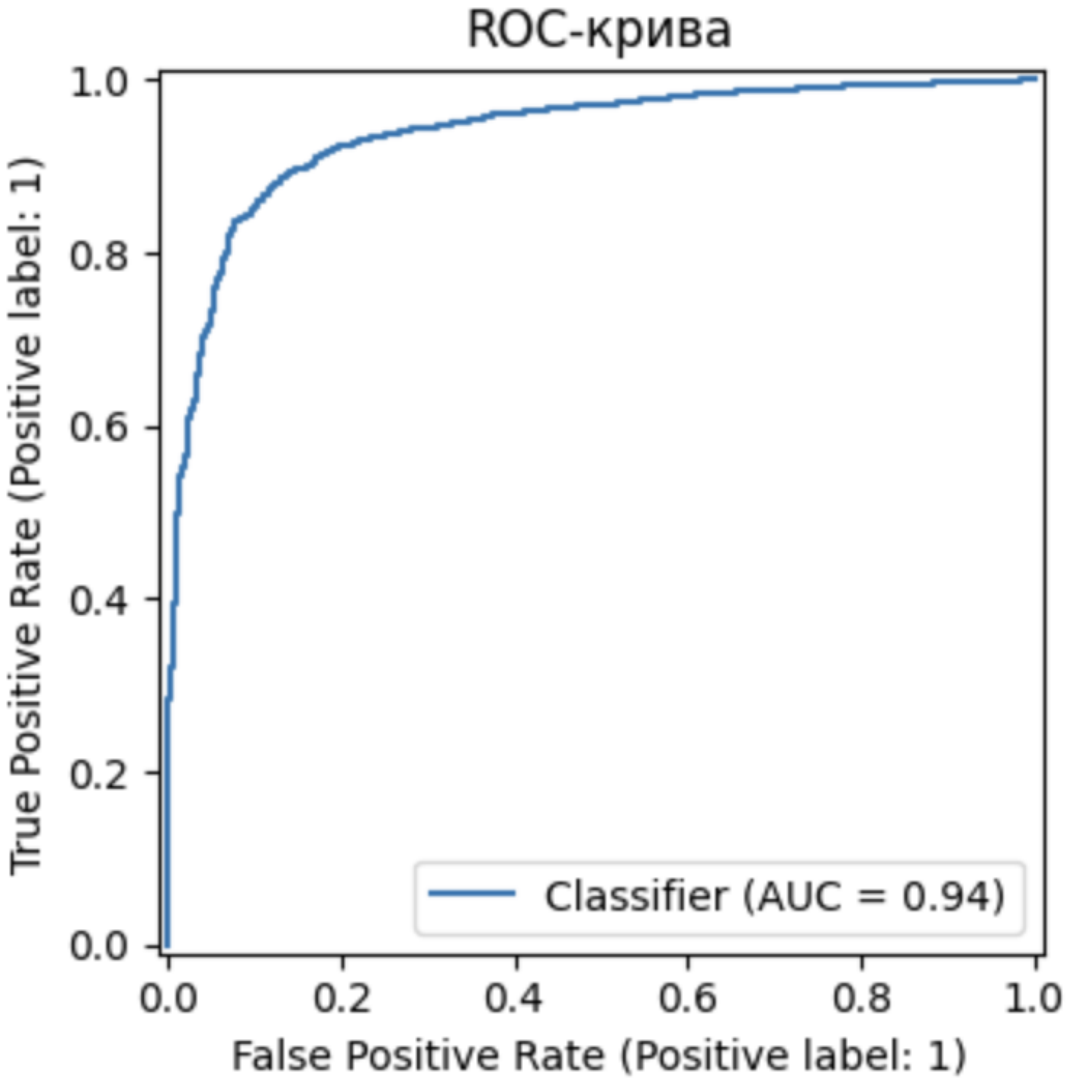
from sklearn.metrics import (roc_auc_score, f1_score, precision_score, recall_score, confusion_matrix, RocCurveDisplay) import numpy as np import matplotlib.pyplot as plt # --- базовий поріг 0.50 ---------- proba = pipe.predict_proba(X_test)[:, 1] y_pred = (proba >= 0.50).astype(int) print("ROC-AUC :", roc_auc_score(y_test, proba)) print("F1-score:", f1_score(y_test, y_pred)) print("Precision:", precision_score(y_test, y_pred)) print("Recall :", recall_score(y_test, y_pred)) print("Confusion matrix [[TN FP] [FN TP]]:\n", confusion_matrix(y_test, y_pred)) # --- підбір порогу для макс. F1 ---------- thr_grid = np.linspace(0.05, 0.95, 91) f1_vals = [f1_score(y_test, (proba >= t).astype(int)) for t in thr_grid] best_t = thr_grid[np.argmax(f1_vals)] best_f1 = max(f1_vals) print(f"\nОптимальний поріг: {best_t:.2f} | F1 = {best_f1:.3f}") # --- візуально ---------- fig, ax = plt.subplots(1, 2, figsize=(10,4)) # 1) F1 vs threshold ax[0].plot(thr_grid, f1_vals, marker='o', ms=3) ax[0].axvline(best_t, ls='--', c='red', label=f'best = {best_t:.2f}') ax[0].set_xlabel('Поріг'); ax[0].set_ylabel('F1'); ax[0].legend() ax[0].set_title('Крива F1-score') # 2) ROC-крива RocCurveDisplay.from_predictions(y_test, proba, ax=ax[1]) ax[1].set_title('ROC-крива') plt.tight_layout(); plt.show()

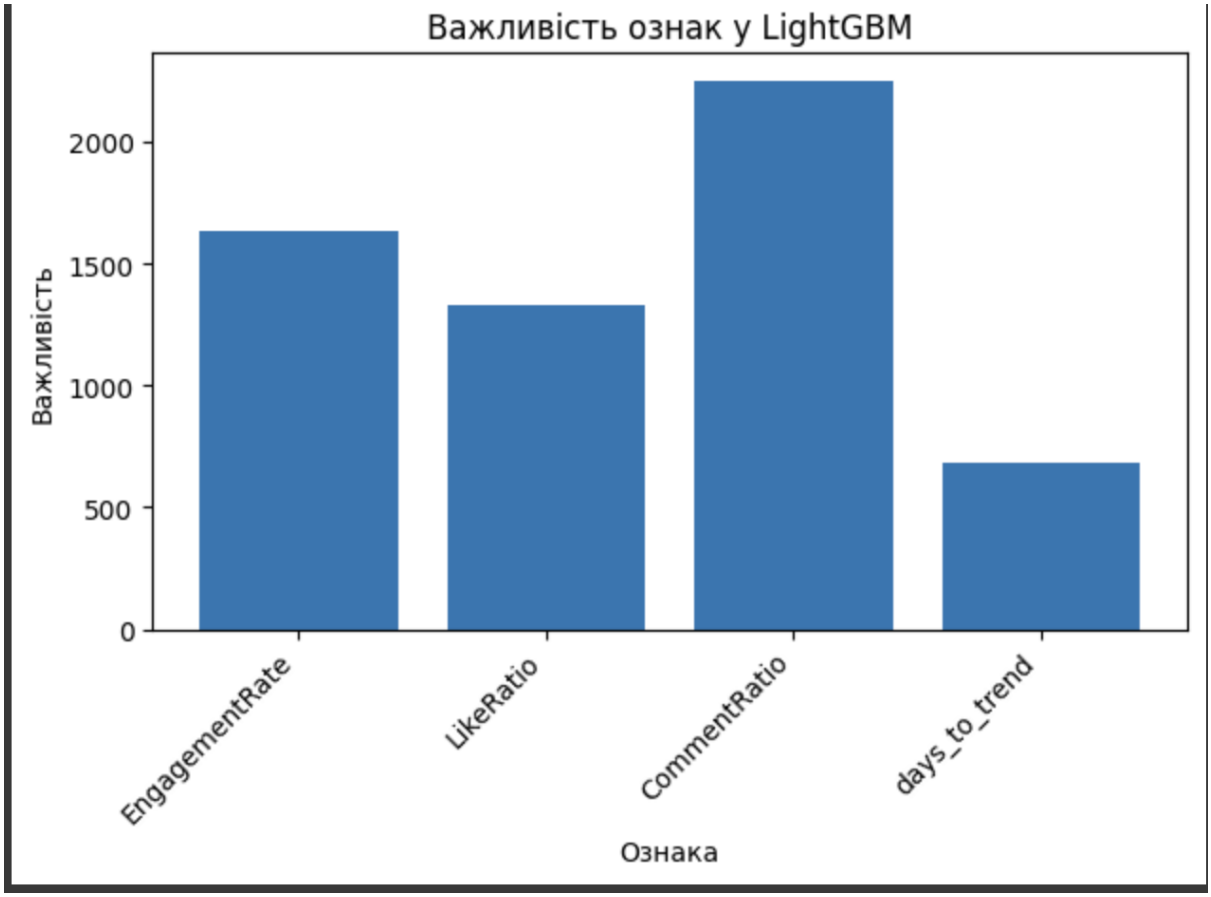
import joblib, pathlib, json MODEL_FILE = 'trend_hit_v2.pkl' FEATURES = NUM + CAT + [TXT] # перелік, який бачить Flask joblib.dump(pipe, MODEL_FILE) pathlib.Path('features.json').write_text(json.dumps(FEATURES, ensure_ascii=False)) print('✅ Збережено:', MODEL_FILE)

Додаток В


Матриця невідповідностей (threshold = 0.50)
Факт
True 0
True 1
0
0
535
1291
Pred 0
Pred 1
Прогноз
1200
1000
800
600
400
200
0


Крива F1-score
best = 0.35
F1
Поріг
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.4
0.6
0.8


ROC-крива
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.94)
0.0
0.2
0.4
0.6
0.8
1.0


Важливість ознак у LightGBM
Важливість
2000
1500
1000
500
0
EngagementRate
LikeRatio
CommentRatio
days_to_trend
Ознака